Volume 28 Number 2 July 2004 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

The Slovene Society Informatika, Ljubljana, Slovenia

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Transformations for Architectural Restructuring

Vincenzo Ambriola
Dipartimento di Informatica, Università di Pisa,
via F. Buonarroti 2, 56127 Pisa, Italy
e-mail: ambriola@di.unipi.it

Alina Kmiecik
Instytut Informatyki, Politechnika Łódzka,
ul. Sterlinga 16/18, 90-217 Łódź, Poland
e-mail: akmiecik@ics.p.lodz.pl

Keywords: software architecture, model transformations, restructuring, software quality improvement

Received: December 1, 2003

Model-driven engineering reaches more and more followers and gradually grows up as an incoming solution to the software-intensive systems production. Architectural modeling (or design) seems to play a fundamental role in this kind of development not only because of its very nature but also because of its impact to the final product structure and behavior as well as the user requirements satisfaction. Incremental and iterative character of architectural design sets a special attention to the aspects of architecture model restructuring, where architectural transformations are the key architect's instrument for introducing quality dedicated changes. Since architectural design constantly grows in complication because of systems complexity, there is a substantial need to support the architect during model architectural transformations.

This paper presents our efforts in defining model-level architectural transformations. It provides a definition and a classification of architectural transformations and describes the semantics of three selected transformations: for component moving, for component splitting and for class splitting. To provide some view of transformations definition complexity it lists informal description for T_SPLIT_CLASS preconditions and post-conditions and presents their formal OCL documentation in Appendix. An example of employing the transformations to improve the architecture of an industrial Geographic Information Web System (WebGIS) is also given.

1 Introduction

Looking at civil engineering we can notice that the major attention during the development of a product (e.g., a building or a mechanical machine) is focused to architectural design. The reason for a special treatment of architectural design comes from the fact that very few things can be done when a building or a machine is already built. What is more, any architectural change applied to a physical construction is expensive, risky and can cause several side-effects that in turn may lead to a serious failure or even a damage. Therefore, in civil engineering, any changes to the product architecture are rather applied to the abstract model than to the real physical product.

In software engineering we observe a completely different approach. Common fascination in re-engineering results in a situation where the majority of software architecture transformations relate to the code – in fact, an already developed product. Software architects may gain the impression that it is better to develop a building first and then move the walls and bricks, in order to improve its structure and properties. What is more, this "re-" approach boils down architectural transformations to the refactorings of low-level structures (i.e., implementation classes) and their interconnections, leaving out the issues related to component organization and deployment. This, in turn, limits the extent of such transformations and makes them incomplete, especially from the perspective of "4+1" architectural view model [11].

In our research we wish to take advantage from the lesson learned in civil engineering and to define architectural transformations by means of changes applied to the software model rather than to its code. Following the "4+1" view model and the architecture-centric development process [8] we adapt UML (Unified Modeling Language) based software architecture representation and identify architectural transformations in terms of UML model transformations. We use the Object Constraint Language (OCL) as a formalism for describing transformations pre-conditions and post-conditions. Although we are aware that UML is still not commonly considered an Architectural Description Language (ADL), we claim that it has some valuable points that none current ADL has – i.e., it is widely used in industry and has a rich tool support including model-to-code as well as code-to-model transformations. Thanks to these we: 1) benefit from a common development practices, 2) capitalize on the large amount of research material, 3) do not fall in

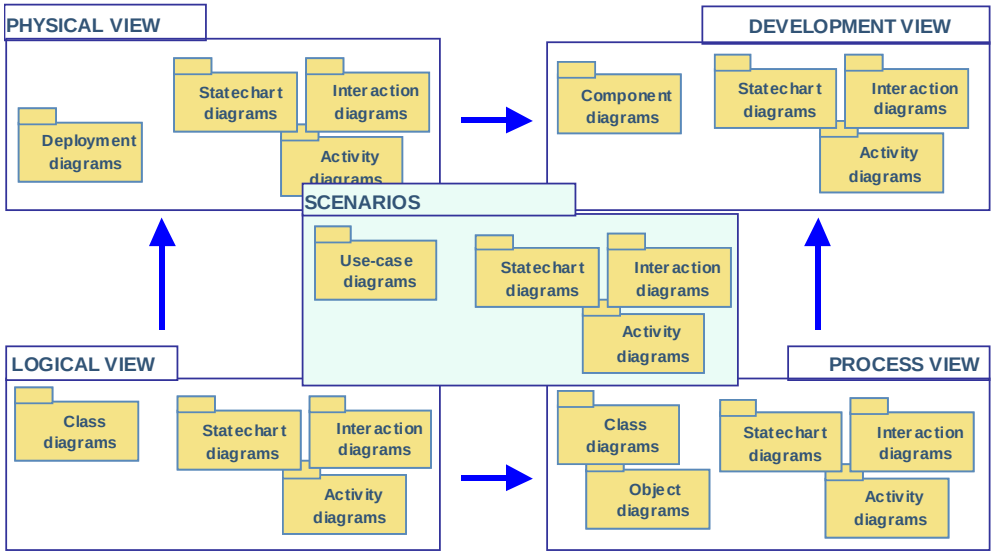


Figure 1: The "4+1 view model and its mapping to UML diagrams.

the trap of imaginary thinking by keeping contact to the ground and 4) are able to provide a sensible comparison of our transformations results from the perspective of running systems.

In this paper we present the outline for model-level architectural transformations and give a closer look to three transformations: for component moving, for component splitting, and for class splitting, to which we define set of pre-conditions and post-conditions. We also present a case study where we apply these transformations to an industrial Geographic Information Web System (WebGIS). The WebGIS example in a real context demonstrates the usefulness of model-level transformations during software architecture improvements. The case study also shows that in some cases model-level architectural transformations are the only way to perform complex software architecture modification.

The rest of the paper is organized as follows: Section 2 discusses underlying UML based software architecture representation; Section 3 provides the definition and classification of model-driven architectural transformations. The semantics for three selected transformations is given in Section 4. Section 5 introduces the WebGIS example: it shortly describes a system, presents the architectural bottleneck, and describes the architectural improvements achieved by means of model-level architectural transformations. Section 6 discusses the results of the WebGIS case study. Section 7 points out related works. Conclusions and future work close the paper.

2 Subject of transformations

Following the most popular definition proposed by Bass [2] we consider software architecture as *"a structure or structures of the system which comprise software components, the externally visible properties of those components and relationships among them"*.

The "4+1" view model [11] defines five views, which together present the overall architectural concept of the software. They are: a logical view, a process view, a development view, a physical view and the scenarios. The UML dedicated Rational Unified Process (RUP) [8] generally adapts this concept with some minor changes to the names (the physical view is called deployment view and the development view is called implementation view). As a result, in UML driven projects it is possible to represent particular views of "4+1" architectural model with a certain set of static and dynamic UML diagrams: the logical and process views may require the class diagrams, the object diagrams and related interaction diagrams, the activity diagrams and the state machine diagrams; the development view can be described by component diagrams and related interaction diagrams, activity diagrams and state machine diagrams; finally, the physical view description is completed with a set of deployment diagrams, interaction diagrams, activity diagrams and state machine diagrams[1].

The adopted "4+1" view model presented in Figure 1 hints at one important property of software model – there exists strong coherence between architectural views that does not allow to judge and modify them in separation. It especially comes obvious when we put together UML diagrams describing particular views. For example, the elements from component diagram are located to the nodes of deployment diagram that define the system topology. At the same time, they are also related to elements of class diagram, which compose their internal structure. This architectural graph is additionally complicated by dynamic aspects surrounding particular structural diagrams and shaping the behavior of modeled structural elements (see also the simplified UML metamodel given in Figure 2 and Figure 3 to take a closer look to UML model elements relationships).

[1] We intentionally do not mention use-cases to enhance understandability of software architecture representation as well as transformations classification.

On the other hand, the only way to perform controlled transformations to such a complex graph is to identify transformation drivers (that is, a node or an edge that an overall graph change should start from). We have noticed that for each view there is exactly one UML element (classifier) that is the central point of a given presentation: a node, a component and a class (a stereotyped class in a case of process view) for the physical, the de-

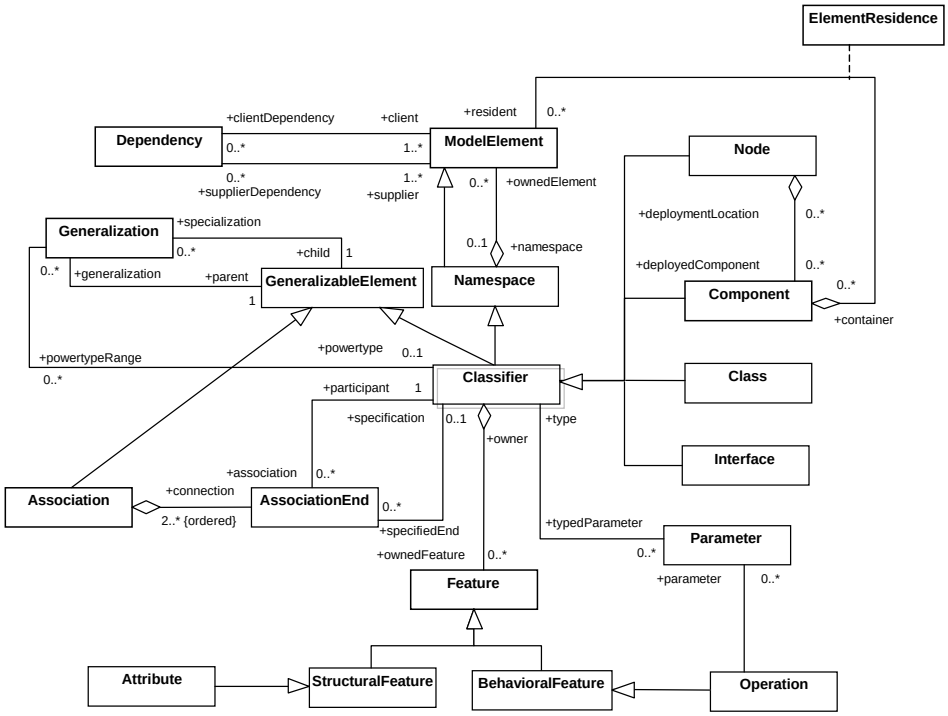


Figure 2: The simplified UML meta-model: structural aspects.

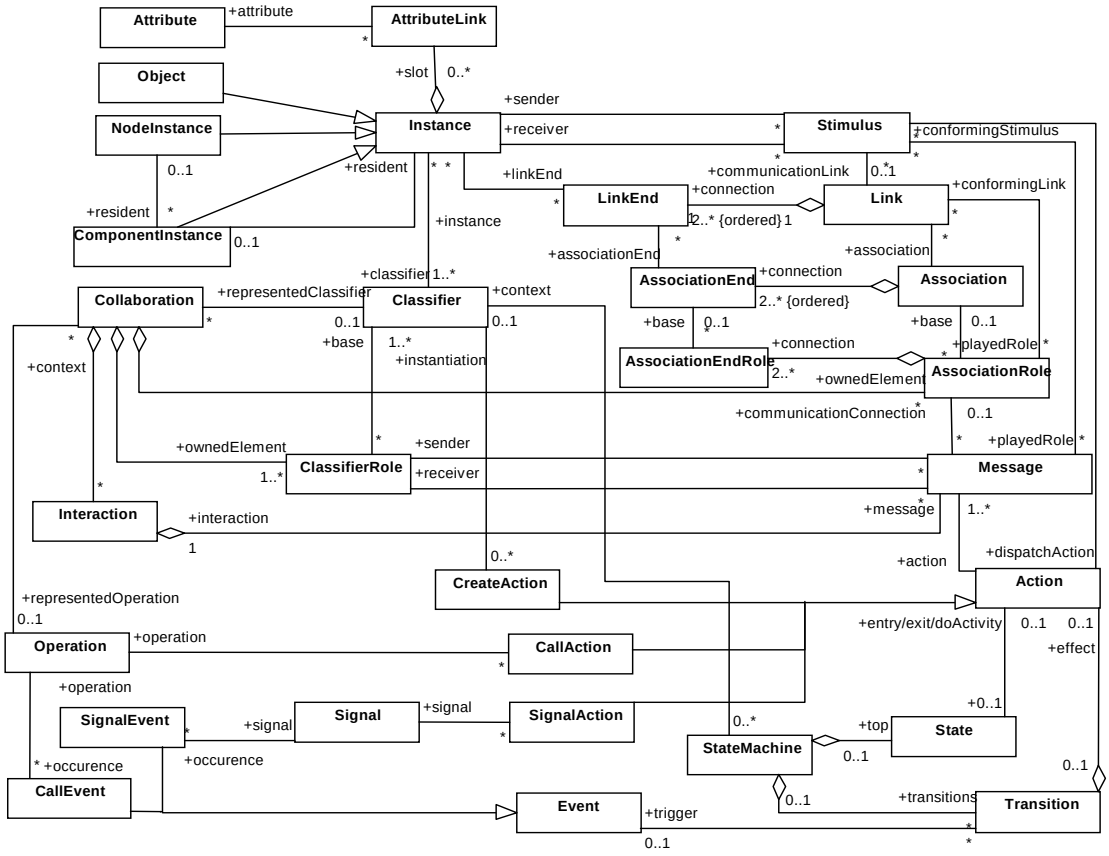


Figure 3: The simplified UML meta-model: behavioural aspects.

velopment and the logical/process view respectively. Since these three classifiers are closely related to <<realize>> and <<deploy>> relationships, they all together compose the nesting hierarchy (similar to a Russian doll) with the high level dedicated to deployment issues, the middle level concerning components and the low level for logical organization and processes. We found out valuable to classify our transformations with respect to such a hierarchy.

3 Architectural transformations

In [1] we identified some key architectural transformations and proposed their classification to the high level, the middle level and the low level transformations, according to the three levels of UML elements nesting hierarchy. We also found out that these transformations are the core of architectural design and a good mechanism for software characteristics management i.e., for software quality improvement and non-functional requirements (NFRs) satisfaction.

In this paper we continue on this line defining architectural transformations by means of UML 1.4 meta-model [16] elements modification. Since we found out that the great majority of software inadequacies (that is, the unfeasibility of the software to meet the user needs) take their roots from non-functional requirements absence and bad quality rather than from lack in functionality or wrong modeling practices, we take for grant that the input UML models for architectural transformations are consistent (i.e. no element in a model exists without relationships to other model elements) and have all functional requirements well modeled. Therefore we set up model consistency and system functionality preserving constraints on architectural transformations.

The last decision restricts the set of previously defined transformations to those that do not influence the system functioning (that is, its externally observable behavior). Moreover, in order to guarantee software functionality invariance we also assume that only the elements of *Classifier* kind may be the direct subject of architectural transformations. Of course, this does not mean that other model elements like relationships, state machines or collaborations and, what follows – the internal behavior of the system - do not change. In truth, these elements are highly influenced when transformations are executed to acceptable model elements. As a consequence, the role of architectural transformations discussed in this paper comes down to architectural modification for quality improvement and non-functional requirements satisfaction. Nevertheless, the requests for quality enhancement and non-functional requirements fulfillment are not a condition set on the transformations themselves, which are considered here only the instruments used to gain certain quality goals. Instead, we propose to fit each architectural transformation with the likely values of its impact to particular quality attributes and use these values as indicators, in the process of composing suitable sequence of transformations with respect to their optimal influence to the software quality. We define *architectural transformation* as:

a transformation applied to the software architecture model that results in a new software model offering the same functionality.

The proposed definition reminds a definition of refactoring [7]. In truth, each model-level architectural transformation can be identified as a specific refactoring dedicated to the software abstraction higher than a code structure. Nevertheless, it is important to notice that the software architecture model transformations are not any kind of UML model refactorings [15] at all. First of all they differ in a goal that for architectural transformations is focused to the software architecture change and indirectly – the general product quality improvement rather than to the code readability or system maintainability enhancement. Secondly, they differ in the extent of their influence. The refactoring methods take their roots from re-engineering concept and thus they apply only to the low level of software representation (i.e. classes, their relationships and behavior). They do not take care of

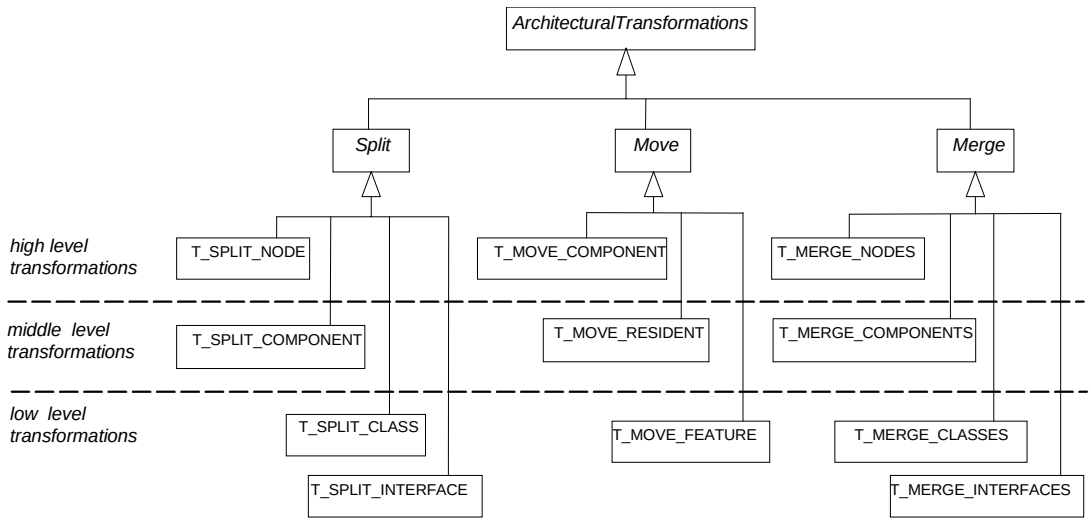


Figure 4: The catalogue of architectural transformations.

deployment and implementation issues, which make up, among others, the architectural core. However, this approach is unacceptable for the software architecture modification, which needs to encompass design, implementation, deployment as well as processing issues to be considered as a whole. What is more, since no model element exists without relationships to elements from neighboring upper and lower levels of software architecture representation, the architectural transformations need to take a focus on a fact that even minor changes at one level may invoke a cascade of changes to the remaining levels of the software representation.

The definition given above characterizes architectural transformations in general and indicates the constraints that must hold for each transformation. These constraints are further encoded in pre- and post-conditions of particular transformation. Figure 4 shows the catalogue of proposed architectural transformations. The elements in italics are not the transformations at all and represent groupings, which collect the transformations with similar functioning, constraints, or rules.

4 Concretizing architectural transformations

Since we present a UML-driven viewpoint to the software architecture, it seems to be a natural consequence that we define architectural transformations by means of UML meta-model (a simplified UML meta-model is shown on Figure 2 and Figure 3) and use UML built-in Object Constraint Language [16] to specify their pre- and post-conditions. By using OCL we entirely integrate our approach with UML-line and can consider transformations as a UML meta-level operational extension.

Because of the paper length restrictions, we cannot present the whole set of architectural transformations and we only discuss in short the transformations that we further use in Section 5 for the case study. However, to give the reader some view of architectural transformations definition complexity we provide in Figure 5 an informal description of the T_SPLIT CLASS pre-conditions and post-conditions (see the Appendix for formal OCL definition of this transformation). For a complete set of architectural transformations, their definition, the pre-

preconditions:

1. *F set is not empty and does not cover the whole set of Features owned by C*
2. *no Feature from F set is not used outside C*
3. *any Operation from F set can be related only to the self-message*
4. *no Operation from F set can be related to the CallAction of a State or Transition of a StateMachine aggregated to the Class other than C or the Operation owned by a Class other than C*
5. *C does not own other Classes as a Namespace*
6. *C does not realize Interfaces*

postconditions:

1. *C2 is of C stereotype and has the same values for its attributes as C has but the name*
2. *C and C2 are participants of a binary association*
3. *an AssociationEnd corresponding to C2 Class in a binary Association established between C and C2 Classes has an attribute isNavigable set to true if C is a base of any ClassifierRole which is sending a message with a CallAction whose Operation belongs to C2*
4. *the AssociationEnd of binary association established between C and C2 and corresponding to C has an attribute isNavigable set to true if C is a base of any ClassifierRole which is sending a message with a CallAction whose Operation belongs to C*
5. *C is a client only to these classes, to which it sends messages*
6. *C2 is a client only to these classes, to which it sends messages*
7. *C2 participates to Associations which reflect the Associations to which C Class is participant of*
8. *C2 is a resident of all Components to which C is a resident of*
9. *for each Object that origins from C there exists an Object that origins from C2 and that is linked to it according to an Association between C and C2*
10. *if an Object O that origins from C is a resident of ComponentInstance(s) then an Object that origins from C2 and has a link to O is also a resident of that*
11. *if C is a base for any ClassifierRole R in a collaboration then C2 is a base for any ClassifierRole R2 in this collaboration and R and R2 are related to the AssociationEndRoles contained in an AssociationRole corresponding to Association between C and C2*
12. *each Message with a CallAction related to Operation from C2 has its receiver set to ClassifierRole based on C2*
13. *each Stimulus with a CallAction related to an Operation from C2 Class has a receiver set to if C aggregates StateMachine S then C2 aggregates StateMachine*
14. *a StateMachine aggregated in C has no transition which has an event trigger handled by C2*
15. *a StateMachine aggregated in C2 has no transition which has an event trigger handled by C*
16. a StateMachine aggregated in C contains a stateVertex of a StateMachineState type, which references to StateMachine of C2
17. a StateMachine aggregated in C contains a stateVertex of a StateMachineState type, which references to StateMachine of C2

Figure 5: The natural language (informal) description of the T_SPLIT_CLASS transformation.

conditions, the post-conditions and an explanation we refer to [9].

4.1 Transformation for class splitting

First we present a low level transformation for class splitting (T_SPLIT_CLASS). It relies on partitioning a C *Class* given to the input of transformation to C and C2 *Classes* with respect to the certain subset of *Features* owned by C *Class*. The splitting line (that means, the subset of *Features* to be moved from C *Class* to a new C2 *Class*) can be recognized with some behavior knowledge (i.e., the collaborations related to C *Class*). However, the selection of *Features* to be moved from C *Class* is not a subject of the T_SPLIT_CLASS transformation. We shift this responsibility onto a higher-level algorithm that is in charge to adjust the *Feature* set with respect to the particular quality requirements.

At first glance, T_SPLIT_CLASS has a direct impact on the software structure only. However, this impression turns out to be false, when we take a closer look. The transformation for class splitting significantly influences several dynamic issues. For instance, all *Objects* originating from a C *Class* have to be split with regard to a set of *slots* corresponding to the *Attributes* listed in a splitting line; if C aggregates a *StateMachine*, then it must be modified and a new S*tateMachine* for C2 *Class* must be created in order to meet a new software system organization and preserve overall model consistency; the collaborations, to which C *Class* participates need to be changed with regard to the *ClassifierRoles* and the *Interactions* and the set of *Instances* and *Stimuli* as well. The low level transformations usually highly influence other levels of software architecture representation in order to ensure overall model consistency. In this case only the middle level is affected by the insertion of a new <<reside>> *Dependency* between C2 *Class* and the *Components* to which C is *resident* of as well as establishing connections between the *Instances* originating from these *Classifiers*. However, for the other low level transformations it is common that they modify dependencies even at a high level of software architecture representation (see T_MOVE_FEATURE in [9]).

4.2 Transformation for component splitting

T_SPLIT_COMPONENT is a representative of middle level transformations. It is used to crumble the scope of a given component implementation. Since we are working on a model, this transformation boils down to the *Component* classifier splitting with respect to the set of its *residents*. The arguments of the transformation are a C *Component* being split and a subset of C *residents* which are to be moved from it.

As in the case of class splitting, T_SPLIT_COMPONENT forces many changes to the dynamic model elements. From the meta-model point of view this transformation:

1. creates a new C2 *Component*, destroys aggregations between *Component* C and a certain subset of its *residents* and sets up new aggregations between these *residents* and a C2 *Component*;
2. modifies some *Relationships* by substituting their participants from C to C2 *Component*, if necessary;
3. creates the *ComponentInstances* of C2 *Component* by splitting the *ComponentInstances* originating from C *Component* according to the set of the *Instances* originating from the *residents* moved from C to C2;
4. alters all *Collaborations* to which C is participant of (i.e., to which C is a *base* for a *ClassifierRole* owned in a given *Collaboration*) by adding the new *ClassifierRoles* based on C2 *Component*, changing the *types* of some *AssociationEndRoles*, changing the *sender* or the *receiver* of particular *Messages*, changing the participants of some *Links* with respect to the changes done to their corresponding *Associations*, and modifying *Stimuli* related to the modified *Messages*;
5. creates a new *StateMachine* and relates it to C2 *Component* (this change is done only in a case when C aggregates at least one state machine). It moves certain *Transitions* and *StateVertexes* from the *StateMachine* of C to the *SateMachine* of C2 following the premises based on transitions *triggers*. For both *StateMachines* it also creates *SubmachineStates* reflecting the opposite *StateMachines* for symmetry and functionality preserving purposes;

Transformation for component splitting not only affects components, but also a higher level organization. For instance, for the system model consistency it adds a new <<deploy>> *Dependency* between C2 and a *Node* classifier, to which C is applied.

4.3 Transformation for component moving

In a group of high level architectural transformations we can find, among others, the T_MOVE_COMPONENT transformation, that is used to change the deployment of a given component. It has given a source *Node* classifier N, a target *Node* classifier N2 and a *Component* classifier C as arguments. When we look at deployment diagram after execution of this transformation, we will write down the change in a *client* of <<deploy>> *dependency* to which C is a *supplier* or the change in the nesting of N and N2, dependently of the adopted form of presentation. The side-effect of transformation may be also a change in *dependencies* and *associations* between the *Nodes*. Additionally, the StateCharts diagrams presenting the *state machines* of N and N2 *Nodes* may be updated as well as some collaborations, to which N and N2 provide the *roles* or *instances* may be altered. These modifications rely mainly on changing a *base* classifier of some *roles*, which previously were based on N *Node* and currently come down from N2 *Node*. Nevertheless, no *instance* originating from N or N2 *Node* is modified, even it may participate in quite different *links*.

T_MOVE_COMPONENT transformation concerns high level representation of software architecture and does not need to influence other levels to keep the model consistency.

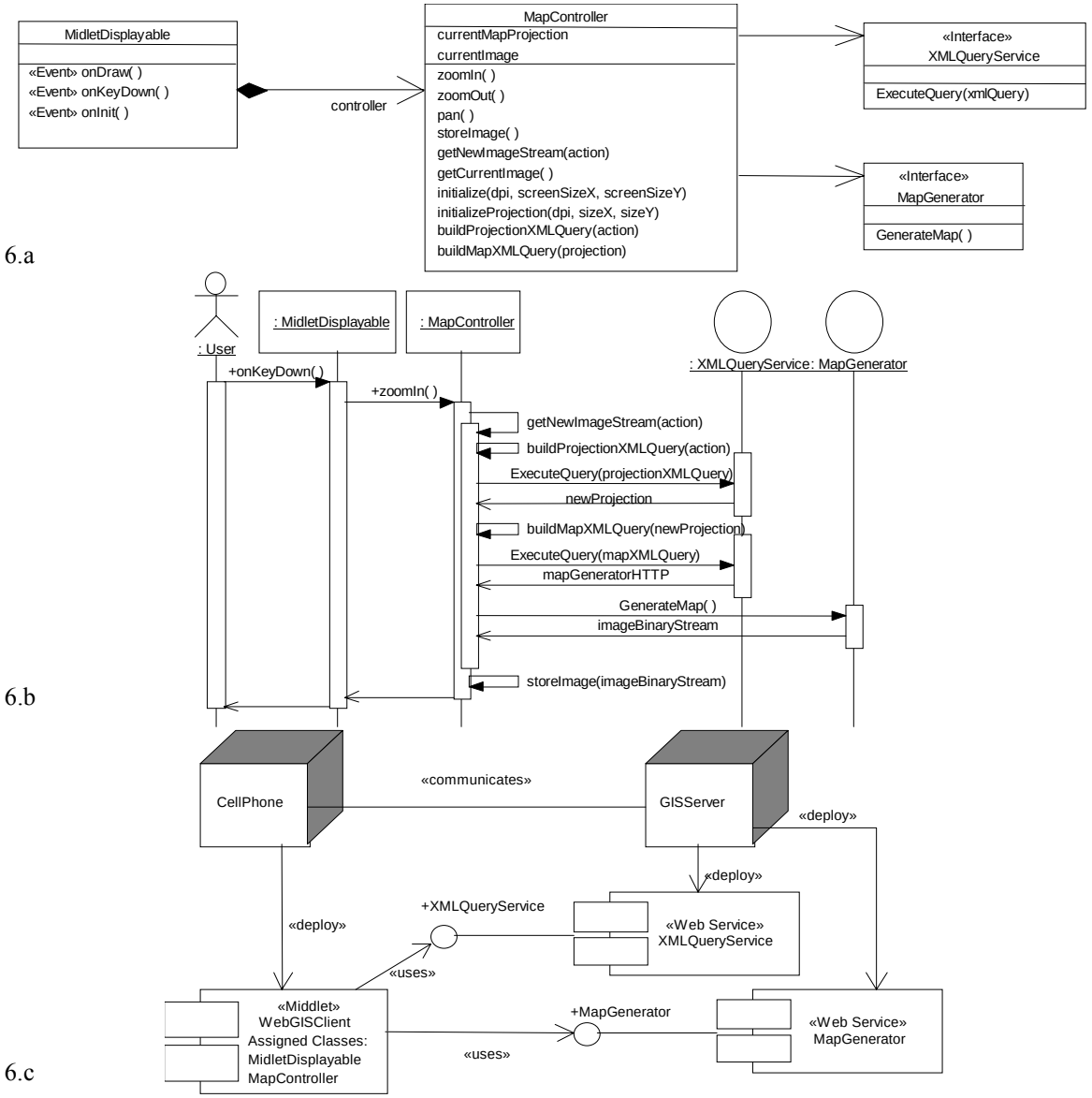


Figure 6: The simplified architecture of the WebGIS system.

5 An example

One of the reasons we decided to move towards UML-based software architecture representation and define architectural transformations by means of UML model transformations was the opportunity of empirically proving that model-level architectural transformations can be successfully used in a real context. In this section we take advantage from this approach and present an industrial case study where we use our transformations to remove an architectural bottleneck.

5.1 Architectural bottleneck

In our case study we used an industrial WebGIS system based on a client-server architecture with a fat client. The role of the server is to provide Web services which are exploited by the client applications to serve the basic GIS functionality. A client communicates to the server using SOAP protocol and sends an XML query each time it needs to use a service. The query is further forwarded by *XMLQueryService* component to the server services. As a reply the client gets the XML answer generated by a suitable service. This architecture has been successfully implemented and works very well for desktop and WWW clients. Since the WebGIS system consists of 14 services, for the sake of brevity we limit this paper discussion to *MapGenerator* service.

The role of *MapGenerator* service is to provide images of some cartographic map according to the client parameters sent as an XML query (e.g., dpi, screen size, cartographic projection type, and so on). For the clients that already connect to WebGIS the *MapGenerator* service reaches very good speed, by generating up to 25 images per second (at a 400x400 pixels resolution).

The company in charge of WebGIS decided to extend the kind of WebGIS clients in order to provide a cartographic map viewer able to work on GPRS cellular

phones. Developers took advantage from the work done so far and reused the existing system architecture. The very simplified UML model of this architecture reconstructed from the code and completed with the developers knowledge is shown in Figure 6. It composes of three diagrams: 6.a - a class diagram with architecturally significant classes and interfaces, 6.b – a sequence diagram describing the essential interactions between the classes and interfaces from a class diagram, 6.c – a deployment diagram with the *client* and the *server* nodes, components deployed to them and some residents nested in the components as well.

As we can see from the figure, the *CellPhone* node has only one component deployed – the *WebGISClient* middlet written in Java that has two resident classes: *MiddletDisplayable* and *MapController*. *MiddletDisplayable* is a user interface class responsible for both map display and handling of user-generated events. *MapController* holds the current map state and provides the logic for map navigation. It also uses two interfaces exposed by the server services: *XMLQueryService* and *MapGenerator*, which provide the operations for map projection and map generation.

However, although this architecture has been empirically proven and works very well for other client types, it turned out to be unsuitable for cellular phones: a new map loading takes about 10 seconds, which is not an acceptable response time for the user. For almost a month developers have been searching for the system bottleneck until they realized that the problem was rooted in a GPRS functionality. The GSM cellular phone system provider informed them that GPRS waits approximately 3 seconds each time it initializes an HTTP connection and just after that time it allows any data transfer. As we can see from Figure 6, *WebGISClient* component owning *MapController* class connects three times to the server (to which the components exposing *XMLQueryService* and *MapGenerator* interfaces are deployed) in order to get a map. Since each connection takes at least 3 seconds, the total time explains the critical delay that slows down the client application.

5.2 Using model-level architectural transformations

It became obvious that the existing architecture is no longer suitable for cellular phones. Since the bottleneck was found in the number of calls between the client and the server, the way to significantly improve the system performance was to minimize the number of connections between these nodes during the map transfer. The intuitive solution for the problem was to move the part of *WebGISClient* component responsibilities to *WebGISServer* node.

The first step toward the improved WebGIS architecture was to separate the responsibilities of *MapController* class. We chose the suitable subset of features to be moved from *MapController*, by following the sequence diagram presented on Figure 5. In general it composed of all these operations which use the interfaces exposed by the server components. Having a splitting line for *MapController* class we were able to execute T_SPLIT_CLASS transformation. We obtained *MapController* and *MapControllerB* classes (see Figure 7.a).

Then we applied the T_SPLIT_COMPONENT transformation to the *WebGISClient* component. It resulted in *WebGISClient* and *WebGISClientB* components *MiddletDisplayable* and *MapController* classes were left on *WebGISClient* component, and *MapControllerB* class became a resident of a newly created *MapControllerB* component. It is important to notice that *MapGISClientB* component was forced to expose an interface *IMapControlerB* that was realized by *MapControllerB* class in order to allow *MapController* class to use operations from *MapControllerB*. The resulted component diagram is shown on Figure 7.c.

Finally we moved *WebGISClientB* component from *Cell*Phone to *WebGISServer* node using T_MOVE_COMPONENT transformation (see Figure 7.d). Thanks to these architectural modifications the cellular phone client has to communicate only once in order to get a new map. A suitable sequence diagram describing the interactions within the modified architecture is shown on Figure 7.b.

The application of three architectural transformations significantly enhanced the system performance. After the implementation of the improved architecture the cellular phone user has to wait only 3 seconds for a map.

6 Discussion

The WebGIS bottleneck concerned the bad component organization and their physical deployment in a case of cellular phone client. It could be detected only when all the three levels of architectural representation were analyzed at the same time. Although the critical point of software behavior was explicitly shown on a sequence diagram (Figure 6.b), it was not considered as such before the classes participating in this interaction had been deployed to certain components.

The proposed architectural modification influenced all three levels of software architecture representation: the classes, because of the need for *MapController* responsibility separation; the components, because of the need for *WebGISClient* splitting; and the nodes - because of the need for changing the component deployment location. For this reason it was difficult to think about the architectural improvement from the code viewpoint, where the implementation and not the architectural aspects are in the area of interests and no information about the components deployment is available.

What is more, from a code level perspective there was no premises that the components organization decreases the system performance. The fact that the client and the server components are written in different languages (Java and C# respectively) additionally complicated the code-level acting. For example, simple code transformations would not have been able to perform the operation of moving Java component from cellular phone to the C# based server, where the library supporting middlets is not available. On the other hand, refactoring methods could have done the good work at a class level but would have

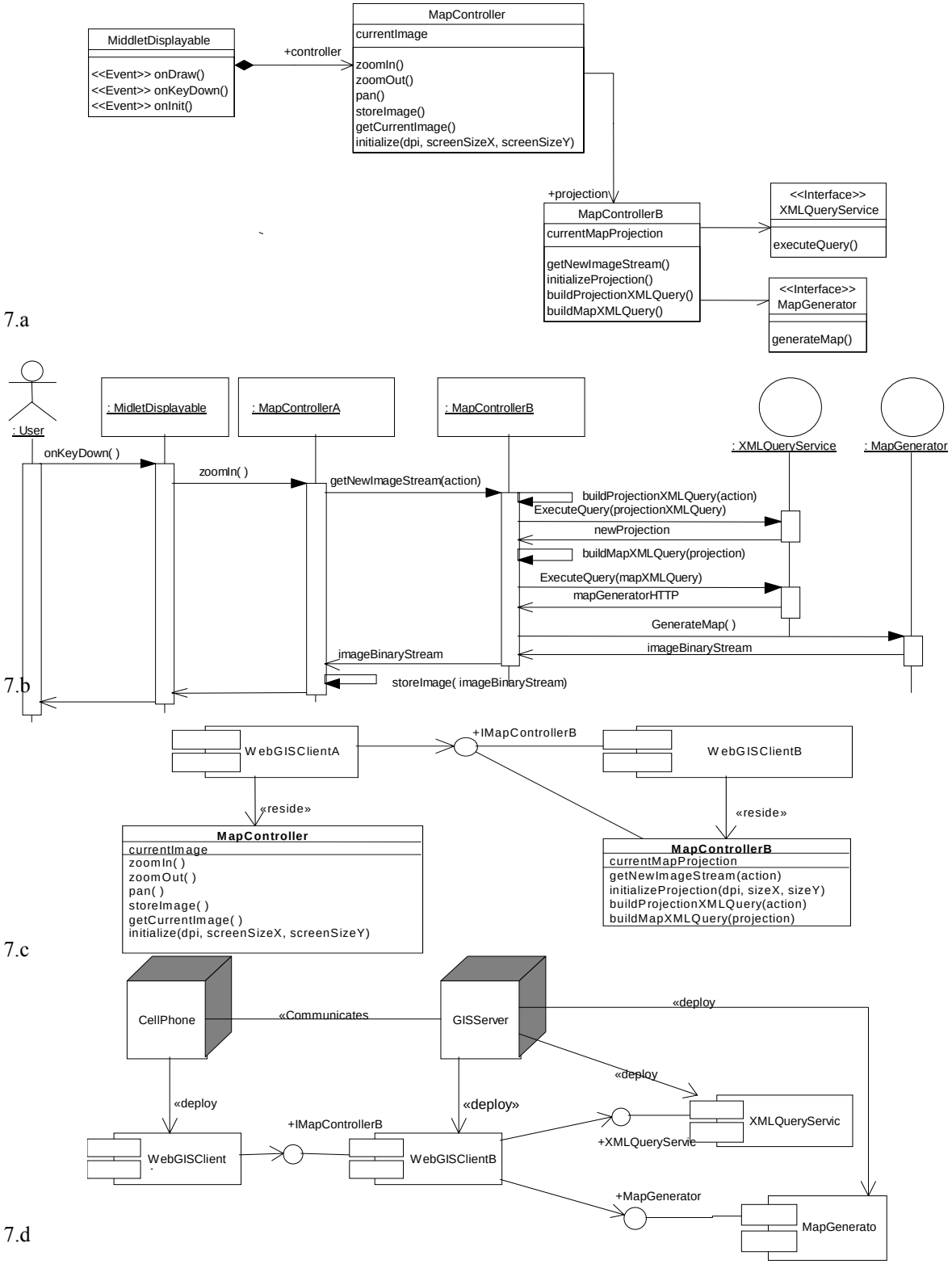


Figure 7: Software architecture after the architectural improvement.

completely missed the higher-level modifications. Thus, in a case of WebGIS system only the model-level approach provided us the appropriate point of view and the flexibility needed to perform necessary architectural improvements.

Finally, the architectural improvements could be successfully performed during forward engineering which would avoid project stopping and thus – speed up the time-to-market.

7 Related works

The software architecture is the subject of intensive research for more than a decade and several approaches to architectural modification have been presented for that time. However it seems that the main research effort has been spent on inventing a general framework for software architecture improvement rather than a definition of a concrete architectural transformations for this goal satisfaction. For instance, Bosch in [3] presents the six step architectural design method. The architectural transformations are considered as one of the steps toward an improved software architecture. Similarly to us, Bosch defines "architectural transformation" as "*a transformation that leads to a new version of architecture that has the same functionality but different values for its properties*" [4]. He also proposes five group-based classification for architectural transformations which may be considered to support one of the following refinement actions: imposing architectural style, imposing architectural pattern, applying design pattern, converting non-functional requirements to functionality and distributing requirements. However he does not define any concrete transformation nor present a mechanics for imposing architectural styles or patterns leaving these issues to the software architect competence.

Krikhaar [10] proposes two-phase process for the software architecture improvement: the first phase "extracts" an architecture from the software, calculates quality metrics and analyses the impact of potential changes to the architecture; the second phase transforms the system implementation according to the set of recipes corresponding to the architectural changes. He follows the re-engineering line and defines architectural transformations as code-level transformations. Considering Krikhaar two-phase process we can relate our model-level transformations to the impact analysis phase rather than to his transformations.

Carriere, Woods and Kazman [5] also discuss architectural transformations from the architectural re-engineering perspective. They "extract" software structure from a code and describe architectural elements by means of their static and dynamic features. Then they define transformations in terms of these features modification and map them to a code transformations.

Our understanding of architectural transformations is close to the one proposed by Fahmy and Holt [6] who define architectural transformations as graph rewriting operations applied to the software structure. However, in their research they focus rather on the comprehension of the software structure and its compatibility with the architect's mental model than on the architectural change for requirements satisfaction and the quality properties improvement.

The UML-model approach to the software architecture representation ties us also to the area of model engineering, which is full with proposals for model-level transformations. Nevertheless, the majority of work concerns meta-model level conversions (e.g. from UML model to a code) [12][13] or the transformation-aided transition of UML models between the phases of development process [14]. The work sharing our viewpoint to model-level transformations is presented in [15], where the authors define UML level counterparts of Opdyke's refactorings as behavior-preserving transformations and use OCL to formalize their preconditions and post-conditions. Some interesting proposals for model transformations specification are also given in the latest OMG Query/View/Transformation RFP [17].

8 Conclusion

The case study presented in Section 5 demonstrates the usefulness of model-level architectural transformations in a real context. It shows also that in some cases only a model-level approach can ensure the appropriate point of view and the flexibility needed to perform necessary architectural improvements. Additionally, it indicates that model-level architectural transformations cannot only be considered as a support during forward engineering but also can be successfully applied to re-engineering line.

Our ongoing work is focused on the implementation of a tool for transforming UML software architecture models. Such a tool will allow us to empirically check the appropriateness of our transformation and experiment with their influence to software quality values. We also find out necessary to formally prove that the transformations defined in [9] do not influence software functionality and can be safely used during re-engineering.

Our future work will concern the following issues: 1) investigating the transformations impact to particular software quality attributes, 2) definition of the rules for building a minimization function for particular software quality goals and employing the model-level architectural transformations to this function calculation, 3) architectural transformations automation. We also intend to integrate our tool with the existing design environments and this way provide a computer-aided support for software architecture improvement.

References

[1] Ambriola V., Kmiecik A., Architectural Transformations, *Proc. of the 14th Int. Conference on Software Engineering and Knowledge Engineering*, ACM Press, 2002, pp. 275-278.

[2] Bass L., Clements P., Kazman R., *Software Architecture in Practice*, Addison-Wesley, 1997.

[3] Bosch J., *Design and Use of Software Architectures*, Addison-Wesley, 2000.

[4] Bosch J., Molin P., Software Architecture Design: Evaluation and Transformation, *Proc. of the Engineering of Computer Based Systems Conference*, IEEE Press, 1999, pp.4-11.

[5] Carrière S.J., Woods S., Kazman R., Software Architecture Transformation, *Proc. of the Int. Conference on Reverse Engineering*, 1999, pp. 13-23.

[6] Fahmy H., Holt R.C., Using Graph Rewriting to Specify Software Architectural Transformations, *Proc. of Automated Software Engineering*, IEEE Press, 2000, pp. 187-187.

[7] Fowler M., *Refactoring. Improving the Design of Existing Code*, Addison-Wesley, 1999.
[8] Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process*, Addison-Wesley, 1998.
[9] Kmiecik A., Ambriola V., *Transformations for Architectural Model Restructuring*, Technical Report, University of Pisa, 2004 (to appear).
[10] Krikhaar R., Postma A., Sellink A., Stroucken M., Verhoef C., A Two-Phase Process for Software Architecture Improvement, *Proc. of the Int. Conference on Software Maintenance*, IEEE Press, 1999, pp.371-380.
[11] Kruchten P., Architectural Blueprints – the "4 +1" View Model of Software Architecture, *IEEE Software*, 12, 6, November 1995, pp.42-50.
[12] Oldevik J., Solberg A., Elvesæter B., Berre A., Framework for model transformation and code generation, *Proc. of the 6th Int. Enterprise Distributed Object Computing Conference*, IEEE Press, 2002, pp.181-191.
[13] Peltier M., Bezivin J., Guillaume G., MTRANS: A General Framework, based on XSLT, for Model Transformations, *Proc. of the Workshop on Transformations in UML*, 2001.
[14] Schönberger S., Keller R., Khriss I., Algorithmic Support for Model Transformation in Object-Oriented Software Development, *Concurrency and Computation: Practice and Experience*, vol. 13, 5, April 2001, pp. 351-383.
[15] Sunye G., Pollet D., Traon Y., Jezequel J., Refactoring UML Models, *Proc. of UML 2001*, LNCS 2185, Springer Verlag, 2001, pp. 134-148.
[16] OMG Unified Modeling Language Specification, version 1.4, OMG document formal/01-09-67, http://www.uml.org, 2001.
[17] OMG MOF 2.0 Query/View/Transformation RFP revised submission, http://qvtp.org, 2003.

APPENDIX: Formal transformation definition

Below we present the formal definition of the T_SPLIT_CLASS transformation written in OCL and verified manually using OCL specification [16]. Since we do not have enough space for introducing to OCL language notation, we refer directly to OMG specification [16].

```
T_SPLIT_CLASS(C: Class, F: Set(Feature))

let   top = C->namespaces()->select(n | n.stereotype.name = "systemModel")
let   collaborations: Set(Collaboration) = C.namespace.ownedElement->
      select(c | c.oclIsKindOf(Collaboration))
let   state_machines: Set(StateMachine) = top->allStateMachines()->
      select(sm | sm.context.oclIsKindOf(Class) or sm.context.oclIsKindOf(Operation))
let   messages: Set(Message) = collaborations->forAll(collab | collab.interaction->
      collect(self.message))
let   collaborations: Set(Collaboration) = C.namespace.ownedElement->
      select(c | c.oclIsKindOf(Collaboration))
let   association_end_C: AssociationEnd = C2.association->
      select(ae |  ae.association.connection.participant->includes(C))
let   association_end_C2: AssociationEnd = C.association->
      select(ae |  ae.association.connection.participant->includes(C2))
let   objects: Set(Object) = top->allObjects()->select(oe |  oe.classifier =C)
let   component_instances: Set(ComponentInstance) = top-> allComponentInstances()->
      select(ci | ci.resident->exists(o | o.classifier = C)

pre:
      C.feature->size()>F->size()
  and F->size() > 0
  and F->forAll(f | f.supplierDependency.client->forAll(cl | (if cl.oclIsKindOf(Class) then
      cl = C) or (if cl.oclIsKindOf(Feature) then cl.owner = C)))
  and F->forAll(f | messages->exists(m | m.action.operation = f implies
      message.sender = message.receiver))
  and F->forAll(f | state_machines->exists(s | s.allStates()->
      exists(state | state.entry.operation = f or state.exit.operation = f or
      state.doActivity.operation = f) or s.transitions->exists(t | t.effect.operation = f)
      implies (if s.context.oclIsKindOf(Class) then s.context = C) or
      (if s.context.oclIsKindOf(Operation) then s.context.owner = C)))
  and C.ownedElement->excludes(c | c.oclIsKindOf(Class))
  and not C.clientDependency->exists(d | d.oclIsKindOf(Abstraction) and
      d.stereotype.name = "realize" and d.supplier->exists(s | s.oclIsKindOf(Interface)))

post:
      C2.stereotype = C.stereotype
  and C2.isActive = C.isActive and C2.isAbstract = C.isAbstract and
      C2.isLeaf = C.isLeaf and C2.isRoot = C.isRoot
  and C.association.association->exists(a | a.connection->size() = 2 and
      a.connection->exists(ae | ae.participant = C2) )
  and collaborations->exists(c | c.ownedElement->exists(oe | oe.base =C2 and c.interaction->
      exists(in |in.message->exists(m | F->includes(m.action.operation) and  m.sender = oe))
      implies association_end_C.isNavigable = true
```

```
  and collaborations->exists(c | c.ownedElement->exists(oe | oe.base = C and c.interaction->
      exists(in | in.message->exists(m | F->includes(m.action.operation) and  m.sender=oe))
      implies association_end_C2.isNavigable = true
  and collaborations->forAll(c | c.ownedElement->forAll(oe | oe.base = C and c.interaction->
      forAll(in | n.message->forAll(m | m.sender <> oe implies C.clientDependency->
      forAll(cd | cd.supplier->excludes(m.receiver.base)))))
  and collaborations->forAll(c | c.ownedElement->forAll(oe | oe.base =C2 and c.interaction->
      forAll(in | n.message->forAll(m | m.sender <> oe implies C2.clientDependency->
      forAll(cd | cd.supplier->excludes(m.receiver.base)))))
  and C2.oppositeAssociationEnds->forAll(ae | ae.participant = C or
      ae.participant.oppositeAssociationEnds->exists(ae2 |ae2.participant = C) ) and
      C2.oppositeAssociationEnds->forAll(ae | ae.participant = C or
      C.oppositeAssociationEnds->exists(ae2 | ae2.participant = ae.participant and
      ae.isSimilarTo(ae2))) and C2.association->forAll(ae | ae.association.connection->
      exists(ae2 | ae2.participant = C) or C.association->exists(ae2 | ae2.isSimilarTo(ae)))
  and C2.implementationLocation->forAll(r | r.resident->includes(C)) and
      C.implementationLocation->forAll(r | r.resident->includes(C2))
  and objects->forAll(o | o.linkEnd->exists(le | le.link.connection->
      includes(o2 | o2.classifier = C2) ))
  and component_instances->forAll(ci | ci.resident->forAll(r | r.isKindOf(Object) and
      r.classifier = C implies ci.resident->exists(r2 | r2.oclIsKindOf(Object) and
      r2.classifier = C2 and r2.linkEnd.link.connection->includes(r))))
  and collaborations->forAll(c | c.ownedElement->forAll(oe | oe.oclIsKindOf(ClassifierRole)
      and oe.base = C  implies  c.ownedElement->exists(oe2 | oe2.oclIsKindOf(ClassifierRole)
      and oe2.base = C2 and c.ownedElement->exists(ar |  ar.oclIsKindOf(AssociationRole) and
      ar.connection->exists(c1 | c1.type = oe ) and ar.connection->exists(c2 | c2.type=oe2)
      and ar.base = C.association.association->select(a | a.connection->size() = 2 and
      a.connection->includes(C2))))))
  and collaborations->forAll(c | c.interaction->forAll(in | in.messages->
      forAll(m | m.action.operation.owner = C2 implies m.receiver.base = C2 and
      m.communicationConnection.connection->includes(m.sender)and
      m.communicationConnection.connection->includes(m.receiver))))
  and collaborations->forAll(c | c.interaction->forAll(in | in.messages->
      forAll(m | m.conformingStimulus->forAll(s | s.action.operation.owner = C2
      implies s.receiver.classifier = C2 and  s.receiver.linkEnd->
      includes(le | le.link.connection->includes(s.sender))))))
  and C.behavior->size() = C2.behavior->size()
  and not C.behavior->exists(sm | sm.transitions->exists(t | t.trigger.operation.owner = C2
      or t.trigger.signal.reception->exists(r | r.owner = C2 ))
  and not C2.behavior->exists(sm | sm.transitions->exists(t | t.trigger.operation.owner = C
      or t.trigger.signal.reception->exists(r | r.owner = C ))
  and C.behavior->forAll(sm | sm.allStates()->exists(s | s.oclIsKindOf(SubmachineState) and
      C2.behavior->includes(s.submachine)))
  and C2.behavior->forAll(sm | sm.allStates()->exists(s | s.oclIsKindOf(SubmachineState) and
      C.behavior->includes(s.submachine)))union(self.transitions->collect(self.target))

context Namespace :: namespaces(): Set(Namespace)
post: result = self.namespace->union(self.namespace.namespaces())

context Namespace :: allStateMachines() : Set(StateMachine)
post: result = self.ownedElement->select(oe| oclIsKindOf(StateMachine))->
      union(self.ownedElement->select(ns | ns.oclIsKindOf(Namespace))->allStateMachines())

context StateMachine:: allStates(): Bag(State)
post: result =self.transitions->collect(self.source)->
      union(self.transitions->collect(self.target))

context Namespace :: allObjects() : Set(Object)
post: result =self.ownedElement->select(oe| oe.oclIsKindOf(Object))->
      union(self.ownedElement->select(ns | ns.oclIsKindOf(Namespace))->allObjects())

context Namespace :: allComponentInstances() : Set(ComponentInstance)
post: result = self.ownedElement->select(oe | oe.oclIsKindOf(ComponentInstance))->
      union(self.ownedElement->select(ns| ns.oclIsKindOf(Namespace))->allComponentInstances())

context Classifier :: collaborations(): Set(Collaboration)
post: result = self.namespace.ownedElement->select(c | c.oclIsKindOf(Collaboration))

context Collaboration :: messages(): Set(Message)
post:  result = self.interaction->collect(self.message)

context AssociationEnd :: isSimilarTo(ae: AssociationEnd)
post: result = ((self.aggregation = ae.aggregation) and (self.name = ae.name) and
      (self.changeability = ae.changeability) and (self.ordering = ae.ordering) and
      (self.isNavigable = ae.isNavigable) and (self.multiplicity = ae.multiplicity) and
      (self.targetScope = ae.targetscope) and (self.visibility = ae.visibility ))
```

Software Engineering: The Trend

Ayaz Isazadeh
Department of Computer Science, Tabriz University, Tabriz, Iran
Phone: +98 411 334 4015, Fax: +98 411 334 2102
E-mail: isazadeh@tabrizu.ac.ir

Keywords: software engineering, requirements engineering, formal methods

Received: October 27, 2003

This paper presents the author's view of the current trend in the world of software engineering. Software engineering has already taken the responsibility of providing the necessary tools for organizing, structuring, and making efficient use of the huge volume of information currently floating around. The responsibility of software engineering will increase dramatically by the exponential increases expected in the volume of information. The volume of information, in every branch of science and technology, is getting so high that without a software engineering tool cannot be of any use. This is an extreme power, and responsibility, for software engineering.

The paper provides an analysis of the increasing power and responsibilities associated with software engineering as an engineering discipline, points out the catastrophic results of failures, as well as significance of accuracy and correctness in this discipline, and concludes that software engineering cannot afford to go wrong. The paper then investigates the significance of mathematical foundations and formal approaches to software engineering, specially in the area of software requirements specification. Finally, the paper recommends using a visual formalism, designed based on a representationistic approach, as a sound and simple foundation, for a sound and successful software engineering practice. An example of such a formalism concludes the paper.

1 Introduction

Living in an age of information, we have collected and stored a huge volume of information in different media, all around the world. And yet, we are just starting to realize the vast dimensions of the universe, the huge volume of information out there, and the astronomical world of unknowns. Considering what we do not yet know of the universe, the huge volume of information currently floating around will be increasing dramatically! The volume of information we will be dealing with, will be beyond our wildest imagination.

All scientists are in search of information. The search for information is structured as different sciences. In every branch of science the key objective is information. Who can deal with all the information in all sciences? The answer is *software engineering*. Software engineering has already taken the responsibility of providing the necessary tools for organizing, structuring, and making efficient use of the huge volume of information currently floating around. The responsibility of software engineering will increase dramatically by the exponential increases expected in the volume of information. The volume of information, in every branch of science and technology, is getting so high that without a software engineering tool cannot be of any use. Software Engineering is, indeed, heavily involved in all sciences. It appears that no science can live without software engineering tools. This is an extreme power, and responsibility, for software engineering. More research, than currently underway, is required to assure that this powerful too is put to proper work. In this paper, as an small step in this research direction, I investigate the significance of formal approaches and mathematical foundations for software engineering.

2 An Engineering Discipline

A hot debate has started years ago and still going on: Is *software engineering* a *science* or an *engineering discipline*? The debate in some prestigious universities is still going on whether software engineering belongs to the *school of engineering* or *faculty of science*. Software engineering is, in fact, an *interdisciplinary* field; it requires mathematics for analysis and proof of correctness, engineering for costs, risks, and tradeoffs, and management for personnel, facilities, and progress.

Software engineering provides the most powerful tools of all sciences. Software engineering is now, and will be more so in the future, providing the very infrastructure of every science. That is, indeed, an extreme power for software engineering. Extreme power, however, requires extreme care. That is why software engineering, today, must be considered as an engineering discipline with all the associated responsibilities.

2.1 Problems

The major problem with software engineering practice, currently, is inaccuracy and ambiguity in system requirement specifications. The Ariane 5 Flight 501 Failure was caused by a poor software engineering practice. The causes of the failure were faults in the capture of the overall Ariane 5 application/environment requirements, and faults in the design and the dimensioning of the Ariane 5 on-board computing system [36, 34].

Verrazano Narrows Bridge in New York City, the largest suspension bridge ever built, completed within budget, just on target date. IBM OS project, involving over 5000 man-years of work, completed, finally, well beyond the target date [54]. Why *software engineering* cannot be planned and completed like any other engineering project? Because, software engineering is more complex, and software engineers are not as experienced. Specifically, software engineering is a young discipline and, therefore, the corresponding professional do not yet have adequate education and training, compared to the other engineering professionals. A closer look at the *formal approaches* to software engineering may lead to the solution.

3 Formal Approaches

Research on improving the quality of software systems includes using formal methods (e.g., ESTEREL [2], VDM [32], Z [9, 53], Statecharts [18, 19, 23]) for specifying software behaviors and possibly refining the specifications to design and implementation. Despite the controversial issues concerning the practicality of formal methods, there are some undeniable facts:

1. Most of the defects in software systems can be traced back to the requirements phase. Specifically, studies in Bell Labs and IBM have shown that 80% of all defects in software systems are rooted in the requirements phase [50].

2. Accurate requirements specification of software systems, therefore, will improve quality and increase reliability of the software. Accuracy in software systems definition and requirements specification, in turn, demands using formal methods. Formality and mathematics are, generally, becoming more and more useful in all phases of software engineering. Manfred Broy, for example, describes how mathematics can provide a scientific foundation for the modeling aspects, description techniques, and development methods of software engineering, leading to a deeper understanding of the development process [6].

3. For over 20 years, IBM received failure reports on CICS [9]; it was developed without using a formal method. Finally CICS was formally specified using Z [53]. An overall objective was to reduce the total number of errors, detected both in the development cycle and by customers, and thereby produce a better quality product. Some 2000 pages of specifications were produced. This experience proved that formal methods can be helpful; it became a considerable source of education and discussion in the software engineering community [42].

4. Formal methods have not been practical for large-scale complex systems. The IBM CICS experience proved that formal methods can be helpful, but not necessarily practical. Based on experience with the A-7 project [25], John Guttag and others [17] conclude that one problem with formal methods is size. The difficulties of managing a large volume of formal specifications have made formal methods impractical for large-scale systems.

5. Using formal methods is, indeed, difficult. Part of this difficulty is due to the way in which formal specifications are presented. Large volumes of specifications presented textually, many pages of mathematical/logical statements, are indeed difficult to produce, read, understand, and verify, specially for the user side. We may accept that the specifiers are technical people and are supposed to be experts in producing formal specifications. However, the users who are supposed to at least read, understand, and verify the specifications are not normally too enthusiastic to get involved with such formal texts. The *presentation* of formal specifications, therefore, is an important factor for practicality of formal methods. This is one factor that the software industry has been resisting against using formal approaches in software engineering practices. Visual formalism are introduced as a attempt to simplify presentations of formal specification.

Visual formalisms have the advantage of specifying software systems, formally, using graphical notations. The specification then is accurate, because it is formal, and simple to understand, because it is visual. Visual formal methods, therefore, are already getting popular in software requirements engineering and will be more so in the future.

The State Transition Diagrams (STD) of Finite State Machines (FSM) [28] has a sound mathematical foundation and, therefore, are the best formal technique for systems definitions and requirements specifications. However, in these machines the number of states grows exponentially as the scale of the system grows linearly. This growth leads to a problem known as *blow-up in the number states* for large-scale systems. There has been some relatively successful attempts made, by defining Extended Finite State Machines (EFSM), to solve this problem. The most popular of these attempts is made by Harel, introducing Statecharts.

3.1 Statecharts

Introduced by David Harel [18, 19, 22, 23], Statecharts is an extension of Finite State Machines (FSM) designed as a

formal method for behavior specification of complex systems. Harel and others [21] have also developed a set of tools called STATEMATE for the specification, design, analysis, and documentation of large and complex reactive systems. STATEMATE uses Statecharts for behavior specification of a system under development. In addition, STATEMATE provides *module-charts* and *activity-charts* for specifying the structural and functional view of the system, respectively.

Harel describes Statecharts as:

> *state diagrams + depth + orthogonality + broadcast communication.*

Depth refers to a clustering of some states into a superstate and thus forming a hierarchy of states.

Orthogonality refers to a composition of two or more states into a superstate called an AND-state. If a system enters into an AND-state, it must enter in all the AND components (i.e., immediate substates) of the AND-state. Similarly, if a system exits from an AND-state, it must exit from all the AND components of the AND-state.

In contrast to AND-states, there are also OR-states in Statecharts. If a system enters into an OR-state, it must enter in only one of the immediate substates of the OR-state.

A transition label, in Statecharts, can have three components: *event, condition,* and *action*. In Figure 1, for example, if the system is in the state S1 and the event *e* occurs, while the condition expression *c* evaluates to true, then the transition from S1 to S2 takes place and the action *a* is generated. An action generated by a transition can be used as an event in another transition. The action, therefore, must be communicated between the components. The communication mechanism used in Statecharts is broadcasting. For example, when an event occurs or an action is generated, in a statechart, it is sensed throughout the statechart.

The problem with Statecharts is the *complexity of scale*. In conventional finite state machines, as mentioned above, the number of states grows exponentially as the scale of the system grows linearly. This growth leads to a blow-up in the number states for large-scale systems. Drusinsky and Harel [11, 12] prove that Statecharts is exponentially more succinct than finite state machines. The proof is based on the cooperative concurrency mechanism (i.e., orthogonality) of Statecharts and applies for any model that uses this mechanism, such as Petri Nets [41] or CSP [10, 26]. If we assume that an increase in the scale of a system results in additional orthogonal components in the corresponding statechart, then the number of states in the statechart has a linear relationship with the scale of the system; in Harel's words "Statecharts represents the scale of the system" [20]. Orthogonality is, indeed, a powerful feature in Statecharts. However, it is not clear that any increase in the scale of a system does result in additional orthogonal components. For example, if an increase in the scale of a system, corresponds to additional complexities in the existing orthogonal components, then the increase in the number of states would still be exponential.

Another problem with Statecharts is the *global name space*. There is no "visibility" control mechanism in Statecharts. (The term *visibility* is defined in terms of *declaration*, *scope*, and *binding*; a visibility control mechanism, essentially, refers to a mechanism that controls scope [52].) When an event occurs, it is sensed throughout the system and, therefore, it must have a unique name. Managing the name space in the global environment of Statecharts, for large-scale software systems, can be difficult. Name management, in general, is one of the fundamental issues in software engineering [33].

3.2 Other Extensions of State Machines

The following is a list of some other extensions of state machines or variations of Statecharts. Leveson's RSML and Selic's ROOMcharts, as described below, provide some mechanisms for reducing the complexity of managing name space. Besides that, none of these machines provide any solutions for the complexity of scale.

- David Carr introduces an executable graphical notation, called Interaction Object Graphs (IOGs) [7], for specification of user interface. This notation combines the data flow and constraint specification of Interface Representation Graphs (IRG) [43] and Statecharts. IOGs, therefore, extends Statecharts to show data relationships as well as control flow.

- Jahanian and Mok introduce another specification language, called Modechart [31], for real-time systems. They also define the semantics of Modechart in terms of Real Time Logic (RTL) [30]. Modechart is originated based on the mode concept of Parnas [25, 48] and Statecharts of Harel; its emphasis, however, is on the specification of absolute timing properties.

- Alan Shaw [46] describes an executable notation, based on communicating real-time state machines (CRSM's), for specifying concurrent real-time systems. CRSM's are state machines that communicate with each other using a CSP-like synchronous scheme [26, 27]. Shaw also provides an algorithm for simulating CRSM's and some techniques for reasoning about the system behavior.

- Charles Hendricksen [24] describes another extension of finite state machines, called the *Augmented State Transition Diagram* (ASTD), and an associated CASE tool called *State-Graph*. ASTD has been used in the definition, design, and implementation of some applications including a PBX phone system and some complex user interface programs.

- Nancy Leveson and others [35] describe their approach to behavior specification of a real aircraft *Traffic Alert and Collision Avoidance System* (TCAS). The

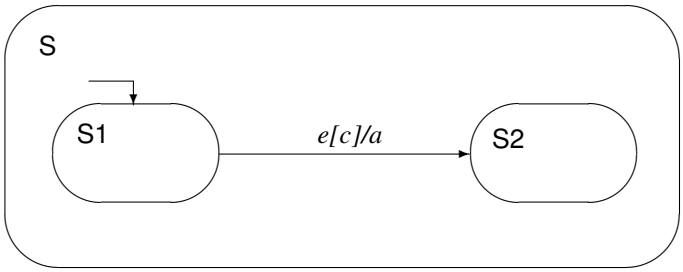


Figure 1: An OR-state.

specification language used for this system is a variation of Statecharts called *Requirements State Machine Language* (RSML). In RSML, physically distinct components are modeled as separate communicating statecharts. The overall system requirement specification can be viewed as a directed graph (not a statechart), where each node represents a component and each edge represents an intercomponent communication channel. The broadcast communication mechanism of Statecharts is used within each component. Intercomponent communication is provided as directed messages transmitted between components over unidirectional channels. An event, therefore, is local to the component in which it occurs. The event does not affect any other component, unless directly transmitted to another one. Therefore, RSML provides a visibility control mechanism that reduces the complexity of name management. The mechanism, however, has a negative consequence: the direct communication method used for intercomponent communications complicates the specifications.

– ROOM [45] is a specialized high-level modeling language, designed for distributed real-time systems and supported by a modeling environment, the ObjectTime toolset. Software behavior in ObjectTime is expressed by ROOMcharts, which is an extension of Statecharts. ROOMcharts replaces the AND-states of Statecharts by encapsulated entities called *actors* (similar to RSML). It also replaces the broadcast communication of statecharts by a port-to-port message-passing mechanism for communications between the actors. As a result, it reduces the complexity of managing name space.

  ROOM starts with the high level design of software components and leads to their implementations. ROOM is not designed for software requirement specification; it is basically a design notation. However it can be used for requirements analysis at a very high level design phase by prototyping and trying out alternative designs.

3.3 OOAD Methods

Some object-oriented analysis and design (OOAD) methods describe the behavior of objects and classes using variations of extended finite state machines. This includes a considerable amount of work in the area of inheritance and refinement of software behavior. Some of this work is outlined below:

– James Rumbaugh and others [44] use an extension of state diagram, based on Harel's Statecharts, to describe the *dynamic model* of Object Modeling Technique (OMT).

– Neal Walters [49] expands on Rumbaugh's work by providing mechanisms for object collaborations: the invocation of object services and interchange of data between objects. Emphasizing the importance of predicting system behavior for validating completeness and analyzing performance, he describes a method for building dynamic models of object-oriented systems using STATEMATE.

– Derek Coleman and others [8] introduce an extension of Statecharts called *Objectcharts* for object-oriented design. They use Objectcharts to specify the behavior of objects and expect that the future work will provide firm semantics for Objectcharts, enabling the behavior of object-oriented systems to be deduced from the specifications of the corresponding objects.

– Finally, the most widely publicized object-oriented method, the Unified Modeling Language (UML) [39, 37, 38], in spite of all its success stories, has not been able to gain the trust of software engineering community for not having a sound mathematical foundation.

  In general, a software engineering methodology, specially in the area of software system definition and requirements specification, cannot succeed unless it has a sound mathematical foundation. The idea of *information hiding*, first proposed by Parnas [40] gave birth to the object oriented methodologies. By mid 90's, these methodologies were going out of momentum for what believed to be their diversity of standards. Introduction of UML, as *the* standard object-oriented notation, was the last attempt made by Object Modeling Group (OMG), to unify the diverse OO methodologies and thereby save the object-oriented approach to software engineering. However, in spite of all the advantages that the unified standard notation could offer, the notation went under attack for not having a sound mathematical foundation.

There is currently a body of work underway on the importance of, and the need to provide, a sound mathematical foundation for UML. Some of this work is as follows:

- Martin Glinz investigate the suitability of UML as a semiformal requirements specification language and, using a case study, identify and demonstrate various problems and deficiencies of UML, particularly concerning use case models and system decomposition [15].
- Robert France [14] describes the role that formal specification techniques can play in the development of well-defined standard modeling languages.
- Breu and others [4, 5] are investigating the possibility of integrating different UML description techniques on a sound mathematical foundation by using a common semantic basis for all notions used by UML.
- Fernandez and others [13] use algebraic specification formal theory to formalized the UML Statechart diagrams and thereby verify the specifications.
- Grosu and others [16] are investigating the formal foundation of UML for Real-Time systems (UML-RT).
- Morgan Björkander describes a two-language merger, combining UML's expressive power and SDL's semantics strengths, to provide a modeling paradigm for visual software engineering that is supposed to be more effective than either language alone [3].

Each of the methods described above has strong features. They provide the accuracy of formality and simplicity of visualism. They simplify the presentation of formal specifications using their graphic notations, solving the problem of presentation. None of these methods, however, provide any solution for the problem of *blow-up in the number states*. This problem, therefore, makes visual formal methods impractical for large-scale systems. That is, of course, if we try to specify the system as a whole. But, how practical is it to accurately and fully describe a system while we can only work with the "representations" of the system? Next section is devoted to this question.

4 Representations

Generally speaking, one can only describe one's "view" of the world. A user's attempt to specify a system, at best, can only result in the specification of his or her "view" of the system. Intuitively, A *view* is a description of the behavior of the system observable from a specific point of view. For a formal definition of *view* see [29]. We also refer to a *view* of a system as a *representation* of the system. "Representation", however, is a much more general term and applies for all entities. Therefore, it deserves a formal definition here.

Formally, r is the (ρ, t)-*representation* of e, if there exists a function ρ and a point in time t, such that $\rho(e,t) = r$. Notice that different functions may produce different representations of a given entity; and a function may produce different representations of a given entity at different times. Using a convention that r refers to the current value of the function, we can eliminate t, simplifying the definition. Thus, we can say r is the ρ-*representation* of e, if there exists a function ρ, such that $\rho(e) = r$. In this case, we can also simply refer to r as a *representation* of e.

With this introduction, therefore, the notion of representation is defined by

$$\begin{aligned} \rho(e,t) = r, \quad & \text{where} \\ \rho \quad & \text{is the } \textit{representation function}, \\ t \quad & \text{is the } \textit{representation time}, \\ e \quad & \text{is the } \textit{representandum}, \text{ and} \\ r \quad & \text{is the } \textit{representation}. \end{aligned} \tag{1}$$

Having defined the notion of representation, we can now define "information" by stating, r is *information* about e or r provides some *information* about e, if r is a representation of e. Any representation of an entity, therefore, provides some information about the entity.

Similarly, any representation of a system provides a partial specification of the system.

Furthermore, any representation of a system specifies a view of the system.

If the inverse of ρ is also a function, then $\rho^{-1}(\rho(e)) = e$ and given r we can reproduce e ($\rho^{-1}(r) = e$), in which case r is a *perfect representation* of e. A perfect representation of e provides all the information about e. For example, on the set of positive integers, if $\rho(e) = e^2$, then $\rho(5) = 25$ and $\rho^{-1}(25) = 5$ and, therefore, 25 is a perfect representation of 5. As another example, in the *Theory of Algorithms and Data Structures*, a graph G is routinely represented by its adjacency matrix M [1]. If we define ρ as $\rho(G) = M$, then $\rho^{-1}(M) = G$ and, therefore, M is a perfect representation of G.

Most representations, however, are not perfect. For example, let us consider the function ρ defined on positive integers as

$$\rho(e) = \begin{cases} \text{odd}\,, & \text{if } e \text{ is an odd number} \\ \text{even}\,, & \text{otherwise} \end{cases}$$

The inverse of this function is not a function and, therefore, the corresponding representations are not perfect. By this definition, all odd numbers are represented as "odd". The representation "odd" is not enough to reproduce the representandum. However, there are cases, where it can be useful just to know whether the number is odd or even. In

general, there are cases, where we want only as much information about an entity as we need and not more. This is because, it serves no purpose to collect and carry around more information about the entity than we need; besides, not all the information on the entity is necessarily available. In other words, for many entities, imperfect representations of the entity are all that we have to work with. In addition, a representation of a representandum can be further represented, creating a hierarchy of representations. In fact, each layer of the communication protocols is described by one level of the corresponding hierarchical representations.

Our notion of representation and the consequent definition of information and specification are consistent with, and describe, most of the familiar concepts. For example:

- *Out-of-date and Up-to-date Information:* If the value of t, in the Equation (1), is equal to an old time, then the corresponding information is said to be *out-of-date*. For the *up-to-date* information t must specify the current time.
- *Volume of Information:* In data communication, a message m is represented as $\rho(m)$, transmitted to the destination, where the original message m is reproduced by $m = \rho^{-1}(\rho(m))$. The volume of information contained in the message is $\log_2(n)$ bits, where n is the size of domain of ρ. That is, the number of bits required to code a message out of n possible messages is $\log_2(n)$ [47]. The logarithmic base 2 is for our choice of "bit" as the unit of measuring information, considering that a bit consists of 2 states. If we choose a another mechanism, consisting of b states, then the logarithmic base will be b.
- *Misinformation (Error) or Inconsistent Specification:* A representation may not reflect the true state of the world, in which case it is misrepresentation, misinformation, or error.

  Formally, $r = \rho(e)$ is misinformation if $\rho^{-1}(r)$ is inconsistent with e. In particular, where perfect representations are required, $r = \rho(e)$ is misinformation if $\rho^{-1}(r) \neq e$.

In addition, our notion of representation provides a new view of the world. I call this view of the world *representationism*, which is the way in which we deal with entities, in representing, manipulating, transferring, and reproducing them. The representationistic approach to study an entity, in general, is to extract representations of the entity, study them, and compose the results to construct knowledge of the entity. Notice that this is different from the well known *reductionistic*, (in contrast to *holistic*) approach in the philosophy of science. In reductionism, a whole is dissected into pieces, each piece is studied and analyzed separately, and then the results are synthesized and integrated. There are similarities, but also major differences: The pieces in reductionism are smaller parts of the whole, while the representations in representationism are simple and mostly imperfect representations of the entire whole. There we have a picture of a section, while here we have a picture of the whole, but from a specific point of view.

Critics of science have portrayed reductionism as an *obsessional disorder, declining toward a terminal stage, as one writer recently dubbed "reductive megalomania"* [51]. This criticism does not apply for representationism. Because, the representationistic approach to study an entity, reduces the problem by describing it in terms of simple representations, but unlike reductionism, keeps the entity intact.

An example of a representationistic approach to system specification is a formalism called Viewcharts.

4.1 Example

The Viewcharts formalism [29], designed based on a representationistic approach, is the most recent attempt to resolve the complexity of scale. In Viewcharts, the behavior of a system is specified, formally and visually, as a composition of "views". Intuitively, as discussed above, A *view* (or *behavioral view*) is a complete description of the behavior of the system observable from a specific point of view. Using this notion of view, the formalism is designed to specify the behavioral requirements of large-scale complex systems on a *need-to-specify* basis. In Viewcharts, one does not have to specify the full behavior of a system and, therefore, is not concerned with the complexity or scale of the system. A complex system may consist of many different sub-systems and components, distributed worldwide, and it may exhibit a combination of many different and identical behavioral views. Current research and industrial advances in networking and distributed systems indicate that software systems will continue to get larger and more complex. One cannot envision producing an integrated behavioral requirements specification for an arbitrarily large and complex system. However, if we define the behavior of a system in terms of behavioral views, then all we need to do is to specify the views of our interest. The Viewcharts formalism allows these views to be specified independent of each other.

Furthermore, views in Viewcharts limit the scope of broadcast communication, solving the problem of *global name space*.

It is, therefore, expected that the Viewcharts formalism will be practical in large-scale systems behavioral specifications.

5 Conclusion

Software engineering is involved, deeply, in every science and technology. Software engineering provides the very infrastructure of every science and technology. With all these responsibilities, software engineering cannot afford to go wrong. Software engineering, therefore, has no choice but to go formal. Software engineering requires professionals, educated and trained in working on a sound foundation. A software engineer, before writing down the first line of

code, must be able to do different kinds of mathematical analysis on his/her design and prove that the system-to-be-developed is specified correctly, consistently, without ambiguity, and once it is developed, is going to work as specified. The universities, professional schools, and education centers, in this field, are responsible for training the software engineers, prepared to face the challenging tasks ahead. And, that is the trend.

In addition to an analysis of the responsibilities associated with software engineering and the corresponding trend, the contribution of this paper can be summarized as an approach to the way in which *information*, in general, and *software systems*, in particular, can be defined and specified. The paper has proposed a representationistic approach to software definition and specification. In this approach we do not deal with a system as a whole, we deal with different representations of the system and, thereby, simplify the system definition, specification, and analysis.

Acknowledgements

I would like to thank the Research Counsel and the Office of Research Affair at Tabriz University for their financial support.

References

[1] Aho A. V., Hopcroft J. E., and Ullman J. D. (1975). *The Design and Analysis of Computer Algorithms*. Addison Wesley, Reading, MA.

[2] Berry G. and Cosserat L. (1984). The ESTEREL synchronous programming language and its mathematical semantics. In *Seminar on Concurrency*, Springer-Verlag, Vol. 197 of *Lecture Notes in Computer Science*, pp. 389–448.

[3] Björkander M. (2000). Graphical programming using UML and SDL. *IEEE Computer*, 33(12):30–35.

[4] Breu R., Grosu R., Huber F., Rumpe B., and Schwerin W. (1998). Systems, views and models of UML. In Schader M. and Korthaus A., editors, *The Unified Modeling Language, Technical Aspects and Applications*, Physica Verlag, Heidelberg, pp. 93–109.

[5] Breu R., Hinkel U., Hofmann C., Klein C., Paech B., Rumpe B., and Thurner V. (1997). Towards a formalization of the unified modeling language. In *Proceedings of ECOOP'97*, Springer Verlag, LNCS, pp. 344–366.

[6] Broy M. (2001). Toward a mathematical foundation of software engineering methods. *IEEE Transactions on Software Engineering*, 27(1):42–57.

[7] Carr D. A. (1997). Interaction object graphs: An executable graphical notation for specifying user interface. In Palanque P. and Pterno F., editors, *Formal Method for Computer Human Interaction*. Springer-Verlag, pp. 141–156.

[8] Coleman D., Hayes F., and Bear S. (1992). Introducing Objectcharts or how to use Statecharts in object oriented design. *IEEE Transactions on Software Engineering*, 18(1):9–18.

[9] Collins B. P., Nicholls J. E., and Sorensen I. H. (1987). Introducing formal methods: The CICS experience with Z. Technical Report TR12.260, IBM Hursley Park.

[10] Davies J. (1993). *Specification and Proof in Real-Time CSP*. Distinguished Dissertations in Computer Science. University of Cambridge Press, Cambridge.

[11] Drusinsky D. and Harel D. (1988). On the power of cooperative concurrency. In *Proceedings of International Conference on Concurrency (CONCURRENCY'88)*, Springer-Verlag, Vol. 335 of *Lecture Notes in Computer Science*, pp. 74–103.

[12] Drusinsky D. and Harel D. (1994). On the power of bounded concurrency I: Finite automata. *Journal of the Association for Computing Machine (ACM)*, 41(3):517–539.

[13] Fernández J. and Toval A. (2000). Can intuition become rigorous? foundations for UML model verification tools. In Titsworth F. M., editor, *Proceedings of the International Symposium on Software Reliability Engineering*, IEEE Press, pp. 344–355.

[14] France R. (1999). A problem-oriented analysis of basic UML static requirements modeling concepts. In *Proceedings of the 1999 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, ACM Press, pp. 57–69.

[15] Glinz M. (2000). Problems and deficiencies of UML as a requirements specification language. In *Proceedings of Tenth International Workshop on Software Specification and Design (IWSSD'00)*, pp. 11–22.

[16] Grosu R., Broy M., Selic B., and Stefanescu G. (1998). Towards a calculus for UML-RT specifications. In Kilov H., Rumpe B., and Simmonds I., editors, *Proceedings of the Seventh OOPSLA Workshop on Behavioral Semantics of OO Business and System Specifications*, pp. 135–152.

[17] Guttag J., Horning J., and Wing J. (1982). Some notes on putting formal specifications to productive use. *Science of Computer Programming*, 2(1):53–68.

[18] Harel D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274.

[19] Harel D. (1988). On visual formalisms. *Communications of ACM*, 31(5):514–530.

[20] Harel D. (1995). Private communication. Weizmann Institute of Science, Rehovot, Israel.

[21] Harel D., Lachover H., Naamad A., Pnueli A., Politi M., Sherman R., Shtull-Trauring A., and Trakhtenbrot M. (1990). STATEMATE: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403–414.

[22] Harel D. and Naamad A. (1995). The STATEMATE semantics of Statecharts. Technical report, i-Logix, Inc., 22 Third Avenue, Burlington, Mass.

[23] Harel D. and Pnueli A. (1985). On the development of reactive systems. In Apt K. R., editor, *Logics and Models of Concurrent Systems*. Springer-Verlag, New York, pp. 477–498.

[24] Hendricksen C. S. (1989). Augmented state-transition diagrams for reactive software. *ACM SIGSOFT Software Engineering Notes*, 14(6):61–67.

[25] Heninger K. L., Kallander J. W., Shore J. E., and Parnas D. L. (1978). Requirements for the A-7E aircraft. Technical Report NRL 3876, Naval Research Laboratory, Washington, DC.

[26] Hoare C. (1978). Communicating sequential processes. *Communications of ACM*, 8(21):666–677.

[27] Hoare C. (1985). *Communicating Sequential Processes*. Prentice Hall, Englewood Cliffs, NJ.

[28] Hopcroft J. E., Motwani R., and Ullman J. D. (2001). *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Boston, MA.

[29] Isazadeh A., Lamb D. A., and Shepard T. (1999). Behavioural views for software requirements engineering. *Requirements Engineering Journal*, 4(1):19–37.

[30] Jahanian F. and Mok A. K. (1986). Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, 12(9):890–904.

[31] Jahanian F. and Mok A. K. (1994). Modechart: A specification language for real-time systems. *IEEE Transactions on Software Engineering*, 20(12):933–947.

[32] Jones C. B. (1990). *Systematic Software Development using VDM*. Prentice Hall International Series in Computer Science. Prentice Hall.

[33] Kaplan A. and Wileden J. C. (1995). Formalization and application of a unifying model for name management. In *Proceedings of ACM SIGSOFT'95*, pp. 161–172.

[34] Lann G. L. (1997). An analysis of the Ariane 5 Flight 501 failure - a system engineering perspective. In *Proceedings of the IEEE Workshop on Engineering of Computer-Based Systems (ECBS'97)*, IEEE Computer Society Press, pp. 339–346.

[35] Leveson N. G., Heimdahl M. P. E., Hildreth H., and Reese J. D. (1994). Requirements specification for process control systems. *IEEE Transactions on Software Engineering*, 20(9):684–707.

[36] Lions J. L. (1996). ARIANE 5; Flight 501 Failure. Report by the Inquiry Board available online at: http://www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html.

[37] OMG (1999). UML Notation Guide, Version 1.3. Object Management Group, available online at http://www.rational.com/uml.

[38] OMG (1999). UML Semantics, Version 1.3. Object Management Group, available online at http://www.rational.com/uml.

[39] OMG (1999). Unified Modeling Language, Version 1.3. Object Management Group, available online at http://www.omg.org.

[40] Parnas D. (1972). On the criteria to be used in decomposing systems into modules. *Communications of ACM*, 15(2):1053–1058.

[41] Peterson J. L. (1977). Petri Net. *Computing Surveys*, 9(3):223–252.

[42] Phillips M. (1989). CICS/ESA 3.1 experiences. In Nicholls J. E., editor, *Z User Workshop*, pp. 179–185.

[43] Rouff C. and Horowitz E. (1991). A system for specifying and rapidly prototyping user interfaces. In Karat J., editor, *Taking Software Design Seriously*. Academic Press, pp. 257–272.

[44] Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W. (1991). *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ.

[45] Selic B., Gullekson G., and Ward P. T. (1994). *Real-Time Object-Oriented Modeling*. Wiley, New York.

[46] Shaw A. C. (1992). Communicating real-time state machines. *IEEE Transactions on Software Engineering*, 18(9):805–816.

[47] Tague-Sutcliffe J. (1995). *Measuring Information: An Information Services Perspective*. Academic Press, San Diego.

[48] van Schouwen A. J. (1990). The A-7 requirements model: Re-examination for real-time systems and an application to monitoring systems. Technical Report 90-276, Telecommunications Research Institute

of Ontario (TRIO), Department of Computing and Information Science, Queen's University, Kingston, Canada.

[49] Walters N. (1992). Using Harel's Statecharts to model object-oriented behavior. *ACM SIGSOFT Software Engineering Notes*, 17(4):28–31.

[50] White M. S. (1994). Requirements: A quick and inexpensive way to improve testing. Testing Techniques Newsletter (TTN), On-Line Edition, ttn@soft.com, http://www.cs.ucl.ac.uk/research/renoir/newsletter/backissues/renl34.

[51] Wilson E. O. (1999). *Consilience: The Unity of Knowledge*. Random House, New York.

[52] Wolf A. L., Clarke L. A., and Wileden J. C. (1988). A model of visibility control. *IEEE Transactions on Software Engineering*, 14(4):512–520.

[53] Wordsworth J. B. (1992). *Software Development with Z*. International Computer Science Series. Addison-Wesley.

[54] Zelkowits M. V., Shaw A. C., and Gannon J. D. (1979). *Principles of Software Engineering*. Prentice-Hall, Englewood Cliffs, NJ.

Computer-Aided Reuse Tool (CART)

Zina Houhamdi
Department of Computer Science, University of Biskra, BP 145, Biskra, 07000, Algeria.
E-mail: z_houhamdi@yahoo.fr

Belkacem Athamena
Department of Automatics, University of Biskra, BP 145, Biskra RP, 07000, Algeria.
E-mail: b.athamena@caramail.com

Keywords: Software reuse, facet classification, object-oriented library, thesaurus, CASE tools.

Received: April 17, 2004

Software reuse has been claimed to be one of the most promising approaches to enhance programmer productivity and software quality. One of the problems to be addresses to achieve high software reuse is organizing databases of software experience, in which information on software products and processes is stored and organized to enhance reuse.
Object-oriented software libraries expand in size more rapidly than other type of software library. This paper presents a simple approach for aiding reuse in software development using object-oriented library. Our approach improves the effectiveness of code searching by reorganizing the library with facet classification scheme and thesaurus. Information in specification models, such as Data Flow Diagrams (DFDs), is extracted through object abstraction and then used as a query input. We are currently implementing a Computer-Aided Reuse Tool (CART) based on the approach.

1 Introduction

Improving software productivity and quality is one of primary emphasis of research in software engineering [12, 2]. Many studies have shown that software reuse can improve software productivity and quality significantly. Research on software reuse can be divided into at least three categories: some researchers study how to construct reusable software components [4,6,11,13], some study system frameworks for reusable software asset libraries [8,9,16], other study classification and retrieval strategies for reusable software libraries to achieve effective reuse [3,5,7].

Computer-Aided Software Engineering (CASE) tools usually come with a repository for specification in different development phases. The repository is designed as a browser-like or query-reply tool for search and retrieval of specification contents (elements). These kinds of tools, however, provide less help in reuse than those booting direct search on a real software library. On the other hand, object-oriented libraries probably grow more quickly than any other type of software libraries. Traditional ways of searching for components, such as consulting a user's manual, using a browser or examining the source code, not only waste time in searching, but provide little help in understanding particular components. This problem is even more serious when searching through a large object-oriented library, which may comprise millions lines of code or thousands of classes.

In this paper, we present an approach that integrates reuse techniques with an actual object-oriented library to promote reuse in CASE. Our approach boots an automatic mechanism on both classification and retrieval processes of software components. The classification scheme is basically the facet scheme from [10, 16, 17], and the retrieval mechanism is a query-reply. The classification is performed using the keywords of a thesaurus. The query inputs are generated by extracting information from the entities in specification models, such as a data flow diagram (DFD). A query is instantiated by a user, and its content can be selected by the users or generated automatically. Replies to queries improve reuse by helping users compare entities and components en the library more precisely. We are currently implementing a Computer-Aided Reuse Tool (CART) based on our approach.

Section 2 gives a brief review of the facet scheme. Section 3 discusses the classification process in detail and presents some interesting results of queries. Section 4 presents a design for an automatic code extractor, and section 5 describes the structure of our CART. Section 6 is the conclusion of the paper.

2 The facet scheme

The facet classification scheme was first proposed by Prieto-Diaz in [17]. The facet scheme has many advantages. In particular, it is suitable for collections of similar reusable components that are large, contain groups, and are growing continuously. A typical example is GTE Data Service's AMP (Asset Management Program), whish was introduced in 1991 [16].

A facet is a viewpoint toward software components. Viewpoints may include the functions the components perform, the objects they manipulate, the system types to whish they belong, and so on. The value for a facet of a component is called *facet value*. The set of facet values

for a component is called *facet descriptor*. The characteristics of a component may be described using its facet descriptor and a component may be understood through its facet descriptor.

In designing classification scheme, there are at least two factors that must be considered. First, the scheme needs a set of proper facets to represent essential features of a component. This set may be determined by referring to system requirements after classifying all components. Unnecessary or insufficient facets may cause some trouble in the classification process. Unnecessary facets may make the facet values hard to assign or make it impossible to index a component by facet descriptor. Insufficient facets may map many components to the same facet descriptor, so hat ambiguities present when locating and retrieving components. Second, each facet needs a set of proper facet values to represent all possible (distinguished) features, or indices, of components. Usually, the definition of facet values relies on domain analysis and expert knowledge. When the collection of components is large, it is difficult to predefine all possible facet values for further indexing. Thus, the facet scheme must be simplified for a large object-oriented library.

3 A facet scheme for large object-oriented library

3.1 The use of keywords

A word may have more than one meaning and two different words may have the same meaning. A thesaurus is used in vocabulary control. It helps to clarify concepts. Table 1 shows several examples. In the table, the words *consume*, *feed*, *partake*, and *chew* have the same meaning, *eat*. On the other hand, the word *consume* has two different meaning, *eat* and *waste*.

Keywords	Synonyms
Eat	Eat, consume, feed, partake, chew
Waste	Consume, expend, exhaust, reduce, eat
Taste	Relish, enjoy, eat, try, sample
Fewness	Diminish, reduce, lessen, lack, need

Table 1: Partial Thesaurus

Using keywords in a thesaurus as the universe of facet values to simplify classification has the following advantages. First, it saves time in determining the universe of facet values, and thus simplifies the classification domains. When a domain is changed, the universe remains unchanged. A facet value is described with a set of keywords, of which each represents one distinct meaning. A facet descriptor described in this way is space-less, but easier to implement.

From the viewpoint of specification, usually more than one word is needed to describe the features of components on each facet. For example, the information on the *function* facet may be (*copy*, *from*), (*copy*, *replace*, *all*) or (*copy*, *replace*, *from*). Although the primitive function is *copy*, these three components are all distinct. As another example, the *object type* facet of components may have the information (*linked*, *list*) or (*double*, *linked*, *list*), indicating that they manipulate *lists* of different kinds. These cases occur frequently, especially when processing a large collection.

The facet scheme in our approach is centered an automated thesaurus developed from [14]. Our facet scheme consists of three facets:

- *Function* refers to the function performed. Function names such as *store*, *changeRequest*, and *updateString* are used to generate facet values.
- *Object type* refers to the template of objects to which the method belongs. Therefore, type names such as *set*, *bag*, and *textItemEditor* are used to represent this facet.
- *System type* refers to domains, which are functionally identifiable, application-independent, and usually include more than one component. The name of a domain, for example, *collection-text* or *graphics-interface*, may be used for the value.

Each component of the software (code) library is given (defined) a set of keywords as a facet value. These keywords are generated from information extracted from the above names. One example is the information (*linked, list*), where *linked* belongs to the three keywords *relations*, *junction* and *combination*, and *list* belongs to four keywords *numeration*, *class*, *list* and *record*. If (*linked*, *list*) is used to define a specific facet of a component, the internal representation of the facet is (*relations, junction, combination, numeration, class, list, record*). The next section describes how to proceed with classification. Section 3.3 describes a retrieval example to illustrate the capabilities of the facet scheme.

3.2 The classification process

Based on the previous discussion, we designed a Library Automation Classification System (LACS) to classify a large object-oriented library. Figure 1 is the overall data flow of our LACS.

In the LACS, each facet has an automatic classification procedure. Each classification procedure is designed according to the properties of components and the needs of each facet. The strategies for generating facet values from a sample object-oriented library, Smalltalk-80, are the following:

- Function facet: The method name can represent the rough meaning of a method, for example, *copy, changeRequest, storeone, doesNotUnderstand, storeString* and so on. Method names are quite useful in helping users to understand the methods. Thus, our procedure in the LACS first splits the method name and then passes each word through the thesaurus to get the corresponding keywords. For example, the name of the method *changeRequest* can be split into two words, *change* and *request*. The

value of the function facet of this method is (*variance, money, improvement, change, command, request, worship, inquiry*).

- Object type facet: This facet refers to the template of objects to which methods belong. An object type is a class name in the library. The class name of the sample library can describe the features of a class. Therefore, the procedure for function facet can be modified slightly to get the values of this facet.

System type facet: The domain that methods belong to are defined in the user manual of Smalltalk-80. The domain names in the user manual are also passed through the thesaurus to get facet values.

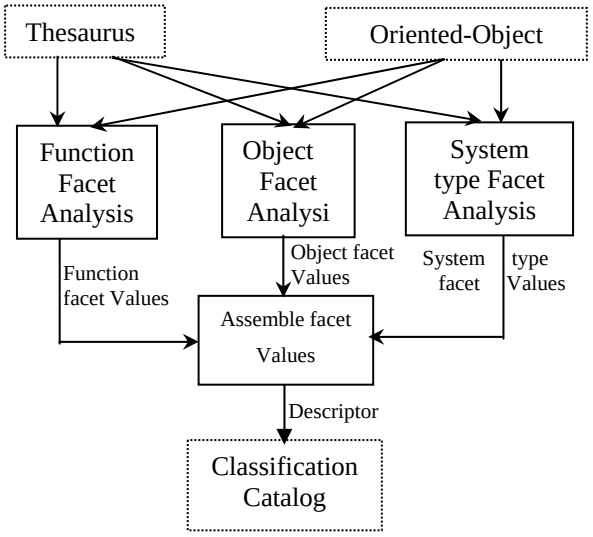


Figure 1: Data Flow Diagram of LACS

The classification process in the LACS is centered on a thesaurus and consists of the following sequence of tasks. First, the necessary information is collected (method name, class name and system type name). Second, each name is decomposed into words, which are passed through the thesaurus to get corresponding keywords (facet values). These facet values for each component form the facet descriptor. Third, the facet descriptors generated by the LACS are stored in the classification catalog.

In the prototype system, the thesaurus consists of 886 keywords and up to 34837 words. The classification catalog contains 349 classes and 3722 methods. The number of actual keywords (facet values) used for function, object type, and system type facets is 723, 356, and 70 respectively. The average number of repetitions of keywords, the number of facet values a keyword appears, for function, object type, and system type facets is 35.39, 5.37, and 3.83 respectively. The average number of keywords contained in a facet value for each method, class name, and system type name is 6.87, 5.47, and 5.47 respectively. Note that the function facet has much higher the average number of facet values a keyword appears (35.39). Perhaps this is why libraries based on functions offer a lower degree of reusability.

The LACS doesn't record inheritance directly. A class in an object-oriented library may inherit method(s) from other classes. The LACS builds the structure of class hierarchies additionally.

3.3 A retrieval example

The service functions provided by the classification catalog are query-reply functions: A user can query to find software component(s) from the classification catalog. The query form is similar to the form of facet descriptor: [(function), (object type), (system type)]. The following examples show what the queries and replies are.

1. Query: A user wants to find possible methods of displaying characters. The query is [(*display, character*), (), ()], where the empty cell "()" means "don't care".

Reply: The reply displays possible methods in classes. There are 42 possible tuples retrieved. However, some tuples seem irrelevant to the user requirement. This is caused by insufficient strictness in the query constraints. A small amount of extra information (constraints) can reduce the size of reply. The next two queries will illustrate the outcome of using a more narrowly constrained query.

Class Name	Method Name
ArihmeticValue	Sign
Behavior	ShowVariableMenu
Browser	ShowHierarchy
ChangerScanner	ScanClassExpression ScanExpression
CharacterScanner	StopConditionFor
CompiledCode	SignExtend
ComposedText	DisplayFromCharacter RightMarginForDisplay
ComposedTextView	DisplaFromCharacter ShowSelectionBoxOn
…….	…….
GraphicsContext	DisplayCharacterOfIndex RoundedDisplayCharacterOf Ind
IOAccessor	OpenFileWriteOnly
LoopNode	Condition
MenuTracker	DisplaExtraInformation DisplayExtraInformationFor
Parser	BlockExpression Expression PrimaryExpression TypeExpression
…….	…….
TextCollector	Show
TextLines	RightMarginForDisplay

2. Query: The user has the same requirements as above, and he guesses the system type to be graphics. So, the query becomes [(*display*, *character*), (), (*graphics*)].

Reply: There are seven tuples, which satisfy the query constraints. The additional information on the system type facet reduces the size of the reply from 42 to seven tuples.

Class Name	Method Name
CharacterScanner	StopConditionFor
ComposedText	DisplayFromCharacter RightMarginForDisplay
DeviceFont	DisplayCharacter
GraphicsContext	DisplayCharacterOfIndex RoundedDisplayCharacterOfInde
TextLines	RightMarginForDisplay

3. Query: If the user guesses that the object type should be graphics, the query may become [(*display, character*), (*graphics*), (*graphics*)].

Reply: there are two tuples, which satisfy the query constraints. The additional information on the object type facet reduces the reply from seven to two tuples.

Class Name	Method Name
GraphicsContext	DisplayCharacterOfIndex RoundedDisplayCharacterOfIndex

Obviously, the replies to queries are determined mainly by the input information (constraints) of each facet. The more precise a user wants a result to be, the more useful information he must provide. This query-reply tool is not complicated, but it still requires that the user type in the details of the query.

4 Object abstraction during specification

4.1 Object abstraction

DFDs have been used widely for system specifications. During requirements analysis, an analyst may not know exactly whether a process in a DFD requires further decomposition. There were some rules to help make such decisions. For example, a process need not be decomposed if its specification can be completely described on a piece of paper or if it has only one input and output. Without enough information, system analysts always decompose processes by experience. This way may cause some processes to be decomposed too much and others too little. Both are undesirable.

Object abstraction of DFDs may provide one kind of help for the above problems. Intuitively, data flows in a DFD can be mapped to objects, data stores to objects, processes to the methods of objects, and external entities to the objects containing original functions as their methods [1]. The specification of an entity, such as a data flow, data store, or external entity, may come from other modeling tools, such as the Entity-Relationship Diagrams (ERD) used in structured analysis (SA) or the object diagrams of Object-Oriented Analysis (OOA). With these object abstractions (mapping), users can query the system whether there exist(s) certain components in the library. For example, they can ask whether there are classes or objects in the library that correspond to a specific data flow or data store in the DFD, or a method corresponds to a specific process, and so forth.

The query information may come directly from the specification, instead of being entered by the user as in section 3.3. An automated tool for processing these queries can help users, when they are working on system analysis, system design, or programming. For example, if the reply indicates that the entity being worked has a matched (or say, qualifies) component in the library, the user may wish to stop decomposition. Therefore, referencing the replied information can speed up the design of the both logical and physical models. In addition, reusing qualified code naturally reduces implementation time.

Figure 2 is a sample DFD from [15]. The process *Evaluation bounds violation* has the target object *blood, pressure temp* and *pulse*. If the system type facet information is *Monitoring System*, a query [(*Evaluation, bounds, violation*), (*blood, pressure temp* and *pulse*), (*Monitoring, System*)] will be extracted to see whether the process *Evaluation bounds violation* has corresponding codes in the library. This query-reply can help an analyst decide whether to decompose the process or not.

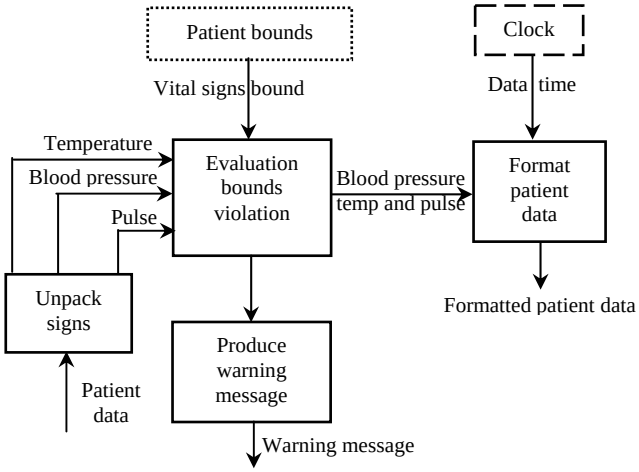


Figure 2: Partial Data Flow Diagram of SA

Another example, shown in figure 3, is an object diagram in Rambaugh's OOA [18]. The diagram is designed for a diagram editor. Let *diagram editor* be set as the system type facet information. A query [(*box*), (), (*diagram, editor*)] Will be extracted to check the existence of code for *box*. Another example, shown in figure 3, is an object diagram in Rambaugh's OOA [18]. The diagram is designed for a diagram editor. Let *diagram editor* be set as the system type facet information. A query [(*box*), (), (*diagram, editor*)] Will be extracted to check the existence of code for *box*.

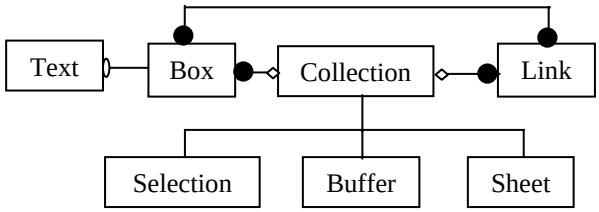


Figure 3: Partial Object Diagram in Rambaugh's OOA

4.2 An Automatic Code Extractor

An automatic Code Extractor (ACE) is an automated tool containing the functions discussed previously. An ACE simplifies the generating action of queries and makes replies more precise by integrating these actions with CASE tools. The more useful the information extracted from the repository is, the more precise the reply derived by an ACE will be. However, each CASE tool may follow a different methodology, such as SA/SD or OOA/OOD. The information extracted different methodologies will be different.

For a SA methodology, DFD is the kernel of SA. The function facet information can be extracted from the name string of a process. The object type facet information is extracted from the data flow name, data store name, or external entity name (which will be specified in the ERD). The system type facet information cannot be extracted from the repository directly. One way to handle this problem is to use an additional tool, like a customizer [19] in Excelerator/IS. To add an additional field to let users input the necessary information. For OOA methodology, the function facet information can be extracted from the service name. The object type facet information can be extracted from the class name or object name. The system type facet information also requires an additional field to let users enter the information.

The role of an ACE within the waterfall model is shown in figure 4. Our current prototype system uses a simple searching strategy, which contains the following steps:

1. Obtain the relevant data from the repository according to the user request and assemble the data into a query.
2. Replace the words in the query by keywords in the thesaurus to produce the real query.
3. Enter the query into classification catalog to get the reply and return it to the user.

If too many components satisfy the query constraints, the system acquires more information from the user interactively. If there is no hit, i.e., no component matches the query (constraints), the user has three choices. One is the user to accept the fact, another is to enter other information for the next query, and the other is to search through the class hierarchy for inherited methods.

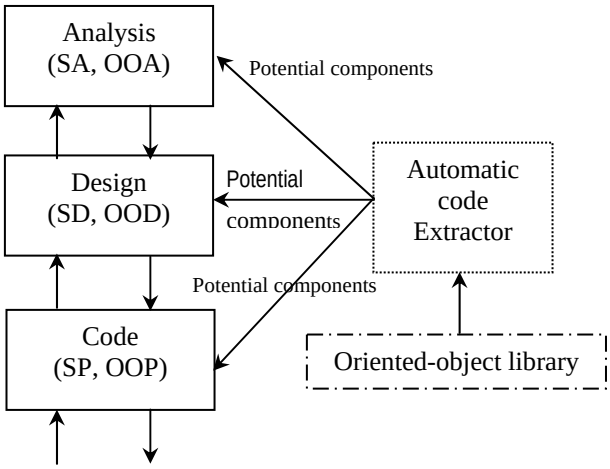


Figure 4: ACE supports for Waterfall Model

Consider the example in figure 2 again. After the step 1, the query [(*Evaluation, bounds, violation*), (*blood, pressure temp* and *pulse*), (*Monitoring, System*)] is extracted. After step 2, the keywords of the words in the above query are as shown in the following table and the real query is [((*judgment, measurement*), (*limit, leap, promise, circumscription*), (*undueness, …, impiety*)), ((*nobility*), (*rarity, …, impulse*), (*measurement, heat*), (*regularity, oscillation*)), ((*inquiry*), (*unity, …, arrangement*))].

The searching process of step 3 is through a union operation of the keywords of each word and then an intersection operation of the result of each word in the query. It is the same as [[((*judgment* ∪ *measurement*) ∩ (*limit* ∪ *leap* ∪ *promise* ∪ *circumscription*) ∩ (*undueness* ∪ …∪ *impiety*)) ∩ ((*nobility*) ∩ (*rarity* ∪ …∪ *impulse*) ∩ (*measurement* ∪ *heat*) ∩ (*regularity* ∪ *oscillation*)) ∩ ((*inquiry*) ∩ (*unity* ∪ …∪ *arrangement*))].

Words	Keywords
Evaluation	Judgment, measurement
Bounds	Limit, leap, promise, circumscription
Violation	Undueness, illegality, misuse, non-observation, overstepping, disobedience, impurity, impiety
Blood	Nobility
Pressure	Rarity, measurement, adversity, weight, influence, compulsion, propulsion, power, impulse
Temp	Measurement, heat
Pulse	Regularity, oscillation
Monitor	Inquiry
System	Unity, order, rule, crossing, arrangement

After step 3, there are no hits. The user chooses to accept this fact. This means that the process *Evaluation bounds violation* needs to be decomposed further.

The above searching process can narrow the range of possible components. The components in the reply should include at least one component the user needs if

The extraction catches his idea and the repository contains the desired component(s). Sometimes, a query may retrieve too many potential components. Combining different searching processes or extracting more useful information may make a reply more precise.

5 The framework of CART

As described in section 1, CART is designed to facilitate software reuse in Case tools. Figure 5 shows an overall data flow diagram of CART. CART mainly consists of five parts: the CASE repository, thesaurus, classification catalog, LACS, and ACE. The CASE repository provides information related to user requests. The thesaurus is used for vocabulary control. The classification catalog is the result of code abstraction with enhanced facet scheme. The LACS described in section 3 is used while constructing the classification catalog. In CART, the role of the LACS is the maintenance of the classification catalog, such as the addition or deletion of software components. The dashed-square in figure 5 is the ACE; it is responsible for locating and retrieving components for users.

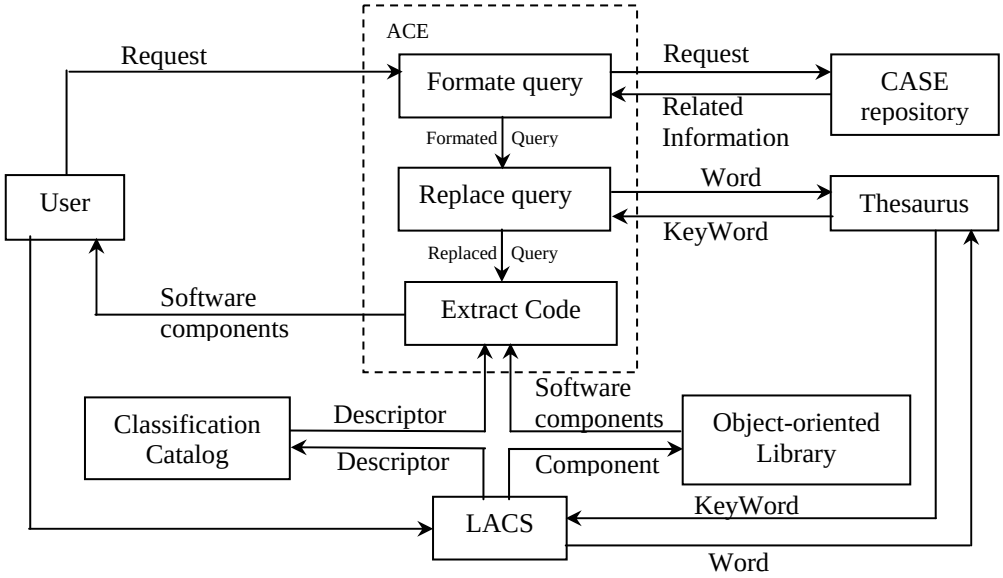


Figure 5: Data Flow Diagram of CART

With these parts, CART can provide the closest related components in the library once the user (a system analyst, system designer, or programmer) makes a request. For example, when the user is specifying his requirements (or design) with DFD, he can ask the ACE to extract his previous specification as the raw query information. Through the help of automated thesaurus, the information is transformed into the query to search for possible components. We are still in the process of implementing CART. Our sample library is Smalltalk-80 and our CASE tool is Excelerator/IS.

6 Conclusion

This paper presents an approach to facilitate software reuse in case that is based on an object-oriented library. This approach boots the facet scheme, which automates the classification process from a (large) object-oriented library. The reuse process is automated with CASE tools to provide more assistance to the user in each phase of the software development cycle. We are still in the process of implementing CART. The next step in our research will be toward "knowledge-based CART", where the knowledge may come from use habit. The system may improve the query by acquiring the knowledge while the user is working on the system.

References

[1]. Aliabiso, B "*Transformation of Data Flow Analysis Models to Object-Oriented Design*". OOPSLA'99, pp. 335-353.

[2]. Anderson, K.J. "*State-of-art in software Reuse Technologies*". COMPSAC'99 Panel on software reuses.

[3]. Burton, B. A., Aragon, R. W., Bailey, S. A. and Mayes, L. A. "*The reusable Software Library*". IEEE Software Engineering, July 1998, pp. 25-33.

[4]. Caldiera G. and Basili V. R. "*Identifying and qualifying Reusable Software Components*". IEEE Computer, Feb. 1997, pp. 61-70.

[5]. Helm, R. and Maarek, Y. S. "*Integrating Information retrieval and Domain Specific Approaches for Browing and retrieval in Object-oriented Class Libraries*". OOPSLA'99, pp. 47-91.

[6]. Houhamdi, Z. and Ghoul, S. "*A Reuse Description Formalism*". ACS/IEEE International Conference on Computer Systems and Applications, AICCSA'2001, Lebanese American University, Beirut, Lebanon. 2001.

[7]. Houhamdi, Z. and Ghoul, S. "*A Classification System for software reuse*". Fifth International Symposium on Programming System, ISPS2001, USTHB Computer science Institute, Algiers, Algeria, 2001.

[8]. Houhamdi, Z. "*A Specification language for software reuse*". CSS/IEEE Alexandria Chapter. 11th International Conference On computers:

Theory and Application, ICCTA2001, Head of Electrical Control, Alexandria, Egypt, 2001.

[9]. Houhamdi, Z. "*Developing a Reuse Library*". CSS/IEEE Alexandria Chapter. 11th International Conference On computers: Theory and Application, ICCTA2001, Head of Electrical Control, Alexandria, Egypt, 2001.

[10]. Houhamdi, Z. "*An adaptative approach to reuse software*". SCS/IEEE 2001. The third Middle East Symposium on Simulation and Modeling, MESM'2001, Amman University, Amman, Jordan, 2001.

[11]. Houhamdi, Z. "*Software Reuse: a new classification approach*". The International Symposium on Innovation in Information and Communication Technology, ISIICT'2001, Philadelphia University, Amman, Jordan, 2002.

[12]. Jones, T.C. "*Reusability in Programming: A survey of state of the art*". IEEE Transaction on Software Engineering, Vol. 10, No 5, Sep. 1999, pp. 488-493.

[13]. Lenz M., Schmid H. A. and Wolf P. W. "*Software Reuse through Building Blocks*". IEEE Software Engineering, July 1999, pp. 34-42.

[14]. Manser, M. H. "*Pocket Thesaurus of English Words*". The Hamlyn Publishing Group Limited, 1988.

[15]. Pressman, R. S. "*Software Engineering: A Practitioner's Approach*". McGraw-Hill, Inc., 2nd Edition, 1995, pp. 171.

[16]. Prieto-Diaz, R. "*Implementation Faceted Classification for Software Reuse*". Communication of ACM, Vol. 34, No 5, May 1991, pp. 89-97.

[17]. Prieto-Diaz, R. and Freeman, P. "*Classifying Software for Reusability*". IEEE Software Engineering, Jan. 1997, pp. 6-16.

[18]. Rambaugh, J. and Addy, F. "*Object-Oriented Modeling and Design*". Prentice-Hall International, Inc., 1998, pp. 191

[19]. *"The customizer Reference Guide"*. Preface, Intersolv Technology, Excelerator Series.

Public-Key Inter-Block Dependence Fragile Watermarking for Image Authentication Using Continued Fraction

Chin-Chen Chang and Wen-Chuan Wu
Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi 621, Taiwan, R.O.C.
E-mail:{ccc, wenn}@cs.ccu.edu.tw
Http://www.cs.ccu.edu.tw/~ccc

Yu-Chen Hu
Department of Computer Science and Information Management,
Providence University, Taichung 433, Taiwan, R.O.C.
E-mail: ychu@pu.edu.tw
Http://www1.pu.edu.tw/~ychu

Keywords: fragile watermark, public key encryption, and continued fraction

Received: March 1, 2003

In this paper, a novel fragile watermarking scheme based on the continued fraction technique is proposed. The goal of this scheme is to verify the integrity of a watermarked image. The feature value of the image is generated from the calculation of the continued fraction. The feature value is regarded as the important information to be hidden in the image. If the watermarked image is suspected of being tempered with, the watermark extraction procedure could be used for verification. We do not to use another image as the watermark image in our scheme. The proposed scheme can solve the weaknesses in the public key blockwise-independent watermarking scheme proposed by Wong in 1998. This is because the proposed scheme uses the continued fraction technique to achieve inter-block dependence for image authentication. According to experimental results, the image quality of each watermarked image is good, and the visual quality of the watermarked image is not affected when the proposed watermark insertion procedure is carried out.

1 Introduction

Because of the rapid growth of electronic commerce over the Internet, there is an urgent need for effective and efficient copyright protection techniques and security verification methods against unauthorized data duplication, especially the illegal distribution of images and video data [7,8,9]. Under such circumstances, digital watermarking has become a critical and imperative subject. In general, from the viewpoint of visual imperceptibility, watermarking techniques can be classified into two categories: visible watermarking techniques and invisible watermarking techniques. Invisible watermarking techniques are mainly adopted because images with visible watermarks can be easily targeted and simply removed [3,15,16].

In addition to visual imperceptibility, digital watermarking techniques can also be classified into two kinds in terms of their robustness: "robust" and "fragile." Robust watermarking techniques are usually used for copyright and ownership verification. Their major focus is on whether or not the watermark is robust enough to prevent removal. Recently, Wang et al. proposed a scheme based on wavelet transformation [14] to augment the robustness.

In contrast, fragile watermarking techniques [1,6,11,12,13] are generally used to detect alterations in watermarked data. When signed data have been tampered with, fragile watermarking techniques are responsible for notifying the user about authentication and integrity. Recently, some watermarking schemes [11,12,13] have been introduced for image integrity and ownership verification. Among them, a public key watermarking scheme for image verification and authentication [12], proposed in 1998, is worthy of special attention. It uses a cryptographic hashing function, such as the MD5, and a public-key cipher to judge if a watermarked image has undergone tampering.

However, as Barreto et al. pointed out in [1], Wong's scheme [12] has some flaws. In this scheme, attackers can forge a watermarked image to pass verification by launching the cut-and-paste attack, the birthday attack, or other counterfeiting attacks. Moreover, Holliman and Memon [4] have also concluded that the use of contextual information can mend some of the weaknesses of watermarking schemes with dependent image blocks. For this reason, they adopted the strategy of hash block chaining (HBC1 and HBC2) to solve the problem of tampering in [1].

In this paper, the continued fraction technique is employed to make image blocks interdependent. We use the image features themselves to calculate, block-by-block, the values used in the continued fraction. Then the continued fraction is hidden in the image by using the proposed image insertion procedure. In other words, we do not need another image as the watermark. By virtue of the use of contextual information, as previously mentioned and indicated by Holliman and Memon [4], the proposed scheme can also resist some attacks, such as the cut-and-paste attack, the birthday attack, and other counterfeiting attacks (the transplantation attack, for example). The experiment provides a clear demonstration that authentication and integrity can be achieved.

The rest of this paper is organized as follows. In Section 2, we will review Wong's scheme and Barreto et al.'s scheme. The drawbacks of Wong's scheme and the improvement suggested by Barreto et al. will also be included. After that, in Section 3, we will propose our scheme that uses the continued fraction technique to achieve inter-block dependence for fragile watermarking. Then the experimental results will be given in Section 4. Finally, the conclusions will be discussed in Section 5.

2 Previous works

To begin with, let us briefly review the public key watermarking scheme proposed by Wong [12] and show its insecurity. After that, we shall also introduce the secure public key block-wise fragile authentication watermarking scheme proposed by Barreto et al. [1].

2.1 Wong's scheme

In 1998, Wong proposed a public key watermarking scheme for image verification and authentication [12]. There are two procedures in this scheme: the insertion procedure and the extraction procedure. The insertion procedure hides the representative watermark in an image. The main goal of this scheme is to ensure the image's ownership and integrity. When a watermarked image is received, we can use the extraction procedure to judge if the watermarked image has undergone altering. The detailed practice is as follows.

2.1.1 The insertion procedure

Suppose X is a gray-scale image of $M\times N$ pixels to be watermarked with a binary image B. The binary image B is formed by tiling the watermark image A until it becomes the same size as X. First, X is partitioned into a set of non-overlapping blocks of 8×8 pixels, and each block X_r (denoting the r^{th} block of X) is watermarked separately. The binary image B is partitioned off into blocks of the same size, too. Here, B_r denotes the r^{th} block of the binary image B.

Then a cryptographic hash function is used to compute the hash value $H_r \cong H(M,N,X_r^*)$, where M and N are the image width and height, respectively, and the block X_r^* is where the least significant bit of each element in X_r is set to be zero, because a pixel in the image processing is composed of a byte, that is to say, eight bits. For this reason, the least significant bit (LSB) is generally denoted the first bit of the byte. Then the exclusive-or operation is used to compute the value $\hat{H}_r = H_r \oplus B_r$. Finally, $\hat{H}_r$ is encrypted with the private key to generate a digital signature S_r. Now, S_r can be inserted into the LSB of the block X_r^* to form a watermarked block. This way, the image X can be successfully watermarked with the binary image B by putting all the B_r values into their corresponding X_r blocks.

2.1.2 The extraction procedure

The corresponding watermark extraction procedure is as follows. Let X' be the watermarked image of $M\times N$ pixels. First, X' is partitioned into non-overlapping blocks of 8×8 pixels, and each block is denoted by X_r'. Then the block X_r^* is formed by clearing the LSB of each element in X_r'. After that, the LSB of X_r' is extracted and decrypted with the public key to get the decrypted block D_r. Using the hashing function operation, we can get the block $H_r \cong H(M,N,X_r^*)$, which we then put through the exclusive-or operation with the help of D_r to get $B_r' = H_r \oplus D_r$. Finally, we compare B_r' with B_r. If they are equal, the watermark is verified. Otherwise, the r^{th} block of the watermarked image X' has been modified.

2.2 Barreto et al.'s scheme — hash block chaining (HBC)

As we have just mentioned, Wong's scheme is subject to the counterfeiting problem. An attacker can produce a fake yet correctly watermarked image by launching the cut-and-paste attack, the birthday attack or other counterfeiting attacks. Wong's scheme can be easily circumvented by the cut-and-paste attack and the birthday attack.

To improve the security of Wong's scheme, Barreto et al. adopted the strategy of hash block chaining (HBC1 and HBC2) [1] to solve the problem of tampering. Barreto et al. brought in contextual information to make the watermarking scheme achieve inter-block dependency in contrast to the inter-block independency in Wong's scheme.

In HBC1, the calculation for each block is not only for that block but also for the next block. Which block is considered the next block depends on the scan type, for example, the zig-zag-scan [2] or raster-scan. Thus, the hash process is $H_r = H(M,N,X_r^*,X_{(r-1) \bmod n}^*,r)$, where n is

the number of blocks in the image *X*. The other steps in the insertion procedure are the same as those in Wong's scheme. If a block X_r' is changed, the signature verification will fail for all the blocks that depend on X_r'.

However, Barreto et al. have also come to the conclusion that HBC1 cannot resist the transplantation attack and, therefore, have offered the so-called hash block chaining version 2, HBC2 for short. In the improved version, the signature of the current block gets hashed. In other words, the hash formula now becomes $H_r = H(M, N, X_r^*, X_{(r-1) \bmod n}^*, r, S_{r-1})$. The main idea is that even the signatures of two identical images will be different when non-deterministic signatures are used (for example, DSA and Schnorr's scheme [10]). Unlike HBC1, HBC2 can withstand the transplantation attack.

3 The proposed scheme

In this section, we shall introduce our inter-block dependent public-key watermarking scheme for image authentication that is built upon the concept of continued fractions. This scheme is composed of two procedures: the watermark insertion procedure and the extraction procedure. Before presenting the detailed steps in the procedures, we will first look at the definition of the continued fraction.

3.1 The definition of continued fraction

In general, we usually use decimal expansion to express a real or rational number. However, there is another way to represent a real number that expresses certain geometric properties in an exceptional way. This other method is the simple continued fraction extension of a real number [5]. Equation 1.1 below is the expression of the continued fraction.

$$\frac{A}{B} = a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots + \cfrac{b_{n-1}}{a_{n-1} + \cfrac{b_n}{a_n}}}}} = [a_0, a_1, a_2, \cdots, a_n]$$

A simple continued fraction exists if all the b_i's are 1 and all the a_i's are integers satisfying $a_i \geq 1$, where $1 \leq i \leq n$. We denote the simple continued fraction of a real number $\frac{A}{B}$ as [a_0, a_1, a_2,…, a_n]. For example, if we have two real numbers, 137 and 112, then $\frac{137}{112} = [1,4,2,12]$. In the case of the steps in the computation of $\frac{137}{112}$, we use the Euclidean algorithm, shown as follows.

$137 = 1 \times 112 + 25$
$112 = 4 \times 25 + 12$
$25 = 2 \times 12 + 1$

We can rewrite the above as fractions, which as follows.

$$\frac{137}{112} = 1 + \frac{25}{112}$$
$$\frac{112}{25} = 4 + \frac{12}{25}$$
$$\frac{25}{12} = 2 + \frac{1}{12}$$

By combining the above fractions, we can write $\frac{137}{112}$ in terms of the successive quotients, as follows, where we have replaced $\frac{25}{112}$ with $\frac{1}{4+\frac{12}{25}}$ and $\frac{12}{25}$ with $\frac{1}{2+\frac{1}{12}}$. This representation of $\frac{137}{112}$ is known as a simple continued fraction, and the terms are the successive quotients in the Euclidean algorithm.

$$\frac{137}{112} = 1 + \cfrac{1}{4 + \cfrac{1}{2 + \cfrac{1}{12}}} = [1, 4, 2, 12]$$

3.2 The insertion procedure

The goal of the insertion procedure is to calculate the feature value of a meaningful gray-scale image and to embed the feature value into the image. Here, the mentioned feature value is represented by a continued fraction, and any rational number can be expressed as a simple continued fraction. In order to make use of the block-by-block clue, we apply the continued fraction to the design of our watermarking scheme.

A	B
C	D

(a) A gray-level image of 16×16 pixels

$$\frac{A}{B} = [a_0^0, a_1^0, \ldots, a_i^0]$$
$$\frac{B}{C} = [a_0^1, a_1^1, \ldots, a_j^1]$$
$$\frac{C}{D} = [a_0^2, a_1^2, \ldots, a_k^2]$$
$$\frac{D}{A} = [a_0^3, a_1^3, \ldots, a_l^3]$$

(b) The continued fractions for the blocks, where, in the expression "a_x^y," *y* denotes the block number and *x* denotes the continued fraction number

Figure 1: Continued fractions for a gray-level image.

We use the mean values of the blocks adjacent to each other as the parameters in the continued fraction expression of a real number. Suppose that Figure 1 is a gray-level image of 16×16 pixels and that *A*, *B*, *C*, and *D* are the mean values of blocks 0, 1, 2, and 3 of 8×8 pixels, respectively. In order to take care of the inter-block dependence problem, the continued fractions to be

calculated are $\frac{A}{B}$, $\frac{B}{C}$, $\frac{C}{D}$, and $\frac{D}{A}$. In our scheme, these continued fractions will serve as secret keys for blocks 0, 1, 2, and 3, respectively. In other words, they separately represent the watermark of those blocks.

The structural insertion procedure of the proposed scheme is as follows. X is a $M\times N$ gray-scale image and M and N are the image width and height, respectively.

Step 1: Partition X into n blocks X_r $(0 \le r \le n-1)$ of 8×8 pixels.

Step 2: For each block X_r, set the LSB of all pixels to zero to obtain the block X_r^*.

Step 3: Calculate the mean value M_r for each block X_r^*.

Step 4: Calculate the continued fraction $F_r = \frac{M_r}{M_{r+1}}$, $(0 \le r \le n-1)$. If r=n-1, $F_{n-1} = \frac{M_{n-1}}{M_0}$. The calculation of F_r follows the zig-zag scan type.

Step 5: Use a cryptographically secure hash function H (in our experiment, we chose MD5 as our hash function) to compute the feature $H_r \cong H(M \| N \| F_r \| r \| S_{r-1})$.

Step 6: Encrypt H_r with the owner's private key to generate a digital signature S_r.

Step 7: Insert S_r into the LSB of the block X_r^* to form a watermarked block X_r'.

Step 8: Repeat steps 4 through 7 above again and again until all the blocks in the image X have been processed.

After the manipulation above, we will get a watermarked image X'. X' can then be safely transmitted and received over the Internet without the danger of being falsified or counterfeited.

3.3 The extraction procedure

The main purpose of the extraction procedure is to verify whether an image has been tempered with. When a legal user receives a watermarked image X', he/she has to take the following steps. X is a $M\times N$ gray-scale image, and M and N are the image width and height, respectively. The detailed algorithm of the extraction procedure is as follows:

Step 1: Divide X' into n blocks X_r' $(0 \le r \le n-1)$ of 8×8 pixels.

Step 2: Extract the LSB of each block X_r' to get S_r', and decrypt it by using the public key to obtain the decrypted block D_r'.

Step 3: Let X_r^* be the block obtained from X_r' by clearing the LSB of all the pixels. Calculate the mean value M_r' for each block X_r^*.

Step 4: Calculate the continued fraction $F_r' = \frac{M_r'}{M_{r+1}'}$, $(0 \le r \le n-1)$. If r=n-1, $F_{n-1}' = \frac{M_{n-1}'}{M_0'}$. The calculation of F_r' follows the zig-zag scan type.

Step 5: Use the same hash function H chosen in the insertion procedure and compute the fingerprint $H_r' \cong H(M, N, F_r', r, S_{r-1})$.

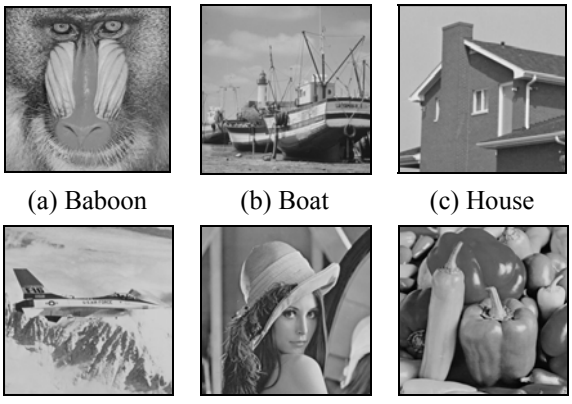
Step 6: If H_r' and D_r' are equal, the r-th block passes verification. Otherwise, the watermarked image X' must have been modified.

Step 7: Repeat steps 4 through 6 above again and again until all the blocks in the image X' have been verified.

From the insertion and extraction procedures above, we see that there is no need for another image to be used as a watermark. This is different than the scheme proposed by Barreto et al. Here, we consider that the usage of a watermark image is unnecessary. We think of the feature H_r as a watermark to be hidden for the block r. Consequently, the proposed scheme will not use another image as the watermark, and verification can be achieved. In addition, because the continued fraction belongs to the category of contextual information, the cut-and-paste attack will cause no harm.

4 Experimental results

In our experiment, a number of standard images, including Lena, Jet, Baboon, Peppers, House, and Boat, were used, as shown in Figure 2. These are gray-scale images of 512×512 pixels. In this paper, due to the limited space of a printed page, we will just include three of the images as examples.

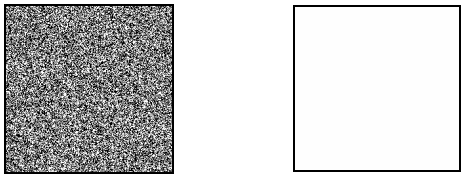


(a) Baboon (b) Boat (c) House

(d) Jet (e) Lena (f) Pepper

Figure 2: Six test images of 512×512 pixels.

In the process of our experiment, we assumed that some images were modified when transmitted over a network. The possible outputs by the extraction procedure are presented in Figure 3. If an image encountered one of several situations, for example, if it was unmarked, scaled, or cropped, or if a wrong public key was used to decrypt it, the extraction procedure resulted in an output that resembled random noise, as shown in Figure 3(a). Succinctly stated, this would mean that the watermarked image was tampered with. On the other hand, Figure 3(b) shows that there was no possibility of the watermarked image being altered and could, therefore, be trusted.



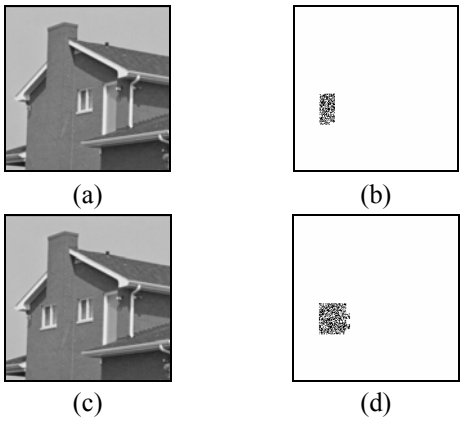
(a) Incorrect (b) Correct

Figure 3: Possible outputs by the extraction procedure.

Figure 4 shows the corresponding watermarked images of the original images shown in Figure 2. The three images are representative. We present the experimental results in Figure 5, Figure 6, and Figure 7. As for the image House, Figure 5(a) and Figure 5(c) show the modified image, with the left window deleted or modified, and Figure 5(b) and Figure 5(d) show the extracted version of House from Figure 5(a) and Figure 5(c), respectively. Because Figures 5(a) and 5(c) are the modified images, the extracted images from them, namely Figures 5(b) and 5(d), have random dots on them.

(a) House (b) Jet (c) Lena

Figure 4: The watermarked images.



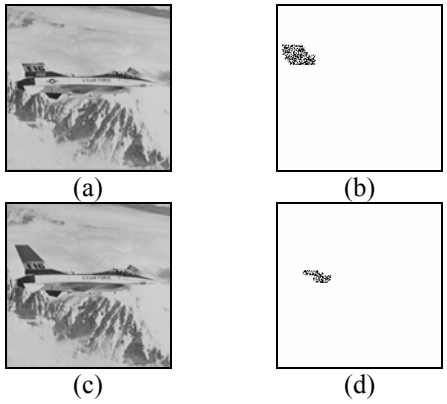
(a) (b)

(c) (d)

Figure 5: The House image processed by the proposed scheme.

The others images, Jet and Lena, used for testing are shown in Figure 6 and Figure 7, respectively. In Figure 6(a), the empennage of the jet aircraft is shown to have been pruned away. The serial number '01568' of the jet aircraft and the hallmark of the American air force are shown to have been pruned away in Figure 6(c), too. Therefore, they generated, by way of the extraction procedure, the noisy outputs shown in Figures 6(b) and 6(d).

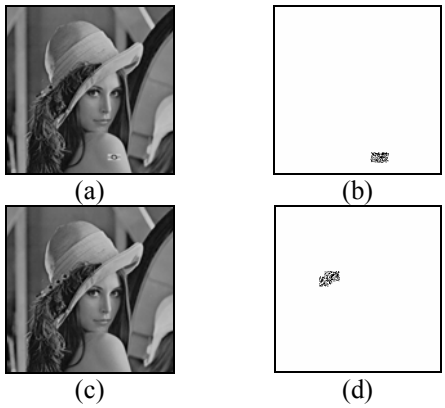
Figures 7(a) and 7(c) simulate the image Lena, of 512×512 pixels, that has undergone tampering over the Internet, and Figures 7(b) and 7(d) show the extracted versions of Lena from Figures 7(a) and 7(c), respectively. In Figure 7(a), there is a tattoo on her arm. Consequently, the corresponding image shown in Figure 7(c) has random dots. As shown in Figure 7(c), there is a modification of the brim of the hat. The hat ribbon has extra ornaments. For this reason, the extraction procedure resulted in an output that resembles random noise, as shown in Figure 7(d).



(a) (b)

(c) (d)

Figure 6: The Jet image processed by the proposed scheme.



(a) (b)

(c) (d)

Figure 7: The Lena image processed by the proposed scheme.

Table 1. The PSNR values of the watermarked images of the proposed scheme.

Image	Baboon	House	Jet
PSNR (db)	51.13	51.14	51.14
Image	Lena	Peppers	Sailboat
PSNR (db)	51.15	51.14	51.14

In the experiments above, we used MD5 as our hash function. The RSA scheme was then used as the signature algorithm. With regards to the time consumed, our scheme took only about 10 seconds to finish all the procedures from signing to verifying on a Pentium 450 with 256MB RAM for the 512×512 gray-scale images. In

addition, Table 1 shows the PSNR values of the six watermarked images shown in Figure 3. Because of the adoption of embedding image's feature in the least signal bit, that is, that bit was the only bit affected, the peak signal to noise ratio (PSNR) values were all higher than 51 dB. If continued fraction could be used for hiding information, we can get better quality of the watermarked image and be hard to be aware of the information hiding in the image. This is worth studying in the future.

5 Conclusion

This paper presented a novel solution of using contextual information, namely using the continued fractions, which is usually perceived as merely a mathematical theory, to the problem of inter-block dependent fragile watermarking. As we have just illustrated, the continued fraction can actually be quite helpful to achieve interdependence among blocks. It also achieves the objective of allowing one to become aware of an alteration to the image.

According to our experimental results, the performance of our scheme in both verification accuracy and efficiency is satisfactory. Even though we tempered the position in the watermarked image, the possible modified place can be straightforwardly pointed out by using the extraction procedure. Due to the use of contextual information, our scheme can also resist the cut-and-paste attack, the birthday attack, and other counterfeiting attacks. Of special note, our scheme does not use another image as the watermark. Hence, images can be processed and verified in a short time.

In addition, we found that the usage of continued fraction could be used for data hiding. The main idea is that the quality of a watermarked image is excellent over 51 dB, and the calculation is speedy and simple.

References

[1] Barreto, P.S.L.M., Kim, H.Y., and Rijmen, V., "Toward Secure Public−Key Blockwise Fragile Authentication Watermarking," *IEE Proceedings-Vision, Image and Signal Processing*, Vol. 149, No. 2, April 2002, pp. 57-62.

[2] Chang, H., "The Scan Sequence of Image Compression and Its Optimization," *Proceedings of International Conference on Communication Technology*, Beijing, China, Vol. 2, May 1996, pp. 679-682.

[3] Craver, S., Memon, N., Yeo, B.L. and Yeung, M.M., "Resolving Rightful Ownerships with Invisible Watermarking Techniques Limitations, Attacks, and Implications," *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 4, May 1998, pp. 573-586.

[4] Holliman, M. and Memon, N., "Counterfeiting Attacks on Oblivious Block-wise Independent Invisible Watermarking Schemes," *IEEE Transactions on Image Processing*, Vol. 9, No. 3, March 2000, pp. 432-441.

[5] Kumanduri, R. and Romero C., "Number Theory with Computer Applications," Prentice Hall, Upper Saddle River, New Jersey, 1998, pp. 243-284.

[6] Li, C.T., Lou, D.C. and Chen, T.H., "Image Authentication and Integrity Verification via Content−Based Watermarks and a Public Key Cryptosystem," *Proceedings of IEEE International Conference on Image Processing*, Vancouver, Canada, Vol. 3, September 2000, pp. 694-697.

[7] Mintzer, F., Braudaway, G.W. and Yeung, M.M., "Effective and Ineffective Digital Watermarks," *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, CA, Vol. 3, October 1997, pp. 9-12.

[8] Memon, N. and Wong, P.W., "Protecting Digital Media Content," *Communications of the ACM*, Vol. 41, No. 7, July 1998, pp. 35-43.

[9] Podilchuk, C.I. and Delp, E.J., "Digital Watermarking Algorithms and Applications," *IEEE Signal Processing Magazine*, Vol. 18, No. 4, July 2001, pp. 33-46.

[10] Schnorr, C.P., "Efficient Signature Generation by Smart Cards," *Journal of Cryptology*, Vol. 4, No. 3, 1991, pp. 161-174.

[11] Wong, P.W., "A Watermark for Image Integrity and Ownership Verification," *Proceedings of Image Science and Technology PIC Conference*, Savannah, Georgia, April 1999, pp. 374-379.

[12] Wong, P.W., "A Public Key Watermark for Image Verification and Authentication," *Proceedings of IEEE International Conference on Image Processing*, Chicago, Illinois, Vol. 1, October 1998, pp. 455-459.

[13] Wong, P.W. and Memon, N., "Secret and Public Key Image Watermarking Schemes for Image Authentication and Ownership Verification," *IEEE Transactions on Image Processing*, Vol. 10, No. 10, October 2001, pp. 1593-1601.

[14] Wang, Y., Doherty, J.F. and Van Dyck, R.E., "A Wavelet−Based Watermarking Algorithm for Ownership Verification of Digital Images," *IEEE Transactions on Image Processing*, Vol. 11, No. 2, February 2002, pp. 77-88.

[15] Yeung, M.M. and Mintzer, F., "An Invisible Watermarking Technique for Image Verification," *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, CA, Vol. 2, October 1997, pp. 680-683.

[16] Zeng, W. and Liu, B., "On Resolving Rightful Ownerships of Digital Images by Invisible Watermarks," *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, CA, Vol. 1, October 1997, pp. 552-555.

A New Efficient Group Signature With Forward Security

Jianhong Zhang, Qianhong Wu and Yumin Wang
State key Lab. of Integrated Service Networks, Xidian Univ, Xi'an
Shannxi 710071, China
jhzhs@hotmail.com, woochanhoma@hotmail.com
ymwang@xidian.edu.cn
Keywords: group signature, forward security, revocation, anonymity, unlikability

Received: April 29, 2003

A group signature scheme allows a group member to sign a message anonymously on behalf of the group. In case of a dispute, the group manager can reveal the actual identity of signer. In this paper, we propose a novel group signature satisfying the regular requirements. Furthermore, it also achieves the following advantages: (1) the size of signature is independent of the number of group members; (2) the group public key is constant; (3) Addition and Revocation of group members are convenient; (4) it enjoys forward security; (5) The total computation cost of signature and verification requires only 8 modular exponentiations. Hence, our scheme is very practical in many applications, especially for the dynamic large group applications.

1 Introduction

Digital signatures play an important role in our modern electronic society because they have the properties of integrity and authentication. The integrity property ensures that the received messages are not modified, and the authentication property ensures that the sender is not impersonated. In well-known conventional digital signatures, such as RSA and DSA, a single signer is sufficient to produce a valid signature, and anyone can verify the validity of any given signature. Because of its importance, many variations of digital signature scheme were proposed, such as blind signature, group signature, undeniable signature etc, which can be used in different application situations.

A group signature was introduced by Chaum and van Heyst [1]. It allows any member of a group to anonymously sign a document on behalf of the group. A user can verify a signature with the group public key that is usually constant and unique for the whole group. However, he/she cannot know which individual of the group signs the document. Many group signature schemes have been proposed [1,2,3,5,6,7,8]. All of them are much less efficient than regular signature schemes. Designing an efficient group signature scheme is still an open problem. The recent scheme proposed by Ateniese et al. is particularly efficient and provably secure [2]. Unfortunately, several limitations still render all previous solution unsatisfactory in practice. Giuseppe Ateniese pointed out two important problems of group signature in [3]. One is how to deal with exposure of group signing keys; the other is how to allow efficient revocation.

In this paper, we propose a novel and efficient group signature scheme with forward security to solve the above two important problems. The concept of forward security was proposed by Ross Anderson [4] for traditional signature. Several schemes have recently been proposed for traditional signatures and threshold signatures that satisfy the efficiency properties. Previous group signature schemes don't provide forward security. Forward secure group signature schemes allows individual group member to join or leave a group or update their private signing keys without affecting the public group key. By dividing the lifetime of all individual private signing keys into discrete time intervals, and by tying all signatures to the time interval when they are produced, group members who are revoked in time interval i have their signing capability effectively stripped away in time interval i+1, while all their signature produced in time interval i or before remain verifiable and anonymous. In 2001, Song [5] firstly presented a practical forward security group signature scheme. Our proposed scheme is a little more efficient than Song's scheme.

The rest of this paper is organized as follows. In section 2, we overview the informal model of a secure group signature scheme and security requirements. After our group signature scheme is proposed in section 3, we give the corresponding security analysis to the scheme in section 4. in section 5, we analyze the efficiency of our proposed scheme and compares the cost with the Song's scheme. Finally, we conclude this paper.

2 Group Signature Model and Security Requirements

The concept of group signature was introduced by Chaum and van Heyst [1]. It allows a group member to sign anonymously a message on behalf of the group. Any one can verify group signature with the group public key. In case of a dispute, the group manager can open the signature to identify the signer.

Participants: A group signature scheme involves a group manager (responsible for admitting/deleting members and for revoking anonymity of group signature, e.g., in case of dispute or fraud), a set of group members,

and a set of signature verifiers, all participants are modeled as probabilistic polynomial-time interactive Turing machines. A group signature scheme is comprised of the following procedure.

Communication: All communication channels are assumed asynchronous, The communication channel between a signer and a receiver is assumed to be anonymous.
A group signature scheme is comprised of the following procedure:

Setup: On inputting a security parameter *l*, this probabilistic algorithm outputs the initial group *PK* and the secret key *SK* for the group manager.

Join: An interactive protocol between the group manager and a user that results in the user becoming a valid group member.

***Sign*:** An interactive protocol between a group member and a user whereby a group signature on a message supplied by a user is computed by the group member.

***Verify*:** A deterministic algorithm for verifying the validity of a group signature given a group public key and a signed message.

***Open*:** A deterministic algorithm that, given a signed message and a group secret key, determines the identity of the signer.
A secure group signature should meet the following requirements:

Correctness: Signature produced by a group member using Sign must be accepted by **Verifying**.

Unforgeability: Only group members are able to sign messages on behalf of the group

Anonymity: Given a signature, identifying the actual signer is computationally hard for any one except the group manager.

Unlinkability: Deciding whether two different signatures were generated by the same group member is computationally hard.

Exculpability: Even if the group manager and some of the group member collude, they cannot sign behalf of non-involved group members.

Traceability: The group manager can always establish the identity of the member who issued a valid signature.

Coalition-resistance: a colluding subset of group members cannot generate a valid group signature that cannot be traced.
To achieving practicability, in this paper, we propose a group signature scheme supporting the above properties and another two attributes, revocation and forward security, as well.

Revocability: the group manager can revoke membership of a group member so that this group member cannot produce a valid group signature after being revoked.

Forward security: When a group signing key is exposed, previously generated group signatures remain valid and do not need to be re-sign.

3 Preliminaries

The building block presented in this subsection is an protocols for proving the knowledge of a discrete logarithm to the setting with a group of unknown order.

Definition 1. Let $\varepsilon > 1$ be a security parameter. A pair $(c,s) \in \{0,1\}^k \times \{-2^{l+k},\ldots,2^{\varepsilon(k+l)}\}$ satisfying $c=h(g\| y\| g^s y^c \| m)$ is a signature of a message $m\in\{0,1\}^*$ with respect to y and is denotes SPK$\{\alpha: y=g^\alpha\}$(m).
An entity knowing the secret key $x\in\{0,1\}^l$ such that $x = \log_g y$ can compute such a signature (c, s) SPK$\{\alpha: y=g^\alpha\}$(m) of a message $m \in\{0,1\}^*$ by

- choosing $r \in \{0,1\}^{\varepsilon(l+k)}$ and computing $t = g^r$
- $c = h(g \| y \| t \| m)$ and
- $s=r-cx$ (*in Z*)

SPK$\{\alpha: y=g^\alpha\}$("") denotes Signature of Knowledge on space message.
The security of all these building blocks has been proven in the random oracle model under the strong RSA assumption.

Our Proposed Group Signature parameter

GM: group manager,
ID_{GM} :Identity of group manager,
ID_B : Identity of group member Bob
n : a RSA modular number.
$h(.)$: a one-way hash function $\{0,1\}^* \rightarrow \{0,1\}^k$
SPK : signature of knowledge.

3.1 System Parameters

The group manager (GM) randomly chooses two large primes p_1, p_2 of the same size such that $p_1 = 2p_1' + 1$ and $p_2 = 2p_2' + 1$, where both p_1' and p_2' are also primes. Let $n = p_1 p_2$ and $G=< g >$ a cyclic subgroup of Z_n^*. GM randomly chooses an integer x as his secret key and computes the corresponding public key $y = g^x \pmod n$. GM selects a random integer e (e.g., $e = 3$) which satisfies $\gcd(e, \varphi(n)) = 1$ and computes d satisfying $de = 1 \bmod \varphi(n)$ where $\varphi(n)$ is the Euler Totient function. $h(\cdot)$ is a coalition-resistant hash function (e.g., SHA-1, MD5). The time period is divided into T intervals and the intervals are publicly known. $(c,s) = SPK\{\gamma : y = g^\gamma\}(")$ denotes the signature of knowledge of $\log_g y$ in G (See [2,6] for details). Finally, the group manager publishes the public key $(y, n, g, e, h(\cdot), ID_{GM}, T)$, where ID_{GM} is the identity of the group manager.

3.2 Join Procedure

If a user, say Bob, wants to join to the group, Bob executes an interactive protocol with GM. Firstly, Bob chooses a random number $k \in Z_n^*$ as his secret key and computes his identity $ID_B = g^k (\mathrm{mod}\, n)$ and the signatures of knowledge $(c,s) = SPK\{\gamma : ID_B = g^\gamma\}(")$, which shows that he knows a secret value to meet $ID_B = g^k (\mathrm{mod}\, n)$. Finally, Bob secretly preserves k and sends $(ID_B, (c,s))$ to the group manager.

After the group manager receives $(ID_B, (c,s))$, he firstly verifies the signatures (*c*, *s*) of knowledge by $(ID_B, (c,s))$. If the verification holds, GM stores $(ID_B, (c,s))$ in his group member database and then generates membership certificate for Bob. Thereby, GM randomly chooses a number $\alpha \in Z_n^*$ and computes as follows.

$$r_B = g^\alpha \bmod n,\ s_B = a + r_B x$$

$$w_{B_0} = (ID_{GM} r_B ID_B)^{-d^T} \bmod n$$

GM sends (s_B, r_B, w_{B_0}) to Bob via a private channel. GM stores (s_B, r_B, w_{B_0}) together with $(ID_B, (c,s))$ in his local database.

After Bob receives (s_B, r_B, w_{B_0}), he verifies the following relations

$$g^{s_B} = r_B y^{r_B} \bmod n$$

$$ID_{GM} ID_B r_B = w_{B_0}{}^{-e^T} (\mathrm{mod}\, n)$$

If both the above equations hold, Bob stores (s_B, r_B, w_{B_0}) as his resulting initial membership certificate.

3.3 Evolving Procedure

Assume that Bob has the group membership certificate (s_B, r_B, w_{B_j}) at time period *j*. Then at time period *j*+1, he can compute new group membership certificate via **Evolving** function $f(x) = x^e (\mathrm{mod}\, n)$ and then his new group membership certificate becomes $(s_B, r_B, w_{B_{j+1}})$ where $w_{B_{j+1}} = (w_{B_j})^e \bmod n$.(**Note that** $w_{B_j} = (g^{s_B} ID_{GM} ID_B)^{-d^{T-j}} \bmod n$).

3.4 Sign Procedure

Suppose that Bob has the group membership certificate (s_B, r_B, w_{B_j}) at time period *j*. To sign a message *m* at time period *j*, Bob randomly chooses three numbers $q_1, q_2, q_3 \in Z_n^*$ and computes

$$z_1 = g^{q_1} y^{q_2} q_3^{e^{T-j}} \bmod n,$$

$$u = h(z_1, m)$$

$$r_2 = q_3 w_{B_j}^u \bmod n,$$

$$r_1 = q_1 + (s_B + k)u$$

$$r_3 = q_2 - r_B u,$$

The resulting group signature on *m* is (u, r_1, r_2, r_3, m, j).

3.5 Verify Procedure

Given a group signature (u, r_1, r_2, r_3, m, j), a verifier validates whether the group signature is valid or not. He computes as follows

$$\begin{aligned}
1)\ z_1' &= ID_{GM}^u g^{r_1} r_2^{h(r_2) e^{T-j}} y^{r_3} \bmod n \\
&= ID_{GM}^u g^{q_1 + (k+s_B)u} q_3^{e^{T-j}} w_{B_j}^{u e^{T-j}} y^{r_3} \bmod n \\
&= ID_{GM}^u g^{q_1} g^{s_B u} q_3^{e^{T-j}} g^{ku} (r_B ID_{GM} ID_B)^{-e^{T-j} d^{T-j} u} y^{q_2 - r_B u} \\
&= ID_{GM}^u g^{q_1} q_3^{e^{T-j}} g^{s_B u} ID_B^u (r_B ID_{GM} ID_B)^{-u} y^{-r_B u} y^{q_2} \\
&= g^{q_1} y^{q_2} q_3^{e^{T-j}}
\end{aligned} \quad (1)$$

2) checks $u' = h(z_1', m)$

and checks whether the equation $u \overset{?}{=} u'$ holds or not. If it holds, the verifier is convinced that (u,r_1,r_2,r_3,m,j)is a valid group signature on *m* from a legal group member.

3.6 Open Procedure

In case of a dispute, GM can open signature to reveal the actual identity of the signer who produced the signature. Given a signature(u,r_1,r_2,r_3,m,j), GM firstly checks the validity of the signature via the **VERIFY** procedure. Secondly, GM computes the following steps:

Step 1: computes $\eta = 1/u \bmod \phi(n)$.

Step2: computes $z_1' = ID_{GM}^u g^{r_1} r_2^{e^{T-j}} y^{r_3} \bmod n$.

Step 3: checks $r_2 / w_B^\eta = (z' / g^{r_1} y^{r_3}{}_1)^{d^{T-j}} \bmod n$.

If there is the corresponding w_B with (r_B, ID_B) satisfying the above Step3, it is concluded that ID_B is the actual identity of the signer.

3.7 Revoking Procedure

Suppose the membership certificate of the group member Bob need to be revoked at time period *j*, the group manager computes the following quantification:

$$R_j = w_B (r_B ID_B)^{d^{T-j}} \bmod \mathrm{n}$$

and publishes duple (R_j, j) in the CRL(the Certificate Revocation List). Given a signature (,u,r_1,r_2,r_3,m,j), when a verifier identifies whether the signature is produced by

a revoked group member or not, he computes the following quantification

Step 1: $z_1' = ID_{GM}^{u} g^{r_1} r_2^{e^{T-j}} y^{r_3} \bmod n$

Step 2: $z_1'(r_2^{-1}R_j^u)^{e^{T-j}} = g^{r_1} y^{r_3} \bmod n$ (2)

For the signature (u, r_1, r_2, r_3, m, j) , if the signature satisfies the above equation (2). We can conclude that the signature is revoked.

4 Security Analysis

In this subsection we show that our proposed group signature scheme is a secure group signature scheme and satisfies forward security.

Correct: we can conclude that a produced group signature by a group member can be identified from **equation (1)** of the above **Verifying Procedure.**

Anonymity: Given a group signature(u,r$_1$,r$_2$,r$_3$,m,j), z_1 is enerated through two random numbers q_1 and q_2 which are used once only and $u = h(z_1, m)$, so that we can infer that u is also a random number generated by random seed z_1 **.** Any one (except for a group manager) cannot obtain any information about the identity of this signer from the group signature(u,r$_1$,r$_2$,r$_3$,m,j).

Unlinkability: Given time period j, two different group signatures(u,r$_1$,r$_2$,r$_3$,m,j)and (u',r'$_1$,r'$_2$,r'$_3$,m', j), we can know that u (or u') is a random number generated by random seed z_1 **,** and u **is** different in each signing procedure and used once only, and u **or** random number q_1 and q_2 are included in r_1 and r_2 . However, an adversary cannot get the relation between the signature (u,r$_1$,r$_2$,r$_3$,m,j)and the signature(,u',r'$_1$,r'$_2$,r'$_3$,m', j).

Unforgeability: In this group signature scheme, the group manager is the most powerful forger in the sense. If the group manager wants to forge a signature at time period j, he chooses (z$_1$, r_2, r_3, j) (or (z_1, r_2, r_1, j)) and computes u=h(z$_1$, m). According to the equation (1), for solving r_1, he needs solve the discrete logarithm so that he cannot forge a group signature.
Furthermore, as an adversary, because an adversary hasn't a valid membership certificate, he cannot forge a group signature satisfying the verification procedure. And in view of the group manager, he cannot forge a valid group signature without knowing private k of group member.

Forward Security: Assume an attacker breaks into a group member's system in time period j and obtains the member's membership certificate. Because of the one-way property of $f(x)$, the attacker cannot compute this member's membership certificate corresponding to previous time period. Hence the attacker cannot generate the group signature corresponding to the previous time.

Assume that the group member Bob is revoked at time period j, the group manager only revokes the group membership certificate of the time period j. then any valid signature with corresponding time period before j is still accepted. Because of the obtained signature (u,r$_1$,r$_2$,r$_3$,m,t),t<j. the signature (u, r$_1$,r$_2$,r$_3$,m,j) is still a valid signature on m and Bob would not need to produce a new signature on m .

Revocation: When a user, say Bob, is expelled from the group starting from the time period i, R_i and i will be published in CRL. Assume a verifier has a signature for period j, where $j \geq i$. To check whether the membership certificate of the group member has been expelled, the verifier simply computes $R_j = (R_i)^{e^{j-i}}$ and checks whether the equation $z_1'(r_2^{-1}R_j^u)^{e^{T-j}} = g^{r_1} y^{r_3} \bmod n$ holds or not. If it holds, it means that the signature has been revoked.

Collision-resistant: Assume that two group members collude to forge a signature. Because they don't know factorization of n and membership certificate of Bob, Furthermore, in Join phase, though the identification for each group member is computed by themselves according to number k , for two conspiracy group members, it is equivalent to forge group manager Schnorr signature to produce a new membership certificate for them. So that they cannot produce a valid membership certificate. Suppose that the group manager and a group member collude to produce the signature of a group member Bob. because they don't know the private key k or $(r_B, s_B w_{B_i})$ of group member Bob respectively, they cannot forge Bob 's signature.
Efficiency**:** for the whole signature phase and verification phase, our scheme only needs 7 modular exponentiations, however, Song's scheme needs more than 20 modular exponentiations. This implies that our scheme is very practical in large group applications.

5 Efficiency Analysis

In this section we show the efficiency of our scheme over that of Song scheme. In a signature scheme, the computational cost of signature is mainly determined by modular exponentiation operator. Let E, M and H respectively denote the computational load for exponentiation, multiplication and hash. Then the following table shows the comparison of computational load of our scheme vs. Song scheme.

Table1: our scheme vs. Song scheme

Scheme	Signing phase computation	Verifying phase computation	Total computation
Song's	22E+1H+6M	14E+1H+6M	36E+2H+12

Scheme			M
Proposed Scheme	4E+3H+5M	4E+3M+1H	8E+8M+4H

Signing phase and verifying phase in our scheme have less computation against Song's scheme. Modular exponentiation is a complicated operator and plays a determinate role in a signature scheme. From the above data, we conclude that our scheme has computational advantage over that of Song. To the best of our knowledge, it takes the much least computation in group signature schemes. Hence, Our proposed scheme is suitable to large group.

6 CONCLUSION

In this paper, we propose a new group signature scheme with forward-security. Our scheme satisfies not only the traditional security properties of the previous group signature schemes, but also forward security. Our scheme is efficient in the sense in that it is independent of the number of the group members and the size of group signature and the size of group key are independent of the number of time periods and the number of revoked members. Our scheme is a practical group signature scheme.

7 REFERENCE

[1] D. Chaum, F. Heyst. (1992) Group Signature. Proceeding EUROCRYPT'91. Springer-verlag, pp. 257-265.

[2] G.Ateniese,J. Camenish, M. Joye, and G. Tsudik. (2000) A Practical and Provably Secure Coalition-Resistant Group signature Scheme. In M.Bellare, editor, Crypto'2000, vol(1880) of LNCS, Springer–Verlag, pp. 255-270.

[3] G. Ateniese and G. Tsudik.(1999)Some Open Issues and New Direction in Group Signature. In Financial Cryptograph'99,

[4] Ross Anderson. (1997) Invited Lecture, 4th ACM Computer and Communications Security.

[5] Dawn Xiaodong Song,(2000) Practical forward secure group signature schemes. Proceedings of the 8th ACM conference on Computer and Communications Security, Pennsylvania, USA, November, pp. 225-234.

[6] J. Camenish and M. Michels. (1999) A Group Signature with Improved Efficiency. K. Ohta and. Pei, editors, Asiacrypt'98.Vol 1514 of LNCS, Springer-Verlag, pp. 160-174.

[7] W. R. LEE, C. C. CHANG. (1998)Efficient Group Signature Scheme Based on the Discrete Logarithm. IEE Proc. Computer Digital Technology, vol.145 (1), pp.15-18.

[8] Constantin Popescu. (20001)An Efficient Group Signature Scheme for Large Groups. STUDIES ININFORMATICS AND CONTROL With Emphasis on Useful Applications of Advanced Technology, Vol.10 (1), pp. 3-9.

[9] Emmanuel Bresson and Jacques Stern.(2001)Efficient Revocation in Group Signature.PKC'2001, LNCS 1992, Springer-verlag, Berlin Heidelberg pp. 190-206, 2001.

[10] Michel Abdalla and Leonid Reyzin.(2000) A new forward secure digital signature scheme. In ASIACRYPT, Springer-Verlag, pp. 116-129.

[11] Y. Tseng, J. Jan. (1998) A novel ID-based group signature, In T.L Hwang and A.K.Lenstra, editors, international Computer Symposium, Workshop on Cryptology and Information Security, Tainan, 1998, pp. 159-164.

[12] C. Popescu. (2000)Group signature schemes based on the difficulty of computation of approxi-mate e-th roots, Proceedings of Protocols for Multimedia Systems (PROMS2000), Poland, pp. 325-331,

[13] S.Kim, S.Park, D.Won, (1998) Group signatures for hierarchical multi-groups, Information Security Workshop, Lecture Notes in Computer Sciences 1396, Springer-Verlag, pp. 273-281.

[14] M.Stadler,(1996)Publicly verifiable secret sharing, Advances in Cryptology, EUROCRYPT'96 lecture Notes in Computer Sciences 1070, Springer-Verlag, 1996, pp. 190-199.

[15] A. Fiat and A. Shamir.(1986) How to prove yourself: practical solutions to identification and signature problems. In Advances in Cryptology-CRYPTO'86, vol. 263 of LNCS, pp.186–194, Springer -Verlag,

[16] S. Goldwasser, S. Micali,and R.Rivest.(1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing, 17(2): 281–308,

[17] J. Kilian and E. Petrank.(1998) Identity escrow. In Advances in Cryptology —CRYPTO'98, vol.1642 of LNCS, pp. 169–185, Springer-Verlag,

[18] A. Lysyanskaya and Z. Ramzan. (1998)Group blind digital signatures: A scalable solution to electronic cash. In Financial Cryptography (FC'98), vol. 1465 of LNCS, pp. 184–197, Springer-Verlag

[19] R.Gennaro, H.Krawczyk,and T.Rabin (2000) RSA-based Undeniable Signature. J. Cryptology, Volume (13)4, pp 397-416

[20] Giuseppe Ateniese, B. de Medeiros,Efficient Group Signatures without Trapdoors, In ASIACRYPT 200

Using Finite-State Transducer Theory for Representation of Very Large Scale Lexicons

Matej Rojc, Zdravko Kačič
Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, Maribor, Slovenia
Phone: +386 2 220 7223, Fax: +386 2 2511 178
matej.rojc@uni-mb.si, kacic@uni-mb.si

Keywords: finite-state transducers, natural language resources, multilingual text-to-speech synthesis, morphology, lexicons

Received: September 17, 2002

In multilingual text-to-speech synthesis systems, many external extensive natural language resources are used, especially in the text processing part. Therefore it is very important that representation of these resources is time and space efficient. It is also very important that language resources for new languages can be easily incorporated into the system, without modifying the common algorithms developed for multiple languages. In this regard the use of large external language resources represents an important problem because of the needed space and slow lookup-time. In the paper a method and results of compiling large lexicons, with an example of compiling German phonetic and morphology lexicons (CISLEX), into corresponding finite-state transducers (FSTs) are presented. Each lexicon consisted of about 300.000 words. Representation of large lexicons using finite-state transducers is mainly motivated by considerations of space and time efficiency. For both lexicons a great reduction in size and optimal access time was achieved. The starting size for German phonetic lexicon was 12.53 MB and 18.49 MB for morphology lexicon. The final size of the corresponding FST was only 2.78 MB for the phonetic lexicon and 6.33 MB for the morphology lexicon. At the same time the look-up time is optimal, since it depends only on the length of the input word and not on the size of the lexicon. Using such representation, the integration of lexicons for new languages into the multilingual TTS system is easy and does not require any changes of algorithms that use such lexicons.

1 Motivation

Finite-state machines are already used in many areas of natural language processing. From the computational point of view, their use is mainly motivated by considerations of space and time efficiency. Linguistically, the finite-state machines [6][8][10][11] allow one to describe easily most of the relevant local phenomena in the language. They provide also compact representation of external language specific resources needed for knowledge representation in the automatic text-to-speech synthesis systems. These features of finite-state machines are of major importance especially when we are dealing with multilingual text processing in text-to-speech synthesis systems (TTS systems).

In multilingual text-processing module for the multilingual TTS system, external natural language resources (e.g., phonetic, morphology lexicons etc.) represent an important problem, regarding the memory usage and time needed for lookup process.

In the following sections we are presenting an approach for compiling such lexicons into finite-state transducers that represent their time and space optimal representation. The effect of using finite-state transducers for representation of external natural language resources is great reduction of the memory usage required by the lexicons and the optimal access time (required for obtaining information) that is independent from the size of the lexicons. The whole compilation process into finite-state transducers will be presented and at the end results obtained for the German lexicons described.

2 Finite-state automata and finite-state transducers

2.1 Finite-state automata (FSA)

Finite-state automata (FSA) [6] can be seen simply as an oriented graph with labels on each arc. Fundamental theoretical properties make FSAs very flexible, powerful and efficient. FSAs can be seen as defining a class of graphs and also as defining languages.

Definition

A finite-state automaton A is a 5-tuple (Σ, Q, i, F, E) where Σ is a finite set called the alphabet, Q is a finite set of states, $i \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $E \subseteq Qx(\Sigma \cup \{\in\})xQ$ is the set of edges.

FSAs have been shown to be closed under union, Kleen star, concatenation, intersection and complementation, thus allowing for natural and flexible descriptions. In addition to their flexibility due to their closure properties, FSAs can also be turned into canonical forms that allow for optimal time and space efficiency.

2.2 Finite-state transducer (FST)

FSTs [9] can be interpreted as defining a class of graphs, a class of relations on strings, or a class of transductions on strings. On the first interpretation, an FST can be seen as an FSA, in which each arc is labelled by a pair of symbols rather than by a single symbol.

Definition
A finite-state transducer T is a 6-tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$ such that:

- Σ_1 is a finite alphabet, namely the input alphabet,
- Σ_2 is a finite alphabet, namely the output alphabet,
- Q is a finite set of states,
- $i \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states,
- $E \subseteq QX\Sigma_1^* X\Sigma_2^* XQ$ is the set of edges.

As with FSAs, FSTs are also powerful because of the various closure and algorithmic properties. In the paper we adhere to the following conventions when describing an FST: final states are depicted by bold circle; ε represents the empty string; the initial state (labelled 0) is the leftmost state appearing in the figure.

2.3 Use of FSMs for time and space optimal Lexicon representation

When representing lexicons by automata, in general, many entries share the same codes (strings, representing some piece of information). The number of codes is then small compared to the number of entries. Newly developed lexicons are more and more accurate and the number of codes can increase considerably. The increase in number of codes also increases the smallest possible size of such lexicons. During the construction of the automaton one needs to distinguish different codes, therefore space required for an efficient hashing of the codes can also become costly.

Available lexicons that were used in this experiment suggest that the representation by automata would be less appropriate. Since morphological and phonetic lexicons can be viewed as a list of pairs of strings, their representation using finite-state transducers [10] seems to be very appropriate. The results given at the end of this paper also confirm this assumption. Representation of lexicons using finite-state transducers on the other hand also provides reverse look-up capability.

In the multilingual TTS system morphological and phonetic lexicons represent part of the external natural language dependent resources used by multilingual text-processing engine. It is desired that language independent modules for morphology analysis and grapheme-to-phoneme conversion inside the multilingual text-processing engine use common algorithms for multiple languages. This is possible when external natural language dependent resources are represented as finite-state transducers. Integration of new lexicons for new languages in the whole TTS system is then very easy, since only compilation procedure (off-line) has to be performed.

3 Compilation process of large scale lexicons into finite-state transducers

3.1 Lexicons preparation

The methods used in the compilation of large scale lexicons into finite-state transducers (FST) assume that the lexicons are given as large lists of strings and not as a set of rules as considered by Mehryar Mohri [3] for instance. Obviously morphological and phonetic lexicons can be viewed as a list of pairs of strings and their representation using finite-state transducers seems to be very appropriate. In Fig. 1 some items from German phonetic and morphology lexicons are shown.

As with automata, direct construction of the sequential transducer representing a large-scale lexicon, is not possible because the construction leads to a blow up for a large number of entries. To avoid this, splitting the lexicon into several parts is performed. Then the construction of the corresponding sequential transducers including minimization operation follows. Using union, determinization, and minimization operations, only one transducer representing the whole lexicon is obtained at the end (Fig.2).

3.2 Determinization of finite-state transducers

The algorithm used is close to the powerset construction used for determinizing automata [3]. The main difference is that here one needs to provide states of the sets with strings. These strings correspond to a delay in the emission that is due to the fact that outputs corresponding to a given input can be different. Therefore only the longest common prefix of outputs can be kept and subsets represent actually pairs (state, string). The pseudo-code for the algorithm to determinize a transducer T_1 is given in Fig. 3.

"Abte
"E p - t @
"Abten
"E p - t @ n
"Abtissin
E p - t "I - s I n
"Abtissinnen
E p - t "I - s I - n @ n

```
"Ackern
"E - k 6 n
"Aderchen
"E: - d 6 - C @ n
"Aderchens
"E: - d 6 - C @ n s
.....
```

a)

```
"Abte
abt.mask(NS1,NP12)#0:amM:gmM:nmM
"Abten
abt.mask(NS1,NP12)#0:dmM
"Abtissin
"Abtissin.fem(NS0,NP5)#0:aeF:deF:geF:neF
"Abtissinnen
"Abtissin.fem(NS0,NP5)#0:amF:dmF:gmF:nmF
"Acker
acker.mask(NS2,NP11)#0:amM:gmM:nmM
"Aderchen
"Aderchen.neut(NS2,NP0)#0:aeN:amN:deN:dmN:gm
N:neN:nmN
"Aderchens
"Aderchen.neut(NS2,NP0)#0:geN
.....
```

b)

Figure 1: German phonetic (a) and morphology lexicons (b). German morphology lexicon is coded according to CISLEX specification [5].

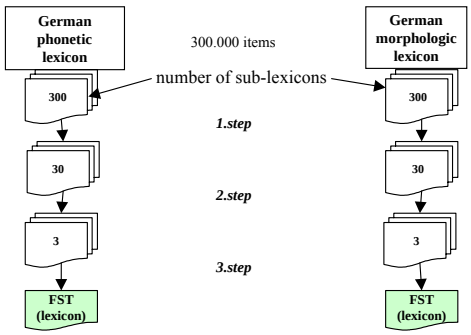


Figure 2: Lexicons preparation.

At each step a new state q_2 is considered as can be seen in line 5. State q_2 is a final state only if it contains a pair (q,w), where q is final in T_1. String w is the final output at the state q_2. In line 10, each input label a of the transitions leaving the states of the subset q_2 is considered. A transition is constructed from state q_2 to state $\delta_2(q_2,a)$ with output $\sigma_2(q_2,a)$. Output $\sigma_2(q_2,a)$ represents the longest common prefix of the output labels of all the transitions leaving the states q of q_2 with input label a, when left concatenated with their delayed string w. State $\delta_2(q_2,a)$ is the subset made of pairs (q',w'). Here q' is a state reached by one of the transitions with input label a in T_1 and $w' = [\sigma_2(q_2,a)]^{-1} w\sigma_1(q,a,q')$ is the delayed string that could not be outputed earlier in the algorithm. String $[\sigma_2(q_2,a)]^{-1} w\sigma_1(q,a,q')$ is a well defined string since $[\sigma_2(q_2,a)]$ is a prefix of all $w\sigma_1(q,a,q')$ as can be seen from line 10.

In Fig. 5 we can see the result of using the determinization algorithm on transducer from Fig. 4 (obtained using union operation). In this example the number of states of the determinized transducer T_2 is already less than in T_1. Experiments showed that this method is very efficient in constructing transducers for representation of large lexicons. The disadvantage of this algorithm is that the outputs are pushed toward final states, which creates a long delay in emission. But fortunately sequential transducers can be minimized as we will show in the next section. An important characteristic of the minimization algorithm is that it pushes back outputs as much as possible toward the initial state. In such a way we can eliminate the problem just mentioned.

Determinize_transducer(T_1,T_2)

1 $F_2 \leftarrow \emptyset$

2 $i_2 \leftarrow \bigcup_{i \in I_1} \{(i, \varepsilon)\}$

3 $Q_2 \leftarrow \{ i_2 \}$

4 while $Q \neq \emptyset$

5 do $q_2 \leftarrow$ head[Q]

6 if (there exists $(q, w) \in q_2$ such that $q \in F_1$)

7 then $F_2 \leftarrow F_2 \cup \{q_2\}$

8 $\phi_2(q_2) \leftarrow w$

9 for each a such that $(q,w) \in q_2$ **and** $\delta_1(q,a)$ defined **do**

10 $\sigma_2(q_2, a) \leftarrow \bigwedge_{(q,a) \in J_1(a)} \left[w \cdot \bigwedge_{q' \in \delta_1(q,w)} \sigma_1(q,a,q') \right]$

11 $\delta_2(q_2, a) \leftarrow \bigcup_{(q,w,q') \in J_2(a)} \{(q', [\sigma_2(q_2,a)]^{-1} w \cdot \sigma_1(q,a,q'))\}$

12 if ($\delta_2(q_2, a)$ is a new state)

13 then Enqueue(Q,$\delta_2(q_2,a)$)

14 Dequeue(Q)

Figure 3: Pseudocode for determinization algorithm [3].

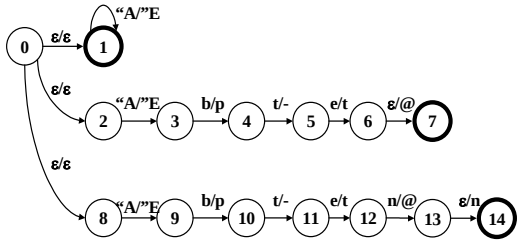


Figure 4: Union operation done on a few word items in the German phonetic lexicon (T_1).

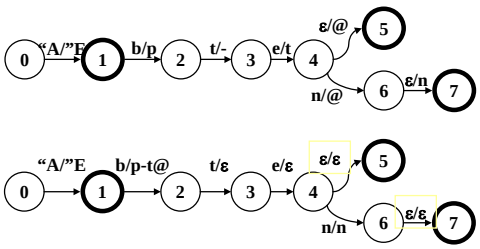


Figure 5: Finite-state transducers T_2 (above) and T_3 (below) obtained after performing determinization and prefixation algorithms on finite-state transducer showed in Fig. 4.

3.3 Minimization of finite-state transducers

Sequential transducers allow very fast look-up. But transducers can also be minimized. Minimization algorithms help to make them also space efficient [1][2][4][7]. The whole minimization procedure for sequential transducers consists actually of two different algorithms. One is algorithm for computation of the prefix of a non-deterministic automaton [4] and the other is classical algorithm for minimization of automata [1][2]. In this section we will present the algorithm for computation of the prefix, as it is independent of the concept of sequential transducers and will describe the entire algorithm that allows derivation of minimal sequential transducers.

In the algorithm described, we use the following notation:

- G^T the transpose of G (the automaton obtained from G by reversing each transition);
- *Trans[u]* the set of transitions leaving $u \in V$;
- *Trans^T[u]* the set of transitions entering $u \in V$;
- *t.v* the vertex reached by *t* and *t.l* its label, for any transition *t* in *Trans[u]* (resp. in *Trans^T[u]*), $u \in V$;
- *out-degree[u]* the number of edges leaving $u \in V$;
- *in-degree[u]* the number of edges entering $u \in V$;
- E the set of edges of G.

1. First we compute π_u, the greatest common prefixes of all its leaving transitions:

$$\pi_u \leftarrow \Big(\bigwedge_{\substack{t \in \mathrm{Trans}[u] \\ t.v \in scc}} t.l\, X_{t.v}\Big) \wedge \Big(\bigwedge_{\substack{t \in \mathrm{Trans}[u] \\ t.v \notin scc}} t.l\Big) \quad \text{if } u \notin F,$$

$$\pi_u \leftarrow \varepsilon \quad \text{else;}$$

2. Then if $\pi_u \neq \varepsilon$, we can make a change of variables: $Y_u \leftarrow \pi_u X_u$. This second step is equivalent to storing the value π_u and solving the system modified by the following operations:

$$\forall t \in \mathrm{Trans}[u],\quad t.l \leftarrow \pi_u^{-1}\, t.l,$$
$$\forall t \in \mathrm{Trans}^T[u],\quad t.l \leftarrow t.l\, \pi_u.$$

The number of times these two operations are performed can be limited by storing in array N the number of empty labels leaving each state u of the strongly connected component *scc*. While $N[u] \neq 0$, there is no use to perform these operations as the value of π_u is ε. Also in the case that $N[u] = 0$ right after the computation of π_u, the π_u will remain equal to ε, as changes of variables will only affect suffixes of the transitions leaving u. This information can be stored using an array F, in order to avoid performing *step 1* in such situations or when u is a final state. In the algorithm we use a queue Q containing the set of states u with $N[u] = F[u] = 0$ for which the two operations above need to be performed, and an array *INQ* indicating for each state u whether it is in queue Q.

The above operations are started by initializing N and F to 0 for all states in *scc*, and by enqueuing in queue Q an arbitrarily chosen state u of the strongly connected component *scc*. Each time the transition of a state v of *Trans^T[u]* is modified, v is added to Q if $N[v] = F[u] = 0$. The property of *SCC's* (strongly connected component) and the initialization of N and F assure that each state of *scc* will be enqued at least once. *Steps 1* and *2* are operated until queue $Q = \varnothing$. This must happen as, except for the first time, *step 1* is performed for a state u if $N[u] = 0$. After the computation of the greatest common prefix we can have $N[u] = 0$ and then u will never be enqueued again, or $N[u] \neq 0$ and then a new non empty factor π_u of *P(u)* has been identified. It is obviously then, that each state u is enqueued at most *(|P(u)|+2)* times in Q, and after at most *(|Pmax|+2)* steps we have $Q = \varnothing$.

Prefix_Computation(G)

```
1   for each u ∈ V(G^SCC)
2   do for each v ∈ SCC[u]
3      do N[v] ← INQ[v] ← F[v] ← 0
4   Q ← v
5   INQ[v] ← 1
6   while Q ≠ ∅
7      do v ← head[Q]
8         Dequeue(Q)
9         INQ[v] ← 0
10        p ← GCP(G,v)
11        for each t ∈ Trans^T[v]
12           do if(p ≠ ε)
13              then if(t.v ∈ SCC[v] and N[t.v] > 0
                    and t.l = ε and F[t.v] = 0)
14                 then N[t.v] ← N[t.v] – 1
15                 t.l ← t.l p
16              if(N[t.v] = 0 and INQ[t.v] = 0 and
                    F[t.v] = 0)
17                 then Enqueue(Q,t.v)
18                 INQ[t.v] = 1
```

Figure 6: Pseudocode for the prefixation algorithm on finite-state transducers [4].

Once $Q = \varnothing$, it is easy to see that the system of equations has a trivial solution: $\forall u \in scc$, $X_u = \varepsilon$. It has a unique solution. Therefore, the system is resolved. Concatenating the factors π_u involved in the changes of variables corresponding to the state u gives the value of $P(u)$. The set of operations (2) are obviously equivalent to multiplying the label of each transition joining the states u and v, ($v \in scc$), at right by $P(v)$ and at left by $[P(u)]^{-1}$ if u is in scc. Thus the transformations described above do modify the transitions leaving or entering states of scc as desired. The above pseudocode gives an algorithm that computes $p(G)$ from G. In the algorithm, $V(G^{SCC})$ represents the set of states of the component graph of G. For each u in $V(G^{SCC})$, $SCC[u]$ stands for the strongly connected component corresponding to u. The function $GCP(G,u)$ called in the algorithm is such that it returns p, which is the greatest common prefix of all transitions leaving u ($p = \varepsilon$ if $u \in F$). It replaces each of these transitions by dividing them at left by p and counts and stores in $N[u]$ the number of empty transitions. If $N[u] = 0$ after the computation of the greatest common prefix or if u is a final state, the $F[u]$ becomes the value 1.

The computation of the greatest common prefix of n $(n>1)$ words requires at most $(|p|+1).(n-1)$ comparisons, where p is the result of this computation [4]. This operation consists of comparing the letters of the first word to those of the $(n-1)$ others until a mismatch or end of a word occurs. The same comparisons allow to obtain the division at left by p and the number of empty transitions. In case only one transition leaves v, the computation of the greatest common prefix can be assumed to be in $O(1)$. Therefore, the cost of a call of the function GCP for a state v ($\in V - F$) is $O((|p|+1)(out\text{-}degree(v)-1)+1)$. Here p is the greatest common prefix of the transitions leaving v.

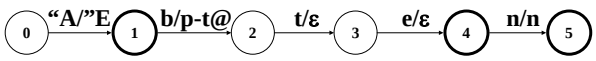
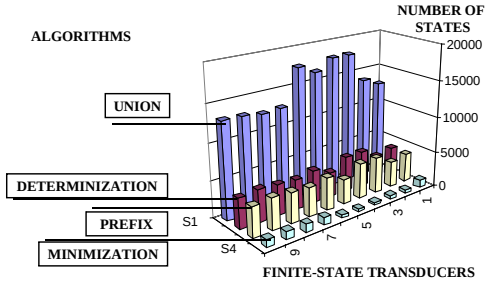


Figure 7: Finite-state transducer T_4 obtained using minimization algorithm in the sense of automata from T_3.

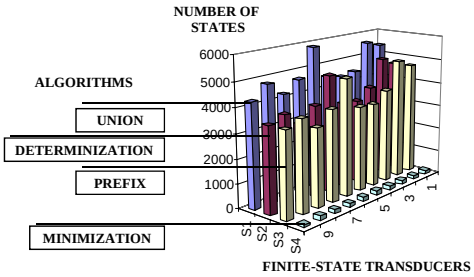
Given a sequential transducer T, the application of the *prefix computation* algorithm [4] to the output automaton of T has no effect on the states of T or on its transition function. Only the output function σ of T is changed. A minimal ST, that computes the same function as T, can be obtained by applying the *prefix computation* algorithm to the output automaton of T, and also the minimization algorithm in the sense of automata, to the resulting transducer [1][2]. Fig. 5 (transducer T_3) shows the result obtained after performing prefix computation algorithm on sequential transducer T_2 in particular case. The application of the *prefix computation* algorithm on T_2 leads to the transducer T_3, which computes the same function. Only outputs differ from those of T_2. In Fig. 7 the final obtained transducer is presented using minimization algorithm in the sense of automata.

4 Results

For the lexicons compilation the German large scale phonetic and morphology lexicons (CISLEX) [5] were used. In compilation process a large set of proprietary programs written in C++ that perform efficiently many operations on finite-state transducers and finite-state automata including determinization, minimization, union, intersection, compaction, prefixation, local extension and others were used.



a)



b)

Figure 8: Achieved reduction of the number of states obtained in the first step of compilation process – 10 randomly chosen transducers (*a:* phonetic lexicon. *b:* morphology lexicon.)

During construction of corresponding finite-state transducers, the following algorithms were used: union, determinization, prefix computation and classical minimization algorithms of finite-state automaton (Aho, Sethi, and Ullman; Hopcroft; Watson) [1][2][12]. Prefix computation algorithm was used before minimization algorithms. It pushes the output labels towards the initial state as much as possible.

All lexicons represent part of the language dependent external knowledge for morphology and grapheme-to-phoneme modules in the multilingual text-to-speech processing system. The starting sizes of phonetic lexicon and morphology lexicon were 12.52 MB and 18.49 MB. Both lexicons contained 300.000 items. Final size of corresponding finite-state transducer was 2.78 MB (120.386 states) for the first one and 6.33 MB (183.123 states) for the second.

In the first step of compilation, the great reduction of the number of states was achieved, what is evident from Fig. 8. This is also the reason, why we follow the procedure described under subsection 2.2 (Fig. 2). The number of

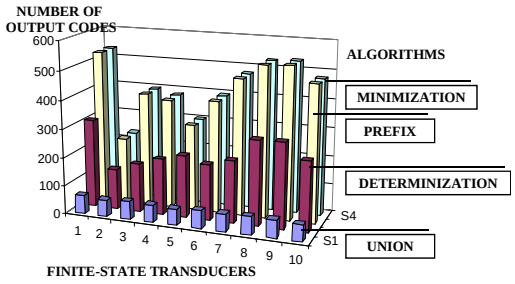
states decreased already after determinization algorithm as expected. The number of states obviously does not change after performing *prefix computation* algorithm. This algorithm works only on the output automaton of the corresponding transducer. It pushes back outputs as much as possible toward the initial state. The effect of *prefix computation* algorithm can be noticed only at the end of the compilation process, when much smaller finite-state transducers are obtained than in the case when only classical minimization algorithm after determinization would be performed. The answer for that can be found from the Fig. 5 (transducers T_2 and T_3). In the transducer T_3 we have after performing *prefix computation* algorithm newly created ε/ε transition labels. This empty transition labels are result of pushing back outputs toward the initial state. That's why after performing minimization algorithm in the sense of automata much smaller transducers are obtained.

In the Fig.9 we see that the number of output codes has increased after determinization and *prefix computation* algorithm were performed. In case that the *prefix computation* would not be performed, final number of codes would be significantly smaller, but the final transducer would also be much bigger (more states and transitions). According to the experiments it only makes sense to have more codes and much smaller transducer.

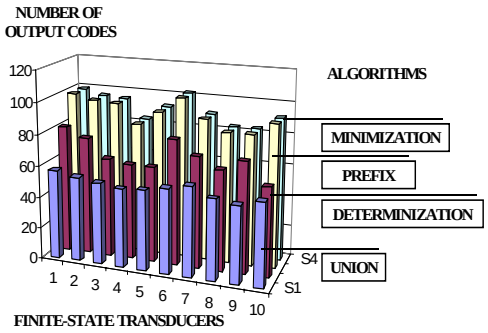
It is also interesting that in compilation of morphology lexicon, much more output codes is generated as in the case of phonetic lexicon (Fig. 9). The reason is that the morphology lexicon comprises much more information than the phonetic lexicon.

In the second step of the compilation process, the same situation regarding the state reduction can be observed as in the first step (Fig. 10). Only reduction of the number of states is smaller and there is no significant increase of the number of output codes (Fig. 11).

In Table 5 the final results for the obtained finite-state transducers for German phonetic and morphology lexicons are given. The number of input codes is the same for both lexicons and the number of output codes is two-times bigger in case of morphology lexicon. The reason is that the information field in the morphology lexicon is substantial longer (Fig. 1).

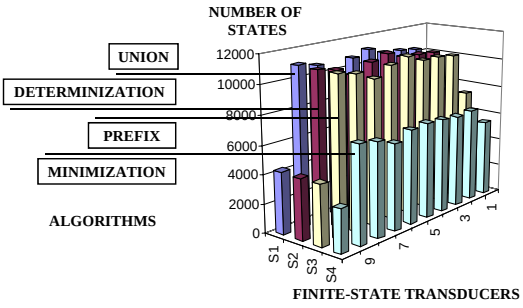


a)

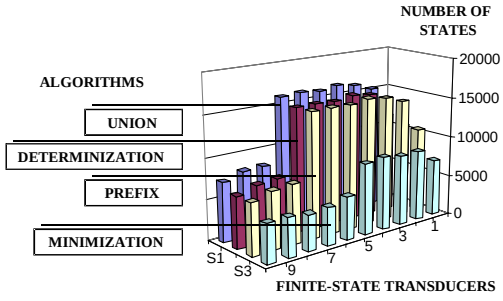


b)

Figure 9: Increasing number of output codes in the first step of compilation process – 10 randomly chosen transducers (*a:* phonetic lexicon. *b:* morphology lexicon.)



a)



b)

Figure 10: Achieved reduction of the number of states obtained in the second step of compilation process – 10 randomly chosen transducers (*a:* phonetic lexicon. *b:* morphology lexicon.)

	FST_1	**FST_2**
Number of input codes	**61**	**61**
Number of output codes	**34.879**	**87.204**
Size of output vocabulary	**343 KB**	**3.6 MB**
Number of states	**112.498**	**169.613**
Number of transitions	**200.801**	**325.839**
Size of ASCII file	**6.6 MB**	**11.53 MB**
Size of bin file	**2.78 MB**	**6.33 MB**

*Table 5: The f*inal finite-state transducers representing German phonetic (***FST_1***) and German morphology lexicon (***FST_2***).

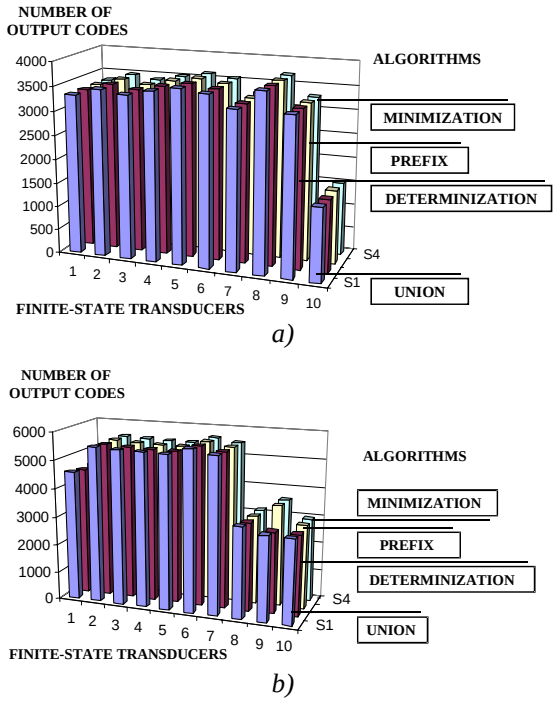


Figure 11: Increasing number of output codes in the second step of compilation process – 10 randomly chosen transducers (*a:* phonetic lexicon. *b:* morphology lexicon.)

5 Conclusion

Performing *determinization* of finite-state transducer obviously results in significant decrease in number of states. One disadvantage of the determinization algorithm is that the outputs are pushed toward final states that create a long delay in emission. But using *prefix calculation* algorithm before classical minimization algorithms for automata, the problem can be efficiently resolved. An important characteristic of this algorithm is that it pushes back outputs as much as possible toward the initial state. As expected, the number of states remains unchanged after performing *prefix calculation* algorithm. The efficiency of this algorithm can be seen only after performing classical minimization algorithm, when much smaller number of states is obtained than in case if only determinization and minimization algorithms would be performed. From table 5 it can be seen that finite-state transducers can efficiently represent large lexicons. They provide fast look-up time, double side look-up, and compactness.

6 References

[1] Bruce William Watson, *Taxonomies and Toolkits of Regular Language Algorithms*, PhD Thesis, Eindhoven University of Technology and Computing Science, 1995.

[2] Watson, B.W., *A taxonomy of finite automata minimization algorithms*, Computing Science Report 93/44, Eindhoven University of Technology, The Nederlands, 1993.

[3] Mehryar Mohri, *On Some Applications of Finite-State Automata Theory to Natural Language Processing*, Natural Language Engineering 1, Cambridge University Press, 1996.

[4] Mehryar Mohri, *Minimization Algorithms for Sequential Transducers*, Theoretical Computer Science, 234:177-201, March 2000.

[5] Guenthner, F.&P. Maier, *Das CISLEX Woerterbuch system*, CIS-Bericht-94-76.

[6] Aho, Alfred V., John E. Hopcroft, and Jeffrey D. Ullman, *The design and analysis of computer algorithms*. Addison Wesley: Reading, MA 1974.

[7] Bauer, W, *On minimizing finite automata*, EATCS Bulletin, 35 1988.

[8] Berstel, Jean and Cristophe Reutenauer, *Rational Series and Their Languages*, Springer-Verlag: Berlin-New York 1988.

[9] Crochemore, Maxime, *Transducers and repetitions*, Theoretical Computer Science, 45 1986.

[10] Hopcroft, John E. and Jeffrey D. Ullman, *Intoduction to Automata Theory, Languages, and Computation*. Addison Wesley: Reading MA 1979.

[11] Kuich, Werner and Arto Salomaa, *Semirings, Automata, Languages*, Number 5 in EATCS Monographs on Theoretical Computer Science. Springer Verlag, Berlin, Germany 1986.

[12] Mehryar Mohri, *Language Processing with Weighted Transducers*, In Proceedings of the 8th annual Traitement Automatique des Langues Naturelles (TALN 2001). Tours, France, July 2001.

The Parameters Tuning for Evolutionary Synthesis Algorithm

Gregor Papa and Jurij Šilc
Computer Systems Department
Jožef Stefan Institute
Jamova c. 39
SI-1000 Ljubljana
Slovenia
gregor.papa@ijs.si, jurij.silc@ijs.si, http://csd.ijs.si

Keywords: evolutionary, scheduling, allocation, genetic operators, tuning

Received: January 29, 2003

This paper covers the evaluation and fine-tuning of different values of genetic operator's parameters in the process of optimizing the designs of the integrated circuits. We investigated the interdependence of various values of these parameters in the use over the set of test-bench circuits, as well as their influence on the quality of the final solution and the convergence speed. Due to the increasing usage of the evolutionary optimization in the area of the integrated circuit design, there is a need to find a proper combination of genetic operators parameters' value to make optimal solutions. Therefore, it is important to perform this kind of evaluation for each new problem to be solved.

1 Introduction

The area of evolutionary computation is very popular but there is always a problem of defining a proper value of parameters of genetic operators. A standard genetic algorithm uses four different parameters that have to be defined in advance, before the algorithm is actually used. These are: the number of generations, the size of the population, the probability of crossover, and the probability of mutation [1].

There are some proposals for setting of these parameters according to the problem size and according to the area of the problem. But these proposals are not always applicable or are not suitable for all problems. Also, there are no proposals for any additional operators, used in some optimizations, which improve the performance of the algorithm.

To find some dependencies between the parameters and the problem that has to be solved, we made the evaluation, similar to that in [7]. We study an evolutionary approach that automatically generates circuit designs. We managed to point to some interesting dependencies between parameters themselves and to determine what values should be used in our optimizations when working with evolutionary-oriented algorithms.

2 ECSA algorithm

The facts presented in the introduction paragraphs and promising results of different evaluations [4, 9, 10] took us to the Evolutionary Concurrent Scheduling and Allocation (ECSA) design approach [8]. This approach considers scheduling and allocation constraints, allows short design time and can find globally optimal solutions. The input description of the integrated circuit (IC) is transformed into two basic (initial) schedules, obtained by As-Soon-As-Possible and As-Late-As-Possible algorithms. Functional units (FUs) used in first case are those fastest for each operation and in second case those slowest for each operation. These two schedules present some kind of boundary solutions, since all other solutions are executed in-between the time limits defined by these two schedules. Namely, no other solution can be faster or slower, considering different combinations of used units.

Each solution has to be properly encoded (into the chromosome), i.e., each operation's start time and FU have to exist in the chromosome. Initial population is built upon the two initial solutions, which are multiplied to form the population with so-called boundary solutions. The optimal solution has to be somewhere in-between the boundaries, therefore genetic operators (crossover, mutation, variation) transform those encoded solutions. With transformations their start times and allocated FUs are changed. The final solution obtained by genetic operators is also influenced by simulated annealing algorithm [6], which improves the solution if it stopped somewhere near the globally optimal point.

2.1 Encoding

The chromosome string consists of the numbers that represent the starting time of each operation and the allocated unit for each operation, where the position in the string depends on the order of the operations in the input IC description. This means that the chromosome consists of pairs of time/space information for each operation. And the genetic operators can influence both parts of that information, either together or separately.

The selected encoding type is chosen because of its convenience. When strings have to be further transformed, checked and analyzed, there is no need for any additional conversion of their values. In addition, the used implementation of genetic operators can check the changed values (their feasibility) instantly, without any transformation. The correctness of the transformation can therefore be checked within the function itself.

2.2 Cost function

One of the most important parts of the algorithm is its cost function. To obtain the cost (Eq. 1) of a certain circuit, the algorithm has to evaluate the required number of resources. In contrast to the other multi-objective functions that give more than one final solution, this one already includes the decision making part, which chooses one solution form all the solutions on the Pareto front.

$$
\begin{aligned}
Cost &= \sqrt{\sum_{i=1}^{N}(cost_{f_i})^2 + cost_r^2 + cost_b^2 + cost_t^2} \\
cost_{f_i} &= w_{f_i} F_i \\
cost_r &= w_r n_r \\
cost_b &= w_b n_b \\
cost_t &= w_t T
\end{aligned}
\qquad (1)
$$

The elements of the function above are calculated as follows.

The number n_{fi} is the highest number of the i-th functional unit needed in a separate control step.

The number n_r is the highest number of variables needed in a separate control step. We consider variables that are needed by the functional unit as input data, variables that are returned as output data, and variables that are not used at the moment but will be used in some of the later control steps or must be available until the end of the execution of all operations.

The number n_b is the highest number of data transmissions (into or from the functional units) in a separate moment.

The execution time, T, is the time needed to execute all the operations of the schedule.

The weights w_{fi}, w_r, w_b, and w_t are the weights of functional units, registers, buses and time, respectively, to be considered in the IC quality-evaluation cost function. The first three weights are proportional to their silicon area in the IC, while w_t reflects our IC speed constraints.

According to the different approaches of multi-objective functions [3] and their efficiency we chose the presented distance function with the variable weight of separate criteria. With this approach it is possible to simplify the conditions or to expose some criteria. As mentioned before, the solution that is closest to the origin of the search space can be found.

2.3 Genetic operators and parameters

In each iteration, e.g., generation, of the algorithm there are four genetic operators that transform the chromosome. They consider data dependencies and the given library of available FUs. Each time after genetic operators transform the chromosome, the chromosome is checked to meet all constraints, considering data dependencies and unit types.

2.3.1 Selection

Upon the cost function values the worse solutions are aborted in the selection step and to ensure equally large population, these solutions are replaced with the best solutions. This ensures best solutions of the given generation to be surely involved in the next generation creation (elitism).

2.3.2 Crossover

In crossover task two approaches are used, each expressing the dominancy of the characteristics. After two crossover points are determined, in the first case the unit information is changed between the two chromosomes and start times are adapted, and in the second case the start times are changed and suitable unit is allocated. So the dominancy is expressed either in FUs or operations start times.

2.3.3 Mutation

Here, we also have two similar approaches to transform the chromosome. In both cases the starting time is changed. Either it is moved to later control steps with the use of faster FUs or it is moved to earlier control steps, if data dependencies allow that, with slower units.

2.3.4 Variation

After two operations are selected and when they are of the same type (e.g., additions), their FUs are switched. If needed also their start times are updated.

3 Test-bench circuits

3.1 Differential equation

Relatively small circuit of differential equation [11] has only 11 operations, but 4 different operation types (6 multiplications, 2 additions, 2 subtractions, 1 comparison), see Figure 1. This circuit is useful when testing libraries with different implementations of the same operation types.

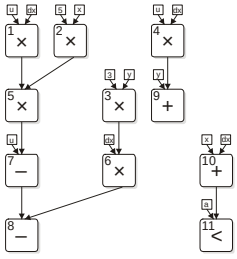


Figure 1: Differential equation

3.2 Elliptic filter

This filter [5] consists of 34 operations, but only two operation types: 26 additions and 8 multiplications (Figure 2). The circuit is suitable for comparison due to its size and operation dependencies, since they form two independent similar critical paths both influencing the circuit delay.

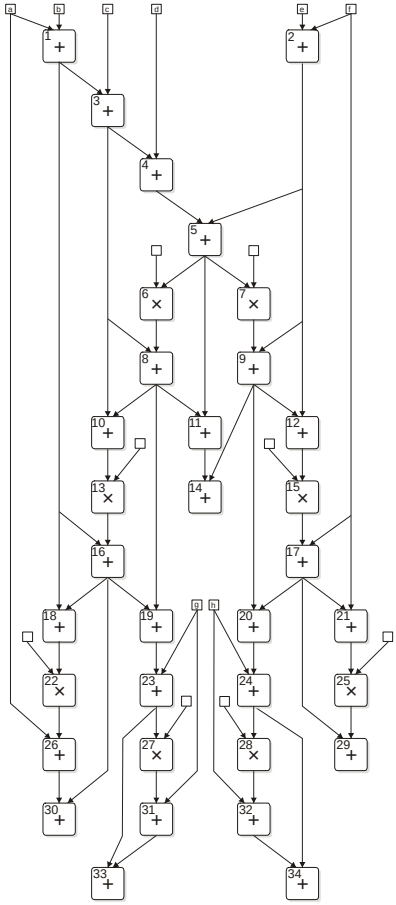


Figure 2: Fifth-order elliptic filter

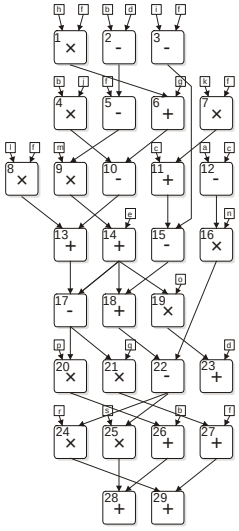


Figure 3: Bandpass filter

3.3 Bandpass filter

One of the implementations of the bandpass filter [5] is the circuit used for our evaluation. It consists of 29 operations; 11 multiplications, 10 additions and 8 subtractions (Figure 3). Due to data dependencies almost all operations influence the circuit delay.

3.4 Least mean square filter

This filter for signal adaptation (noise reduction) is based upon least mean square method [2]. It consists of 47 operations; 24 multiplications and 23 additions (Figure 4). This test-bench circuit is useful due to its size and unique data dependencies.

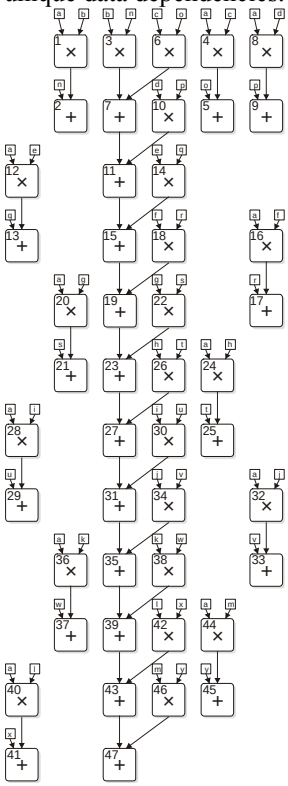


Figure 4: Least mean square filter

4 Evaluation

Considering 18750 different schedules of each circuit and different combinations of parameters, we statistically compared the results according to their cost function (Eq. 1). For each of described four test-bench circuits we made a set of 3125 different combinations of parameters (generations, populations, crossover, mutation and variation). We repeated the optimization process with each combination five times to reduce the influence of statistical error and to get the average fitness of solutions obtained by each combination of parameters.

The solutions with fitnesses of top 20% of all fitnesses for a certain circuit were defined as high quality solutions and solutions with bottom 20% of fitnesses were defined as low quality solutions

To ensure most solutions being time-constrained (executed in shortest possible time) the weight w_t was set to extremely high value.

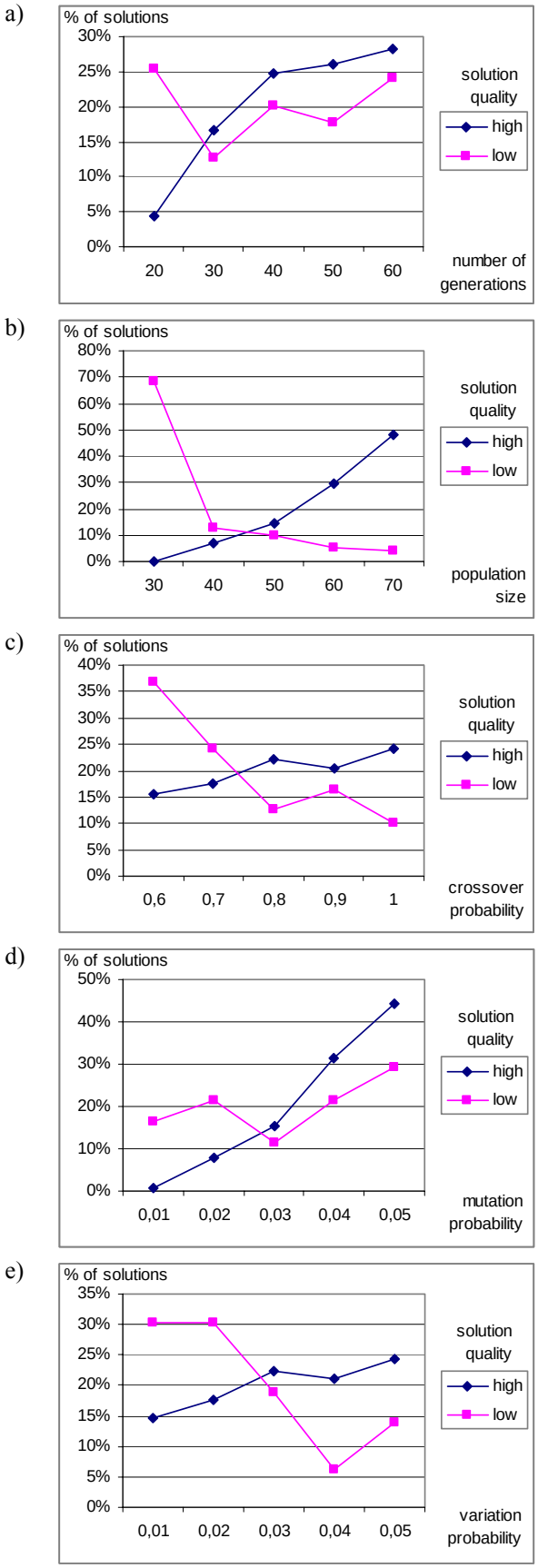


Figure 5: Differential equation

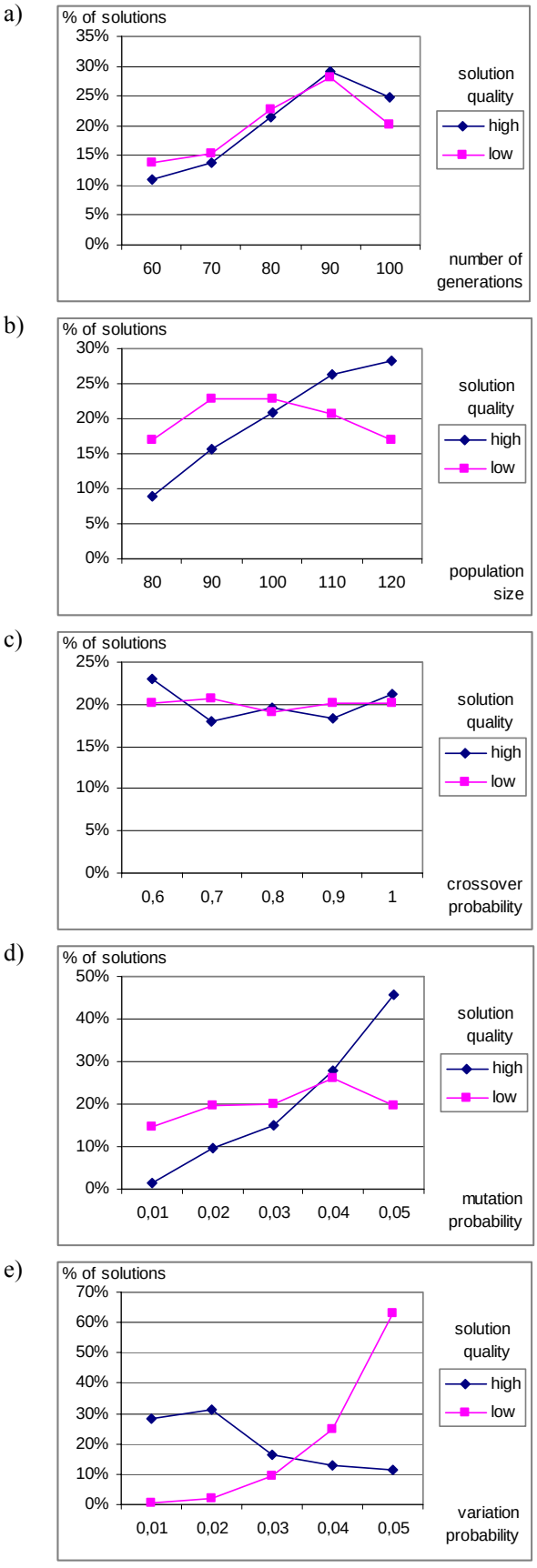


Figure 6: Fifth-order elliptic filter

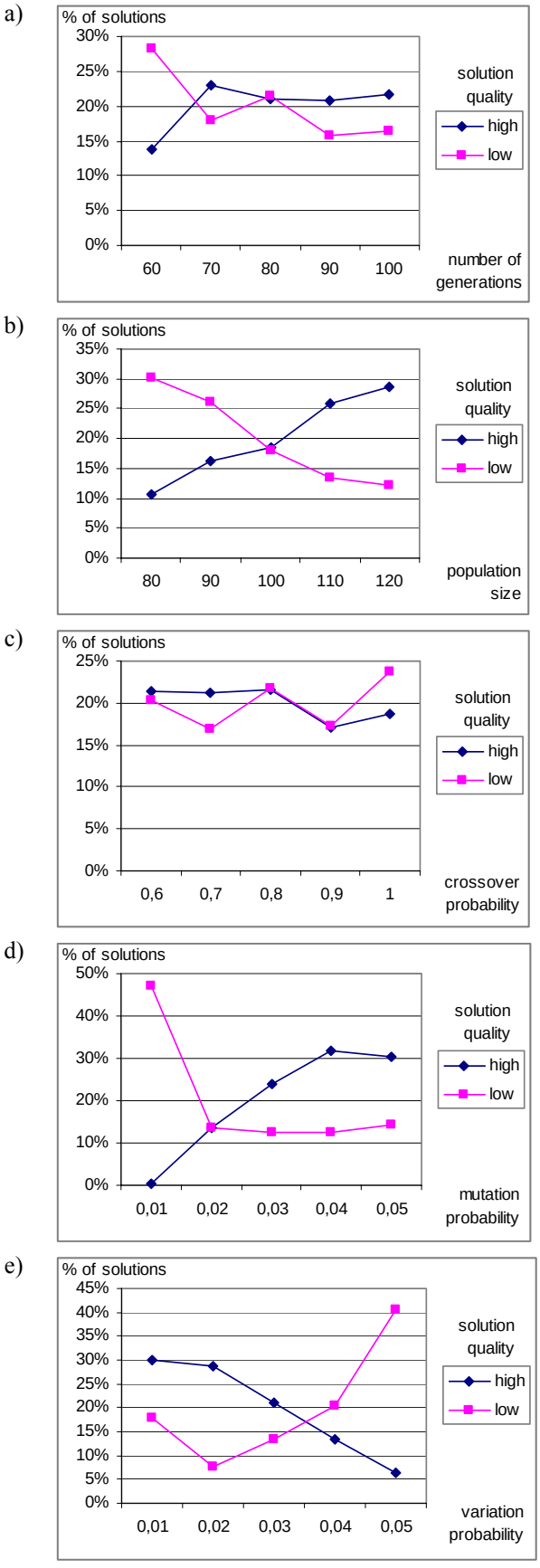


Figure 7: Bandpass filter

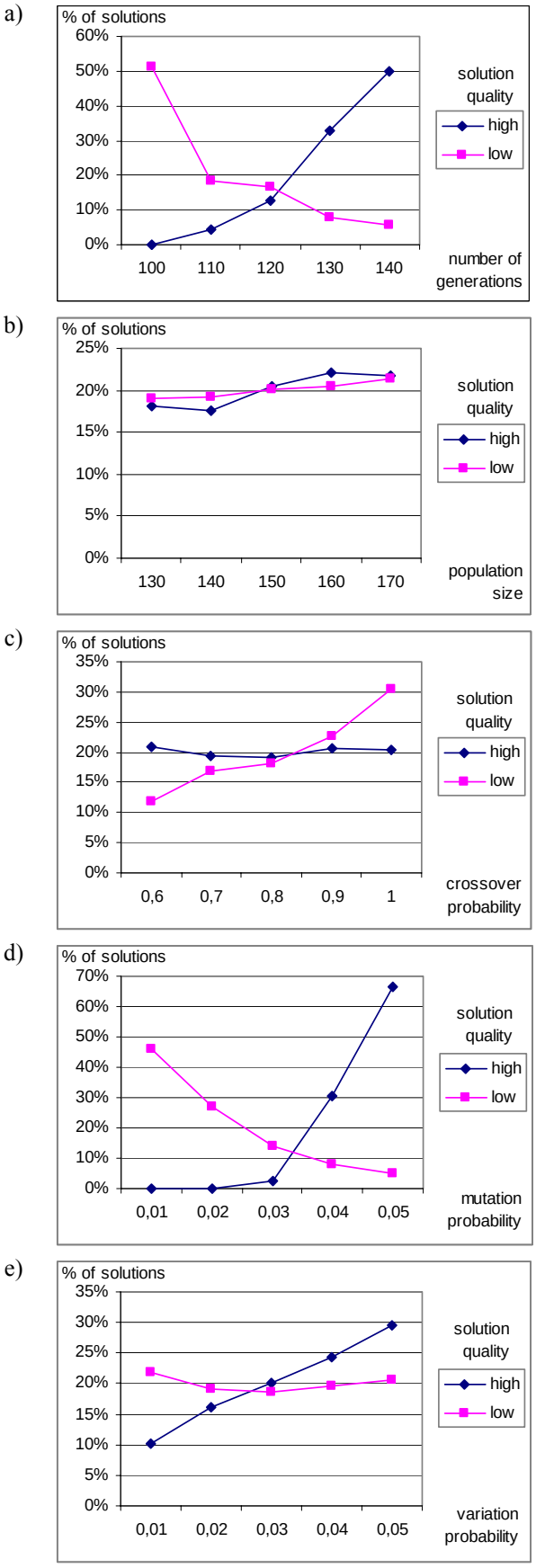


Figure 8: Least mean square filter

As presented in Figures 5, 6, 7, and 8, solutions with high quality are mostly obtained by the following values of parameters: probability of crossover is 0.7, probability of mutation is 0.04, and probability of variation is 0.03. Besides, considering the circuits sizes the number of generations and population size should be set to 3 times and 4 times of a circuit size, respectively.

The values of parameters in this combination are named as optimal values. These optimal values are determined upon the percentage of solutions with certain parameters among high quality solutions. The parameter value, to be considered as optimal, should have at least 25% share among high quality solutions, while it should have less than 10% share among low quality solutions. Of course, there are some minor deviations but in general we can define some average values of genetic operator's parameters when working with high-level IC design.

5 Conclusion

As presented there is a lot of work to fine-tune the proper values of the genetic operators. To achieve compatible results in optimization of the used circuits it is appropriate to use the values obtained by our investigation.

Generally, the quality of solution is always influenced by parameters and the problem itself. Therefore, it is important to perform this kind of evaluation each time we are in search of the optimal values of the genetic operators for some new problem to be solved.

References

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford University Press, 1996.

[2] J. Benesty, P. Duhamel, A Fast Exact Least Suare Adaptive Algorithm, IEEE Transactions on Signal Processing 40, 1992, pp. 2904-2920.

[3] C. A. Coello Coello, A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems 1, 1999, pp. 269-308.

[4] B. Filipič, J. Štrancar, Tuning EPR spectral parameters with a genetic algorithm. Applied soft computing 1, 2001, pp. 83-90.

[5] G. W. Grewal, T. C. Wilson, An Enhanced Genetic Algorithm for Solving the High-Level Synthesis Problems of Scheduling, Allocation, and Binding, Intl. Journal of Computational Intelligence and Applications, 1, 2001, pp. 91-110.

[6] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220, 1983, pp. 671–680.

[7] F. G. Lobo, The parameter-less genetic algorithm: Rational and automated parameter selection for simplified genetic algorithm operation, Ph.D. thesis, University of Lisbon, Portugal, 2000.

[8] G. Papa, Concurrent operation scheduling and unit allocation with an evolutionary technique in the process of integrated-circuit design, Ph.D. thesis, Faculty of Electrical Engineering, University of Ljubljana, Slovenia, 2002.

[9] G. Papa, B. Koroušić-Seljak, B. Benedičič, T. Kmecl, Universal Motor Efficiency Improvement using Evolutionary Optimization, IEEE Transactions on Industrial Electronics 50, 2003, pp. 602-611.

[10] G. Papa, J. Šilc, Automatic Large-Scale Integrated Circuit Synthesis Using Allocation-Based Scheduling Algorithm, Microprocessors and Microsystems 26, 2002, pp. 139-147.

[11] P. G. Paulin, J. P. Knight, E. F. Girczyc, HAL: A Multiparadigm Approach to Automatic Data Path Synthesis, Proc. 23rd ACM/IEEE Design Automation Conference, Las Vegas, USA, June 1986, pp. 263-270.

Extending CC4 Neural Networks to Classify Real Life Documents

Enhong Chen and Zhenya Zhang
Department of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui, 230027, P.R. China
Phone: +86 551 3602824,
Fax: +86 551 3603388
Cheneh@ustc.edu.cn, zzychm@mail.ustc.edu.cn

Hans-Dieter Burkhard and Gabriela Lindemann
Institut für Informatik, Humboldt Universität Berlin
D-10099 Berlin, Unter den Linden 6, Germany
hdb, lindeman@informatik.hu-berlin.de

Keywords: CC4 neural network, document classification, dimensionality reduction

Received: March 26, 2002

The CC4 neural network is a kind of fast pattern-learning techniques that can be used for document classification. In essence, the underlying classification mechanism of CC4 neural networks is equivalent to the use of the Hamming distance measure for classification in which the radius of generalization r *of CC4 neural network plays an important role in defining the sphere of influence for each training sample. If we rely only on the titles and summaries returned from standard search engines, it could be appropriate to represent the Web documents as binary vectors. However, when classifying real life documents, binary representation of documents may not be an effective one and may reduce the classification precision. The paper presents a method to classify documents with their term frequency (TF) vector. In this method a way to transform the real value of each element to a binary number that required by CC4 is put forward. Usually, the dimensionality of the TF vector representation is very large. Therefore, before transforming the real value of each element to a binary number a step called dimensionality reduction, i.e., construction of indexes of much lower dimensionality called the k-index of documents will be performed. Then each* k-*index of documents is transformed to a 0/1 sequence. This kind of sequences should keep as much the original distance information of documents when measured within the Hamming distance space. Experimental results show that the CC4 performs better when CC4 uses our proposed method to classify news documents than it does when only depending on binary representation of documents.*

1 Introduction

In recent years, we have seen an enormous growing number of text documents available on the Internet, digital libraries, and news sources. However, effective use of the documents requires fairly sophisticated tools for locating, classifying, and retrieving only those of interest to an individual. This problem is known as document classification or document categorization.

Over the years there are many algorithms developed for document classification [1, 2, 4, 5, 9]. Unfortunately, most rule based statistical methods are slow in pattern learning process The Corner-Classification (CC) as a classification technique has an instantaneous nature and has been used to solve some problems, such as data-compression, time-series prediction and intelligent Web meta-search engine design [6, 7, 10]. Unlike most other artificial networks, the CC neural networks do not need repeated training or weight adjustments; the training sample needs to be shown to the network just once. CC4 as the fourth version of the CC-method is an effective implementation technique, which learns instantaneously and can be used as a classification method to classify the results returned by standard search engines [6, 10]. These returned documents include the titles and summaries alone rather than the complete contents of the Web pages. The neural network is built using keywords from the Web pages. Each keyword maps into a 0 or 1, so the length of the input vector is equal to the number of the keywords chosen. The neural network can assign all the Web pages to either one of the predefined classes or assign a relevancy value to the page that is a fraction between 0 and 1.

In essence, the underlying classification mechanism of CC4 neural networks is to use the Hamming distance measure for classification in which the radius of generalization r of CC4 neural network plays an important role in defining the sphere of influence for each training sample. If we rely only on the titles and summaries returned from standard search engines, it

could be appropriate to represent the Web documents as binary vectors. However, when classifying real life documents, binary representation of documents may not be an effective one and may reduce the classification precision. As a more effective term weighting method for document representation, documents are represented as an L-dimensional vector whose elements are indicated by the occurrence frequency of the corresponding terms, where L is the number of the size of the dictionary used. By this representation method any document is represented to be a normalized vector tf =(tf[1], tf[2], ..., tf[L]) called TF vector, where $0 \le tf[i] \le 1$ is the normalized occurrence frequency of the term corresponding to i, $1 \le i \le L$, and L is the size of the dictionary used.

Now let's give two examples to analyze the reason why CC4 can not work well on real life documents if the documents are simply represented with binary vectors and then classified with CC4 neural networks, because the underlying Hamming Distance criterion could reduce the classification accuracy. In the first example, suppose that we have two documents represented as TF vectors tf_1 and tf_2, and there exists an i_0 such that $tf_1[i_0] = tf_2[i_0] = 0.90$, and for other j, $1 \le j \le L$, $j \ne i_0$, $tf_1[j] \times tf_2[j] = 0$ and $tf_1[j] + tf_2[j] \ne 0$. The Hamming Distance of tf_1 and tf_2 is L-1 and they will be classified into two different classes by the CC4 neural network. If we use similarity criterion of cosine similarity, however, the distance of the two vectors is $1 - 0.90^2 = 0.19$. The two documents may therefore be judged to be of the same class. In the second example, suppose that $tf_1[i_0] = 0.98$, $tf_1[i_1] = 0.02$, $tf_2[i_0] = 0.02$, $tf_1[i_1] = 0.98$, and $tf_1[j] = tf_1[j] = 0$, where $1 \le j \le L$, $j \ne i_0$, i_1. The two documents will be classified to be of the same class by the CC4 neural network. However, they should be in different classes when measured with cosine similarity criterion.

This paper proposes a method to use CC4 neural networks to classify real life documents. We choose the TF vector representation of documents. Because CC4 neural network require its input to be binary vectors, we should look for a way to transform the real value of each element to a binary number. Considering that the dimensionality of the TF vector is too large, the paper proposes an approach for constructing much lower dimensional indexes called k-index of the original vectors of documents. Then each k-index of documents is transformed to a 0/1 sequence. To avoid the problems mentioned in the above two examples, such sequences should keep as much as possible the original distance information of documents when measured within the Hamming distance space.

In the following section, we will first briefly introduce CC4 neural networks. Then the MDSNN based data indexing method will be presented in Section 3. Section 4 describes the method of the construction of binary sequences used as the input of the CC4 neural network. In Section 5 theoretical analysis of classification behavior of CC4 neural networks is given. Our experimental results and analysis are shown in Section 6. Section 7 comprises our concluding remarks and future research directions.

2 The CC4 Neural Network

The CC4 algorithm is a new type of corner classification training algorithm for three-layered feed forward neural networks [6, 11]. For clarity, we call the neural network trained with CC4 algorithm CC4 neural network. It has three layers that are Input Layer, Hidden Layer, and Output Layer. The neurons between the three layers are fully connected. The weight of the link between any two neurons connected is assigned to be +1 when the input to the neuron is a 1, and is assigned to be –1 otherwise. The number of the input layer neurons is one more than the size of an input vector. The additional neuron is the bias neuron. The weight from this neuron to any neurons in the hidden layer is always r-s+1, where s is the number of 1's in the input vector and r is the generalization radius for the CC4 neural network. The hidden layer consists of n neurons, where n is the number of input training samples or vectors. Each training vector is memorized by one hidden neuron. The number of output layer neurons depends on the number of different classes. In our work on document classification, the number of output neurons equals to the number of bits used to encode classes. Thus, the CC4 neural network maps an input binary vector X to an output binary vector Y.

As mentioned before, an important feature of the CC4 neural networks is the radius of generalization r. After a test signal has been processed by input layer and hidden layer neurons, output layer neurons receive their inputs and calculate the output vector. The output vector is the same with the vector that is outputted by the CC4 neural network for a training vector whose Hamming distance is not greater than the generalization radius r. This shows that when the Hamming distance of the test vector and a training sample is not greater than r, then the test sample will be judged to be of the same class with that of the training sample by the CC4 neural network. The computation of the CC4 neural networks can be described as following in three steps:

- Construct the input vector *input* of the CC4 network:
  the frontmost N elements are directly from a sample and the last element is 1;
- for (j = 0; j < H; j++) //H is the number of //hidden neurons
  { hidden[j] = 0; // *hidden* is a vector used to //memorize the hidden neuron states
  for(i = 0; i < N; i++) hidden[j] = hidden[j] + W[i*H][j] * input[i];
  // W is the matrix representing the
  // connection weights from neurons
  // of the input layer to the hidden
  // layer ones
  if (hidden[j] > 0) hidden[j] = 1 else hidden[j] = 0;

```
    }
• for (j = 0; j < M; j++)//  M is the number of
                              //output neurons
    { output[j] = 0; // output is used to save the output
                      // neuron states
      for (i = 0; i < H; i++) output[j] = output[j] +
                              U[i*M][ j]*temp[i];
                    // U is the connection weight  matrix
                    // from hidden layer to output layer
      if (output[j] > 0) output[j] = 1 else output[j] = 0;
    }
```

Now let's briefly analyze the way that Hamming distance is employed in computation of CC4 algorithm. We first give a function calculating the Hamming distance between two n-dimensional binary vectors

$X = (x_1, x_2, \ldots, x_n)$, $Y = (y_1, y_2 \ldots, y_N)$. Let $f_x = \sum_{i=1}^{n} x_i$, and $X' = (x_1', x_2' \ldots x_n')$ be the vector corresponding to X, where $x_i' = 1$ if $x_i = 1$, otherwise $x_i' = -1$. Then the function $g(X, Y) = f_x - X' \cdot Y$ can be used to calculate the Hamming distance between vectors X and Y, where $X' \cdot Y$ is the inner product of vectors X' and Y.

Suppose that $T_1, T_2 \ldots T_H$ are all sample binary vectors used for training the CC4 network. After the training of the CC4 neural network, the connection matrix between input and hidden layer is as follows:

$$\begin{pmatrix} W_1 & W_2 & \cdots & W_H \\ r - f_{T_1} + 1 & r - f_{T_2} + 1 & \cdots & r - f_{T_H} + 1 \end{pmatrix}.$$

where W_i is the transpose of vector T_i' . For any vector X to be classified, the input signal for the i*th* neuron in hidden layer is $W_i \cdot X + r - f_{T_i} + 1 = r + 1 - (f_{T_i} - T_i' \cdot X) = r + 1 - g(T_i, X)$, where $i = 1, 2, \ldots, H$, and r is generalization radius for the network. If the Hamming distance $g(T_i, X)$ between vector X and a training sample binary vector T_i for CC4 network is within generalization radius r, that is $g(T_i, X) \le r$, then $r + 1 - g(T_i, X) \ge 1 > 0$. According to the computation of the CC4 neural network the output of the i*th* neuron in hidden layer should be 1. So X will be judged to be of the same class with that of the training sample T_i by the CC4 neural network. Otherwise we have $g(T_i, X) \ge r + 1$, so $r + 1 - g(T_i, X) \le 0$ and X will be judged to be of a different class with that of the training sample T_i by the CC4 neural network.

3 Low Dimensional Index Generation for Documents

Mapping n-dimensional data objects into lower dimensional space while preserving distances between original data before performing indexing, mining and visualizing operations to the data is a typical useful method for textual and multi-media documents[3]. Multi-Dimensional Scaling (MDS) is a readily used method to index original data in low dimensional space when knowing the distances of pairs of data objects. However, MDS is inefficient when new data objects need to be indexed with respect to the old data and they have to be indexed again with new data. To overcome this problem, we have proposed a BP neural network where the incremental data indexing approach - named MDSNN - is based on. In this method a small data set used as sample data set is first indexed with MDS. In the case if the size of the sample data set is very small, the time spent on this step is very low. Then indexing results are provided as training samples and supervisor signals to train the neural network. The trained neural network is used to construct indexes of new documents. Such index is denoted as k-index, where k is the expected dimensionality of the resulting index space. The quality of indexing is measured by the so-called *Stress* [8] function:

$$Stress = \sqrt{\frac{\sum_{i,j} (d_{ij}' - d_{ij})^2}{\sum_{i,j} d_{ij}^2}}$$

where d_{ij}' is a distance between P_i and P_j, d_{ij} is a distance between the two objects O_i and O_j, and $P_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$ is a k-dimensional point called k-index of O_i and it denotes the image of the original object O_i.

Definition 3.1

K-index: Suppose that there exists a mapping that maps any n-dimensional original data d into a point p in a k-dimensional space, where $n > k$, then point p is called the k-index of d.

To index all objects in k-dimensional space means that all distances among objects should be kept as much as possible in the lower dimensional space. For this purpose, *Stress* is used to find a good index in the k-dimensional space for each original object. When finishing the indexing operation *Stress* reaches a minimum value. Suppose that we have all distance information for n objects O_i, where $i = 1, 2, \ldots, n$. The following steps show a typical implementation of the MDS method: 1) Randomly assign each object O_i to a k-dimensional point $P_i = (P_{i1}, P_{i2}, \ldots, P_{ik})$, i.e. the coordinate vector representation of a k-dimensional point; 2) For each point P_i, compute its distances from the all other points P_j, where $i \ne j$. Then update the coordinate vector of P_j to decrease the *Stress* function value with the method of non-linear least squares; 3) Iteratively perform step 2 until the *Stress* value becomes stable.

Based on MDS our proposed MDSNN method is as follows:

1) Build the k-dimensional indexes of training sample data using the MDS method;
2) Construct the sample data set and supervisor signal set for the BP neural network with the results obtained in step 1;
3) Train the neural network with the data from step 2;

4) Build the index of newly incoming data with the trained neural network.

4 Binary Sequence Construction from Indexes

Through MDSNN all documents are mapped to points in the *k*-dimensional space while their distance information is kept as much as possible. In the following, we will extend the CC4 neural network so that it can accept *k*-dimensional indexes of documents as its input.

Because the CC4 can only receive binary numbers to be its input, every *k*-index of documents must be transformed to a 0/1 sequence and the sequence should keep as much the original distance information of k-indexes of documents in the Hamming distance space.

In the following, we will define the notion of *L*-binary sequences of real values first, and then *L*-binary sequences for *k*-indexes.

Definition 4.1

***L*-binary sequence**: Let x be a real value such that $x \in [a, b]$, where [a, b] is the domain for all possible real values considered to be transformed. S is an *L*-binary sequence for x when all first k elements of S are ones and the rest $L - k$ elements are all zeroes, where L is the length of the sequence S.

Let $m = \frac{b-a}{L}$, then $k = [\frac{x-a}{m}]$.

For example, let $x = 0.72$, $x \in [0, 1]$, $L = 10$, then $m = 0.1$, $k = 7$. We get an *L*-binary sequence 1111111000 for $x = 0.72$ at interval [0, 1].

Definition 4.2

***L*-binary sequence for *k*-indexes**: Suppose that the *k*-index of a document is $(x_1, x_2, \ldots, x_k) \in [a, b]^k$, L is a given positive integer and S_i is the *L*-binary sequence for x_i, then $S = < S_{11}, S_{12}, \ldots, S_{1L}, S_{21}, S_{22}, \ldots, S_{2L}, \ldots, S_{k1}, S_{k2}, \ldots, S_{kL}>$ is the *L*-binary sequence for the *k*-index of data d, where $i = 1, 2, \ldots, k$, $j = 1, 2, \ldots, L$. S_{ij} is the j-th element of the *L*-binary sequence for the *k*-index of x_i. □

Because we use a normalized term frequency vector method to represent documents, every element of a vector is assumed to be in the interval [0, 1], which means that a = 0 and b = 1. To a given *k*-index of a document its *L*-binary sequence can be constructed as follows:

1) Assign a value to L to fix the length of the *L*-binary sequence for *k*-indexes;
2) Initialize S to be an empty sequence;
3) For each element e_i of the *k*-index of a document d do
   3.1) *Step*= [1/*L*];
   3.2) *Length* = [e_i/*Step*];
   3.3) Assign $<S_{i1}, S_{i2}, \ldots, S_{iL}>$ to be the *L*-binary sequence for the *k*-index of element e_i, where $S_{ij} = 1$ for $j = 1, 2, ..., Length$, and $S_{ij} = 0$ for $j = Length +1, ..., L$;
   3.4) Append $<S_{i1}, S_{i2}, \ldots, S_{iL}>$ to the tail of the sequence S.

When training a CC4 neural network to classify documents, each training document is firstly indexed with MDSNN and then the corresponding *L*-binary sequence for its *k*-index is constructed to be the input of the CC4 neural network. The topic or target class of the document is used to generate the supervisor signal of the CC4 neural network. For new text documents MDSNN is called to construct the *L*-binary sequences for their *k*-indexes and then these new documents are classified with the CC4 neural network.

5 Theoretical Analysis of Classification Behavior of CC4 Neural Networks

In the following, we will theoretically analyze the classification behavior of CC4 neural networks. For the ease of description a neuron is said to be in active state if its output is 1, otherwise to be in blocked state. As stated previously, in our work on document classification the number of output neurons is equal to the number of bits used to encode classes. Suppose that there are six classes, they are encoded from 0 to 5, respectively. In binary encoding form 3 bits are needed to encode them, i.e. from 000 to 100.

Definition 5.1

Binary Matrix: An $m \times n$ matrix $A = (a_{ij})$ is called a binary matrix if $\forall a_{ij} \in \{0, 1\}$.

Definition 5.2

B-composed and Fully B-composed Matrix: Let A be an $m \times n$ binary matrix, where $1 \le m \le 2^n$. Denote the i-th row of A to be a_i, then $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})$. If $(a_{i1}, a_{i2}, \ldots, a_{in})$ is regarded as a binary integer, where the leftmost digit is a_{i1} and the rightmost digit is a_{in} and $a_{i1} \times 2^{n-1} + a_{i2} \times 2^{n-2} + \ldots + a_{in} \times 2^0 = i$-1holds for all $1 \le i \le m$, then we call A B-composed. Specifically, A is called to be fully B-composed if $m = 2^n$.

Definition 5.3

Binary Extension: Let A be a B-composed $m \times n$ binary matrix, where $1 \le m \le 2^n$. Let $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})$ to be the i-th row of A. Suppose that B is a B-composed $m \times (n+1)$ binary matrix. Let $b_i = (b_{i1}, b_{i2}, \ldots, b_{in})$ to be the i-th row of B. B is called an Binary extension of A if:

$$(b_{i1}, b_{i2}, \ldots, b_{i(n+1)}) = \begin{cases} (0, a_{i_1}, a_{i_2}, \ldots a_{i_n}) & \text{for} \quad i \le 2^n \\ (1, a_{i_1}, a_{i_2}, \ldots a_{i_n}) & \text{for } 2^n < i \le 2^{n+1} \end{cases}$$

Lemma 5.1

Suppose that *A* is a B-composed *m*×*n* binary matrix, then the number of the ones is not greater than the number of the zeros in each column of A.

Proof:

When $n = 1$, A is (0) or $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, the conclusion obviously holds.

As an induction hypothesis, we assume that the conclusion holds for $n = k$. Now let n be k+1. Suppose that B is a sub-matrix obtained by deleting the last column of *A*. *B* is B-composed when $m \le 2^k$ and the number of the ones is not greater than the number of the zeros in each column of *B*. If $m > 2^k$, then let B_1 be the sub-matrix consisting of the uppermost 2^k rows of *B*, and B_2 be the sub-matrix consisting of the rest rows of *B*. Apparently, B_1 is fully B-composed and B_2 is B-composed. Thus the number of the ones is not greater than the number of the zeros in each column of B_1 and B_2. Hence the number of the ones is not greater than the number of the zeros in each column of *B*. Obviously, matrix *A* can be obtained through a Binary-extension operation on *B*. When $m \le 2^k$, all elements of the first column of A are zeros. When $m > 2^k$, the number of the ones in the first column is $m - 2^k$. Let c be the difference of the number of the zeros to that of the ones in the first column, then $c = 2^k - (m - 2^k) = 2^{k+1} - m$. For $m \le 2^{k+1}$ follows that $c \ge 0$. We conclude that the conclusion holds when $n = k$+1. This completes the proof for this theorem.

Theorem 5.2

Every supervisor signal for training CC4 neural networks can be supposed as a row vector. All these vectors can be arranged in form of a matrix *U* where every row corresponds to the output of the network training with a special training set. If we substitute all zero elements with -1, then the obtained matrix *V* is the matrix of the weights of connections from the hidden to output layer.

Proof:
It is obvious by observing the training process of CC4 neural networks.

Theorem 5.3

All inputs that cannot be recognized by the CC4 neural network will be classified into Class 0.

Proof:
From the calculation process of CC4 neural networks, any input vector whose Hamming Distances to all learning vectors are greater than the radius of generalization will bring all the hidden neurons in blocked statuses. Apparently, when these states are propagated to the output layer, all the output layer neurons will come also in blocked states. According to the encoding method Class 0 corresponds to the situation that all output layer neurons are in blocked states.

Theorem 5.4:

When the generalization radius of the CC4 neural network is great enough all input vectors will be classified into Class 0.

Proof:
When the generalization radius of the CC4 neural network is great enough any input vector will bring all the hidden neurons in active states. By Lemma 5.1 and Theorem 5.2 all the input signals of output layer neurons are not greater than 0. Thus all output layer neurons come in blocked states. According to the encoding method Class 0 corresponds to the situation that all output neurons are in blocked states.

In the following, the notion of δ-neighbourhood of a *k* dimensional point *X* is given first, and a theorem from which the classification nature of CC4 can be explained will thereafter be presented.

Definition 5.4

δ-neighborhood: Given two vectors $X = (x_1, x_2, \ldots, x_k) \in [0, 1]^k$ and $Y = (y_1, y_2, \ldots, y_k) \in [0, 1]^k$. X' and *Y'* are the *L*-binary sequences for X and Y, respectively. If the Hamming distance $| x_i' - y_i'| \le \delta/k$, where x_i' and y_i' are the *L*-binary sequences for x_i and y_i, respectively, and δ is a positive integer, then *Y'* is said to belong to the δ-neighborhood of *X'* and this is denoted as $Y' \in N_\delta(X')$. □

Theorem 5.5

Suppose that the *k*-index of $X = (x_1, x_2, \ldots, x_k) \in [0, 1]^k$ is the center of the training document set for class C. Given δ, let $r = \delta/k$. If for any document $Y = (y_1, y_2, \ldots, y_k) \in [0, 1]^k$ the Hamming distance of the *L*-binary sequences for x_i and y_i is *n*, i.e. $| x_i' - y_i'| = n$, then $n \le r$ iff $Y' \in N_\delta(X')$.

Proof:
First, we know that $r = \delta/k$, hence $rk = \delta$. For $n \le r$ and $k>0$ follows that $nk \le rk = \delta$. Hence the Hamming distance of the *L*-binary sequences of the *k*-indexes of *X* and *Y* is no more than δ, i.e., $| X' - Y' | = \sum|x_i' - y_i'| \le nk \le \delta$. Thus $Y' \in N_\delta(X')$.

Conversely, given that $Y' \in N_\delta(X')$, we have $| X' - Y' | \le \delta$. Thus $|x_i' - y_i'|k = nk \le \delta$, hence $nk \le \delta$, and $n \le \delta/k$ Thus we can get n ≤ r, and this completes the proof of the theorem.

Obviously, the distribution of points in *k*-dimensional space corresponding to the k-indexes of documents is completely determined once all documents are mapped into the *k*-dimensional space. Because documents come from several different classes, these points are thus expected to be partitioned into several different subspaces and each subspace corresponds to a different class. What CC4 does is to decide which subspace every document should belong to based on its *L*-binary sequence for its *k*-index. For the ease of clarity, the class that the first center used to train CC4 belongs to is called the first class, and so on. By Theorem 5.5 we know that more and more points will be covered by the δ-neighborhood of every training data center with the increase of the radius of generalization when training CC4 and thus improve the classification precision of the trained CC4. The precision will reach its highest value at a certain generalization radius. Afterwards, with the increase of the radius of generalization, more and more points will be covered by the δ-neighborhoods of other class centers. Hence the classification precision will decrease till the generalization radius reaches a threshold value r_0. Then the δ-neighborhood of the first class center will cover all points and the classification precision thus stays at a fixed level, i.e. around the percentage of test samples of the first class as stated in Theorem 5.4. This classification behavior of CC4 will be demonstrated in the following experiments.

6 Experimental Results and Analysis

In the following, we will perform some experiments to compare the classification performance of CC4 that uses binary vectors and *L*-binary sequences for *k*-indexes of documents. For the sake of clarity, the CC4 neural networks using *L*-binary sequences for *k*-indexes is called LKCC4, where LK stands for an *L*-binary sequence for a *k*-index. The CC4 neural network directly using binary representation of documents as its input is called InitialCC4. The binary input of InitialCC4 constructed for each document is an *L*-dimensional binary vector t =(t[1], t[2], ..., t[L]), where L is the number of the size of the dictionary used, and element $t[i]$ = 1 if the i-th keyword in the dictionary occurs at least one time in the document, $t[i]$ = 1 otherwise.

Our experiments are performed on documents downloaded from UCI KDD Archive site *http://kdd.ics.uci.edu*. We randomly select 10 groups of downloaded news data and pick up randomly 50 news in each group as our experimental data. We set k to be 3. Thus all documents are mapped into points in a 3-dimensional space. To test the performance of our MDSNN indexing model we select different numbers of documents to train the neural network. Fig. 1 shows *Stress* values obtained with the different number of documents as training samples. The detailed experimental setup is given below:

1) Build a dictionary based on the terms obtained from each news group and use it to construct the TF vectors of all documents.
2) In each news group determine the number of training documents by assigning a value to *ratio* such that the size of the training set *SampleNum* is identical to the size of whole entire document set * *ratio.*
3) Calculate the center vector of the training document set in each group.
4) Take the center vectors as the training data set of MDSNN, and the news in each group as the test data set. So we can generate the *k*-indexes of the test data using MDSNN.
5) Construct the L-binary sequences for the *k*-indexes of the test data as inputs to LKCC4 for classification.

Given different ratios of training documents Figs 2-5 show the influences of the radius of generalization on the classification precision of LKCC4 and InitialCC4. It can be observed, that when given the same ratio value the highest classification precision of LKCC4 will be much greater than that of InitialCC4. From Figs 2 – 5 we can also observe, that when the radius of generalization is greater than a certain threshold value, the classification precision of LKCC4 and InitialCC4 will both stay at a fixed value which is around the percentage of test samples belonging to the first class. To confirm our observation, we also perform further experiments by picking up 10, 20, 30, 40 and 50 documents, respectively, from the first news group and 50 documents from all other nine groups. The documents used to generate the center used as the training sample for each group are 10 percent of all documents that each group contains. Therefore, the respectively 9, 18, 27, 36 and 45 documents in the first news group could be used as test documents. Table 1 shows the results obtained when the classification precision of LKCC4 converges.

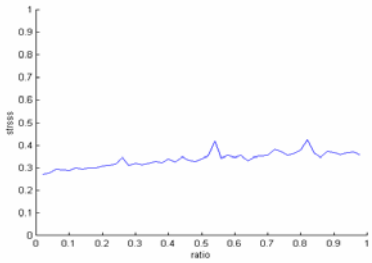


Fig. 1 The Stress value for different ratio of documents used as training samples of MDS-NN

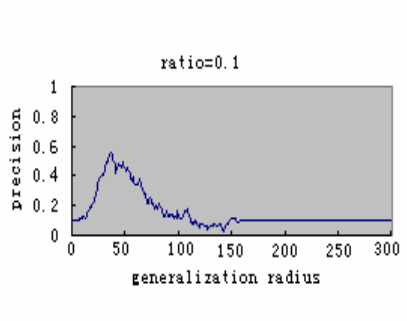


Fig. 2 The influence of radius of generalization on classification precision of LKCC4 (*ratio* = 0.1)

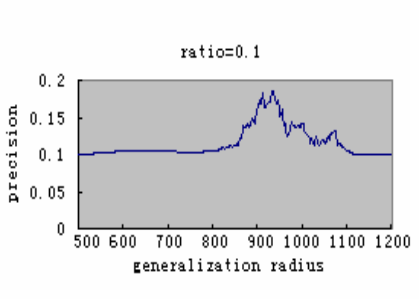


Fig. 3 The influence of radius of generalization on classification precision of InitialCC4 (*ratio* = 0.1)

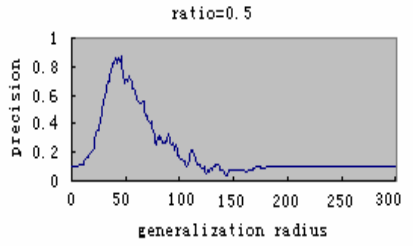


Fig. 4 The influence of radius of generalization on classification precision of LKCC4 (*ratio* = 0.5)

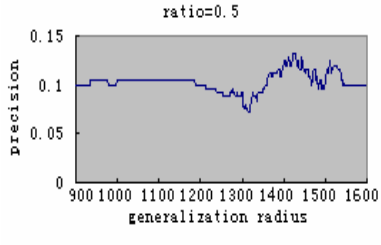


Fig. 5 The influence of radius of generalization on classification precision of InitialCC4 (*ratio* = 0.5)

Table 1. The expected and practical classification precision of LKCC4 network when the radius of generalization reaches the threshold value

No. of documents from the first news group	No. of documents of all other news group	Threshold of generalization radius	Classification Precision of LKCC4
9	414	157	0.022
18	423	159	0.043
27	432	159	0.063
36	441	159	0.083
45	450	159	0.1

7 Conclusion

In this paper we present a method to use CC4 neural networks to classify documents with their term frequency vectors. Firstly, we construct much lower dimensional indexes of the original document vectors to reach the goal of dimension reduction. Then each index of documents is transformed to a binary valued sequence. This kind of sequences should keep as much the original distance information of documents when measured with the Hamming distance. Experimental results show that the CC4 performs better when CC4 uses our proposed method to classify news documents than it does when only depending on binary representation of documents.

In the future the integration of CC4 and incremental indexing methods is one of our research topic. Because in our approach continuously valued data indexes can be discretized as the CC4 neural network input, extending the method to other textual and non-textual data classification applications is a further future effort of our group.

Acknowledgements

The work in this paper was funded by National Nature Science Foundation of China research grant 60005004 , Nature Science Foundation of Anhui Province research grant 01042302 and DAAD.

8 References

[1] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Kary pis, V. Kumar, B. Mobasher and J. Moore, Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3),pp329-341, 1999.

[2] M. Craven, S. Slattery, Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1/2), pp 97-119.

[3] C. Faloutsos, FastMap: A fast algorithm for indexing, data-Mining and visualization of traditional and multimedia datasets. in *Proc. of ACM SIGMOD Conf* , 1995,pp163-174

[4] E.H. Han and G. Karypis. Centroid-based document classification algorithms: Analysis & experimental results. *Technical Report* TR-00-017, Department of Computer Science, University of Minnesota, Minneapolis, 2000.

[5] T. Joachims, Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137-142. Springer Verlag, Heidelberg, 1998.

[6] S. Kak. Better web searches and faster prediction using instantaneously trained neural networks. *IEEE Intelligent Systems*. 14(6):78-81, 1999.

[7] S. Kak, V. R. Admal. Text compression using instantaneously trained neural networks, 2001. http://www.ece.lsu.edu/kak/recent.html

[8] J. B. Kruskal and M. Wish, Multidimensional scaling. SAGE publications, Beverly Hills, CA. 1978

[9] David D. Lewis, Robert E. Schapire, James P. Callan and Ron Papka, Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96*, pages 298-306. ACM Press, New York, 1996.

[10] B. Shu and S. Kak. A neural network-based intelligent metasearch engine. *Information Sciences*, vol. 120, pages 1-11, 1999

[11] K.-W. Tang and S. Kak, A new corner classification approach to neural network training, Circuits, Systems, and Signal Processing, vol.17, pages 459-469, 1998.

A Pattern Mapping Based Digital Image Watermarking

Chwei-Shyong Tsai
Department of Information Management National Taichung Institute of Technology,
Taichung, Taiwan 404, R.O.C.
E-mail: tsaics@ntit.edu.tw

Chin-Chen Chang
Department of Computer Science and Information Engineering National Chung Cheng University,
Chiayi, Taiwan, 621, R.O.C.
ccc @cs.ccu.edu.tw

Keywords: Discrete wavelet transformation, digital watermarking technique, intellectual property right

Received: February 6, 2002

This paper proposes a digital watermarking technique based on discrete wavelet transformation (DWT) to effectively protect the intellectual property rights of digital images. Basically, the proposed technique uses the DWT frequency coefficients of a digital image to preserve important image features and to embed a meaningful digital watermark into the image. In the proposed watermark embedding process, the protected digital image is first transformed into the DWT frequency domain and then each digital watermark bit is effectively embedded into four DWT coefficients using the proposed pattern mapping method. From the experimental results, we see that a digital watermark of acceptable quality can be extracted from an attacked watermarked image that has undergone the processes of image cropping, JPEG lossy compression, destructive signal, and even mixed attacks; that is, the proposed digital watermarking technique has an excellent robust property.

1 Introduction

With growing popularity of Internet, and the improvements in digital multimedia technology, users can easily copy vast amounts of multimedia data causing the copyrights of this multimedia data to be significantly threatened. Protecting the intellectual property rights of multimedia data as well as avoiding the invasion of the digital media by unauthorized persons has recently become an important research issue. As for the protection of the intellectual property rights of this multimedia data, these days many researchers [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] are putting a great amount of time into research related to watermarking techniques.

Digital image watermarking is a type of protection technique performed by embedding a special digital watermark into a protected image to protect the copyrights of the property of the image creator. The quality of the image is affected to the minimum degree. The protected target image is called a host image, the host image embedded with the watermark is called a watermarked image, and the image used as the digital watermark is a watermark image.

Most image watermarking algorithms transform a watermark image into data represented in a binary string pattern, which is called a watermark binary string. Some of these directly change the color intensity of pixels in the host image represented in the spatial domain in order to embed the watermark binary string. The advantages of this kind of algorithm are simplicity and quick processing. Generally, the watermark image embedded in the watermarked image will be destroyed by lossy image processing operations such as JPEG lossy image compression. These algorithms also have weak robustness.

Another kind of watermarking algorithms with stronger robustness than the previously mentioned ones convert the host image into the frequency domain. Then the watermark binary string is embedded into the frequency coefficients in the high or middle frequency bands of the frequency domain host image. In the transformed image, the frequency coefficients in the low frequency band describe the subjection of color allocation within the whole image. Thus when the coefficient in the low frequency band is changed, the pixel colors in the matching spatial domain image will be obviously influenced. However, the damage to the watermark image data resulting from the image processing operations will be eliminated if the watermark data is embedded into frequency coefficients in the low frequency. The pattern mapping-based image watermarking method proposed in this paper embeds the watermark image into a combination of frequency coefficients in the low frequency bands and the high frequency bands of the transformed host image.

According to the proposed pattern mapping based image watermarking method, the spatial domain host image is transformed into the discrete wavelet transformation (DWT) frequency domain. Then four frequency coeffi-

cients, three selected from the low frequency band and one selected from the other high frequency bands of DWT frequency domain image, are chosen to contain one bit of the watermark binary string. The four frequency coefficients are transformed to the matching pattern based on the embedded watermark bit. In other words, the proposed digital image watermarking method randomly chooses four frequency coefficients from the frequency band to contain a bit of data and to reduce the effects on the watermark image caused by lossy image processing operations performed on the watermarked image.

2 Pattern Mapping Watermarking Method

Discrete wavelet transformation is a technique that translates an image in the spatial domain into the frequency domain. In the process of DWT, the host image is first divided into four sub-bands LL_1, HL_1, LH_1, and HH_1 (as shown in Figure 1). Then, the low frequency sub-band LL_1 is further decomposed by repeatedly performing the same process until the application purpose is reached. In the proposed scheme, the repeated decomposition process continues on the lower frequency sub-bands containing LL_3, HL_3, LH_3, HH_3, HL_2, LH_2, and HH_2.

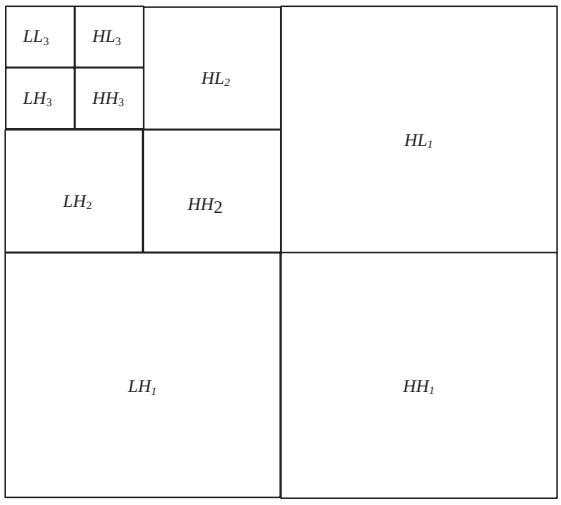


Figure 1: The sub-bands of a DWT frequency domain.

After the image is transformed from the spatial domain to the frequency domain, the original energy is focused on the upper left corner. This range represents the lower frequency of the transformed image, and the corresponding coefficients in this range are more significant. Moving to the lower right corner the range represents the high frequency of the transformed image, and the corresponding coefficients are less significant. Moreover, in the image of the DWT frequency domain, the frequency coefficients in the lower frequency sub-band are not easily affected by the general image processing operation. Therefore, the proposed watermarking method suggests embedding one watermark bit with the cooperation of three coefficients in the low DWT frequency sub-bands LL_3, HL_3, LH_3, HH_3, HL_2, LH_2, and HH_2, and one coefficient in HL_1, LH_1, HH_1.

2.1 Watermark Embedding

Figure 2 shows the block diagram of the proposed watermark embedding process. The first step of data embedding of the pattern mapping based watermarking method is to perform discrete wavelet transformation to transform the host image I in spatial domain to DWT frequency domain H. Next, derive a bit b_i continuously from watermark image S and embed the message of each b_i into four DWT coefficients. Finally, apply inverse DWT (IDWT) to the modified H to form the watermarked image I'.

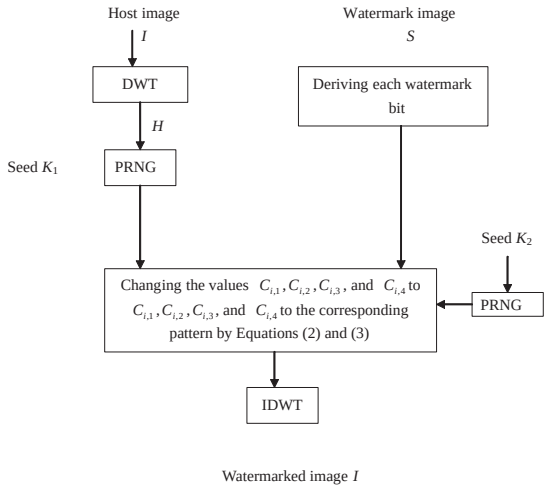


Figure 2: Block diagram of the proposed watermark embedding process.

During the process of watermark embedding, K_1 and K_2 are the two predefined private keys to be used as the seeds of the pseudorandom number generator (PRNG). When embedding the bit b_i, use $PRNG(K_1)$ to pick up three unselected DWT frequency coefficients $C_{i,1}$, $C_{i,2}$, $C_{i,3}$ from the sub-bands LL_3, HL_3, LH_3, HH_3, HL_2, LH_2, and HH_2 of DWT frequency domain of H, and to pick up one unselected DWT frequency coefficient $C_{i,4}$ from the sub-bands HL_1, LH_1, HH_1. If the value of b_i is 0, then change the values $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ to correspond to the 0-pattern. Otherwise, make them to correspond to the 1-pattern.

During pattern mapping, we define a DWT frequency coefficient α as being located within the 0-range or the 1-range area by Equation (1). If $\beta = 0$, we define α as being located within the 0-range area; otherwise, it is within the 1-range area.

Here β is defined as below:

$$\beta = \lfloor \alpha / R \rfloor \bmod 2. \quad (1)$$

In Equation (1), R is a given threshold and is set to be R_L or R_H depending on the sub-band of α. If α belongs to the sub-band LL_3, HL_3, LH_3, HH_3, HL_2, LH_2, or HH_2, R is set to be R_L. On the other hand, R is set to be R_H when α belongs to the sub-band HL_1, LH_1 or HH_1.

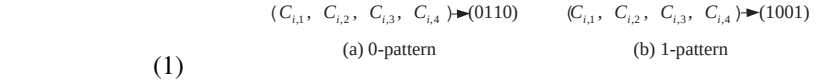
During pattern mapping, Equation (2) is used to change $C_{i,j}$ to $C'_{i,2}$ and it makes the areas of $C_{i,j}$ and $C'_{i,j}$ different. That is, if $C_{i,j}$ is located within the 0-range area, Equation (2) makes the corresponding $C'_{i,j}$ to be located within the 1-range area and vice versa.

$$C'_{i,j} = (\lfloor C_{i,j}/R \rfloor - 1) \times R + \lfloor R/2 \rfloor + (-1)^{r+1} Q. \quad (2)$$

In Equation (2), Q is an integer between 0 and $R/4$, and its value depends on $PRNG(K_2)$. Moreover, Equation (3) is used to change $C_{i,j}$ to $C'_{i,j}$ and it makes the areas of $C_{i,j}$ and $C'_{i,j}$ the same.

$$C'_{i,j} = (\lfloor C_{i,j}/R \rfloor) \times R + \lfloor R/2 \rfloor + (-1)^{Q+1} Q. \quad (3)$$

The pattern mapping based watermarking method modifies the values of $C_{i,j}$'s, $j = 1, 2, 3, 4$, according to the value of bit b_i to cause the corresponding $C'_{i,j}$'s to be located within the 0-range or the 1-range area when embedding data. We define $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ to correspond to the 0-pattern if the corresponding $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ are located within the 0-range, 1-range, 1-range, and 0-range, respectively. Conversely, $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ correspond to the 1-pattern if the corresponding $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ are located within the 1-range, 0-range, 0-range, and 1-range, respectively. When the embedded watermark bit b_i is 0, the pattern matching based watermarking method changes the values of $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ so as to make $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ to locate within 0-range, 1-range, 1-range, and 0-range, respectively; otherwise, the values of $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ are altered to make $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ to be located within the 1-range, 0-range, 0-range, and 1-range, respectively; that is, make the values of $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ correspond to the 1-pattern. Figure 3 demonstrates the condition of location for the corresponding $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ when $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ correspond to 0-pattern or 1-pattern. In Figure 3, 0 and 1 represent the 0-range and the 1-range, respectively. During the embedding process, if the areas of $C_{i,j}$ and $C'_{i,j}$ are different, Equation (2) is applied to change $C_{i,j}$ to $C'_{i,j}$; otherwise, Equation (3) is utilized.



(a) 0-pattern (b) 1-pattern

Figure 3: The related ranges of $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ patterns of $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$

.

2.2 Watermark Extraction

Figure 4 shows the proposed watermark extraction process. The steps of the watermark extraction in the pattern mapping based watermarking method are mainly based on the converse processing steps of watermark embedding. First, from the DWT frequency domain of the watermarked image I', the extraction process utilizes $PRNG(K_1)$ to pick up four unselected coefficients $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$, such that $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$ come from the sub-bands LL_3, HL_3, LH_3, HH_3, HL_2, LH_2, and HH_2, and $C'_{i,4}$ comes from HL_1, LH_1, HH_1, each time by continuously using $PRNG(K_2)$. When $((C'_{i,j} - (-1)^{Q+1} Q)/R) \bmod 2 = 0$, for $j = 1, 2, 3, 4$, $C'_{i,j}$ is defined to be located within the 0-range; otherwise, it is located within the 1-range. Similar to the watermark embedding process, R is a given threshold and is set to be R_L or R_H depending on the sub-band of $C_{i,j}$, and the value of Q is determined by $PRNG(K_1)$. In the watermark embedding process, the location of each $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ is located either within the 0-range, 1-range, 1-range, and 0-range, respectively, or within the 1-range, 0-range, 0-range, and 1-range, respectively. In other words, $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ correspond to either the 0-pattern or the 1-pattern. But the content of the watermarked image is possibly changed on purpose or unintentionally, for example, by using lossy image compression, so that the range to which corresponds is changed as well. Thus, $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ possibly no longer correspond to the two patterns of the 0-pattern or the 1-pattern. The column of possible patterns in Figure 5 lists all 16 possible changed patterns.

In the lower frequency band, every DWT frequency coefficient corresponds to a group of adjacent pixels in image I. That is, the DWT frequency coefficient is the final average result derived from calculating the color value of the adjacent pixels many times. Therefore, it is only when the color value from the pixels of image I changes greatly that the value of $C'_{i,j}$ changes so as to locate it within another scope of range. The possibility of changing the range into which $C'_{i,j}$ is located to another is not high. The watermark extraction process of the pattern mapping based watermarking method performs the following cases:

1. When $C'_{i,1}$ and $C'_{i,4}$ are both located within the 0-range, or $C'_{i,2}$ and $C'_{i,3}$ within the 1-range, regard $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ as corresponding to the 0-pattern, and the extracted bit b_i is 0.

2. When $C'_{i,1}$ and $C'_{i,4}$ are both located within the 1-range, or $C'_{i,2}$ and $C'_{i,3}$ within the 0-range, regard $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ as corresponding to the 1-pattern, and the extracted bit b_i is 1.

3. If the range of each $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ is 1, 1, 0, 0, respectively, or 0, 0, 1, 1, respectively, regard $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ as corresponding to the 0-pattern, and the extracted bit b_i is 0. If the range of each $C'_{i,1}$,

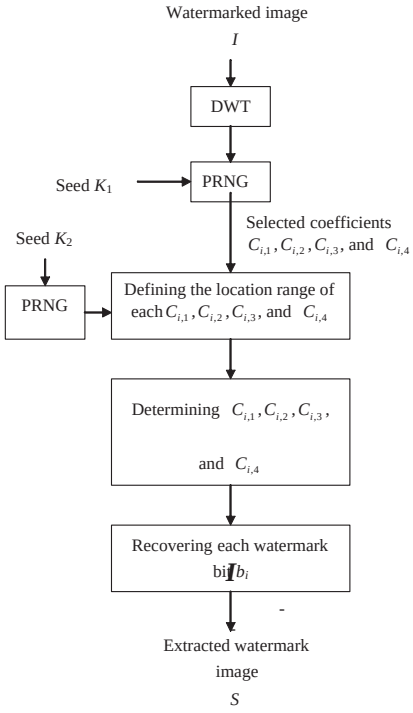


Figure 4: Block diagram of the proposed watermark extraction process.

$C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ is 1, 0, 1, 0, respectively, or 0, 1, 0, 1, respectively, regard $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ as corresponding to the 1-pattern, and the extracted bit b_i is 1.

The above cases of watermark extraction can ensure that the value of b_i can be guessed even if the range in which the location of any one of four $C'_{i,j}$'s is altered.

3 Security Analysis

The proposed pattern mapping watermarking method uses K_1 as the seed of the pseudorandom number generator to determine four DWT frequency coefficients $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, and $C_{i,4}$ which are used to embed the bit b_i. Thus only the person who has K_1 can obtain the watermark image embedded in I'. Furthermore, the value Q is randomly added to or subtracted from every value of $C'_{i,j}$'s. The value Q is assigned by the pseudorandom number generator with the seed K_2. It can avoid the condition that all the values of $C'_{i,j}$'s are added $R/2$. Therefore, only the one who has both private keys K_1 and K_2 can extract the embedded watermark image in the watermarked image.

After the operation of some type of image processing (like lossy image compressing, blurring, sharpening, etc.) is performed on the watermarked image I', some DWT frequency coefficient values of the frequency domain image may be changed so that the scope of the range in which the $C'_{i,j}$ is initially located is also altered. In Figure 5, the column **Possible patterns** shows all possible conditions of $C'_{i,1}$, $C'_{i,2}$, $C'_{i,3}$, and $C'_{i,4}$ after modification. The column **Original patterns** assumes the original correct corresponding pattern. The column **Guessing patterns** shows the incorrect guessing results based on the column **Possible patterns** and taking the watermark extraction step. The column **Related coefficients** lists $C'_{i,j}$'s which lead to incorrect guesses, that is, the guess is incorrect because the scope of the range in which the location of these $C'_{i,j}$'s is changed. The column **Probability** is the possibility of incorrect guessing for such pattern. Suppose p is the average possibility of the change to the range in which every $C'_{i,j}$ is located. So the average possibility that a pattern is guessed incorrectly is $6p^2 - 12p^3$. Consequently, the possibility of incorrect guessing when taking the watermark extraction step to calculate the value of bit b_i is $6p^2 - 12p^3$. For example, when p is 0.1, the possibility of incorrect guessing is 0.056. In the previous section, we concluded that the possibility of change to the range in which $C'_{i,j}$ is located is small, so the possibility of incorrect guessing when calculating the value of b_i is small as well.

Possible *patterns*	Original *pattern*	Guessing *pattern*	Related coefficients	Probability
0000	1-pattern	0-pattern	(C_1, C_4)	$(1-p)^2\times p^2$
1000	0-pattern	1-pattern	(C_1, C_2, C_3)	$(1-p)\times p^3$
0100	1-pattern	0-pattern	(C_1, C_2, C_4)	$(1-p)\times p^3$
0010	1-pattern	0-pattern	(C_1, C_3, C_4)	$(1-p)\times p^3$
0001	0-pattern	1-pattern	(C_2, C_3, C_4)	$(1-p)\times p^3$
1100	1-pattern	0-pattern	(C_2, C_4)	$(1-p)^2\times p^2$
1010	0-pattern	1-pattern	(C_1, C_2)	$(1-p)^2\times p^2$
1001	0-pattern	1-pattern	(C_1, C_2, C_3, C_4)	p^4
0110	1-pattern	0-pattern	(C_1, C_2, C_3, C_4)	p^4
0101	0-pattern	1-pattern	(C_3, C_4)	$(1-p)^2\times p^2$
0011	1-pattern	0-pattern	(C_1, C_3)	$(1-p)^2\times p^2$
1110	1-pattern	0-pattern	(C_2, C_3, C_4)	$(1-p)\times p^3$
1101	0-pattern	1-pattern	(C_1, C_3, C_4)	$(1-p)\times p^3$
1011	0-pattern	1-pattern	(C_1, C_2, C_4)	$(1-p)\times p^3$
0111	1-pattern	0-pattern	(C_1, C_2, C_3)	$(1-p)\times p^3$
1111	0-pattern	1-pattern	(C_1, C_4)	$(1-p)^2\times p^2$

Figure 5: All possible error guesses.

4 Experimental Results

With simple calculation, the proposed watermarking technique can effectively complete the tasks of watermark embedding and extraction. The effectiveness of the proposed watermark embedding and extraction processes can be examined from the following experimental results. As for the platform of the experiment, we used Pentium III 500 CPU, 64MB RAM, Windows 2000 Professional operating system, and Java programming language.

In these experiments, the extracted watermark from the

watermarked images that have undergone various kinds of attacks such as cropping, JPEG lossy compression, blurring and sharpening operations is visually acceptable. To judge the extracted accuracy, a quantitative measurement is required. Here, the watermarked image quality and the extracted watermark are evaluated based on two kinds of ratios that have been popular in related studies. One is Peak Signal to Noise Ratio (PSNR), and the other is Normalized Correlation (NC) ratio, a similarity measurement between the referenced watermark image S and the extracted watermark image S'.

The following equation shows the definition of PSNR:

$$PSNR = 10\log_{10}(\frac{255^2}{MSE})dB. \quad (4)$$

In Equation (4), *MSE* represents the mean-square error. For a gray-level image with $N_1 \times N_2$ pixels, its *MSE* is defined as

$$MSE = \frac{1}{N_1 \times N_2}\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1}(I(i,j) - I'(i,j))^2, \quad (5)$$

where $I(i,j)$ and $I'(i,j)$ denote the (i,j)-th pixel value in the original image and that in watermarked image, respectively.

Equation (6) defines the other ratio NC

$$NC = \frac{\sum_{i=0}^{M_1-1}\sum_{j=0}^{M_2-1} S(i,j)S'(i,j)}{\sum_{i=0}^{M_1-1}\sum_{j=0}^{M_2-1}(S(i,j))^2} \quad (6)$$

In this equation, for an $M_1 \times M_2$ watermark image, $S(i,j)$ and $S'(i,j)$ denote the (i,j)-th pixel values in the watermark image and the extracted watermark, respectively. NC is the cross-correlation normalized by the referenced watermark energy that gives unity as the peak correlation.

In our experiments, the original image, 512×512 gray-level "Lena", corresponds to another 32×32 binary watermark, as shown in Figure 6(a) and Figure 6(b). On the other hand, we set the thresholds of the low frequency sub-band and high frequency sub-band to be 4 and 7 in the experiment. That is, $R_L = 4$ and $R_H = 7$. After the proposed watermark embedding process, the PSNR value of the watermarked image becomes 39.65 dB (as shown in Fig. 6(c)), which indicates better quality of a watermarked image for the human eye. In other words, this proposed scheme satisfies the watermarking requirements. Besides processing good image quality, the extracted watermark is the same as the original one (as shown in Fig 6(d)). Its NC value is 1, and the bit correct rate (BCR) is 100%.

In the following subsections, we will discuss some experiments with analyses of the robustness of a watermarked image that subsequence undergoes various attacks such as cropping, JPEG lossy compression, and destructive signal processing. We organize different experimental results, and NC and BCR values from the extracted digital watermark of the image to create Table 1. The values of PSNR in Table 1 come from comparing the destroyed digital watermarked image to the original host image. The bit correct rate is obtained by comparing the extracted watermark to the original one.

(a) 512x512 original image "Lena"

(b) 32x32 original watermark

(c) The watermarked image, PSNR =39.65dB

(d) the extracted watermark image, NC=1.0, BCR=100%

Figure 6: The original image and the watermark image.

Table 1: PSNR, NC and BCR values under various image processing attacks.

Test results / Attacks	PSNR	NC	BCR
1/4 cropping	11.39	0.9998	0.9873
Half cropping	8.5679	0.9976	0.9496
3/4 cropping(top-left quarter leaving)	6.087	0.9928	0.8843
3/4 cropping(image boundary leaving)	2.9554	0.9976	0.8887
JPEG compression with rate 8.9	36.6906	0.9976	0.9844
JPEG compression with rate 9.8	36.3994	0.9952	0.9756
JPEG compression with rate 10.7	36.1892	0.9904	0.9824
JPEG compression with rate 13.3	35.4638	0.9796	0.9434
JPEG compression with rate18.3	34.4659	0.9026	0.832
Blurring	38.6222	0.9904	0.9764
Sharpening	32.8791	0.9183	0.8447
first-sharpening last-blurring	38.2217	0.9997	0.9987
first-blurring last-sharpening	38.0953	0.9978	0.9981
first JPEG compression last-blurring	36.7352	0.9255	0.8652
first JPEG compression-sharpening last-blurring	36.5627	0.9997	0.9932

4.1 Effects of a Cropping Attack

In the image cropping experiment, we apply three kinds of conditions:(1) Cropping 1/4 of the watermarked image, as shown in Figure 7(a), (2) Cropping half of the watermarked image, as shown in Figure 7(b), (3) Cropping $3/4$ of the watermarked image, as shown in Figures 7(c)-(d).

From Figures 7(a)-(d), we discover that the accuracy of restoring the digital watermark is in inverse proportion to

the size of the cropped image embedded with the watermark. The larger the removed part of an image is, the less accurate the restored digital watermark will be. Experimental results indicate that the digital watermark can still be restored to a certain degree even if $3/4$ of the image is removed.

Figure 7: Watermarks extracted from the various cropped, watermarked image "Lena".

4.2 Experiment and Analysis of the Impact of Lossy Image Compression on the Watermark

This subsection discusses the experiment of extracting the digital watermark after the watermarked image "Lena" undergoes JPEG lossy compression process. We adopted the image optimizing function of ULEAD Photo impact to process the watermarked image with different compression rates and quality rates, and then extracted the digital watermark, as shown in Figure 8. Comparing the experimental data in Table 1 to that of previous research in Table 2 [9], we discover that the proposed method has a better ability of resisting ordinary lossy compression.

Table 2: Changes of NC values under various JPEG lossy compression rates of Hsu and Wu's method [9].

Compression ratio	3.49	4.41	5.18	5.92	6.55	7.16	7.81	8.46	9.05	9.81	10.74
PSNR(dB)	33.78	35.52	34.74	33.84	33.15	32.56	32.07	31.75	31.47	31.41	31.27
NC	1.000	1.000	1.000	1.00	1.000	0.988	0.88	0.796	0.663	0.488	0.413

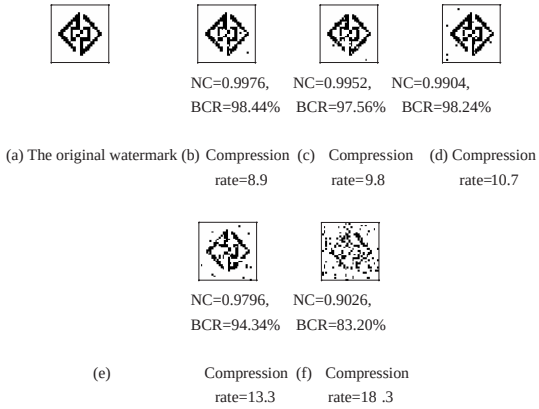


Figure 8: Watermarks extracted from the watermarked image "Lena" that has been processed by JPEG lossy compression with various compression rates.

4.3 Experiment and Analysis of the Impact of the Destructive Signal Process on the Watermark

This subsection discusses the experiment of extracting the digital watermark after the watermarked image undergoes the destructive signal process. In this experiment, we adopted the filter function of Photoimpact respectively conduct blurring and sharpening processes, and then extracted the embedded digital watermark.

Figure 9 is the experimental results from extracting the watermark after the blurring process. The proposed technique can still extract the embedded digital watermark under an acceptable blurring process. Figure 10 is the experimental results for the extracted watermark after sharpening process. According to the experimental results, the proposed technique can still effectively extract the embedded digital watermark after sharpening attacks.

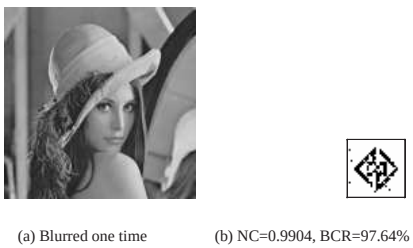


Figure 9: Blurred image "Lena" and the recovered watermark.

The following experiment explores the effect of restoring the watermark from the digital watermarked image which has undergone mixed image processing. In this experiment, the mixed image processing includes the first-sharpening last-blurring operation, first-blurring last-sharpening oper-

(a) Sharpened one time

(b) NC=0.9183, BCR=84.47%

Figure 10: Sharpened image "Lena" and the recovered watermark.

ation, first JPEG compression last-blurring operation, and first JPEG compression-sharpening last-blurring operation. The experimental results are shown in Figures 11-15. Table 1 shows that the NC's are greater than 0.91 and the correct rate is greater than 0.84. The extracted watermarks are still visually recognizable and are highly similar to the original watermark.

(a) First-blurred last-sharpened

(b) NC=0.9997, BCR=99.87%

Figure 11: First-sharpened last-blurred image "Lena" and the recovered watermark.

(a) First-sharpened last-blurred

(b) NC=0.9978, BCR=99.81%

Figure 12: First-blurred last-sharpened image "Lena" and the recovered watermark.

5 Conclusions

This paper proposes a scheme that combines discrete wavelet transformation and pattern mapping and successfully applied them to the digital image watermarking technique. This scheme meets the requirements of current digital image watermarking techniques. In the proposed watermark embedding process, the pattern mapping method can be used to effectively embed each digital watermark bit into the four DWT frequency coefficients of the digital host image, and retains an acceptable quality of watermarked image. On the other hand, with simple calculation, the proposed watermark extraction process can effectively extract the embedded watermark from the watermarked image to complete the verification of image ownership. From the experimental results, a digital watermark of acceptable quality can be extracted from an attacked watermarked image that undergone processes image cropping, lossy compression, destructive signal processing, and even mixed image processing attacks.

(a) First JPEG compression last-blurred

(b) NC= 0.9255, BCR=86.52%

Figure 13: First JPEG compression last-blurred image "Lena" and the recovered watermark.

(a) First JPEG compression-sharpened last-blurred

(b) NC=0.9997, BCR=99.32%

Figure 14: First JPEG compression-sharpened last blurred image "Lena" and the recovered watermark.

Figure 15: Noised image "Lena" and the recovered watermark,NC=0.9976, BCR=99.32%.

References

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu (1996) Techniques for Data Hiding, *IBM System Journal*, Vol. 35, No. 3, pp. 313–336.

[2] A. G. Bors and I. Pitas (1996) Image Watermarking Using DCT Domain Constraint, *IEEE International Conference on Image Processing (ICIP'96)*, Vol. 3, pp. 231–234.

[3] C. C. Chang and C. S. Tsai (2000) A Technique for Computing Watermark from Digital Images, *International Journal of Computing and Informatica*, Vol. 24, pp. 391-396.

[4] C. C. Chang and K. F. Hwang (2000) A Digital Watermark Scheme Using Human Visual Effects, *International Journal of Computing and Informatica*, Vol. 24, No. 4,pp. 505-511.

[5] C. C. Chang, and H. C. Wu (2001) A Copyright Protection Scheme of Images Based on Visual Cryptography, *The Imaging Science Journal*, Vol. 14, pp. 141-150.

[6] I. J. Cox, J. Kilian, T. leighton, and T. Shamoon (1997) Secure Spread Spectrum Watermarking for Multimedia, *IEEE Transactions on Image Processing*, Vol. 6, No. 12, pp. 1073–1687.

[7] I. J. Cox and J. P. M. G. Linnartz (1998) Some General Methods for Tampering with Watermarks, *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 4, pp. 587-593.

[8] S. Craver, N. Memon, B. L. Yeo, and M. M. Yeung (1998) Resolving Ownerships with Invisible Watermarking Technique: Limitations, Attacks, and Implication, *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 4, pp. 573–586.

[9] C. T. Hsu and J. L. Wu (1999) Hidden Digital Watermarks in Images, *IEEE Transactions on Image Processing*, Vol. 8, No. 1, pp. 58–68.

[10] C. T. Hsu and J. L. Wu (1998) Multiresolution Watermarking for Digital Images, *IEEE Transactions on Consumer Electronics*, Vol. 45, pp. 97-110.

[11] M. Kutter, F. Jordan, and F. Bossen (1998) Digital Watermarking of Color Images Using Amplitude Modulation, *Journal of Electronic Imaging*, Vol. 7, No. 2, pp. 326–332.

[12] R. Ohbuvhi, H. Masuda, and M. Aono (1998) Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications, *IEEE Journal on Selected Areas in Communications*, Vol. 16, No, 4, pp. 551–560.

[13] G. Voyatzis and I. Pitas (1999) Protection Digital Image Copyrights: A Framework, *IEEE Computer Graphics and Applications*, Vol. 1, pp.18-24.

[14] M. D. Swason, M. Kobayashi, and A. H. Tewfik (1998) Multimedia Data Embedding and Watermarking Techniques, *Proceedings of IEEE*, Vol. 86, No. 6, pp. 1064–1087.

[15] R. B. Wolfgang and E. J. Delp (1996) A Ownership for Digital Image, *Proceedings of the 1996 International Conference on Image Processing*, Lausanne, Switzerland, Vol. 3, pp. 219–222.

Development of Diabetes Mellitus Mathematical Models From Patient's Clinical Database

A.Karim El-Jabali
Assistant Professor,
Department of Electrical Engineering, Al-Isra University,
Amman, Jordan
akjab873@hotmail.com

Keywords: Diabetes Modeling; Identification; Time Series; Nonlinear Regression

Received: May 5, 2003

The need for an exact mathematical and computer aided simulation model in physiological research is increasing. This paper deals with the building and identification of diabetes dynamic models. Real clinical data was used to demonstrate how to derive mathematical models that govern the dynamics of diabetes, namely three Multivariable models: Nonlinear Least Squares Regression (NLSR), Auto-Regressive with Exogenous Input (ARX), and Auto-Regressive Moving Average with Exogenous Input (ARMAX) are identified in this process. Their quantitative performance is assessed by four statistical measures which have shown that the ARX is more accurate than the two other models in predicting the near future values of glucose concentration. The general form of the simulated models can be recommended for use in computerized drug delivery systems, as well as for developing an interactive software package for patient and medical staff education

1 Diabetes: The Problem Size

As the number of diabetics grows worldwide, the diabetes mellitus takes an ever-increasing interest in national health care policies and budgets. The latest estimate of World Health Organization of the people affected by diabetes in 2003 is 370 million and about 9% of the global total number of deaths [1]. Gross health care cost of treating diabetes in the USA in 1997 was US$ 98 billion, apart from other intangible costs (pain, anxiety, inconvenience and generally lower quality of life etc). Effective data show that an overall health care cost of diabetes is about 10% of annual health care budgets. The above mentioned figures rise annually as diabetes prevalence increases.

2 Problem Definition

Glucose in blood is controlled mainly by two hormones: the insulin and the glucagon. Inadequate amount of these hormones leads to low (hypoglycemia) or high (hyperglycemia) blood sugar levels. Glucagon and insulin are produced by alpha and beta cells of the pancreas respectively. Cortisone, growth hormone, and catecholamine are other hormones that influence blood sugar levels.

When blood sugar rises after a meal, the beta cells release the insulin which helps glucose penetrate the body cells, thus lowering blood levels of glucose to the normal range. As blood sugar lowers, the alpha cells secrete glucagon, a signal that prompts the liver to release stored glycogen and turn it into glucose, thus raising blood sugar levels to normal range. The normal range of blood sugar is about 60 mg/dL to 120 mg/dL, depending on the last meal a person ate [2].

Diabetes develops when the body cannot use glucose for fuel. This is due the inability of the pancreas to produce enough insulin, or because the available insulin is not effective. As a result, glucose builds up in the blood instead of getting into body cells.

The treatment of diabetes aims to keep blood sugar in a close-to-normal range and to minimize the frequency and severity of glycemic excursions. To achieve that goal, diabetics may use multiple injections of insulin every day, or use insulin pump, make frequent tests of blood glucose, dieting and perform exercise, and get due guidance from health care professionals [3]. In this paper, three algorithms of insulin/glucose interaction that can be used in the treatment are presented.

3 Mathematical Modeling

As biomedical processes are better established and available computing power increases, mathematical models and computer simulations are increasingly utilized and applied. Biologically realistic mathematical models serve as the basis for the majority of the methods used in quantitative physiologic analyses of medical data [4].These models help to test the mechanisms related hypotheses that govern these complex systems, reveal contradictions or incompleteness of data and hypotheses, and allow prediction of system performance under untested or presently un-testable conditions. They may also predict and supply the values of experimentally

inaccessible variables. These models should be as representative and comprehensive as possible, but not complicated. The first two requirements give good insight into what is being investigated by providing a concise summary of the observed behavior. The complexity of the model may be restricted by the need for implementation, particularly because applying more complex models is time-consuming and requires considerable effort.

The general potential of mathematical models is apparent when there is sufficient knowledge about the system. This knowledge allows the formulation of solid hypotheses [5]. As the ability to acquire data expands and the sophistication of computing methods increases, more effective and broader applications of these models are expected. This includes, but not restricted to, estimation, prediction, calibration, and optimization [6]. Hence, estimation is made to obtain the value of the model's parameters (coefficients) for a particular combination of values of the predictor variables, while prediction is used to test the model's validity.

However, mathematical models of diabetes are of wide diversity. The most widely used in literature are the compartment models that consist from linear and nonlinear differential equations [7, 8, 9]. One common model in this category is the so called Automated Insulin Dosage Advisor model, or simply AIDA model [10]. It consists of four differential equations along with twelve auxiliary relations that determine the parameters of the model. AIDA model was developed for patient and medical staff education. Other classes of models like probabilistic [11] and non-compartment [12] were also used. Recent spectrum of diabetes models is reported in [13, 14].

In the last decade, several research groups used, and continue to use, modern identification techniques that are based on artificial intelligence, statistical and quantitative methods. These include time series methods [15], fuzzy logic [16], neural nets [17,18] and a combination of these and other methods[19]. An example of artificial neural networks- based models (ANN) is that developed by Pender [17] which had shown good performance in predicting blood glucose levels 2 hours ahead. All these models of glucose metabolism and insulin kinetics, including that developed in this paper, are intended to estimate the time course of glucose concentration and are expected to be conclusive to a generalized model.

4 Data and Methods

The data used in this study are based on those of the Artificial Intelligence in Medicine symposium AIM-94[20] .It consists of a protocol of type I diabetic patient observations over a period of 75 days. This period of time is quite sufficient to capture a detailed record of excursions, and provides ample information about the patient's diabetes profile. Meanwhile, the Present blood Glucose Level (PGL), dosages of insulin injections: Short-Term acting Insulin (STI), Mid-Term acting (MTI) and Long-Term acting (LTI) Insulin, and qualitative indication about the amounts of food intake (Meal) and physical effort exerted (Exercise) are recorded. The total number of readings is 560 for each variable. Figure (1) shows twenty readings of glucose from which it is clear that blood glucose fluctuates in a wide range. The time between readings is about three hours. The goal of this study is to establish effective equations i.e. a mathematical model of diabetes dynamics that is consistent with the data. The efficiency of the developed model is to be tested by its ability to determine the next glucose level for specified parameter values, and to perform parameter estimation in which model outputs are fit to the clinical data. The inputs to the system u(s) in all developed models include PGL, STI, MTI, LTI, Meal, and Exercise, while the output is the Next Glucose Level (NGL). In this study the data is divided into two subsets:

- Estimation Data Set – used to estimate the parameters of the mathematical models
- Validation Data Set –used to decide whether the models correctly predict the actual output

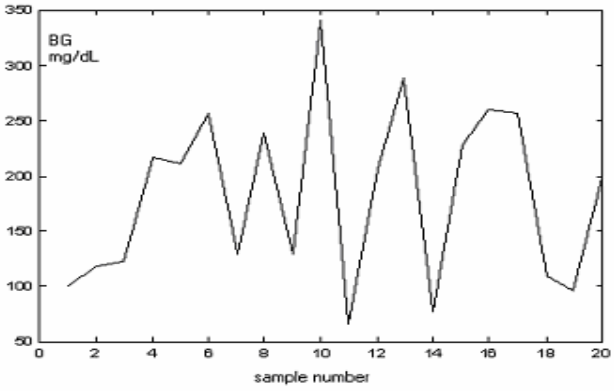


Figure 1: The first twenty readings of blood glucose.

5 Prediction Strategies

Model validation is the core of the identification problem, but there is no absolute procedure for approaching it [21]. One primary tool is to compare the results obtained from model structures with different orders. Another method is to test prediction properties of the model. Hence, prediction is necessary to evaluate how well the proposed time series model is capable of predicting future values of the data. This prediction method depends on the model structure. The validation data set is usually different from that used for parameter estimation.

Figure (2) illustrates the task of one step prediction in which the previous inputs u(k-i) and outputs y(k-i) of a process are given and the process output $\hat{y}$ (k) at the next sample instant is predicted.

To compute the k-step ahead prediction of the output y(t), the suggested model uses the information in the input u(s) up to time s= t, and information in the output y(t) up to time s = t-k , as input values and the results of the prediction is compared with the output data y(t)

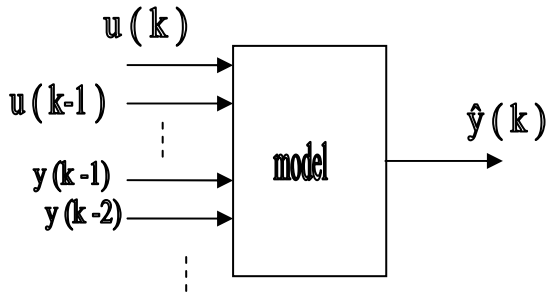


Figure 2: One step prediction of the output.

One method to obtain multi step prediction [21,22] is to use recursively the model shown in figure (2) i.e. at the first step the prediction of one step into the future is accomplished, next the same model is used to predict a further step ahead by replacing k with k+1 and utilizing the result ŷ (k) from the previous prediction step as an input to the second stage of the model. This procedure can be repeated to predict the required number of future steps altogether. The drawback of this approach is that the prediction relies on previous predictions and thus the prediction errors may accumulate

6 Measures of Prediction Errors

To assess the performance of all models the difference between the Predicted Next Glucose Level (PNGL) and Actual Next Glucose Level (ANGL) from the data set is defined as an Error

Error = ANGL – PNGL **(1)**

and the following measures of performance were introduced

Mean Square Error (MSE)

MSE = ∑(Error.^2)/length(ANGL) **(2)**

Mean Absolute Error (MAE),

MAE = ∑ |Error|/length(ANGL) **(3)**

Average Error (AE)

AE = ∑ (Error)/length(ANGL) **(4)**

Percentage Relative Error

PRE = |((PNGL - ANGL)./ ANGL)|.*100 **(5)**

Three different multivariable models of glucose/insulin dynamics are developed, assessed and compared: Nonlinear Least Square Regression NLSR, Auto-Regressive with Exogenous Input (ARX) and its Moving Average version (ARMAX).

7 Nonlinear Least Square Regression (NLSR)

Since the nature of the dynamics of diabetes, as many other biomedical phenomena, is nonlinear [12] the corresponding nonlinear regression is a more realistic favorite to find a fitting model. A nonlinear model is any model of the form

$$\mathbf{Y} = f(x_i, \alpha_i) + \varepsilon \quad (6)$$

in which the functional part of the model f is not linear with respect to systems variables x_i and the unknown parameters α_i , while ε represents random disturbances.

The least squares method is used to estimate unknown parameters by minimizing the sum of the squared deviations between the data and the model. These estimates are actually optimal [22]. Nonlinear least squares regression extends linear least squares regression for use with a much larger and more general class of functions. Almost any function that can be written in a closed form can be incorporated in a nonlinear regression model.

Being a "least squares" procedure, NLSR has some of the advantages (and disadvantages) that the linear least squares regression has over other methods. One common advantage is the efficient use of data. That is, it can produce good estimates of unknown parameters in the model with relatively small datasets. The major cost of moving to nonlinear least squares regression from other simpler modeling techniques is the need to use iterative optimization procedures to compute the parameter estimates. This necessarily requires the user to provide starting values for the unknown parameters before the software can begin optimization [23]. Selecting an appropriate nonlinear model is always an iterative process. Much of the need to iterate stems from the difficulty in initially selecting a function that describes the data well. Details about the data are often not easily visible as originally observed, especially in the case of multivariable systems. In reaching the best model the starting model was the linear model

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 + \alpha_5 x_5 \quad (7)$$

where y, x_1, x_2, x_3, x_4, and x_5 are the NGL, PGL, STI, MTI, LTI and Meal respectively, while α_i are unknown coefficients to be estimated. This linear model gives a Percentage Relative Error (PRE) value of 35.712% in predicting the 5 readings ahead of the next glucose level. Moreover, the MSE is of 10^8 orders and MAE is 69.0618.These facts show that the linear model is poor and is not an adequate candidate to describe the dynamics of diabetes. The next step in obtaining a more accurate model is the addition of nonlinear terms to the previous model. Trying different combinations of nonlinear formulae lead to the following equation:

$$\mathbf{y = x_1 + \alpha_1 x_5 + \alpha_2 [\exp(-\alpha_3 (x_2 + x_3 + x_4))\alpha_4 x_1] + (\alpha_5 x_2 + \alpha_6 x_3 + \alpha_7 x_4) x_1} \quad (8)$$

Model validation is the next and possibly the most important step in the model building sequence. To get this task accomplished, another part of data was applied to test the model's potential to predict the actual measured output. The result, i.e. the predicted values of the output ŷ (k) i.e. PNGL and their actual values y (k) i.e. ANGL are shown in figure (3). Moreover, the difference between the actual and predicted values is obtained in figure (4). The performance of the model is computed for 5, 10, and 60 steps ahead, using MSE, MAE, AE, and PRE. The computed values of the

performance criteria of NLSR are summarized in Table (1).

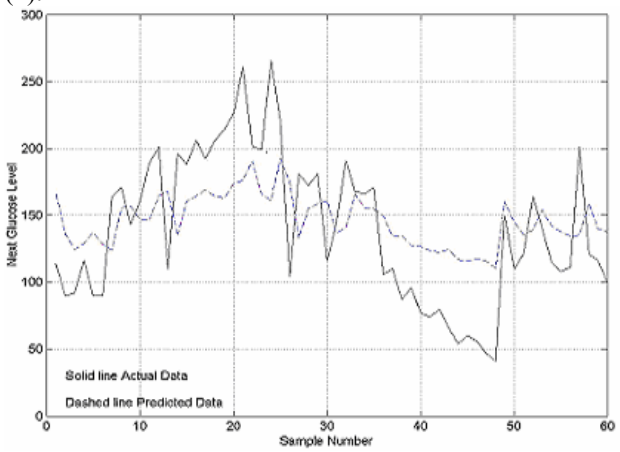


Figure 3: Actual and Predicted Data for Non-Linear Model.

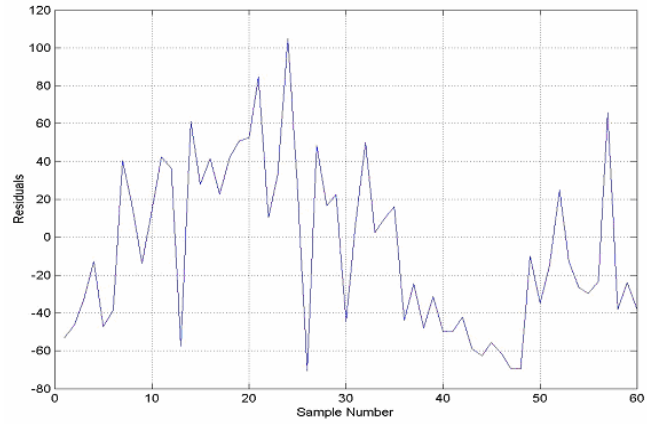


Figure 4: Errors between Actual and Predicted Data for Non-Linear Model.

8 Time Series Models

Time series methods aim to explore the main characteristics of the system through the frequent measurements of the input and output variables. When several variables are taken together, the series becomes multivariate. The model in this case tries to capture the trends and the various inter-relationships between the different series. A time series is not usually a very compact representation of a time evolving phenomenon. Therefore, it is necessary to condense the information available and find a parameterization that contains the most relevant features to the underlying system. The amount of information that can be recovered from time-delayed copies of finite sets of noisy measurements is, however, quite a complicated question that has no general answer right now [24,25,26].

One formal requirement for almost all time series methods is stationarity. The most common definition of a stationary process found in textbooks (often called strong stationarity) is that all conditional probabilities are constants with time. If nonstationarity is detected, the time series is often discarded as unsuitable for a detailed analysis, or split into segments that fall short to be regarded as stationary [25,26].There are many methods to investigate the stationarity of the data [22,25,27].The application of these methods suggested by Harvey [27] on data under consideration, approved the further developing of time series models for diabetics.

Indeed, the application of these methods to explore the underlying dynamics of biomedical systems is widespread [28, 29, 30]. Interdisciplinary works which combine approaches from different disciplines in solving problems are just one feature of research of the last decade. It has been manifested in many publications which use time series and artificial neural networks concepts to beat the complex problems like diabetes [30,31]. Among the many forms of time series models are multivariable Auto-Regressive with Exogenous Input (ARX) Model and the multivariable Auto-Regressive Moving Average with Exogenous Input (ARMAX) Model .Both forms are used in this paper to derive models of diabetes mellitus.

8.1 ARX model

A common question that justifies the use of ARX models is whether present and future glucose values can be predicted from recent blood glucose history. The fact that there is significant statistical inter-dependence among the individual successive glycemic measurements indicates

Table 1: NLSR performance evaluation for 5, 10, and 60 steps prediction.

Model	NLSR model			
Performance	**MSE(x10^3) mg/dL**	**MAE mg/dL**	**AE mg/dL**	**PRE %**
5 step ahead	**1684.5**	**38.428**	**-38.428**	**-1.9710**
10 step ahead	**1217**	**31.475**	**-17.433**	**-2.0698**
60 step ahead	**1905**	**38.496**	**-6.0382**	**14.371**

the motivation to use the patient's record to find a model that at least forecasts the near future values of glucose. The latter might be helpful for both the patient and physician in dealing with the complications of the disease. In this study, the problem is a multidimensional with five inputs and one output. A multivariable ARX model is given by

$$A(q)y(t) = B(q)u(t-nk) + e(t) \qquad \textbf{(9)}$$

where u(t),y(t), and e(t) are input, output and random disturbance vectors respectively. *nk* is the number of delays from input to output and $A(q)$, B(q) are the corresponding polynomials [27] in the delay operator q^{-1}. To reach the best polynomial, low order models were initially used, then the order was discretely increased. This procedure leads to the following best model with parameters as shown in Table (2).

The model uses three past values of glucose history (A-matrix), and four past values of different types of insulin(B1, B2, and B3), meal (B4) and exercise (B5) to model the dynamics. Then, the performance of the obtained ARX model was assessed by a validation data set that differs from the estimation data set. The new set was applied to test the model's ability to predict the actual measured glucose level. The predicted model output and the actual data are shown in figure (5).The error (residuals) between the predicted and actual data is shown in figure (6).The performance of the model is computed for 5, 10, and 60 steps ahead using MSE, MAE, AE, and PRE. The computed values of the performance criteria of ARX are given in Table (3).

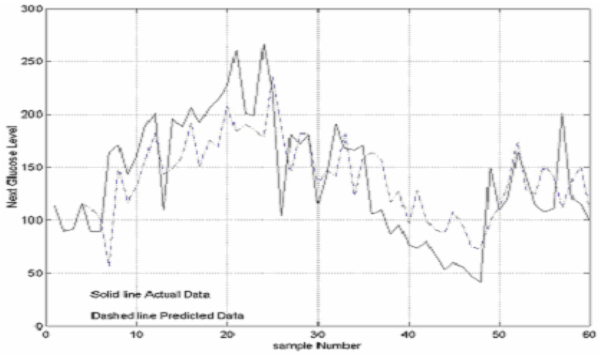


Figure 5: Actual and Predicted Data for ARX Model.

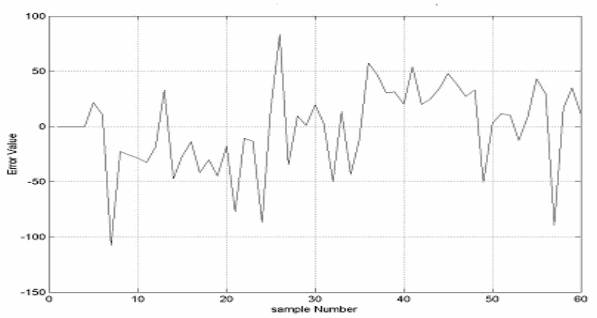


Figure 6: Error between Actual and Predicted Data for ARX Model.

Table 2: the coefficients of ARX model.

	q^0	q^-1	q^-2	q^-3	q^-4
A	**1**	**- 0.2787**	**- 0.1093**	**0.1125**	**0**
B1	**0**	**0**	**0.1813**	**0.3393**	**- 0.1359**
B2	**0**	**0**	**-2.709**	**1.813**	**- 0.195**
B3	**0**	**0**	**-0.8138**	**2.086**	**2.087**
B4	**0**	**0**	**0.5189**	**1.762**	**1.026**
B5	**0**	**0**	**-4.937**	**45.88**	**7.849**

Table 3: ARX performance evaluation for 5, 10, and 60 steps prediction.

Model	ARX model			
Performance	**MSE(x10^3) mg/dL**	**MAE mg/dL**	**AE mg/dL**	**PRE %**
5 step ahead	**0.00375**	**4.3295**	**4.3295**	**0.2405**
10 step ahead	**1.4140**	**21.668**	**-15.262**	**1.5025**
60 step ahead	**1434.8**	**29.739**	**-1.563**	**10.26**

8.2 ARMAX model

A general multivariable ARMAX model structure is

$$A(q)y(t) = B(q)u(t-nk) + C(q)e(t) \qquad \textbf{(10)}$$

where, $A(q)$, $B(q)$, and *nk* are as for ARX case while $C(q)$ is the polynomial representing the weight of the disturbance e(t) .The orders of A, B, and C are to be selected and their coefficients to be estimated by least square regression algorithm. Starting from low orders and carrying iterations for each order, the best performance model is reached. Model coefficients are given in table (4). Hence the introduction of weighting error matrix C (q) reduces the required past values of the

input variables to three. As in the case of ARX, the obtained ARMAX model was tested by a new data set. The predicted output by the model and the actual data are shown in figure (8).The error (residual) between the predicted and actual data is shown in figure (9). As in the case of NLSR and ARX, the performance is assessed for 5, 10, and 60 steps ahead. The performance measures of ARMAX model are summarized in Table (5).

Table 4: the coefficients of ARMAX model.

	q^0	q^-1	q^-2	q^-3
A	**1**	**0.05073**	**- 0.04447**	**0.2591**
B1	**0**	**0.005721**	**0.3144**	**0.4302**
B2	**0**	**-0.5042**	**- 2.41**	**0.6396**
B3	**0**	**0.01116**	**- 0.3916**	**2.815**
B4	**0**	**1.083**	**1.123**	**1.281**
B5	**0**	**30.55**	**21.67**	**37**
C	**1**	**0.3856**	**0.1649**	**0.09075**

9 Results

Models obtained in this study performed best when using MSE, MAE, and PRE as a criterion for predicting future values of blood glucose levels. NLSR model has given good results in short term prediction. It generates a small error (see table 1) using all measures of performance in predicting 5 and 10 readings ahead of the NGL output. The PRE of prediction, for example was -1.971 % and -2.0698% respectively. The negative sign of the PRE indicates that the predicted value is less than the actual reading. The model's prediction ability deteriorates for long term prediction (60 readings ahead) with corresponding relative percentage error of 14.371%

Using the time series ARX model, with third order input polynomial and fourth order output polynomial, it was found that (see table 3)the relative percentage error PRE is 0.2405% and 1.5025% in predicting the 5 and 10 future values of NGL respectively. Moreover, it was observed that the developed model's output replicates the system's output (patient's glucose levels) for the first four prediction samples. This fact is shown more clearly in figure (7). This is equivalent to about ten future hours, which is a good indication of the model validity. After that, (for 60 readings ahead), the model becomes less reliable and its performance tends to go far away from the actual data. Hence the other performance criteria act in identity with the PRE.

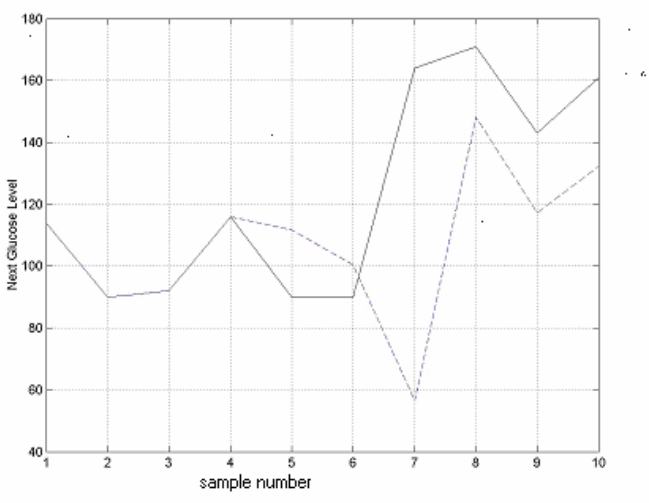


Figure 7: The first ten actual and predicted data for ARX model.

The performance of the developed ARMAX model with third order input polynomial, third order disturbance polynomial, and third order output polynomial is not any better than that of the ARX model. It gives a slightly increased PRE, (see table 5) in the prediction of the near future values of NGL. In addition, none of the first four values are in agreement with the original value of the systems output as it is with the ARX model (see figure 8).

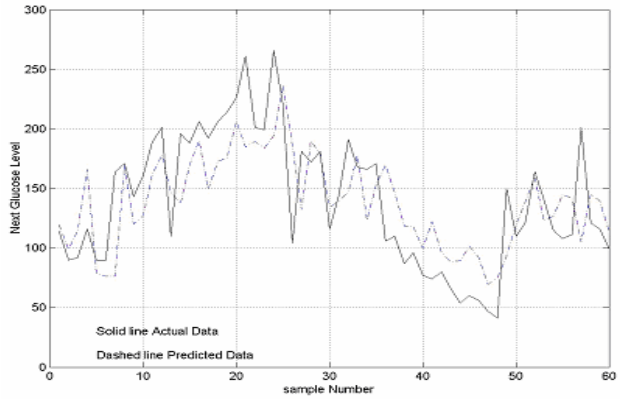


Figure 8: Actual and Predicted Data for ARMAX Model .

Table 5: ARMAX performance evaluation for 5, 10, and 60 steps prediction.

Model	ARMAX			
Performance	**MSE(x10^3) mg/dL**	**MAE mg/dL**	**AE mg/dL**	**PRE %**
5 step ahead	**0.6459164**	**19.033**	**14.6018**	**0.9131**
10 step ahead	**1.2788**	**25.828**	**-8.4937**	**2.0166**
60 step ahead	**1387.5**	**30.461**	**-1.52**	**12.34**

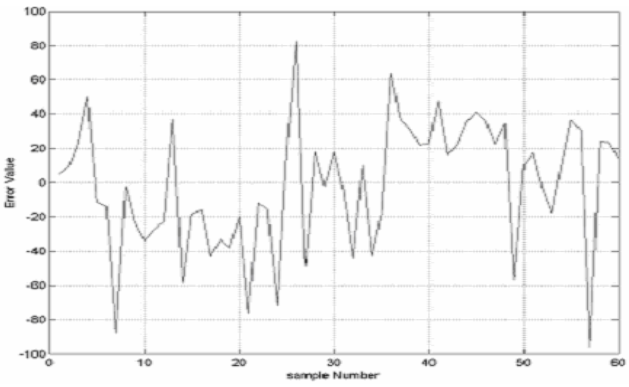


Figure 9: Error between Actual and Predicted Data ARMAX Model.

10 Argumentation

The developed time series models in this paper focus on the use of a diabetic record of direct measured variables. In contrast to many previously developed heuristic rule-based expert systems and linear models of diabetes mellitus [17, 32, 33], these models not only can model diabetes mellitus using relatively simple interpreted physiological terms, but also can be of good potential in predicting future values of glucose concentration. The models account for meal and exercise qualitatively using a two-levels scale (0 –for the case of going with no food or when no exercise had been done, and 1- for the case of eating or having done exercise). In this case, and in contrast to AIDA model [34], determining the values of these parameters is not difficult in real life. Moreover, the developed models are simple and readily understandable to a health care worker or patient.

Exploring the predictability of future blood glucose levels, from the time course of the patient's records, provides a good tool to judge whether the proposed models have acceptable performance. Although, the deviation between the observed and predicted blood glucose values is used in Tresp et al [30], the authors did not give any quantitative evaluation of the performance. Furthermore, many ANN- based studies [17, 29] used only one quantitative measure of the performance, usually the normalized MSE. The assessment of the models developed in this study is accomplished through using four quantitative measures: MSE, MAE, AE, and PRE. These criteria gave identical indication of the ability of these models to predict the future values of blood glucose concentration. Moreover, the accuracy of prediction is in general better than that provided in other studies [17, 28]. For example, the PRE of the model developed in Pender [17] ranges from 3% to 12.4% of the actual blood glucose level depending on the validation data set, while the results of the developed current models (see tables 1,3, and 5) provide less PRE for the case of 5 and 10 prediction steps. Discrepancy between the observed and predicted blood glucose values became systematically worse as time progressed from the date of the original parameter estimation.

All that said, the three mathematical models are considered to be of good performance. The NLSR model is the worst and can be further improved by adding more components or by adjusting one or more of them. This problem is not evident in the case of the time series models. The ARX model showed its power to predict near future values of the glucose concentration with acceptable accuracy (see table 3), while ARMAX model(see table 5) is less accurate in obtaining the expected value of glucose, though still good. Both models are of good use for programmed insulin delivery devices and for educational purposes. However, these models can not recommend the amount of insulin to be delivered in response to the elevated glucose concentration. Developing an appropriate controller to fulfill this task needs more research.

11 Conclusions and Future Work

Describing complex medical processes like diabetes by a relatively simple and accurate time series models is possible. Meanwhile, many computerized systems of drug delivery and implantable pumps have been developed and all try to give patient an adequate amount of insulin in proper time. The presently developed time series algorithms show that these systems can use the available data on the patient, current status and history to predict the next value(s) of glucose level. Likewise they can be utilized in computerized trial design, and to assist clinical investigators. Indeed, effective use of models can significantly reduce the time and costs involved in drug development and administration. The obtained models open the way for seeking a general model that can be used for developing an interactive software package similar to AIDA. This goal can be achieved by comparing the similar models derived from data base of many patients. Furthermore, this software can account for more factors that affect the patient's health. The long-term goal of this research is the development of an optimal controller that will use blood glucose measurements, qualitative information about food intake, physical exercise and past insulin infusion values to compute the proper insulin dose that compensate the elevated glucose concentration. Other problems that can be investigated in future work are the different time intervals between measurements and the missing data in the time series.

12 References

[1] World Health Organization; Fact Sheet N° 236; September 2002.

[2] R.Cotran; Pathological Basis of Disease, 6th edition; W.B. Saunders Company; Ontario, Canada; 1999.

[3] Jean Venable R, Goode, Pharm D.; New Advances in the Treatment of Diabetes; Medscape Portals; New Jersey; 2001

[4] Modeling in Biomedical Research: An Assessment of Current and Potential Approaches: Applications to Studies in Cardiovascular/Pulmonary

Function and Diabetes; Proceedings of National Institutes of Health Conference; USA; 1989.

[5] David Foster, Arthur J. Atkinson, Principles of Pharmacokinetic Data Analysis: Modeling and Simulation, Philadelphia, Pennsylvania; 2002

[6] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/, 2003

[7] Candas B, Radziuk J: An adaptive plasma glucose controller based on a nonlinear insulin/glucose model; IEEE TA BME 41:116-124, 1994.

[8] Thomas Briegel, V.Tresp;A Nonlinear State Space Model for the Blood Glucose Metabolism of Diabetic; Automatisierungstechnik; vol. 50 No 5; Oldenbourg Verlag; Germany; 2002.

[9] R.S.Parker, F.J.Doyle , and N.A.Peppas; A Model-based Algorithm for Blood Glucose Control in Type 1 Diabetic Patients; IEEE TA BME Vol.46,No 2, 1999.

[10] Lehmann ED; Interactive educational simulators in diabetes care; Medical Informatics; Vol 22;pp 47-76; USA;1997.

[11] Andreassen S, Benn J, Hovorka R, Olesen K, Carson E; "A probabilistic approach to glucose prediction and insulin dose adjustment: description of a metabolic model and pilot evaluation study"; Computer Methods and Programs in Biomedicine Vol 41; pp.153-163, 1994, Elsevier Sc. Publ. Ireland.

[12] Cobelli C, Toffolo G, Ferrannini E; A model of glucose kinetics and their control by insulin, compartmental and noncompartmental approaches. Mathematical Biosciences, 71:291-316, 1996; Elsevier Sc Publ. Ireland.

[13] Naylor JS, Hodel AS, Albisser AM, Evers JH, Strickland JH, Schumacher DA: Comparison of parameterized models for computer-based estimation of diabetic patient. glucose response. Medical Informatics, Vol 22, PP.21-34, 1997.

[14] Brian Hipszer; A type 1 Diabetic Model; MSc. thesis, Drexel University; 2001.

[15] Riccardo Bellazzi, Paolo Magni, Giuseppe De Nicolao; Bayesian Analysis of Blood Glucose Time Series from Diabetes Home Monitoring ; IEEE TA BME Vol.47,No 7, 2000.

[16] A.Karim El-Jabali A. Kailani, M. Abbas. "Dynamic simulation of fuzzy controller of glucose concentration based on patient's clinical database"; Modeling and Simulation Conference MS'02; Melbourne, Australia 2002

[17] Pender J; "Modelling of blood glucose levels using artificial neural networks". Phd Dissertation. University of Strathclyde, 1997. Scotland.

[18] V. Tresp, T. Briegel and J. Moody; Neural Network Models for Blood Glucose Metabolism of Diabetic; IEEE TA on Neural Networks, Vol.10, No.5,pp. 1204--1213, 1999.

[19] R.J.Frank, N.Davey, S.P.Hunt ;Time Series Prediction and Neural Networks;Department of Computer Science, University of Hertfordshire, Hatfield, 1997. UK

[20] M. Kahn; Artificial Intelligence in Medicine AIM-94; Washington University; 1994

[21] Ljung, L.; System Identification-Theory for the User; Englewood Cliffs; 1999.

[22] O.Nelles; Nonlinear System Identification; Springer –Verlag Berlin, Germany, 2001.

[23] Moscinski J., Ogonowski Z; Advanced Control with Matlab & Simulink; Ellis Horwood; 1995.

[24] Raimondas Ciegis; Some Algorithms in Mathematical Modeling of Diabetes Mellitus; International Journal Informatica; Lithuanian Academy of Sciences; Vol. 6 No 1, 1995, Lithuania.

[25] Thomas Schreiber; Interdisciplinary application of nonlinear time series methods; Physics Reports; vol. 308, 1999.

[26] Silipo, R.; Deco, G.; Vergassola, R.; Bartsch, H; Dynamics extraction in multivariate biomedical time series; Biological Cybernetics, Vol. 79 Issue 1, pp.15-27,1998

[27] Andrew C. Harvey; Time Series Models, 2nd edition; Harvester Wheatsheaf; 1992.

[28] Bremer, Troy; Gough, David A.; Is blood glucose predictable from previous values? Diabetes, Vol. 48 Issue 3, 1999.

[29] Prank, Klaus; Jurgens, Clemens; Huhlen A, Brabant G; Predictive neural networks for learning the time course of blood glucose levels ; Neural Computation, Vol. 10 Issue 4., 1998.

[30] Tresp, V., Moody, J., Delong, W ; Neural network modeling of physiological processes; In Computational Learning Theory and Natural Learning Systems, vol. 2, T. Petsche, M. Kearns, S. Hanson, R. Rivest (editors). Cambridge, MA: MIT Press, pp. 363--378, 1993.

[31] S. L. Ho, M. Xie and T. N. Goh ; A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction; Computers & Industrial Engineering; Vol. 42, PP. 371-375, 2002.

[32] Deutsch T, Carson E, Harvey F, Lehmann E, Sonksen P, Tamas G, Whitney G, Williams C; Computer-assisted diabetic management: a complex approach. Computer Methods and Programs in Biomedicine; Vol. 32: PP. 195-214, 1990.

[33] Lehmann E, Deutsch T, Roudsari A, Carson E, Benn J, Sonksen P; A metabolic prototype to aid in the management of insulin-treated diabetic patients. Diabetes Nutrition and Metabolism; Vol.4 (Suppl. 1), PP. 163-167, 1991.

[34] Guyton J, Foster R, Soeldner J, Tan M, Kahn C, Koncz L, Gleason R. A model of glucose-insulin homeostasis in man that incorporates the heterogenous fast pool theory of pancreatic insulin release; Diabetes; Vol.27,PP.1027-1042, 1978.

Parallel Modular Exponentiation Using Signed-Digit-Folding Technique

Der-Chyuan Lou and Chia-Long Wu
Department of Electrical Engineering
Chung Cheng Institute of Technology
National Defense University
Tahsi, Taoyuan 33509, Taiwan
E-mail: dclou@ccit.edu.tw
Web Address: http://www.ccit.edu.tw/~elec/teach/t3.htm
Phone: 886-3-3809331; Fax: 886-3-3801407

Keywords: Computer arithmetic, modular exponentiation, public-key cryptography, signed-digit recoding, redundant number system, Galois fields.

Received: July 1, 2003

Fast modular exponentiation algorithms are often considered of practical significance in RSA cryptosystem. In this paper, a new modular exponentiation algorithm is proposed which based on the binary algorithm, signed-digit representation, and the folding-exponent technique. As the "signed-digit recoding algorithm" has less occurrence probability of the nonzero digit than binary number representation. Taking this advantage, we can effectively decrease the amount of modular multiplications. By using the technique of recording the common parts in the folded substrings, the "folding-exponent algorithm" can improve the efficiency of the binary algorithm, thus can further decrease the computational complexity of modular exponentiation. As the modular squaring operation in GF(2^{n}*) finite field can be done by a simple shift operation when a normal basis is used, and the modular multiplications and modular squaring operations in our proposed signed-digit recoding scheme can be executed in parallel, by using our proposed generalized* r*-radix signed-digit folding algorithm, we can decrease the computational complexity to* $2^{n-1}\{(\frac{4r+3}{4(r+1)^2})\times\frac{k}{2^n}\}+2^n+(k-\frac{k}{2^n})+(2^n-1)$ *multiplications where* k *denotes the digit-length of the exponent and* n *denotes the folding time of the exponent, respectively. Furthermore, if we have the folding time* n*=1 to minimize the total multiplication complexity, we can obtain the optimal overall computational complexity as* $\frac{k}{2}+\frac{k}{2}[\frac{4r+3}{4(r+1)^2}]+3$ *multiplications in* r*-radix signed-digit recoding system.*

1 Introduction

The motivation of studying high-speed and space-efficient algorithms for modular exponentiation (ME) comes from the applications in cryptography. Taking the RSA cryptosystem [1] for example, the public and private keys are functions of a pair of large prime numbers, and the encryption and decryption operations are accomplished by modular exponentiation. This modular exponentiation problem can be described as follows. Given M (message), E (public key), and N (the product of two large primes), compute ciphertext $C=M^E$ mod N. For the computation of modular exponentiation of M^E mod N, the very intuitive way is to break the modular exponentiation operation into a series of modular multiplications. That is, we first multiply M by itself (E-1) times, and then the result is obtained right after executing modulo N operation.

As efficient computation of the modular exponentiations $C=M^E$ mod N is very useful for cryptosystem, we need fast multiplication designs or novel exponentiation algorithms such as the Montgomery reduction method [2], high-radix method [3], addition chains method [4], binary method [5], residue number conversion method [6, 7], signed-digit recoding method [8], exponent-folding method [9], non-standard arithmetic method [10], and compression-based method [11]. Moreover, a detailed survey of fast exponentiation techniques has been described in [12].

The rest of the paper is organized as follows. In Section 2, we review and introduce some famous works of the modular exponentiation. The proposed high-radix signed-digit-folding (HRSDF) algorithm for fast modular exponentiation is discussed in Section 3. The computational complexity of the proposed algorithm is detailed analysed in Section 4. Finally, we conclude our work in Section 5.

2 The Modular Exponentiation

Most modern cryptosystems are based on modular exponentiation. Exponentiation of large integers with large exponent and modulus (longer than 512 bits) is one of the most important operations in several well known cryptographic algorithms. In 1976, Diffie and Hellman [13] had proposed the first public-key algorithm as

"Diffie-Hellman key exchange scheme". In 1978, RSA public key cryptosystem was invented by Rivest *et al*. [1] and is widely used in secure electronic communication. The ElGamal scheme can be used for both digital signatures and encryptions [14].

The modular exponentiation used in the RSA public-key cryptosystem can be expressed as follows. Given message M, public-key E, and N, compute ciphertext $C=M^E \bmod N$. We focus the fast exponentiation problem, and this problem is dependent on the algorithm being used and its implementation. For examples, a fixed number is raised to different powers in the ElGamal cryptosystem and Diffie-Hellman key exchange scheme, so pre-computing some powers can save time at the expense of more storage.

The modular exponentiation is composed of repetition of modular multiplications. Therefore, modular exponentiation can be time consuming, and is often the dominant part of modern cryptographic algorithms for key exchange, electronic signatures, and authentication. Two different approaches are often used to reduce the execution time of the modular exponentiation operation. One approach is simply to reduce the number of modular exponentiation. The other approach is to reduce the execution time of each modular multiplication. In this paper, we are concentrate on the first approach to effectively reduce the number of modular exponentiation.

2.1 The Binary Method

The binary method is also known as the "square-and-multiply" method [5]. The basic idea of binary method is to compute M^E using the binary expression of exponent E. Assume k denotes the bit-length of the exponent E, the exponent E can be expressed in binary representation as $E = (e_{k-1}e_{k-2}...e_1e_0)_2$ and $E=\sum_{i=0}^{k-1} e_i \times 2^i$, where $e_i \in \{0,1\}$.

The exponentiation operation is broken into a series of squaring and multiplication operations [15] by the use of the binary method. There are two commonly used algorithms [5][16] in binary method can convert the modular exponentiation of $C = M^E \pmod N$ into a sequence of modular multiplications, i.e., the LSB method and the MSB (most significant bit) method. The LSB (least significant bit) algorithm computes the exponentiation starting from the least significant bit of the exponent and proceeding to the left, which is shown as follows.

LSB (Right-to-Left) Algorithm
Input: Exponent: $E = (e_{k-1}e_{k-2}...e_1e_0)_2$;
Message: M;
Output: Ciphertext: $C = M^E$;
$C = 1$; $S = M$;
begin
for $i = 0$ ***to*** k-1 ***do*** /*scan from right to left*/
begin
if $(e_i = 1)$ $C = C \times S \bmod N$; /*multiply*/
$S = S \times S \bmod N$; /*square*/
end;
end.

If we use the LSB algorithm to calculate $C = M^{37}$ for example, we can first transfer the decimal representation 37_{10} into binary representation $(100101)_2$ for exponent part. We then can get $C=M \times M^4 \times M^{32} = M^{(1+4+32)} = M^{37}$.

The MSB algorithm computes the exponentiation starting from the most significant bit of the exponent and proceeding to the right, which is shown as follows.

MSB (Left-to-Right) Algorithm
Input: Exponent: $E = (e_{k-1}e_{k-2}...e_1e_0)_2$;
Message: M;
Output: Ciphertext: $C = M^E$;
$C = 1$;
begin
for $i = k$-1 ***to*** 0 ***do*** /*scan from left to right*/
begin
$C = C \times C \bmod N$; /*square*/
if $(e_i = 1)$ $C = C \times M \bmod N$; /*multiply*/
end;
end.

If we use the MSB algorithm to calculate $C=M^{57}$, we can first transfer decimal representation 57_{10} into binary representation $(111001)_2$. We then calculate $C = ((((M^2 \times M)^2 \times M)^2)^2 \times M = ((((M^3)^2 \times M)^2)^2)^2 \times M = ((((M^7)^2)^2)^2 \times M = (M^{56}) \times M = M^{57}$.

As the LSB and MSB algorithms have the same computations for both multiplication and squaring operations, therefore they have the same computational complexity. But there are few differences existing between these two algorithms, to be specifically, the scan patterns of these two algorithms are different and squaring operations are executed in different procedures. Both LSB and MSB algorithms have two same states. The first state is to execute the multiplication operation as the bit "1" being scanned, the second state is to execute the squaring operation when the bit "0" being scanned.

Take k-length binary exponent for example, for the average case, we assume the occurrence probabilities for both bits "1" and bits "0" are the same. Then, the expectation hamming weights for bits "1" and "0" are both $k/2$.

Therefore, the computational complexities for both LSB and MSB algorithms are $2\times(k/2)+ 1\times(k/2) = 1.5k$ multiplications to evaluate M^E. Notice, those two algorithms although using different scan manners can still get the same result to evaluate the modular exponentiation $C = M^E \pmod N$ as we put modulo N operation after every multiplication process.

Considering the hardware implementation, the operation in LSB algorithm requires one more storage register S than in MSB algorithm. Therefore, the MSB algorithm is more appropriate for hardware design. As there are two independent variable registers S and C in

the LSB algorithm, the value in register S will not influenced by the value in register C. Therefore, we choose the LSB algorithm to implement modular exponentiation operation in our proposed algorithm.

2.2 The Exponent-Folding Algorithm

In 1996, Lou and Chang [9] proposed a fast exponentiation method using exponent-folding technique. Let the exponent E have the binary representation $(e_{k-1}e_{k-2}...e_1e_0)_2$ i.e. $E=\sum_{i=0}^{k-1} e_i \times 2^i$, where $e_i \in \{0,1\}$ and k is the bit-length of the exponent E. Here we depict the Exponent-Folding algorithm as follows.

In the first phase of the Exponent-Folding algorithm, by folding the exponent E in half n times, and E is then divided into 2^n equal sized sub-strings. Let each sub-string of E be denoted as E_i for i= 1, 2, ..., 2^n, i.e. $E = E_{2^n} \| E_{2^n-1} \| E_{2^n-2} \| \ldots \| E_2 \| E_1$, where "$\|$" is the concatenation operator. Hence

$$M^E = \prod_{i=1}^{2^n} S^{(i-1)(\frac{k}{2^n})}(M^{E_i}), \quad (1)$$

where $S^{(m)}(z)$ represents performing m squares on the related value z, and E_i is denoted as $(e_i^{\frac{k}{2^n}-1} e_i^{\frac{k}{2^n}-2} \ldots e_i^1 e_i^0)_2$. In the second phase, we perform bit-wise **AND** operation between two bit-strings E_i and E_{i+1} and bit-wise **XOR** operation between two bit-strings E_i and E_{com_i}, we define the following variables:

$E_{com\_i} = E_{com\_i+1} = E_i$ **AND** E_{i+1} for i = 1, 3, ...,2^n-3, 2^n-1, (2)

$E_{excl\_i} = E_{com\_i}$ **XOR** E_i for i = 1, 2, ..., 2^n. (3)

Then, E_i can be represented as:

$$E_i = E_{com\_i} + E_{excl\_i}. \quad (4)$$

In the third phase, the exponentiation of the consecutive pairs of $M^{E_{2^n}}, M^{E_{2^n-1}}, ..., M^{E_1}$ can be computed as:

$$M^{E_i} = M^{E_{com\_i}} \times M^{E_{excl\_i}}, \quad (5)$$

Exponent-Folding Algorithm
Input: M, E_i, E_{i+1}
Output: $M^{E_i}, M^{E_{i+1}}$
$C_1 = C_2 = C_3 = 1$; $S = M$;
begin
  for b = 1 ***to*** $k/2^n$ ***do*** /*scan the folding parts*/
  begin
    if $(e^b_{excl\_i} = 1)$ ***then*** $C_1 = S \times C_1 \bmod N$;
    if $(e^b_{excl\_(i+1)} = 1)$ ***then*** $C_2 = S \times C_2 \bmod N$;
    if $(e^b_{com\_i} = 1)$ ***then*** $C_3 = S \times C_3 \bmod N$;
    $S = S \times S \bmod N$;
  end;
  $C_1 = C_1 \times C_3$; $C_2 = C_2 \times C_3$;
end.

and $M^{E_{i+1}} = M^{E_{com\_i}} \times M^{E_{excl\_i+1}}$ for i = 1, 3, ...,2^n-3, 2^n-1. (6)

By the definition of the "E_i **AND** E_{i+1}" operation shown in Eq. (2), it is oblivious that this operation will directly record the common bits for every segment in E_i and E_{i+1} using logical "**AND**" operation and put this result into E_{com_i}. Meanwhile, from the definition of the "E_{com_i} **XOR** E_i" operation shown in Eq. (3), the difference bits of "comparisons between E_{com_i} and E_i" for every segment are recorded in E_{excl_i}.

The Exponent-Folding algorithm [9] proposed by Lou and Chang uses the definitions shown above to change the register C into three different registers C_1, C_2 and C_3. Here we use register C_3 to store the common part (E_{com_i}), and we use registers C_1 and C_2 to store the difference bits for "E_{com_i} **XOR** E_i" and "E_{com_i} **XOR** E_{i+1}", respectively.

The last phase of the Lou-Chang Exponent-Folding algorithm can be depicted as follows, we need compute the result of $C_s \equiv C_1^{2^{(k-\frac{k}{2^n})}} \times C_2 \pmod N$ before we output the results every time. Note that, without completing this computation procedure, instead obtaining the correct result, we will only have the results of E_i and $E_{(i+1)}$ segments stored in registers $C_1$and C_2, respectively.

On average, the hamming weights of E_i, E_{com_i} and E_{excl_i} are $k/2^{n+1}$, $k/2^{n+2}$ and $(k/2^{n+1} - k/2^{n+2})$, respectively. Eq. (2) and Eq. (3) confirm that the three "***if***…***then***…" statements in the Exponent-Folding algorithm will be true under mutually exclusive situations. Thus, in the Exponent-Folding algorithm, $\{S \times C_1, S \times S\}$, $\{S \times C_2, S \times S\}$, $\{S \times C_3, S \times S\}$ and $\{S \times S\}$ are all performed $k/2^{n+2}$ times.

Here we assume $e^b_{excl\_i}$, $e^b_{excl\_i+1}$ and $e^b_{com\_i}$ are computed in advance. Let P denotes the required number of multiplications for evaluating common multiplicand multiplications of $X \times Y$ and $X \times Z$. It is oblivious that the P is not greater than two. On average, the number of multiplications needed in the Exponent-Folding algorithm is:

$$F(P) = 2^{n-1}\{(P \times \frac{3k}{2^{n+2}}) + 1 \times \frac{k}{2^{n+2}} + 2\} + (k - (k - \frac{k}{2^n}) + (2^n - 1). \quad (7)$$

Note that, the item $2^{n-1}\{(P \times \frac{3k}{2^{n+2}}) + 1 \times \frac{k}{2^{n+2}}\}$ in Eq. (7)

Generalized Signed-Digit Recoding Algorithm
Input: $(r_{k-1}r_{k-2}...r_1r_0)_2$, $k \ge i \ge 0$, $r_i \in \{0,1,\bar{1}\}$
Output:$(e_ke_{k-1}e_{k-2}...e_1e_0)_{SDr}$,
$k \ge i \ge 0$, $e_i \in \{0,1,\bar{1},2,\bar{2},...\}$
begin
  $c_0 = 0$; $e_{k+1} = 0$; $e_k = 0$;
  for i = 0 to k ***do***
  begin
    $c_{i+1} = \lfloor (c_i + r_i + r_{i+1})/2 \rfloor$;
    $e_i = c_i + r_i - 2c_{i+1}$;
  end;
end.

equals to the constant $\{P\times\frac{3k}{2^3}+1\times\frac{k}{2^3}\}$, and won't be influenced by different n. Meanwhile, the value of the item $2^{n-1}\{2\}+(k-\frac{k}{2^n})+(2^n-1)$ in Eq. (7) becomes bigger as n grows.

Therefore, it can get the optimal case for exponentiation when $n = 1$ (i.e. by folding the exponent exact one time). When $n = 1$ and $P = 2$ in Eq. (7) for evaluating M^E, it will only need $1.375k+3$ multiplications.

For the worst case, the numbers of multiplication required in the Exponent-Folding algorithm is:
As $n = 1$ and $P = 2$, then the Exponent-Folding algorithm

$$F(P)=2^{n-1}[P\times\frac{k}{2^n}+2]+(k-\frac{k}{2^n})+(2^n-1). \quad (8)$$

takes $1.5k+3$ multiplications.

3 The Proposed New Algorithm

3.1. Signed-Digit Recoding Algorithm

The signed-digit (SD) (redundant) representations number system was first proposed by Avizienis [17] to make it possible to perform carry-free addition. Recently, many signed-digit number systems have been used to increase the efficiency of computer arithmetic [18][19][20][21] [22]. A signed digit representation of an integer a in radix r is a sequence of digits $a = (\cdots, a_2, a_1, a_0)$ with $a_i \in \{0,\pm1,\cdots,\pm(r-1)\}$.

Moreover, redundant representations of this form have been used successfully in many arithmetic applications, including the exponentiation problem in a group.

In 1993, Arno and Wheeler [23] proposed the signed-digit representations for minimal hamming weight arithmetic. For simplicity, we abuse the symbol "r" and refer to S_r as the set of all signed digit radix r representations of elements of Z. The mapping $\pi : S_r \rightarrow$ Zdefined by:

$$\pi(a)=\sum_{i=0}^{\infty}a_i r^i \quad (9)$$

associates an integer with each element $a \in S_r$.

The hamming weight of an element $a \in S_r$, denoted $w(a)$, is defined to be the number of nonzero terms in a. The original Booth recoding technique [24] scans the bits of the multiplier one bit at a time, and adds or subtracts the multiplicand to or from the partial product, depending on the current bit and the previous bit. In 2000, Joye and Yen [8] proposed new methods for producing optimal binary signed-digit representations. We generalized this signed-digit recoding arithmetic algorithm as follows.

As for signed-digit number with radix r (here r is greater than 0), assume we abuse the symbol "r", the symbols $\{\bar{r}, 0, r\}$ can be used for the digit set, in which r and $\bar{r}$ in digit position k represent $+r^k$ and $-r^k$, respectively. Based on signed-digit recoding arithmetic algorithm, the signed-digit representation can be shown as follows.

$$\begin{aligned} 2r &= (r_{k-1}, r_{k-2}, r_{k-3}, \ldots, r_1, r_0, 0)_2 \\ +\ r &= (r_{k-1}, r_{k-2}, \ldots, r_2, r_1, r_0)_2 \\ \hline 3r &= (s_k, s_{k-1}, s_{k-2}, s_{k-3}, \ldots, s_1, s_0, r_0)_2 \\ -\ r &= (r_{k-1}, r_{k-2}, \ldots, r_2, r_1, r_0)_2 \\ \hline 2r &= (e_k, e_{k-1}, e_{k-2}, e_{k-3}, \ldots, e_1, e_0, 0)_{SD_r} \end{aligned}$$

Figure 1. Signed-digit representation.

Take the SD representation with radix 2 ($r = 2$) for an example to discuss the digit occurrence probability, on average, the probability of the digit "0" appearance is "2/3", and the total occurrence probability of nonzero digits "1" and "$\bar{1}$" is "1/3" [23]. Notice that, again by abusing the symbol "r", in order to obtain signed-digit $\bar{1}$ for our signed-digit representation in Figure 1, the subtraction operation executed between $3r$ and r is a "no-borrow (carry)" subtraction. If we assume that these two possible nonzero digits (1 and $\bar{1}$) shared equal occurrence probability, then we can have the occurrence probability for each element recorded as {Pr(0) = 2/3, $\Pr(1) = \Pr(\bar{1}) = 1/6\}$.

Moreover, Joye and Yen [8] had combined the techniques of LSB algorithm and the signed-digit algorithm to produce optimal binary signed-digit representations as the LSD algorithm. The LSD algorithm scans the signed-digit recoding exponent E from the least significant digit position toward the most significant digit position to break the exponentiation operation M^E into a series of squaring and multiplication operations. Here we generalized the LSD algorithm as follows.

Generalized LSD (Right-to-Left) Algorithm
Input: Message M;
Exponent: $E = (e_k e_{k-1} \ldots e_1 e_0)_{SDr}$
where $e_i \in \{0,1,\bar{1},2,\bar{2},\ldots\}$
Output: Ciphertext: $C = M^E$
begin
$C = 1$; $S = M$;
for $i = 0$ ***to*** k-1 ***do*** /*scan from right to left*/
begin
if $(e_i = r)$ ***then*** $C = S^r \times C \bmod N$;
/* multiply by s r times*/
if $(e_i = -r)$ ***then*** $C = S^{-r} \times C \bmod N$;
/*divide by s r times*/
$S = S \times S \bmod N$; /*square*/
end;
end.

However, there exists a main difference between the LSD algorithm and the LSB algorithm, namely, the LSD algorithm must deals with signed-digit "$-r$"

element. Note that, there are two different operations needed to be done when we scanning elements "r" and "$-r$" in the generalized LSD algorithm. When we have scanned the digit element from E as "r", then the r times multiplication operation $C = S^{r} \times C$ is executed. Otherwise, if we have scanned the digit element from E as "$-r$", then the r times "division" operation $C = S^{-r} \times C$ is executed.

3.2. Definitions and Lemma

The basic idea of our proposed method is try to extract the common substring of the signed-digit recoding exponent E, and then save the number of required for the computation of common substrings. Let each substring of E be denoted as E_i for i= 1, 2, …, 2^n. Using the signed-digit representation with radix r and Exponent-Folding algorithm, we define the following variables:

$$E = (e^{\frac{k}{2^n}-1} e^{\frac{k}{2^n}-2} \ldots e^{b} \ldots e^{1} e^{0})_{\mathrm{SD}r}, \qquad (10)$$

High-Radix-Signed-Digit-Folding (HRSDF) Algorithm
Input: M, E_i, E_{i+1}
Output: $M^{E_i}, M^{E_{i+1}}$
$C_1 = C_2 = \ldots = C_r = C_{r+1} = 1;\ C' = 1;\ S = M;$
begin
 for $b = 0$ ***to*** $(k/2^n)$ ***do*** /*scan the folding parts*/
 begin
  if $(e^b_{com\_i} = r)$ ***then*** $C_r = S^r \times C' \bmod N;$
  if $(e^b_{com\_i} = -r)$ ***then*** $C_r = S^{-r} \times C' \bmod N;$
  if $(e^b_{excl\_i} = r$ ***and*** $-r \le e^b_{excl\_i+1} \le -1)$
   then{ $C_r = S^r \times C_r \bmod N;$
    $C_{r+1} = S^{-r} \times C_{r+1} \bmod N$};
  if $(e^b_{excl\_i} = r$ ***and*** $e^b_{excl\_i+1} = 0)$
  then $C_r = S^{-r} \times C_r \bmod N;$
  ⋮
  if $(e^b_{excl\_i} = 0$ ***and*** $-r \le e^b_{excl\_i+1} \le -1)$
   then $C_{r+1} = S^{-r} \times C_{r+1} \bmod N;$
  if $(e^b_{excl\_i} = 0$ ***and*** $1 \le e^b_{excl\_i+1} \le r)$
   then $C_r = S^r \times C_r \bmod N;$
  ⋮
  if $(e^b_{excl\_i} = -r$ ***and*** $1 \le e^b_{excl\_i+1} \le r)$
   then{ $C_r = S^r \times C_r \bmod N;$
    $C_{r+1} = S^{-r} \times C_{r+1} \bmod N$};
  if $(e^b_{excl\_i} = -r$ ***and*** $e^b_{excl\_i+1} = 0)$
   then $C_r = S^{-r} \times C_r \bmod N;$
  $S = S \times S;$
 end;
 $C_r = C_r \times C';$
 $C_{r+1} = C_{r+1} \times C';$
end.

$E_{com\_i} = E_{com\_i+1} = E_i$ **AND** E_{i+1} for i = 1, 3, …,$2^n - 3$, $2^n - 1$, (11)

$E_{excl\_i} = E_{com\_i}$ **XOR** E_i for i = 1, 2,.., 2^n.(12)

As $E_i = E_{com\_i} + E_{excl\_i}$, we can obtain:

$$M^{E_i} = M^{E_{com\_i}} \times M^{E_{excl\_i}}, \qquad (13)$$

$M^{E_{i+1}} = M^{E_{com\_i}} \times M^{E_{excl\_i+1}}$ for i = 1, 3, …,$2^n - 3$, $2^n - 1$. (14)

Here we detailed describe the calculation processes for the Signed-Digit-Folding method, we first define two bits "00", "0r", "r0" to represent "0", "r", and "$-r$" respectively for e^b (in Eq. (10)), as we scan the signed-digit transformed result of E exact two bits per time during folding procedure, we can therefore guarantee this transformation for exponent E will not influence the calculation result for evaluating M^E.

The E_{com_i} operation will record the common part for every two consecutive segments E_i and E_{i+1}. If the bits in the same position of two consecutive segments shared the same value, we will put this same bit value in E_{com_i}. However, if the bit values in the same position of every two consecutive segments E_i and E_{i+1} are different, we will put "00" in E_{com_i} to show the bit values in two consecutive segments are different. Here our signed-digit representation is defined with radix r and digit set is {$-r$+1,…,0,…,r-1} as defined in Section 3.1. Then, we direct record the differences for "E_i and E_{com_i}" and "E_{i+1} and E_{com_i}" in E_{excl_i} and E_{excl_i+1}, respectively.

Lemma: The relation $a^{-1} \equiv r^{-1} \pmod N$ holds as $r \equiv a \pmod N$.

By using this lemma, we know as the relation "$r \equiv a \pmod N$" holds we can get the multiplicative inverse result of $a^{-1} \pmod N$ form the result of $r^{-1} \pmod N$. Hence, as long as the relation $a = QN + r$ holds, we can have the result of $a^{-1} \equiv r^{-1} \pmod N$. Moreover, the bitwise "**AND**" operation and "**XOR**" operation defined in Eq. (11) and Eq. (12) are depicted as shown in Table 1 and Table 2, respectively.

Table 1. Truth table for bitwise operator "**AND**".
Table 2. Truth table for bitwise operator "**XOR**".

AND	-(r-1)	…	-1	0	1	…	(r-1)
-(r-1)	-(r-1)	Θ	Θ	Θ	Θ	Θ	Θ
…	Θ	…	Θ	Θ	Θ	Θ	Θ
-1	Θ	Θ	-1	Θ	Θ	Θ	Θ
0	Θ	Θ	Θ	0	Θ	Θ	Θ
1	Θ	Θ	Θ	Θ	1	Θ	Θ
…	Θ	Θ	Θ	Θ	Θ	…	Θ
(r-1)	Θ	Θ	Θ	Θ	Θ	Θ	(r-1)

XOR	1-r	…	-1	0	1	..	r-1
-(r-1)	Θ	…	-r	1-r	2-r	..	0
…	…	Θ	…	…	…	…	…
-1	-r	…	Θ	-1	0	..	r-2
0	1-r	…	-1	Θ	1	..	r-1
1	2-r	…	0	1	Θ	…	r
…	…	…	…	…	…	Θ	…
(r-1)	0	…	r-2	r-1	r	..	Θ

3.3. The Proposed HRSDF Algorithm

After we have introduced the definitions and lemma shown above. By using the signed-digit recoding technique and the exponent-folding algorithm, we now introduced a fast high-radix (r-digit recoding) signed-digit-folding modular exponentiation algorithm as follows.

This proposed HRSDF algorithm uses both the signed-digit recoding and the exponent-folding technique for speeding up the modular exponentiation. By using the lemma defined in Section 3.2, we can speed up the proposed HRSDF exponentiation algorithm and have the original HRSDF algorithm revised as follows.

Improved High-Radix-Signed-Digit-Folding (HRSDF) Algorithm
Input: M, E_i, E_{i+1}
Output: $M^{E_i}, M^{E_{i+1}}$
$C' = D' = 1$; $S = M$; $C_1 = C_2 = ... = C_r = C_{r+1} = 1$;
$D_1 = D_2 = ... = D_r = D_{r+1} = 1$
begin
 for $b = 0$ ***to*** $(k/2^n)$ ***do***
 begin
  if $(e^b_{com\_i} = r)$ ***then*** $C_r = S^r \times C' \bmod N$;
  if $(e^b_{com\_i} = -r)$ ***then*** $C_r = S^{-r} \times D' \bmod N$;
  if $(e^b_{excl\_i} = r$ ***and*** $-r \le e^b_{excl\_i+1} \le -1)$
   then { $C_r = S^r \times C_r \bmod N$;
    $D_{r+1} = S^{-r} \times D_{r+1} \bmod N$};
  if $(e^b_{excl\_i} = r$ ***and*** $e^b_{excl\_i+1} = 0)$
  then $C_r = S^{-r} \times C_r \bmod N$;
  ⋮
  if $(e^b_{excl\_i} = 0$ ***and*** $-r \le e^b_{excl\_i+1} \le -1)$
   then $D_{r+1} = S^{-r} \times D_{r+1} \bmod N$;
  if $(e^b_{excl\_i} = 0$ ***and*** $1 \le e^b_{excl\_i+1} \le r)$
   then $D_r = S^r \times D_r \bmod N$;
  ⋮
  if $(e^b_{excl\_i} = -r$ ***and*** $1 \le e^b_{excl\_i+1} \le r)$
   then { $C_r = S^r \times C_r \bmod N$;
    $D_{r+1} = S^{-r} \times D_{r+1} \bmod N$};
  if $(e^b_{excl\_i} = -r$ ***and*** $e^b_{excl\_i+1} = 0)$
   then $C_r = S^{-r} \times C_r \bmod N$;
  $S = S \times S$;
 end;
 $C_r = C_r \times C'$; $C_{r+1} = C_{r+1} \times C'$;
 $D_r = D_r \times D'$; $D_{r+1} = D_{r+1} \times D'$;
end.

To speed up the computation of multiplicative inverse, we use extra r+1 registers (D_1~D_{r+1}) to store the original S^{-1} result. By using the revised algorithm, we can transform the inverse multiplicative operations to normal multiplicative operations to avoid inverse operation at every step when we dealing with the negative number "-r".

In the improved HRSDF algorithm, we put the operation results of positive digit in the registers C_r and C_{r+1}, and we put the operation results of negative digit in the registers D_r and D_{r+1}. C_r and D_r are used to store the operation results in every E_i segment, respectively. Meanwhile, C_{r+1} and D_{r+1} are used to store the operation results in every E_{i+1} segment, respectively.

We define C_r and C_{r+1} for HRSDF algorithm when we dealing with positive digit:

$$C_r \equiv M^{E_i} (\bmod N), \quad (15)$$

$$C_{r+1} \equiv M^{E_{i+1}} (\bmod N). \quad (16)$$

We define D_r and D_{r+1} for HRSDF algorithm when we dealing with positive digit:

$$D_r \equiv M^{E_i} (\bmod N), \quad (17)$$

$$D_{r+1} \equiv M^{E_{i+1}} (\bmod N). \quad (18)$$

Let C_s and D_s store the final results. Since

HRSDF Modular Multiplication (Evaluate for positive folding-exponent part)
Input: M, E_i, E_{i+1}; **Output**: $M^{E_i[r]}, M^{E_{i+1}[r]}$
$C_1 = C_2 = \ldots = C_r = C_{r+1} = 1$; $C' = 1$; $S = M$;
begin
for $b = 0$ ***to*** $k/2^n$ ***do*** /*scan the folding parts*/
 begin
  if $(1 \le e^b_{com\_i} \le r)$ ***then***
   $C' = S \times C' \bmod N$;
  if $(1 \le e^b_{exel\_i} \le r)$ ***then***
   $C_r = S \times C_r \bmod N$;
  if $(1 \le e^b_{exel\_i+1} \le r)$ ***then***
   $C_{r+1} = S \times C_{r+1} \bmod N$;
  $S = S \times S \bmod N$;
 end;
$C_r = C_r \times C'$; $C_{r+1} = C_{r+1} \times C'$;
end.

$$M^{E_i \| E_{i+1}} \equiv (M^{E_i})^{2^{(k-\frac{k}{2^n})}} \times M^{E_{i+1}} (\bmod N), \quad (19)$$

we can define C_s for HRSDF algorithm when we have positive digits:

$$C_s \equiv C_r^{2^{(k-\frac{k}{2^n})}} \times C_{r+1} (\bmod N). \quad (20)$$

We define D_s for HRSDF algorithm when we have negative digits:

$$D_s \equiv D_r^{2^{(k-\frac{k}{2^n})}} \times D_{r+1} (\bmod N). \quad (21)$$

We here to specify this replacement will not influence the final operation result. Let

$M^{E} \equiv M^{A+(-B)} (\mathrm{mod}\ N)$

$\equiv (M^{A} \bmod N)(M^{-B} \bmod N)(\mathrm{mod}\ N)$,

$C_s \equiv M^{A} (\mathrm{mod}\ N) \equiv r_{M^A} (\mathrm{mod}\ N)$;

and $D_s \equiv M^{B} (\mathrm{mod}\ N) \equiv r_{M^B} (\mathrm{mod}\ N)$.

By using the lemma shown in Section 3.2, we can have the following:

- $a^{-1} \equiv r^{-1} (\mathrm{mod}\ N)$,
- $M^{-B} \equiv r_{M^B}^{-1} (\mathrm{mod}\ N)$,

$M^{A} \times M^{-B} (\mathrm{mod}\ N) \equiv r_{M^A} \times (r_{M^B})^{-1} (\mathrm{mod}\ N)$,

- $r_{M^A} \times (r_{M^B})^{-1} \equiv C_s \times D_s^{-1} (\mathrm{mod}\ N)$.

By using the formulas shown above, we can therefore transform the original multiplicative inverse operation into normal positive (multiplication) calculation, and use the operation result for our original multiplicative inverse calculation. Here we can speed the whole modular exponentiation process as we let the multiplicative inverse operation performed only once in the last step before our final output result for the improved HRSDF algorithm.

To further speed up the improved *r*-radix recoding signed-digt-folding algorithm, we can put the multiplicative inverse operation in the proposed algorithm over a finite filed (such as GF(2^n) when normal basis is used [25][26]). By using the computation over Galois Field, the time complexity of multiplicative inverse operation is equivalence to the bit shift operation [12]. For example, *r* times multiplicative inverse operation is then replaced by *r* bit shift operation over Galois Field and therefore the computational complexity is reduced and the total modular exponentiation operation performance can be improved.

HRSDF Modular Multiplication
(Evaluate for negative folding-exponent part)

Input: M, E_i, E_{i+1}; **Output**: $M^{E_i[\bar{r}]}, M^{E_{i+1}[\bar{r}]}$

$D_1 = D_2 = \ldots = D_r = D_{r+1} = 1;\ D' = 1; S = M;$

begin
for $b = 0$ ***to*** $k/2^n$ ***do*** /*scan the folding parts*/
begin
if $(-r \le e^b_{com\_i} \le -1)$ ***then***
$D' = S \times D' \bmod N;$
if $(-r \le e^b_{exel\_i} \le -1)$ ***then***
$D_r = S \times D_r \bmod N;$
if $(-r \le e^b_{exel\_i+1} \le -1)$ ***then***
$D_{r+1} = S \times D_{r+1} \bmod N;$
$S = S \times S \bmod N;$
end;
$D_r = D_r \times D'; D_{r+1} = D_{r+1} \times D';$
end.

3.4. The Parallel HRSDF Algorithm

Parallel HRSDF Exponentiation Algorithm
Input: M, E_i, E_{i+1};

Output: $M^{E_i[r]}, M^{E_{i+1}[r]}, M^{E_i[\bar{r}]}, M^{E_{i+1}[\bar{r}]}$

$C_1 = C_2 = \ldots = C_r = C_{r+1} = 1; D_1 = D_2 = \ldots = D_r = D_{r+1} = 1; C' = D' = 1; S = M;$

begin
for $b = 0$ ***to*** $k/2^n$ ***do*** /*scan the folding parts*/
begin
parbegin
{ ***if*** $(1 \le e^b_{com\_i} \le r)$ ***then***
$C' = S \times C' \bmod N;$
if $(1 \le e^b_{exel\_i} \le r)$ ***then***
$C_r = S \times C_r \bmod N;$
if $(1 \le e^b_{exel\_i+1} \le r)$ ***then***
$C_{r+1} = S \times C_{r+1} \bmod N;$}
{ ***if*** $(-r \le e^b_{com\_i} \le -1)$ ***then***
$D' = S \times D' \bmod N;$
if $(-r \le e^b_{exel\_i} \le -1)$ ***then***
$D_r = S \times D_r \bmod N;$
if $(-r \le e^b_{exel\_i+1} \le -1)$ ***then***
$D_{r+1} = S \times D_{r+1} \bmod N;$}
{ $S = S \times S \bmod N;$}
parend;
parbegin
{ $C_r = C_r \times C';$
$C_{r+1} = C_{r+1} \times C';$
$D_r = D_r \times D';$
$D_{r+1} = D_{r+1} \times D';$ }
parend;
end;
end.

As we can further execute the modular multiplication (when positive digits of folding-exponent are being scanned) and the multiplicative inverse (when negative digits of folding-exponent are being scanned) operations separately, we can apply the "parallel-processing" technique on the HRSDF algorithm to speed up the total exponentiation efficiency. The two separated modular multiplication and the multiplicative inverse operations of HRSDF algorithm are detailed depicted as follows.

As the modular multiplication operation and the modular squaring operation can be concurrently executed, we can have the proposed HRSDF algorithm work more efficient as a parallel version of HRSDF algorithm as follows.

4 Complexity Analyses

In this section, we will detailed describe the theoretical analyses for the performance of the proposed

parallel HRSDF algorithm. We use the number of modular multiplications to express the speed-up efficiency. Let $\kappa_r(n)$ be a random variable on the space of κ-digit radix r integers denoting the minimal signed radix r Hamming weight for the signed-digit recoding folding-exponent.

The occurrence of the digit "0" in our folding-exponent signed-digit recoding is approximately $2/(r+1)$, and the occurrence of the nonzero digits "r" and "$-r$" is close to $(r-1)^2/[r(r+1)]$ and $(r-1)/[r(r+1)]$, respectively [15]. Let k denotes the digit-length of the exponent. In the proposed parallel HRSDF method, the number of modular squaring is $2^{n-1}\{1\times\frac{k}{2^n}\}$. As we take r-radix signed-digit recoding for the folding-exponent representation, there exists total $2(r-1)$ nonzero digit numbers.

The average occurrence probability for nonzero digit is $\frac{(\frac{r-1}{r+1})}{2(r+1)}$, the computational complexity as well as the multiplication number of nonzero digit is $\frac{4r+3}{4(r+1)^2}$. The computational complexity of the proposed method is $2^{n-1}\{\frac{4r+3}{4(r+1)^2}\times\frac{k}{2^n}\}$ plus the computational complexity $\frac{k}{2}[\frac{4r+3}{4(r+1)^2}]+3$, i.e., $\frac{k}{2}[\frac{4r+3}{4(r+1)^2}+1]+3$.

Based on the above analyses, we can easily conclude that the value in item $2^{n-1}\{1\times\frac{k}{2^n}\}$ in the number of modular multiplication will not change (equal to constant $k/2$) even as the folding processing number n grows. While the total number of modular multiplication $2^n+[k-\frac{k}{2^n}(\frac{4r+3}{4(r+1)^2})]+(2^n-1)$ will grow larger as n grows.

Therefore, we can get the performance for the proposed parallel HRSDF algorithm when we fold the exponent simply one time (i.e., n=1). Since the modular squaring in finite field GF(2^n) is only a simple shift operation [20], and it does not increase more complexities for the proposed algorithm. The computational complexity for the parallel HRSDF method is then becomes:

$$2^{n-1}\{\frac{4r+3}{4(r+1)^2}\times\frac{k}{2^n}\}+(k-\frac{k}{2^n})+1=\frac{k}{2}+\frac{k}{2}[\frac{4r+3}{4(r+1)^2}]+3.$$

As the folding times increased, the number of modular multiplication item $2^{n-1}\{\frac{4r+3}{4(r+1)^2}\times\frac{k}{2^n}\}$ will not change (this value equals to constant $\frac{4r+3}{8(r+1)^2}$) even as the folding processing number n grows.

Note, as in finite field GF(2^n) when a normal basis is used, we can get the performance for the proposed parallel HRSDF algorithm when we fold the exponent simply one time. Here the computational complexities based on different radix r are analysed as follows.

If we apply the radix-2 (r=2) recoding the proposed HRSDF algorithm, as n = 1, we can have: $2^{n-1}\{(\frac{4r+3}{4(r+1)^2})\times\frac{k}{2^n}\}+2^n+(k-\frac{k}{2^n})+(2^n-1)=\frac{47}{72}k+3$ ($\approx 0.65277k+3$) multiplications.

If we apply the radix-3 recoding in the proposed HRSDF algorithm, as n = 1, we can have $\frac{79}{128}k+3$ ($\approx 0.6172k+3$) multiplications. If we apply the radix-4 recoding in HRSDF algorithm, as n = 1, we can have $\frac{119}{200}k+3$ ($\approx 0.595k+3$) multiplications. Moreover, if we apply the radix-5 recoding in HRSDF algorithm, as n = 1, we can have $\frac{167}{288}k+3$ ($\approx 0.5789k+3$) multiplications.

5 Conclusions

As we know the modular exponentiation is one of the most important operations in public-key cryptography. The modular exponentiation is more complicated and time-consuming because the modular exponentiation deals with very large operands as 512-bit to 1024-bit integers. Public-key cryptosystems often involve raising elements of some group (e.g. GF(2^n) or elliptic curves) to large powers. Therefore, an efficient software algorithm or hardware implementation of modular exponentiation operation is becoming one of the key factors affecting the best performance of public-key cryptosystems.

In this paper, a new method (HRSDF) for speeding up modular exponentiation was proposed based on the binary method, signed-digit recoding, and folding technique. When incorporating the parallel processing technique and employing the parallel implementation, we can process the squaring and multiplication concurrently. As the modular squaring operation in GF(2^n) finite field is simply a shift operation when a normal basis is used, in the proposed generalized r-radix signed-digit recoding HRSDF algorithm, we can therefore decrease the computational complexity to $2^{n-1}\{(\frac{4r+3}{4(r+1)^2})\times\frac{k}{2^n}\}+2^n+(k-\frac{k}{2^n})+(2^n-1)$ multiplications. Furthermore, if we have the folding time n=1 to minimize the total multiplication complexity, we obtain the optimal overall computational complexity as $\frac{k}{2}+\frac{k}{2}[\frac{4r+3}{4(r+1)^2}]+3$ multiplications.

6 Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments.

7 References

[1] L. Rivest, A. Shamir, and L. Adleman (1978) "A method for obtaining digital signatures and public

key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126.

[2] P. L. Montgomery (1985) "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519-521.

[3] T. Blum and C. Parr (2001) "High-radix Montgomery modular exponentiation on re-configurable hardware," *IEEE Transactions on Computers*, vol. 50, no. 7, pp. 759-764.

[4] Y. Yacobi (1990) "Exponentiating faster with addition chains," *Proceedings of EUROCRYPT' 90*, pp. 222-229.

[5] D. E. Knuth (1997) *The Art of Computer Programming, Vol. II: Seminumerical Algorithms*, 3rd edition, MA: Addison-Wesley.

[6] A. B. Premkumar (2002) "A formal framework for conversion from binary to residue numbers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 2, pp. 135-144.

[7] G Alia and E. Martinelli (2002) "Fast modular exponentiation of large numbers with large exponents," *Journal of Systems Architecture*, vol. 47, no. 14, pp. 1079-1088.

[8] Joye and S.-M. Yen (2000) "Optimal left-to-right binary signed-digit recoding," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 740-748.

[9] D.-C. Lou and C.-C. Chang (1996) "Fast exponentiation method obtained by folding the exponent in half," *Electronics Letters*, vol. 32, no. 11, pp. 984-985.

[10] D.-C. Lou and C.-C. Chang (1998) "An adaptive exponentiation method," *The Journal of Systems and Software*, vol. 42, no. 1, pp. 59-69.

[11] V. S. Dimitrov, G. A Jullien, and W. C. Miller (2000) "Complexity and fast algorithms for multiexponentiations," *IEEE Transactions on Computers*, vol. 49, no. 2, pp. 141-147.

[12] D. M. Gordon (1998) "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, no. 1, pp.129-146.

[13] W. Diffie and E. Hellmen (1976) "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654.

[14] T. ElGamal (1985) "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469-472.

[15] Song Y. Yan (2002) *Number Theory for Computing*, 2nd edition, Springer-Verlag.

[16] I. Koren (2002) *Computer Arithmetic Algorithms*, 2nd edition, A. K. Peters, Natick, MA.

[17] A. Avizienis (1961) "Signed digit number representation for fast parallel arithmetic," *IRE Transaction on Electronic Computers*, EC-10, no. 3, pp. 389-400.

[18] W. Shugang, C. Shuangching, and K. Shimizu (2002) "Fast modular multiplication using booth recoding based on signed-digit number arithmetic," *IEEE Asia-Pacific Conference on Circuits and Systems* (*APCCAS*), vol. 2, pp. 31-36.

[19] N. Besli and R.G. Deshmukh (2002) "A novel redundant binary signed-digit (RBSD) Booth's encoding," *Proceedings of the IEEE Conference on Southeast*, pp. 426-431.

[20] W. Shugang and K. Shimizu (2002) "Residue signed-digit arithmetic circuit with a complement of modulus and the application to RSA encryption processor," *Proceedings of the 9th IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 591-594.

[21] M. Syuto, E. Satake, K. Tanno, and O. Ishizuka (2002) "A high-speed binary to residue converter using a signed-digit number representation," *IEICE Transactions on Information and Systems*, vol. E85-D, no. 5, pp. 903-905.

[22] G.W. Reitwiesner (1960) "Binary Arithmetic," *Advances in Computers*, vol. 1, Academic Education Press, New York, pp. 231-308.

[23] S. Arno and F. S. Wheeler (1993) "Signed digit representations of minimal hamming weight," *IEEE Transactions on Computers*, vol. 42, no. 8, pp. 1007-1010.

[24] A. D. Booth (1951) "A signed binary multiplication technique," *Quarterly Journal Mechanics and Applied Mathematics*, vol. 4, pp. 236-240.

[25] M. A. Hasan (2001) "Efficient computation of multiplicative inverses for cryptographic applications," *Proceedings of the 15th IEEE Symposium on Computer Arithmetic,* pp. 66-72.

[26] N. Takagi, J. Yoshiki, and K. Takagi (2001) "A fast algorithm for multiplicative inversion in GF(2^n) using normal basis," *IEEE Transactions on Computers*, vol. 50, no. 5, pp. 394-398.

MIAOw: a Method to Integrate a Process Algebra with Formal Data

Gwen Salaün and Christian Attiogbé
LINA, Université de Nantes
2 rue de la Houssinière, B.P. 92208, 44322 Nantes Cedex 3, France
{salaun,attiogbe}@lina.univ-nantes.fr

Keywords: Integration Method, Process Algebra, Data Description Techniques, Formal Foundations.

Received: October 10, 2002

In this article, we propose a well-defined method to build the formal foundations underlying the design of a formal language combining a process algebra with a data specification formalism. Our approach is flexible since several languages are possibly involved in the integrated result even though a combination is only restricted to two languages. The different steps of our method are precisely enumerated. Afterwards, a concrete example of a language is formalised following the method. It integrates the CCS process algebra with the Larch algebraic specification language. Finally, we show how this new formalism can be used to specify an example of real system.

1 Introduction

All the aspects of complex software systems can neither be specified nor verified with only one approach. The joint use of several different formal methods, called *multi-formalism*, *integrated* or *mixed* specifications, is necessary for the description of such systems. The main motivation is that the different parts of software systems have to be specified with appropriate formalisms and have to be formally verified with suitable tools. We need to have at our's disposal adequate formal languages to specify the main parts of systems which are data (or static) and behavioural (or dynamic) aspects. These different parts have to be linked in a formal way. Furthermore, it is interesting to allow the developer to specify each facet of the system with suitable formalisms. Then, the specifier has to be free to choose his/her specification languages. Indeed, (s)he could prefer a formalism for different reasons: suitability with respect to the system requirements, existence of development tools, personal expertise.

In this article, we propose a method to formally combine one *process algebra* with one *formal data language*. Guidelines are precisely given to define the formal foundations underlying the construction of such a new language. Our work considers the whole family of process algebras [6] and some well-known data specification languages: algebraic specifications [14], state oriented specifications (like Z [33] and B [1]). Only basic languages (purely behavioural or purely data oriented) are considered, *i.e.* we do not take into account already combined languages like LOTOS [21]. This is justified because basic languages are more suitable to specify a precise facet of a complex system. Moreover, as the authors of [20] said: *"We believe that the best chances for a well-founded combination are with specification techniques that are well researched individually"*. The main contribution of our approach is in a first place to allow people to easily define a new specification language composed of chosen basic languages, and then to use this formalism for the specification of real case studies.

Data languages are used to abstract and model data, whereas process algebras specify dynamic aspects such as concurrency. We consider process algebras extended with value passing in order to allow communications of complex data expressed with data terms. The process algebra is called the *main* formalism (control oriented approach) because it gives the behaviour of the full specification. The syntax of the integration is formalised using a BNF-like notation, and its semantics is operational.

The material of this article issues from a series of works concerning formal combinations of basic languages [29, 31]. We claim that it is essential to have at one's disposal methods to envisage the formalisation of numerous integrated languages following an easy and systematic process. This proposal is related to Fischer's paper focusing on Z and process algebra [16]. Nevertheless, this work was dedicated to fixed languages. One of our previous works [30] suggested a similar method, but one strongly bound to a particular initial outcome (an integrated formal framework). Furthermore, this framework was not as flexible as the current proposal. Ehrig and Orejas [15] proposed an integration paradigm for system specification which provides a unified approach on a conceptual level. The key idea was to consider four different layers which correspond to different kinds of integrated views of system specification. The difference with our work is that we focus on the formal foundations and not only move ourselves at a conceptual level. Finally, Astesiano *et al.* [4] aimed at composing languages, especially a data description language and a paradigm-specific language. Their goal was the definition of languages in a component-based style, focusing on the data definition component. An example of application of

their technique is worked out in [28]. Our work is more comprehensive because we deal equally with dynamic and static aspects.

The remainder of this article is organized as follows. Section 2 describes the guidelines to be followed for the designer to formalise a new formalism combining one process algebra with one data language. Section 3 deals with *evaluation functions* which are useful to interpret data terms appearing in the dynamic specification. In Section 4, we illustrate the method with concrete languages namely the CCS process algebra and the Larch algebraic specification language. Section 5 shows how the formalised language can be used in the specification of a real case study. Some issues about the integration of a process algebra with state oriented languages are studied in Section 6. Finally, Section 7 draws up a conclusion and discusses future work.

2 Guidelines of the Method

An interest of the current work is that it makes it possible to specify the main aspects of complex systems, and it favours the flexibility of integrated formalisms. Next, we accurately detail the guidelines to be followed so that the foundations for a formal integration can be achieved easily. Our method is made up of three parts; each part is also composed of several steps.

1. Study of the process algebra. Above all, the dynamic language to be treated has to be studied thoroughly. This first part in the method is essential for learning or recalling precisely its syntax, its semantics and its communication model. Therefore, the steps are:

a. enumerating all the operators of the language (as an abstract grammar);

b. enumerating the corresponding operational semantic rules;

c. recalling the communication model.

In this part, only a basic version of the process algebra is studied. We stress that only an operational approach is regarded in this work to formalise the meaning of the dynamic specification. The language extended with value passing (existing or to be designed) is emphasized in the next part. In step b, the designer has also to set the necessary notations and variables which are next used to formalise the semantic rules. This task does not explicitly appear in the method because the semantics can be reminded without it.

2. Extension of the process algebra to consider data terms. The second part of the method consists of itemizing each construct of the process algebra, and possibly extending the syntax to take into account the data management. For each operator of the process algebra, the different steps are:

a. studying the possibility of data management;

b. formalising syntactically the behaviour enhanced with data;

c. formalising the meaning of the enhanced operator:

   1. defining the needed environment;
   2. stating the needed notations and variables;
   3. enhancing the previous inference rules (introduced in step 1.b) to manage data terms.

This part is more or less simple, depending on the existence of a value passing version of the process algebra. If this extension is not yet formalised, the designer needs to locate the syntactic extension of the operators on his/her own; otherwise (s)he can recover the existing work.

The environment is a context memorizing different informations used to define the inference rules. Its construction, as well as the notations and variables declaration, is done once and for all, and not performed for each operator of the process algebra.

There are two kinds of inference rules in the operational semantics. The first one corresponds to the construction of the environment from the agent definitions and the data specification part. The second one gives the meaning for each extended operator. For this second type, the global specification is seen as a Labelled Transitions System (LTS) whose evolution is described by the inference rules.

At the end of this part, we have at our disposal a comprehensive abstract grammar and the corresponding inference rules, that are a complete formalisation of the extended process algebra.

In this stage, the meaning bound to the data term is expressed with an *evaluation function*. The evaluation function choice is justified since it is suitable to an operational semantics, and accordingly enables us to remain in a pragmatic and executable context. The evaluation function is used to interpret data appearing in dynamic behaviours. Evaluation functions are computed from data definitions (especially algebraic axioms, or properties for the state oriented languages). The computation steps are itemized in the next part.

3. Definition of the evaluation function. This last part aims at defining the evaluation function used to interpret the data terms managed by the process algebra. The steps are:

a. extracting the needed informations from the abstract declarations;

b. defining the computation of the function;

c. defining the properties of this function.

We have at our disposal two types of evaluation function depending on the kind of data specification languages: axiomatic oriented or state oriented ones. For algebraic specifications, the evaluation function corresponds to a term rewriting function. Concerning state oriented languages, the evaluation function implies modifications of the state

space mainly composed by the properties characterizing the abstract declarations, that are variables, initialisations and invariants.

3 Evaluation Functions

Now, we focus on the way the evaluation function is defined. This section deals with the third part of the method and explains the different ways to compute evaluation functions from data specifications. In Section 4, we rather look at the dynamic aspects and interactions between behaviours and data. Here, explanations are split into two outcomes. We respectively offer several valuable insights into the evaluation function formalisation for algebraic specifications and state oriented languages (Z and B).

Algebraic specifications. Concerning algebraic specifications, a rewriting system is chosen as the evaluation function. This choice is justified since it is suitable for an operational semantics. There are two cases: either the evaluation function is already available (formalised by hand or computed in a tool) and the designer only has to study this existing work following the guidelines of part 3, or (s)he has to build the function step by step. The first case is the most widespread; indeed, languages often come with their rewriting system. Furthermore, this case is more relevant because our goal is to deal with real cases and to point the work towards pragmatic results. For both cases, the evaluation function is mainly computed from algebraic axioms appearing in the datatype declaration (step 3.a). Signatures are used to perform static semantics verification such as consistent typing. The first solution aims at using the own ordering mechanisms of the language. We illustrate this idea on two algebraic languages which are Larch and CASL.

Larch [18] is a multi-site project exploring methods, languages and tools for the practical use of formal specifications. Much of the early work was done at MIT. Larch has at its disposal a theorem prover LP [17] which is able to automatically orient equations into rewrite rules without users having to enter explicit ordering commands (step 3.b). LP provides three types of ordering mechanisms for orienting equations into rewrite rules: two registered orderings (the **dsmpos** and **noeq-dsmpos** orderings), a polynomial ordering and three "brute-force" ordering procedures. Here, we detail the registered orderings. These orderings are chosen because they are sufficient to illustrate the computation of the function. Furthermore, both next kinds of ordering mechanisms are difficult to use and produce a nonterminating set of rewrite rules. LP's registered orderings use information in a registry to orient equations. When no commutative or associative-commutative operators are involved, these orderings guarantee that the resulting rewrite rules terminate (step 3.c). There are two kinds of information in a registry: height information (relating pairs of operators) and status information (assigning relative weights to the arguments of operators with arity greater than one). The reader may refer to [17, 13] in order to have supplementary explanations about the registered ordering mechanism.

CASL [3] (Common Algebraic Specification Language) is a reasonably expressive algebraic language for specifying requirements and design for conventional software. From CASL, simpler languages (*e.g.* for interfacing with existing tools) are to be obtained by restriction. The main features of its design are as follows: many-sorted basic specifications, structured specifications, architectural specifications and specification libraries. CASL specifications have loose semantics as described in [11], but the meaning of CASL terms can be given using rewriting too. The rewriting is performed using a set of rewrite rules deduced from the CASL specifications. This can be achieved following the conceptual steps defined in [23] (step 3.b). Ringeissen and Kirchner [23] deal with the execution of CASL equational specifications with the ELAN rewrite engine [9]. Both basic and structured specifications are considered even though some restrictions are assumed (subsorting and partiality features).

About the second case for describing evaluation functions (by hand), axioms or equations have to be oriented into rewrite rules to obtain the rewriting system (step 3.b). There are two ways to compute this set of rewrite rules. It can be obtained from algebraic axioms by applying general ordering algorithms like those described in [24]. This computation is not automated and must be performed by hand. Most of these algorithms ensure termination and confluence of the computed rewriting system (step 3.c). To apply these algorithms, we restrict ourselves to the initial semantics of the datatypes because the interpretation of a set of axioms by a rewriting system has really sense only in the case of the initial model [8]. For algebraic specification languages with loose semantics, it is possible to restrict them to their initial algebra so as to simplify the process.

State oriented languages. In this part, we focus on the languages Z and B which are viewed through the language aspect (and not the methodological one). After a brief introduction of each language, the different steps enumerated in the method are studied.

B. The B method is a collection of mathematically based techniques for specification, design and implementation of software modules. Systems are modeled as a collection of interdependent abstract machines. An abstract machine is described using the Abstract Machine Notation (AMN). Now, we discuss the computation of the evaluation function from state oriented specification techniques, and especially from B abstract machines. For this issue, we inspire ourselves from the B-Book [1], and from a more recent work of Bert and Cave [7]. In the latter, the authors study several ways to build finite LTS from B abstract systems: enumeration of states, symbolic evaluation and set constraints, abstract interpretation and so on. A LTS is suitable to depict an evaluation function.

The choice of evaluation function can be based on the solutions undertaken in [7]. In this section, we define the evaluation function using the enumeration of states. This is

the easiest way to define the function, and the underlying set of states and transitions. Given a B machine, we need the set of variables, the invariant, the initialisation and the list of operations (step 3.a). The behaviour of the machine is viewed as a LTS (step 3.b). Each state denotes a set of data with values, and more precisely a finite set of variable/value couples. These couples are deduced from the variables and initialisations of the B machine. Names of operations are transition labels and symbolize the evolution from one state to another. The state space of the LTS is the set of states which satisfy the invariant. From the machine initialisation clause (depicted here using *INIT*), the initial states are deduced. In the formulas below, the symbols $\langle\rangle$ and $[]$ are the temporal operators used in the formalisation of some B dynamic aspects [2].

$$\boxed{S_0 = \{x_j \mapsto v_j \mid \langle INIT \rangle (x_j = v_j)\} \ \forall j \in 1..n}$$

Then, from each state satisfying the invariant and for each operation enabled in the initial state, the successors are the states with new values obtained by the application of the operation body (*i.e.* the corresponding generalized substitution).

$$\boxed{\begin{array}{c} S_{k+1} = \{x_j \mapsto v'_j \mid [S_k]\langle OP_i \rangle (x_j = v'_j)\} \\ \forall i \in 1..m \ \forall j \in 1..n \end{array}}$$

According to the B-Book, this rule means that if $(x_j \mapsto v_j) \in S_k$ then $(v_j, v'_j) \in \mathrm{rel}_{x_j}(OP_i)$, where $\mathrm{rel}_{x_j}(OP_i)$ is the binary relation which relates the values of x_j before and after the substitution OP_i. In the current case, the function properties are the well-known ones on transition systems: deadlock, finite states, reachability and so on (step 3.c). Here, we are mainly interested in the fact that there are no deadlocks in the transition system, since the evaluation function can always be applied (moving from one state to another applying any operation of the machine). However, it is possible that the evolution in the LTS has no effect on the data. Another important property is the finite looping of the function. The termination is ensured because the evaluation of one operation induces a single step in the transition system evolution. Last but not least, the transition system is not finite in most of cases, but this is not really a shortcoming, since the generation of the whole automaton is useless here.

Z. Z [33] is a mathematical notation based on set theory and first order predicate calculus. It uses the notion of state schema and operation schema to structure data specifications and operation specifications. A schema S is made up of a declaration part D (set of variables x_i with their types) and a predicate P on the variables. A schema named S is written like that: $S \triangleq [D \mid P]$. The semantics of a state schema is a set of bindings between the variables x_i appearing in the declaration part and their values defined with respect to the predicate part P. A state schema defines a state space. A complete specification has also an initialisation schema which gives the initial values of variables.

Following the step 3.a of our method, from a given data specification in Z, we need the state schema, the initialisation schema and the operation schemas. One has to consider a LTS associated to the Z specification (step 3.b). Since Z follows the model oriented approach, a given Z specification can be viewed as a LTS. Let *Sch* be the set of schema descriptions as mappings from variables x_i to corresponding values v_i, and let *Op* be the set of all operation schema names. The Z operation schemas defined on state schemas can be viewed as labelled transitions between a current state and a next state. Indeed, each operation via its predicate part relates the binding (x_i, v_i) to the binding (x'_i, v'_i) of the next state. Then $\langle Sch, \{\stackrel{op}{\rightarrow} \mid op \in Op\}\rangle$ is the LTS capturing the meaning of the given Z specification. The properties (step 3.c) are as defined above in the step 3.c for B.

To conclude the model oriented discussion, let us remark that the different steps followed for both languages could be gathered in a common approach. Indeed, they have the same underlying semantic model (LTS) introduced above.

4 Illustration with CCS and Larch

Now, we exploit these guidelines with a concrete example: the integration of the CCS process algebra with the Larch algebraic specification language (more precisely, for pragmatic reasons, we use the input language of the theorem prover LP [17]). Each of the above key-points is detailed and formalised to illustrate our systematic process on this mere example.

4.1 Study of the process algebra

In the first part of our method, we want to be more exhaustive than the guidelines. Thus, we do not restrict ourselves to the different steps of the first part, but we give more explanations about the treated language to make the understanding of the CCS concepts easier.

The CCS language (Calculus of Communicating Systems) was suggested by Milner [25, 26]. It relies on a very small, but expressive enough, set of operators. The different constructs of the language are gathered in the grammar of Figure 1 (step 1.a). The symbol 0 denotes an agent which has finished its behaviour. The symbol τ expresses hidden action. The prefixing '.' indicates the precedence of an action on a behaviour. The choice + allows the possible firing of two different behaviours (nondeterminism). The parallel composition | denotes the execution in parallel of different behaviours allowing interleaving and possible synchronizations. The restriction $\setminus$ enforces the synchronization between complementary actions (then the evolution is depicted with a τ action). The agent call is substituted with the behaviour of the process. The summation $\Sigma_{i \in I}\, \mathrm{P}_i$ (possibly infinite generalized choice) and the renaming [ ] are not considered in the remaining of this formalisation to make the understanding and readability of this section easier.

```
CCS-SPEC ::=   AGENT+

AGENT ::=   AGENT-ID def= BEHAVIOUR

BEHAVIOUR ::=
     0 |
     PREFIXING |
     BEHAVIOUR+BEHAVIOUR |
     BEHAVIOUR|BEHAVIOUR |
     BEHAVIOUR\{ACTION+} |
     AGENT-CALL
PREFIXING ::=   ACTION.BEHAVIOUR |
                ACTION.BEHAVIOUR |
                τ.BEHAVIOUR
AGENT-CALL ::=   AGENT-ID
```

Figure 1: CCS Grammar

The + and | symbols of the BNF-like notation must not be confused with those of the CCS notation. Therefore, the nondeterministic choice and the parallel composition of CCS are written larger than the others. AGENT-ID and ACTION are basic lexical entities.

Now, we set the used notations as well as the used variables. For each grammar rule, inference rules formally describe the corresponding behaviour. The format used for their writing is: $\frac{premises}{conclusion}$. The = symbol indicates the Boolean equality of two operands. The $\rightarrow$ symbol denotes the transition relation from a behaviour to another. Informal descriptions of variables appearing in the inference rules are gathered in Table 1.

Variable	Role
F,G	Behaviour
α	Any action (input, output, or τ)
a	Atomical action
L	Restriction set

Table 1: Variable description

The semantics of the language is given in an operational way and is available in [26] (step 1.b). We recall that a LTS is formally defined thanks to a set of states S, a set of labels L and a transition relation with type $S \times L \rightarrow S$. The semantics of the process algebra is considered following this common typing, and the next rules respect this abstract definition of LTS. Now, the inference rules are written and give the meaning of the CCS operators.

PREFIXING ::= ACTION.BEHAVIOUR

$$\frac{}{a.F \xrightarrow{a} F} \quad (1)$$

The current behaviour evolves in F by an input action.

PREFIXING ::= $\overline{\text{ACTION}}$.BEHAVIOUR

$$\frac{}{\overline{a}.F \xrightarrow{\overline{a}} F} \quad (2)$$

The current behaviour evolves in F by an output action.

PREFIXING ::= τ.BEHAVIOUR

$$\frac{}{\tau.F \xrightarrow{\tau} F} \quad (3)$$

The current behaviour evolves in F by an internal action.

BEHAVIOUR ::= BEHAVIOUR+BEHAVIOUR

$$\frac{F \xrightarrow{\alpha} F'}{F+G \xrightarrow{\alpha} F'} \quad (4)$$

If a behaviour F behaves, after the firing of an action α, as the behaviour F', then $F+G$ evolves by α in F'. The rule for the symmetrical case is omitted.

BEHAVIOUR ::= BEHAVIOUR|BEHAVIOUR

$$\frac{F \xrightarrow{\alpha} F'}{F|G \xrightarrow{\alpha} F'|G} \quad (5)$$

If a behaviour F evolves by α in F', then the parallel composition $F|G$ evolves by action α in $F'|G$. The rule for the symmetrical case, where the behaviour G evolves and not the F one, is omitted.

$$\frac{F \xrightarrow{a} F' \quad G \xrightarrow{\overline{a}} G'}{F|G \xrightarrow{\tau} F'|G'} \quad (6)$$

If a behaviour F becomes F' after the firing of the input action a, and a behaviour G becomes G' after the firing of the output action $\overline{a}$, then $F|G$ evolves in $F'|G'$ by τ. The symmetrical rule, where F does an output action and G an input action, is omitted.

BEHAVIOUR ::= BEHAVIOUR\{ACTION+}

$$\frac{F \xrightarrow{a} F' \quad a \notin L}{F\backslash L \xrightarrow{a} F'\backslash L} \quad (7)$$

If a behaviour F becomes F' by the firing of a, and the action a is not in the set L of unobservable actions, then the behaviour F becomes F' by application of a, and the same restriction set L is preserved.

$$\frac{F \xrightarrow{\overline{a}} F' \quad a \notin L}{F\backslash L \xrightarrow{\overline{a}} F'\backslash L} \quad (8)$$

If a behaviour F becomes F' by the firing of $\overline{a}$, and the action a is not in the set L of unobservable actions, then the behaviour F becomes F' by application of $\overline{a}$, and the same restriction set L is preserved. This rule is necessary because L only contains non oriented actions.

$$\frac{F \xrightarrow{\tau} F'}{F\backslash L \xrightarrow{\tau} F'\backslash L} \quad (9)$$

If a behaviour F becomes F' by the firing of τ, then the behaviour F becomes F' by application of τ. This rule is needed since τ is a possible case too. More precisely, when the rule corresponding to an operator is not written with the general action α, we should detail all the possible cases, *i.e.* for an input action, an output one and the action τ (hidden communication).

AGENT-CALL ::= AGENT-ID

$$\frac{F \xrightarrow{\alpha} F'}{AC \xrightarrow{\alpha} F'} \; AC \stackrel{def}{=} F \quad (10)$$

If F evolves in F' after the firing of α, and an agent AC is defined as a behaviour F, then the agent AC becomes F' by α.

The CCS language permits synchronous communication (step 1.c). The communication is restricted to exchanges between two agents. Furthermore, the communication is oriented. CCS allows the specification of open systems. The concurrency is defined by interleaving (and not a true concurrency). The time is only logical.

Concerning the available tools for the CCS language, we refer to CWB[1] [27] and CWB-NC[2] [10]. These model-checkers allow the verification of finite state systems (simulation, temporal formulas properties verification, equivalences and so on).

SPEC ::= **LARCH-SPEC CCS-SPEC**
LARCH-SPEC ::= *see* [18]
CCS-SPEC ::= AGENT+
AGENT ::=
  AGENT-ID $\stackrel{def}{=}$ BEHAVIOUR |
  AGENT-ID(VAR-DECL+) $\stackrel{def}{=}$ **BEHAVIOUR**
BEHAVIOUR ::=
  0 |
  PREFIXING |
  BEHAVIOUR**+**BEHAVIOUR |
  BEHAVIOUR|BEHAVIOUR |
  BEHAVIOUR\{ACTION+} |
  AGENT-CALL |
  if LARCH-PRED
    then BEHAVIOUR else BEHAVIOUR
PREFIXING ::=
  ACTION.BEHAVIOUR |
  $\overline{\text{ACTION}}$.BEHAVIOUR |
  ACTION(VAR-DECL+).BEHAVIOUR |
  $\overline{\text{ACTION}}$**(APPLICATION+).BEHAVIOUR** |
  τ.BEHAVIOUR
AGENT-CALL ::=
  AGENT-ID
  AGENT-ID(APPLICATION+)
APPLICATION ::= **OP-NAME EXPR*** |
  VAR
LARCH-PRED ::= **PRED-NAME EXPR*** |
  VAR
EXPR ::= **VAR** |
  APPLICATION

Figure 2: Extended CCS grammar

4.2 Extension of the process algebra with data terms

In order to take into account data, we interest ourselves in the extension of CCS with value passing (step 2.a). This result is well-known and fully formalised in [26]. The links (at a syntactic level) between dynamic constructs and data are located at the following levels.

- declaration and call of parameterised agents;
- input and output parameterised actions;
- condition of the *if* structure.

We highlight that, in the value passing CCS, a new operator is taken into account that is the *if* structure. Now, we enhance the CCS grammar introduced in step 1.a so that the data can be also considered in the specification (step 2.b). This leads to the grammar formalised in Figure 2.

The LARCH-SPEC part corresponds to the datatype declaration, that is written respecting the syntax of the used language (the reader may consult [18] for the detailed syntax). One line is added for the nonterminal AGENT so that the parameterised agent declaration is taken into account. For the other extended operators, one rule is added for each of them in the BEHAVIOUR part. The APPLICATION rule denotes expressions obtained by application of different operations, or variables. Operation arguments are open to be the result of another operation application. The next nonterminal symbols are basic lexical entities: VAR-DECL, VAR, SORT, OP-NAME, PRED-NAME.

The next step (2.c) consists of defining the meaning of the extended process algebra. In this part, we particularly deal with the operators managing data. This step is split into three sub-steps. The first one (sub-step 2.c.1) aims at stating the necessary environment used to memorize the informations useful for the inference rules definition. We declare the environment E as a couple composed of an evaluation function *eval* deduced from the data specification, and of a set of tuples *CCSE* for CCS agents. The set E is built from the full specification (*i.e.* both Larch sorts and CCS agents).

$$E \triangleq < eval, CCSE >$$

The computation of the evaluation function is deferred in the third part of the method. The set *CCSE* contains informations memorized during the agent declaration. More precisely, for each agent declaration, we store in this set a tuple containing the agent name (or agent constant) AC,

[1] http://www.dcs.ed.ac.uk/home/cwb/
[2] http://www.cs.sunysb.edu/~cwb/

its whole behaviour H and the list of the agent parameters (identifiers) AP.

$$CCSE \triangleq \{< AC_1, H_1, AP_1 >, \ldots, < AC_n, H_n, AP_n >\}$$

These three values, associated with each agent, are useful in presence of agent call. In such a case, the agent constant is substituted by the behaviour corresponding to this call. Moreover, for a parameterised agent, identifiers memorized during the declaration are substituted in the whole behaviour by the terms in parameter.

Afterwards, we state additionnal notations and variables (sub-step 2.c.2); the ones introduced in step 1.b are also considered here. The $\triangleq$ symbol indicates the definitional equality. $Hyp \vdash C$ is a sequent, and allows to deduce C from a set of hypotheses Hyp. One may note that the notation used to write rules and the sequent one have both the same meaning. However, the first is rather commonly used for semantic rules. The $\{\ldots\}$, $<\ldots>$ and $[\ldots]$ notations refer respectively to sets, tuples and lists with comas being used as separator between elements. The *computation* function builds an evaluation function from a data specification. In the current case, the data specification is expressed using Larch datatypes, and the evaluation function is a rewriting engine. These issues will be discussed in the last part of the method. The $Exp[T/V]$ notation depicts the substitution of a variable V by a term T in an expression Exp. New variables are declared in Table 2.

Variable	Role
Sp	Global specification
$LarchSp$	Larch specification
$CCSSp$	Extended CCS specification
$eval$	Evaluation function
$CCSE$	Set for CCS agents
E	Global environment
S,S_i	Set of sorts
Σ	Set of signatures
Ax	Set of axioms
Ag_i	Agent declaration
AT_i	Tuple for an agent
AC_i	Agent constant
x_i	Variable identifier
t_i	Larch term
p	Larch predicate
AP	Agent parameters list

Table 2: Variable description

There are two kinds of inference rules in our operational semantics (sub-step 2.c.3). The first one corresponds to the construction of the E environment from the agent definitions and the algebraic specification part. The E environment, after being completely built, is never modified. The second one gives the meaning of each extended CCS operator. For this second type, the global specification is seen as a LTS whose evolution is described by the inference rules. Both types of inference rules are distinguished next.

Agent declaration. We start with rules defining agent declarations.

`SPEC ::= LARCH-SPEC CCS-SPEC`

$$\frac{\begin{array}{c} Sp \triangleq LarchSp\ CCSSp \\ LarchSp \vdash eval \\ CCSSp \vdash CCSE \\ E \triangleq < eval, CCSE > \end{array}}{Sp \vdash E} \quad (11)$$

If a specification is composed of one part expressed with Larch, and of another one expressed with CCS using Larch value passing, and from the Larch specification part we deduce an evaluation function, and the CCS part produce a set $CCSE$, and the environment E is equal to a couple composed of $eval$ and $CCSE$, then the global specification provides the environment E.

`LARCH-SPEC ::= see [18]`

$$\frac{\begin{array}{c} LarchSp \triangleq (S, \Sigma, Ax) \\ computation(S, \Sigma, Ax) \vdash eval \end{array}}{LarchSp \vdash eval} \quad (12)$$

If a Larch specification is made up of a set of sorts S, a set of signatures Σ, and a set of axioms Ax, and if we apply the *computation* function to the set of sorts, the set of signatures, and the set of axioms, then we obtain an evaluation function, so the $eval$ function is built from the specification written in Larch.

`CCS-SPEC ::= AGENT+`

$$\frac{\begin{array}{c} CCSSp \triangleq Ag_1\ \ldots\ Ag_n \\ Ag_1 \vdash AT_1\ \ldots\ Ag_n \vdash AT_n \\ CCSE \triangleq \{AT_1, \ldots, AT_n\} \end{array}}{CCSSp \vdash CCSE} \quad (13)$$

If the CCS part is composed of a set of agent declarations, and each agent declaration Ag_i gives a tuple AT_i, and the environment $CCSE$ is made up of tuples AT_i, then the set $CCSE$ is deduced from the declaration of all CCS agents.

`AGENT ::= AGENT-ID` $\stackrel{def}{=}$ `BEHAVIOUR`

$$\frac{}{AC \stackrel{def}{=} F \vdash < AC, F, [] >} \quad (14)$$

From a non parameterised agent declaration, a tuple containing the agent constant AC, its behaviour F, and an empty list (no parameters) is built.

```
AGENT ::=
   AGENT-ID(VAR-DECL+) ≝ BEHAVIOUR
```

$$\frac{}{AC(x_1 : S_1, \ldots, x_n : S_n) \stackrel{def}{=} F \vdash < AC, F, [x_1, \ldots, x_n] >} \quad (15)$$

For each new parameterised agent, a tuple containing the agent constant AC, its behaviour F, and an ordered list AP composed of the parameter identifiers is built. The parameter sorts are not memorized since our purpose is not to verify either the well-formedness of expressions or the type correspondence. We only give a dynamic semantics and not a static one.

Semantics of the extended operators. Before explaining the inference rules representing the behaviour associated to each agent, we state some preliminaries. The set E does not appear in the premises of the next rules because we assume that it is built once and for all; therefore it could be used directly in the rule definitions. Each transition from a state to another within the LTS can include the firing of several rules. Here, we just report the added inference rules due to the data extension. All the rules introduced in step 1.b run for this step too.

```
PREFIXING ::=
   ACTION(VAR-DECL+).BEHAVIOUR
```

$$\frac{}{a(x_1 : S_1, \ldots, x_n : S_n).F \xrightarrow{a(x_1:S_1,\ldots,x_n:S_n)} F} \quad (16)$$

The current behaviour evolves in F by an input parameterised action. Similarly to non parameterised action rules of step 1.b, the firing of parameterised atomic actions induces the opening of the system with the external environment. This provides possibilities of modules composition and structuring. There is no state explosion because no values are received. If we want to simulate our specification, the presence of these constructs enforces us to compose the system with other agents to obtain a closed system.

```
PREFIXING ::=
   ACTION(APPLICATION+).BEHAVIOUR
```

$$\frac{}{\overline{a}(t_1, \ldots, t_n).F \xrightarrow{\overline{a}(t_1,\ldots,t_n)} F} \quad (17)$$

The current behaviour evolves in F by an output action.

```
BEHAVIOUR ::= BEHAVIOUR|BEHAVIOUR
```

$$\frac{F \xrightarrow{a(x_1:S_1,\ldots,x_n:S_n)} F' \quad G \xrightarrow{\overline{a}(t_1,\ldots,t_n)} G'}{F|G \xrightarrow{\tau} F'[eval(t_1)/x_1, \ldots, eval(t_n)/x_n]|G'} \quad (18)$$

If a behaviour F becomes F' after the firing of an input parameterised action, and a behaviour G becomes G' after the firing of an output parameterised action, and the environment E is composed of R and $CCSE$, then $F|G$ evolves in $F'|G'$ by τ. The input variables x_i are substituted in the behaviour F' by the terms t_i received during the communication. These terms are evaluated thanks to the *eval* function. The rule corresponding to the symmetrical case, where F does the output action and G the input one, is omitted.

```
AGENT-CALL ::=   AGENT-ID(APPLICATION+)
```

$$\frac{F[eval(t_1)/AP[1], \ldots, eval(t_n)/AP[n]] \xrightarrow{\alpha} F' \quad < AC, F, AP > \in CCSE}{AC(t_1, \ldots, t_n) \xrightarrow{\alpha} F'} \quad (19)$$

If a behaviour F, in which the parameters $AP[i]$ are substituted by the normal forms (obtained after evaluation with *eval*) of the real terms t_i, becomes F' by α, and the tuple $< AC, F, AP >$ is in the set $CCSE$, then the parameterised agent AC becomes F' by α.

```
BEHAVIOUR ::=
   if LARCH-PRED
   then BEHAVIOUR else BEHAVIOUR
```

$$\frac{F \xrightarrow{\alpha} F' \quad eval(p) = true}{\texttt{if } p \texttt{ then } F \texttt{ else } G \xrightarrow{\alpha} F'} \quad (20)$$

If F becomes F' by the firing of α, and the condition p (predicate with closed terms) is evaluated to *true*, then the conditional expression with the behaviour F joined to the block `then` becomes F' after the firing of α. The symmetrical rule, where the condition is evaluated to *false*, is omitted.

4.3 Definition of the evaluation function

Now, we focus on the way the *eval* function, only introduced abstractly up to now, is computed. Here, we recall briefly the process already discussed in Section 3. The evaluation function corresponds to a rewriting system. The evaluation function is mainly computed from the algebraic axioms appearing in the datatype declaration (step 3.a). Here, the own ordering mechanisms of the language are used (step 3.b). The LP theorem prover is able to automatically orient equations into rewrite rules, especially using registered orderings. When no commutative or associative-commutative operators are involved, this ordering guarantees that the resulting rewrite rules terminate (step 3.c).

5 Application on a Case Study: the Password Manager

In this section, we illustrate the formal foundations introduced previously on a real system: the password manager. Our single goal is to show in which way the formalised language can be practically used for a concrete specification. This example is chosen for its simplicity, its reduced size and the inclusion of static and dynamic aspects.

Informal requirements. The system to be specified is made up of several basic users, a privileged user (root) and a password manager. Basic users may modify their password. The privileged user may act as a basic user (and change his/her password), but may create also a user account (*i.e.* add a user identifier and a password). The creation of a user account unfolds as follows: (1) the user identifier is asked, and either it exists and an error is raised, or it does not and the password is asked; (2) if no errors occur the password is asked again and if it is different from the first one an error is raised, otherwise the account is created. The modification of a password by a user happens similarly to the creation case with some slight differences: (1) the user identifier must already exist; (2) the old password is asked and verified before typing the new password.

Language suitability. Languages involved in the combination (CCS and Larch) are well suited for this case study. For the static part, Larch is expressive enough for specifying the data appearing in the case study. Furthermore, Larch has a theorem prover (Larch Prover [17]) which is useful for verifying proofs on this part of the specification. For the dynamic aspects, the CCS formalism provides a set of operators sufficiently expressive for the specification of the current system. Likewise, CCS has several tools such as the model-checker CWB-NC [10]. Finally, we emphasize that CCS is the process algebra we have greater experience with.

Specification. From the analysis of this case study, we have identified two kinds of algebraic sorts: basic types and more complex ones obtained by composition of the former. We have specified datatypes to represent users (`User`) and passwords (`Pwd`). From these basic types, we have defined a set made up of user and password couple (`Spm`). This sort constitutes the static part of the password manager and is described below. The algebraic specification is straightforwardly written in the LP input language.

```
set name SPM
declare Sort Spm
declare op
 empty:  -> Spm
 add:  Spm, User, Pwd -> Spm
 modify:  Spm, User, Pwd -> Spm
 is_declared:  Spm, User -> Bool
 is_correct:  Spm, User, Pwd -> Bool
..
assert sort Spm
 generated by empty, add;
declare var
 s:  Spm
 u, u1, u2:  User
 p, p1, p2:  Pwd
..
assert
 modify(empty,u,p) =
  add(empty,u,p);
 eq_user(u1,u2) =>
  modify(add(s,u1,p1),u2,p2) =
   add(s,u1,p2);
 ~eq_user(u1,u2) =>
  modify(add(s,u1,p1),u2,p2) =
   add(modify(s,u2,p2),u1,p1);
 is_declared(empty,u) = false;
 eq_user(u1,u2) =>
  is_declared(add(s,u1,p1),u2) =
   true;
 ~eq_user(u1,u2) =>
  is_declared(add(s,u1,p1),u2) =
   is_declared(s,u2);
 is_correct(empty,u,p) = false;
 eq_user(u1,u2) =>
  is_correct(add(s,u1,p1),u2,p2) =
   eq_pwd(p1,p2);
 ~eq_user(u1,u2) =>
  is_correct(add(s,u1,p1),u2,p2) =
   is_correct(s,u2,p2);
..
```

About the dynamic part, we illustrate with the password manager specification; basic user and root behaviours are not worth introducing herein. The `Dpm` agent has two possible behaviours which are the password creation and modification. It has as parameter the set of data memorizing users and their passwords. A scenario of an account creation is described in Figure 3.

```
Dpm(st_man:Spm) =def
 createAccount(u:User).
 if is_declared(st_man,u)
  then error.Dpm(st_man)
  else ok.givePwd(p1:Pwd).
  givePwd(p2:Pwd).if eq_pwd(p1,p2)
   then ok.Dpm(add(st_man,u,p1))
   else error.Dpm(st_man)
 + modifyPwd(u:User).
 if ~ is_declared(st_man,u)
  then error.Dpm(st_man)
  else ok.giveOldPwd(p:Pwd).
  if is_correct(st_man,u,p)
   then ok.givePwd(p1:Pwd).
   givePwd(p2:Pwd).if eq_pwd(p1,p2)
    then ok.Dpm(modify(st_man,u,p1))
    else error.Dpm(st_man)
   else error.Dpm(st_man)
```

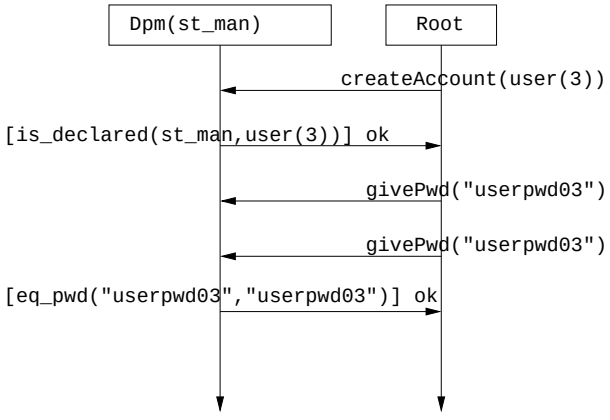


Figure 3: Scenario for an account creation

Verification aspects. Some properties have been proved on this specification, and we give insights into possibilities of proofs that we perform on the password manager specification. First of all, the LP interactive theorem proving

system allows the specifier to load and parse the specification to verify the syntactic correctness, to create rewrite rules using ordering mechanisms and to prove theorems.

The CWB-NC automatic verification tool cannot be used straightforwardly. We should translate our specification into the CCS input language of CWB-NC. This implies an abstraction of our specification by a syntactic and semantic restriction. Concerning the syntax, this abstraction leads to removing of the value passing of the extended CCS and the *if* conditional structure. Moreover, we have to respect the precise syntax of the tool detailed in [10]. Concerning the semantics, we do not take into account the rules with algebraic value passing. Then, verification on the CCS specification part consists of simulation, finding deadlock test and temporal formula written with the mu-calculus and the CTL operators (liveness and safety properties) such as:

```
prop can_modifyPwd =
   min X = <modifyPwd>tt \/ <->X
```

This property verifies that the action `modifyPwd` is performed after a finite number of steps. To complement these partial proofs, we have developed a tool (ISA [5]) which allows the animation of specifications written in value passing CCS. This tool makes it possible to manage data potentially written with any considered data language for which an evaluation function exists. This function is computed as an external module called by ISA. We have also experimented the encoding of this kind of integrated languages into the input formats of higher-order logic tools [32]. Our approach has focused on the formalisation of such languages into PVS [12] and particularly on process equivalence proofs in this homogeneous context.

Links with the formal foundations. To reinforce the correspondence between the concrete use of the integrated language and its formal basis, we show how both are linked through an actual piece of specification. More precisely, we instantiate the inference rule giving the semantics of the last synchronization in Figure 3 (*i.e.* on the `ok` action). First of all, we instantiate the `st_man` parameter with a real value.

```
st_man =
   add(add(empty,user(1),"userpwd01"),
      user(2),"userpwd02")
```

The semantic rule for the synchronization (the first one below) needs the rule for the conditional structure (the second one below) to give sense to the first premise of the first rule. Both rules respectively correspond to the rules (6) and (20) formalised in Section 4. Besides, the rewriting function substitutes the generic evaluation function in this last rule using the following instantiation notation: $eval \equiv \rightsquigarrow_R^*$ (pages 17–18 of [17]), where R is the set of rewrite rules. Thus, we stress that in the second rule, the behaviour `eq_pwd("userpwd03","userpwd03")` is rewritten as *true*.

$$\frac{\begin{array}{c}\texttt{if eq\_pwd}(''\texttt{userpwd03}'',''\texttt{userpwd03}'')\\ \texttt{then } \overline{\texttt{ok}}.\texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\\ \texttt{else } \overline{\texttt{error}}.\texttt{Dpm}(\texttt{st\_man}) \xrightarrow{\overline{\texttt{ok}}}\\ \texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\end{array} \quad \texttt{ok.Root} \xrightarrow{\texttt{ok}} \texttt{Root}}{\begin{array}{c}\texttt{if eq\_pwd}(''\texttt{userpwd03}'',''\texttt{userpwd03}'')\\ \texttt{then } \overline{\texttt{ok}}.\texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\\ \texttt{else } \overline{\texttt{error}}.\texttt{Dpm}(\texttt{st\_man}) \mid \texttt{ok.Root} \xrightarrow{\tau}\\ \texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}'')) \mid \texttt{Root}\end{array}}$$

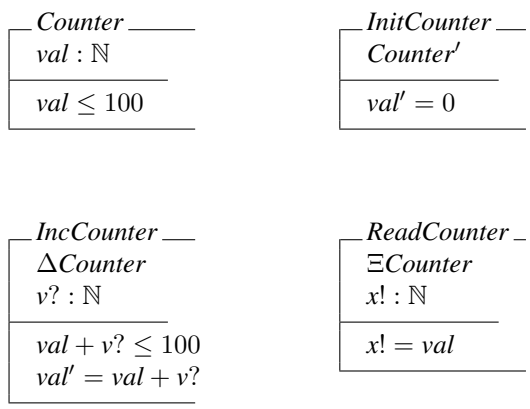
$$\frac{\begin{array}{c}\texttt{if eq\_pwd}(''\texttt{userpwd03}'',''\texttt{userpwd03}'')\\ \texttt{then } \overline{\texttt{ok}}.\texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\\ \texttt{else } \overline{\texttt{error}}.\texttt{Dpm}(\texttt{st\_man}) \xrightarrow{\overline{\texttt{ok}}}\\ \texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\\ \texttt{eq\_pwd}(''\texttt{userpwd03}'',''\texttt{userpwd03}'') \rightsquigarrow_R^* \mathit{true}\end{array}}{\begin{array}{c}\texttt{if eq\_pwd}(''\texttt{userpwd03}'',''\texttt{userpwd03}'')\\ \texttt{then } \overline{\texttt{ok}}.\texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\\ \texttt{else } \overline{\texttt{error}}.\texttt{Dpm}(\texttt{st\_man}) \xrightarrow{\overline{\texttt{ok}}}\\ \texttt{Dpm}(\texttt{add}(\texttt{st\_man},\texttt{user}(3),''\texttt{userpwd03}''))\end{array}}$$

6 Issues with the Model Oriented Integration

In this section, we discuss issues to integrate a process algebra with a state oriented language. Indeed, Section 4 deals with algebraic specifications as data formalism. Accordingly, we give hints and ideas to perform such a combination. The topic of interest is interactions between the process algebra and the data language. Similarly to algebraic specifications, the value passing is an easy way to express these links; as a consequence, we apply the same approach of formalisation in this section. However, there are differences between these families of language.

Local data will be defined using the process declaration parameters. Recursion will make modification of local data possible (with side effect) or just refer to the unchanged state space. Care must be taken while dealing with communication since output parameters are not simple terms but state description (that is a complete set of variables and value bindings). Then, the output parameters (operation applications) should express modification of the state space and emission of data. That behaviour could be also split into two sequential actions. Input events could receive data expressed using variables with types declared in the initial data language. Operation returning a Boolean could be used to write conditions in guards or conditional structures. Another case is the use of the data language syntax to express conditions on local variables.

Now, we illustrate these general ideas on a real combination between Z and CSP [19]. Several proposals of formal combination between process algebras and Z have been already suggested [16, 22]. Nevertheless, we just aim at explaining one way to integrate these languages through a short piece of specification. The Z schemas model a constrained counter using a natural number. Two operations are defined: the incrementation (*IncCounter*) and the consultation (*ReadCounter*).



Afterwards, we specify two processes using a CSP-like notation. The first one (`Controller`) has two possible behaviours: either it increments the counter with a received natural number *vv* (and consequently updates its local data), or it synchronizes itself with the other process exchanging the current value of the counter. The process `Tester` receives the current value of the controller and tests if the counter is greater or equal to 50. Depending on the test result, its behaviour stops or continues.

```
Controller(Counter) ≙
 receiveVal?vv : ℕ ->
  inc(IncCounter[vv/v?]) ->
   Controller(θCounter)
 □ read(ReadCounter[x!/y]) -> comm!y ->
   Controller(θCounter)

Tester ≙
 comm?z : ℕ ->
  (z ≥ 50 -> stop | z < 50 -> Tester)
```

Interactions with data are located at different levels: process parameters, event parameters and guard condition. Processes are possibly parameterised with the schema type (*Counter* in the `Controller` process). Recursion is parameterised with the current state space which is the current binding between variables and their values (denoted with *θCounter*). The Z input and output variables are expressed as usual using the ? and ! notations. Conditions are expressed using Z comparison operators with local or received variables. The last interaction between Z data and CSP undertakes the modification of local data. Operation calls are defined as a parameter to a CSP event (*e.g.* `inc(`*IncCounter*`[`*vv*`/`*v?*`])`). It encompasses the related modification of the bound state space. Substitution is used to perform the links between the input and output variables of Z operations and the ones appearing in process definitions.

We underscore that the way to write syntactic interactions between both languages is the single difficulty to the construction of the integrated formalism. Apart from that, following the guidelines introduced in Section 2 the remainder of the formalisation is pretty straightforward, and could be achieved without too much effort.

7 Concluding Remarks

We claim that there is a lack of methods for designers to build their own integrated languages. The need for methods is really important because with this kind of methods they can obtain formalisms well suited to a precise system to be specified. In this article, we propose a method to build integrated languages constituted of a process algebra and a formal data oriented language. The method caters for the initial motivations. Thus, the new language is especially adapted to model complex systems because it covers the main aspects (static and dynamic) involved in these systems. Let us remark that in this work, we suggest a precise kind of formalisation although we know that other ways exist to reach this goal. Here, the proposed method and formalisation approach are strongly bound.

Although we have started studying verification means on integrated specifications, we should continue our effort in this direction. For instance, an ongoing work is the extension of the ISA tool in a modular and generic way to take into account other dynamic specification languages and to perform property verification (model checking). Likewise, the encoding approach in PVS has to be pushed on. We should undertake more general proofs (not only equivalence ones) on integrated specifications embedded into PVS.

References

[1] Abrial, J. R. 1996, *The B-Book*, Cambridge University Press.

[2] Abrial, J.-R. Mussat, L. 1998, Introducing Dynamic Constraints in B, *in* D. Bert, ed., 'Proceedings of the 2nd International B Conference (B'1998)', Vol. 1393 of *Lecture Notes in Computer Science*, Springer-Verlag, France, pp. 83–128.

[3] Astesiano, E., Bidoit, M., Kirchner, H., Krieg-Brückner, B., Mosses, P. D., Sannella, D. Tarlecki, A. 2002, 'CASL: The Common Algebraic Specification Language', *Theoretical Computer Science* **286**(2), 153–196.

[4] Astesiano, E., Cerioli, M. Reggio, G. 2000, Plugging Data Constructs into Paradigm-Specific Languages: towards an Application to UML, *in* T. Rus, ed., 'Proceedings of the 8th International Conference on Algebraic Methodology and Software Technology (AMAST'00)', Vol. 1816 of *Lecture Notes in Computer Science*, Springer-Verlag, USA, pp. 273–292.

[5] Attiogbé, C., Francheteau, A., Limousin, J. Salaün, G. n.d., ISA, a tool for Integrated Specifications Animation. Available at `http://www.sciences.univ-nantes.fr/info/perso/permanents/salaun/ISA/isa.html`.

[6] Bergstra, J., Ponse, A. Smolka, S., eds 2001, *Handbook of Process Algebra*, Elsevier.

[7] Bert, D. Cave, F. 2000, Construction of Finite Labelled Transition Systems from B Abstract Systems, *in* W. Grieskamp, T. Santen B. Stoddart, eds, 'Proceedings of the Second International Conference on Integrated Formal Methods (IFM'00)', Vol. 1945 of *Lecture Notes in Computer Science*, Springer-Verlag, Germany, pp. 235–254.

[8] Bidoit, M. 1989, Pluss, un langage pour le développement de spécifications algébriques modulaires, PhD Thesis, Université de Paris-Sud – Centre d'Orsay.

[9] Borovanský, P., Kirchner, C., Kirchner, H., Moreau, P.-E. Ringeissen, C. 1998, An Overview of ELAN, *in* C. Kirchner H. Kirchner, eds, 'Proceedings of the International Workshop on Rewriting Logic and its Applications (WRLA'98)', Vol. 15 of *Electronic Notes in Theoretical Computer Science*, Elsevier Science, France.

[10] Cleaveland, R., Li, T. Sims, S. 2000, *The Concurrency Workbench of the New Century (Version 1.2)*, Department of Computer Science, North Carolina State University.

[11] CoFI Semantics Task Group 2000, CASL – The CoFI Algebraic Specification Language – Semantics (version 1.0). Note S-9 in [**?**].

[12] Crow, J., Owre, S., Rushby, J., Shankar, N. Srivas, M. 1995, A Tutorial Introduction to PVS, *in* 'Proceedings of the Workshop on Industrial-Strength Formal Specification Techniques (WIFT'95)', Computer Science Laboratory, SRI International, USA.

[13] Dershowitz, N. 1982, 'Orderings for Term-Rewriting Systems', *Theoretical Computer Science* **17**(3), 279–301.

[14] Ehrig, H. Mahr, B. 1985, *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, Vol. 6 of *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, New-York.

[15] Ehrig, H. Orejas, F. 1998, 'Integration Paradigm for Data Type and Process Specification Techniques', *Bulletin of the European Association for Theoretical Computer Science* **65**, 90–97. Columns: Formal Specification.

[16] Fischer, C. 1998, How to combine Z with a process algebra, *in* J. P. Bowen, A. Fett M. G. Hinchey, eds, 'Proceedings of the 11th International Conference of Z Users (ZUM'98)', Vol. 1493 of *Lecture Notes in Computer Science*, Springer-Verlag, Germany, pp. 5–23.

[17] Garland, S. J. Guttag, J. V. 1991, A Guide to LP, the Larch Prover, Technical report, Palo Alto, California.

[18] Guttag, J. V., Horning, J. J., Garland, S. J., Jones, K. D., Modet, A. Wing, J. M. 1993, *Larch: Languages and Tools for Formal Specification*, Texts and Monographs in Computer Science, Springer-Verlag.

[19] Hoare, C. A. R. 1985, *Communicating Sequential Processes*, Prentice-Hall.

[20] Hoenicke, J. Olderog, E.-R. 2002, Combining Specification Techniques for Processes, Data and Time, *in* M. Butler, L. Petre K. Sere, eds, 'Proceedings of the Third International Conference on Integrated Formal Methods (IFM'02)', Vol. 2335 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 245–266.

[21] ISO 1989, LOTOS: a Formal Description Technique based on the Temporal Ordering of Observational Behaviour, Technical Report 8807, International Standards Organisation.

[22] Jones, C. B., Cooke, D. J. Wing, J. M., eds 2002, *Formal Aspects of Computing*, Vol. 13(2), Springer.

[23] Kirchner, H. Ringeissen, C. 2000, Executing CASL Equational Specifications with the ELAN Rewrite Engine. Note T-9 in [**?**].

[24] Klop, J. W. 1992, Term Rewriting Systems, *in* S. Abramsky, D. M. Gabbay T. S. E. Maibaum, eds, 'Handbook of Logic in Computer Science', Vol. 2, Oxford University Press, Oxford, chapter 1, pp. 1–117.

[25] Milner, R. 1980, 'A Calculus of Communicating Systems', *Lecture Notes in Computer Science* **92**.

[26] Milner, R. 1989, *Communication and Concurrency*, International Series in Computer Science, Prentice Hall.

[27] Moller, F. Stevens, P. 1999, *The Edinburgh Concurrency Workbench (Version 7.1)*, Laboratory for Foundations of Computer Science, University of Edinburgh.

[28] Reggio, G. Repetto, L. 2000, Casl-Chart: A Combination of Statecharts and of the Algebraic Specification Language Casl, *in* T. Rus, ed., 'Proceedings of the 8th International Conference on Algebraic Methodology and Software Technology (AMAST'00)', Vol. 1816 of *Lecture Notes in Computer Science*, Springer-Verlag, USA, pp. 243–257.

[29] Salaün, G., Allemand, M. Attiogbé, C. 2001, Formal Framework for a Generic Combination of a Process Algebra with an Algebraic Specification Language: an Overview, *in* 'Proceedings of the 8th Asia-Pacific Software Engineering Conference (APSEC'01)', IEEE Computer Society Press, China, pp. 299–302.

[30] Salaün, G., Allemand, M. Attiogbé, C. 2002*a*, A Method to Combine any Process Algebra with an Algebraic Specification Language: the π-Calculus Example, *in* 'Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02)', IEEE Computer Society Press, England, pp. 385–390.

[31] Salaün, G., Allemand, M. Attiogbé, C. 2002*b*, Specification of an Access Control System with a Formalism Combining CCS and CASL, *in* 'Proceedings of the 7th International Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA'02)', IEEE Computer Society Press, USA.

[32] Salaün, G. Attiogbé, C. 2003, Formalising an Integrated Language in PVS, *in* J. S. Dong J. Woodcock, eds, 'Proc. of the 5th International Conference on Formal Engineering Methods (ICFEM'03)', Vol. 2885, Springer-Verlag, Singapore, pp. 187–205.

[33] Spivey, J. M. 1992, *The Z Notation: A Reference Manual*, 2nd edn, Prentice Hall International Series in Computer Science.

ERRATA CORRIGE

Upon publication of the following paper:
P.Rocchi – "Unifying the Interpretation of Redundant Information" ***Informatica*** **Volume 28, Number 1 [91-94] (2004)**
We discovered that some equations appear unintelligible due to the involuntary switching of the symbol formats in the electronic text.
The following are corrections to the affected equations.

$$\varepsilon_i \ \gamma \ \varepsilon_j \qquad i, j = 1,2,..n \qquad (2.1)$$

$$\{\varepsilon_u\} \ 3 \ \{\varepsilon_z\} = \varnothing \qquad (3.5)$$

$$\varepsilon_u \ \gamma \ \varepsilon_z \qquad (3.6)$$

$$R > 0 \quad \omega \quad L > L_\alpha \quad Bf\,2 \qquad (4.3)$$

$$R_C \ f \ 2 \qquad (5.1)$$

$$\{\varepsilon\} = [\{\varepsilon_z\}3\{\varepsilon_{zC}\}] \ 4 \ [\{\varepsilon_u\}3\{\varepsilon_{uC}\}] \qquad (5.3)$$

$$\{\varepsilon_u\}3\{\varepsilon_{zC}\} = \{\varepsilon_u\}$$
$$\{\varepsilon_z\}3\{\varepsilon_{uC}\} = \{\varepsilon_z\} \qquad (5.4)$$

$$\{\varepsilon_u\}4\{\varepsilon_z\} = \{\varepsilon\} \qquad (5.5)$$

We apologize for the inconvenience.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 425 1038
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Natalija Polenec

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://ai.ijs.si/informatica/

Referees:

Witold Abramowicz, David Abramson, Adel Adi, Kenneth Aizawa, Suad Alagić, Mohamad Alam, Dia Ali, Alan Aliu, Richard Amoroso, John Anderson, Hans-Jurgen Appelrath, Iván Araujo, Vladimir Bajič, Michel Barbeau, Grzegorz Bartoszewicz, Catriel Beeri, Daniel Beech, Fevzi Belli, Simon Beloglavec, Sondes Bennasri, Francesco Bergadano, Istvan Berkeley, Azer Bestavros, Andraž Bežek, Balaji Bharadwaj, Ralph Bisland, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Pavel Brazdil, Bostjan Brumen, Jerzy Brzezinski, Marian Bubak, Davide Bugali, Troy Bull, Sabin Corneliu Buraga, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Rajkumar Bvyya, Giacomo Cabri, Netiva Caftori, Particia Carando, Robert Cattral, Jason Ceddia, Ryszard Choras, Wojciech Cellary, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, Mel Ó Cinnéide, David Cliff, Maria Cobb, Jean-Pierre Corriveau, Travis Craig, Noel Craske, Matthew Crocker, Tadeusz Czachorski, Milan Češka, Honghua Dai, Bart de Decker, Deborah Dent, Andrej Dobnikar, Sait Dogru, Peter Dolog, Georg Dorfner, Ludoslaw Drelichowski, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Marjan Družovec, Jozo Dujmović, Pavol Ďuriš, Amnon Eden, Johann Eder, Hesham El-Rewini, Darrell Ferguson, Warren Fergusson, David Flater, Pierre Flener, Wojciech Fliegner, Vladimir A. Fomichov, Terrence Forgarty, Hans Fraaije, Stan Franklin, Violetta Galant, Hugo de Garis, Eugeniusz Gatnar, Grant Gayed, James Geller, Michael Georgiopolus, Michael Gertz, Jan Goliński, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Jozsef Gyorkos, Marten Haglind, Abdelwahab Hamou-Lhadj, Inman Harvey, Jaak Henno, Marjan Hericko, Henry Hexmoor, Elke Hochmueller, Jack Hodges, Doug Howe, Rod Howell, Tomáš Hruška, Don Huch, Simone Fischer-Huebner, Zbigniew Huzar, Alexey Ippa, Hannu Jaakkola, Sushil Jajodia, Ryszard Jakubowski, Piotr Jedrzejowicz, A. Milton Jenkins, Eric Johnson, Polina Jordanova, Djani Juričič, Marko Juvancic, Sabhash Kak, Li-Shan Kang, Ivan Kapustøk, Orlando Karam, Roland Kaschek, Jacek Kierzenka, Jan Kniat, Stavros Kokkotos, Fabio Kon, Kevin Korb, Gilad Koren, Andrej Krajnc, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Sofiane Labidi, Les Labuschagne, Ivan Lah, Phil Laplante, Bud Lawson, Herbert Leitold, Ulrike Leopold-Wildburger, Timothy C. Lethbridge, Joseph Y-T. Leung, Barry Levine, Xuefeng Li, Alexander Linkevich, Raymond Lister, Doug Locke, Peter Lockeman, Vincenzo Loia, Matija Lokar, Jason Lowder, Kim Teng Lua, Ann Macintosh, Bernardo Magnini, Andrzej Małachowski, Peter Marcer, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Daniel Memmi, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Armin R. Mikler, Gautam Mitra, Roland Mittermeir, Madhav Moganti, Reinhard Moller, Tadeusz Morzy, Daniel Mossé, John Mueller, Jari Multisilta, Hari Narayanan, Jerzy Nawrocki, Rance Necaise, Elzbieta Niedzielska, Marian Niedq'zwiedziński, Jaroslav Nieplocha, Oscar Nierstrasz, Roumen Nikolov, Mark Nissen, Jerzy Nogieć, Stefano Nolfi, Franc Novak, Antoni Nowakowski, Adam Nowicki, Tadeusz Nowicki, Daniel Olejar, Hubert Österle, Wojciech Olejniczak, Jerzy Olszewski, Cherry Owen, Mieczyslaw Owoc, Tadeusz Pankowski, Jens Penberg, William C. Perkins, Warren Persons, Mitja Peruš, Fred Petry, Stephen Pike, Niki Pissinou, Aleksander Pivk, Ullin Place, Peter Planinšec, Gabika Polčicová, Gustav Pomberger, James Pomykalski, Tomas E. Potok, Dimithu Prasanna, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Vojislav D. Radonjic, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Kai Rannenberg, Wolf Rauch, Peter Rechenberg, Felix Redmill, James Edward Ries, David Robertson, Marko Robnik, Colette Rolland, Wilhelm Rossak, Ingrid Russel, A.S.M. Sajeev, Kimmo Salmenjoki, Pierangela Samarati, Bo Sanden, P. G. Sarang, Vivek Sarin, Iztok Savnik, Ichiro Satoh, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Dennis Sewer, Zhongzhi Shi, Mária Smolárová, Carine Souveyet, William Spears, Hartmut Stadtler, Stanislaw Stanek, Olivero Stock, Janusz Stokłosa, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Leon Strous, Ron Sun, Tomasz Szmuc, Zdzislaw Szyjewski, Jure Šilc, Metod Škarja, Jiři Šlechta, Chew Lim Tan, Zahir Tari, Jurij Tasič, Gheorge Tecuci, Piotr Teczynski, Stephanie Teufel, Ken Tindell, A Min Tjoa, Drago Torkar, Vladimir Tosic, Wieslaw Traczyk, Denis Trček, Roman Trobec, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Tadeusz Usowicz, Romana Vajde Horvat, Elisabeth Valentine, Kanonkluk Vanapipat, Alexander P. Vazhenin, Jan Verschuren, Zygmunt Vetulani, Olivier de Vel, Didier Vojtisek, Valentino Vranić, Jozef Vyskoc, Eugene Wallingford, Matthew Warren, John Weckert, Michael Weiss, Tatjana Welzer, Lee White, Gerhard Widmer, Stefan Wrobel, Stanislaw Wrycza, Tatyana Yakhno, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Ales Zivkovic, Zonling Zhou, Robert Zorc, Anton P. Železnikar

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at USA: http://, Europe: http://ai.ijs.si/informatica, Asia: http://www.comp.nus.edu.sg/ liuh/Informatica/index.html.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2004 (Volume 28) is
– USD 80 for institutions,
– USD 40 for individuals, and
– USD 20 for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, Kranj, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry of Education, Science and Sport, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 28 Number 2 July 2004 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **28** (2004) Number 2, pp. 117–223