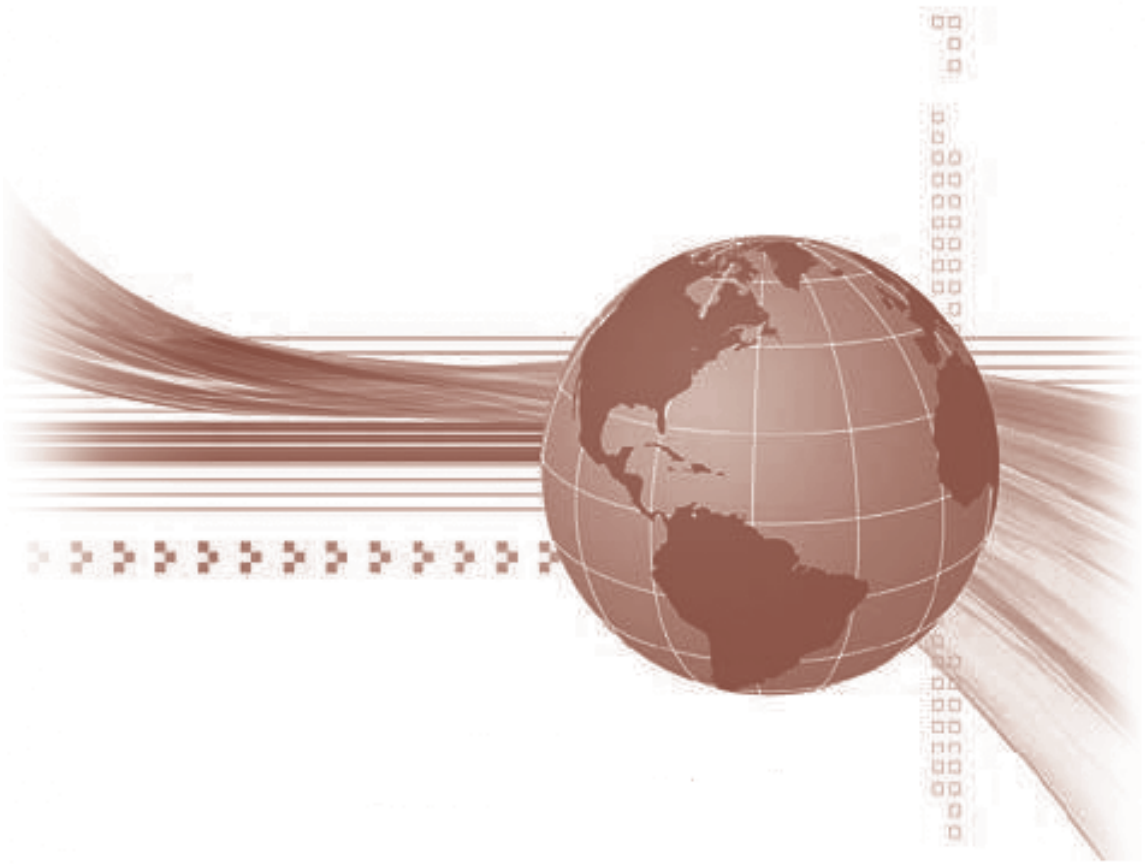
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

2/2010

Anul LV **2010**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

2

Redacția: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

ADAPTIVE RESTRUCTURING OF OBJECT-ORIENTED SOFTWARE SYSTEMS

ISTVAN GERGELY CZIBULA AND GABRIELA CZIBULA

Abstract. In this paper we approach the problem of adaptive refactoring, the process of adapting the class structure of a software system when new application classes are added. We have previously introduced an adaptive clustering method that deals with the evolving structure of any object oriented application. The aim of this paper is to extend the evaluation of the proposed method on the open source case study JHotDraw, emphasizing, this way, the potential of our approach.

1. Introduction

Improving the software systems design through refactoring is one of the most important issues during the evolution of object oriented software systems. Refactoring aims at changing a software system in such a way that it does not alter the external behavior of the code, but improves its internal structure. We have introduced in [1] a clustering approach, named *CARD* (*Clustering Approach for Refactorings Determination*), for identifying refactorings that would improve the class structure of a software system.

But real applications evolve in time, and new application classes are added in order to met new requirements. Consequently, restructuring of the modified system is needed to keep the software structure clean and easy to maintain. Obviously, for obtaining the restructuring that fits the new applications classes, the original restructuring scheme can be applied from scratch on the whole extended system. However, this process can be inefficient, particularly for large software systems. That is why we have proposed in [2] an adaptive method that deals with the evolving application classes set. The proposed method extends our original approach previously introduced in [1].

Received by the editors: January 10, 2009.

2010 *Mathematics Subject Classification.* 62H30, 68N99.

1998 *CR Categories and Descriptors.* I.5.3 [**Computing Methodologies**]:Pattern Recognition – *Clustering*; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Restructuring, reverse engineering, and reengineering*.

Key words and phrases. Software engineering, refactoring, clustering.

The rest of the paper is structured as follows. Our clustering based approach for adaptive refactorings identification is described in Section 2. For the adaptive process, a *Core Based Adaptive Refactoring* algorithm, named *CBAR*, is proposed. Section 3 indicates several existing approaches in the direction of automatic refactorings identification. An evaluation of $CBAR$ on the open source case study JHotDraw is provided in Section 4. Some conclusions of the paper and further research directions are given in Section 5.

2. Core Based Adaptive Refactoring. Background

In the following we will briefly review our clustering approach from [2] for adapting the class structure of a software system when it is extended with new applications classes.

2.1. **Initial Restructuring Phase.** We have introduced in [1] a clustering approach, named *CARD*, for identifying refactorings that would improve the class structure of a software system. First, the existing software system is analyzed in order to extract from it the relevant entities: classes, methods, attributes and the existing relationships between them: inheritance relations, aggregation relations, dependencies between the entities from the software system. After data was collected, the set of entities extracted at the previous step are re-grouped in clusters using a clustering algorithm. The goal of this step is to obtain an improved structure of the existing software system. The last step is to extract the refactorings which transform the original structure into an improved one, by comparing the newly obtained software structure with the original one.

For re-grouping entities from the software system, a vector space model based *clustering* algorithm, named *kRED* (*k-means for REfactorings Determination*), was introduced in [1].

In the proposed approach, the objects to be clustered are the elements (called *entities*) from the considered software system, i.e., $\mathcal{S} = \{e_1, e_2, \ldots, e_n\}$, where $e_i, 1 \leq i \leq n$ can be an application class, a method of a class or an attribute of a class. Each entity is measured with respect to a set of l features (l representing the number of application classes from $\mathcal{S}$), $\mathcal{A} = \{C_1, C_2, \ldots, C_l\}$, and is therefore described by an l-dimensional vector: $e_i = (e_{i1}, e_{i2}, \ldots, e_{il}), e_{ik} \in \Re, 1 \leq i \leq n, 1 \leq k \leq l$. The *distance* between two entities e_i and e_j from the software system $\mathcal{S}$ is computed as a measure of dissimilarity between their corresponding vectors, using the *Euclidian distance*.

2.2. **The Adaptive Refactoring Phase.** During the evolution and maintenance of $\mathcal{S}$, new application classes are added to it in order to met new functional requirements. Let us denote by $\mathcal{S}'$ the software system $\mathcal{S}$ after

extension. Consequently, restructuring of $\mathcal{S}'$ is needed to keep its structure clean and easy to maintain. Obviously, for obtaining the restructuring that fits the new applications classes, the original restructuring scheme can be applied from scratch, i.e., *kRED* algorithm can be applied considering all entities from the modified software system $\mathcal{S}'$. However, this process can be inefficient, particularly for large software systems.

That is why we have extended the approach from [1] and we have proposed in [2] an adaptive method, named *CBAR* (*Core Based Adaptive Refactoring*), that deals with the evolving application classes set. Namely, the case when new application classes are added to the software system and the current restructuring scheme must be accordingly adapted, was handled. The main idea is that instead of applying *kRED* algorithm from scratch on the modified system $\mathcal{S}'$, we adapt (using *CBAR*) the partition obtained by *kRED* algorithm for the initial system $\mathcal{S}$, considering the newly added application classes.

The extension of the application classes set from $\mathcal{S}$ means that the number of entities in $\mathcal{S}'$ increases, and the vectors characterizing the entities, increase, as well. Therefore, the entities have to be re-grouped to fit the new application classes set. Let us consider that the set $C_1, C_2, \ldots, C_l$ of application classes from $\mathcal{S}$ is extended by adding s ($s \geq 1$) new application classes, $C_{l+1}, C_{l+2}, \ldots, C_{l+s}$. Consequently, the set of attributes will be extended with s new attributes, corresponding to the newly added application classes. The vector for an extended entity $e_i \in \mathcal{S}, 1 \leq i \leq n$ is extended as $e'_i = (e_{i1}, \ldots, e_{il}, e_{i,l+1}, \ldots, e_{i,l+s})$ and a set of new entities $\{e'_{n+1}, e'_{n+2}, \ldots, e'_{n+m}\}$ is added to $\mathcal{S}'$. This set corresponds to the entities from the newly added application classes, $C_{l+1}, C_{l+2}, \ldots, C_{l+s}$.

The *CBAR* method starts from the partitioning of entities from $\mathcal{S}$ into clusters established by applying *kRED* algorithm in the initial restructuring phase. Let $\mathcal{K} = \{K_1, K_2, \ldots, K_l\}$ be the initial clusters (restructured application classes) of $\mathcal{S}$, $K_i \cap K_j = \emptyset, i \neq j$, $\bigcup_{v=1}^{l} K_v = \mathcal{S}$. *CBAR* determines then $\mathcal{K}' = \{K'_1, K'_2, \ldots, K'_{l+s}\}$ the new partitioning of entities in $\mathcal{S}'$ after application classes set extension. It starts from the idea that, when adding few application classes, the old arrangement into clusters (partition $\mathcal{K}$) can be adapted in order to obtain the restructuring scheme of the extended software system. The algorithm determines those entities within each cluster K_i ($1 \leq i \leq l$) that have a considerable chance to remain together in the same cluster. They are those entities that, after application classes extension, still remain closer to the centroid (cluster mean) of cluster K_i. These entities form what is called the core of cluster K_i, denoted by $Core_i$. We denote by $CORE$ the set of all cluster cores, $CORE = \{Core_1, Core_2, \ldots, Core_l\}$.

The cores of all clusters K_i, $1 \leq i \leq l$, will be new initial clusters from which the adaptive partitioning process begins. The extended partition $\mathcal{K}'$ should also contain initial clusters corresponding to the newly added application classes. Therefore, the centroids corresponding to clusters $K_{l+1}, K_{l+2}, \ldots, K_{l+s}$ are chosen to be the newly added application classes.

Next, *CBAR* proceeds in the same manner as *kRED* [1] algorithm does. The *CBAR* algorithm can be found in [2, 3]. As experiments show, the result is reached by *CBAR* more efficiently than running *kRED* again from the scratch on the feature-extended entity set.

3. Related Work

There are various approaches in the literature in the field of *refactoring*. But, only very limited support exists in the literature for automatic refactorings detection.

Deursen et al. have approached the problem of *refactoring* in [4]. The authors illustrate the difference between refactoring test code and refactoring production code, and they describe a set of bad smells that indicate trouble in test code, and a collection of test refactorings to remove these smells.

Xing and Stroulia present in [13] an approach for detecting refactorings by analyzing the system evolution at the design level.

A search based approach for refactoring software systems structure is proposed in [9]. The authors use an evolutionary algorithm for identifying refactorings that improve the system structure.

An approach for restructuring programs written in Java starting from a catalog of bad smells is introduced in [5].

Based on some elementary metrics, the approach in [12] aids the user in deciding what kind of refactoring should be applied.

The paper [11] describes a software vizualization tool which offers support to the developers in judging which refactoring to apply.

Clustering techniques have already been applied for program restructuring. A clustering based approach for program restructuring at the functional level is presented in [14]. This approach focuses on automated support for identifying ill-structured or low cohesive functions. The paper [8] presents a quantitative approach based on clustering techniques for software architecture restructuring and reengineering as well as for software architecture recovery. It focuses on system decomposition into subsystems.

A clustering based approach for identifying the most appropriate refactorings in a software system is introduced in [1].

To our knowledge, there are no existing approaches in the literature in the direction of adaptive refactoring, as we have approached in [2].

4. JHotDraw Case Study

In this section we present an experimental evaluation of *CBAR* algorithm on the open source software JHotDraw, version 5.1 [7].

It is a Java GUI framework for technical and structured graphics, developed by Erich Gamma and Thomas Eggenschwiler, as a design exercise for using design patterns. Table 1 gives an overview of the system's size.

Classes	**173**
Methods	**1375**
Attributes	**475**

Table 1. JHotDraw statistic.

The reason for choosing JHotDraw as a case study is that it is well-known as a good example for the use of design patterns and as a good design. Our focus is to test the accuracy of *CBAR* algorithm described in Subsection 2.2 on JHotDraw.

4.1. Quality Measures.

In the following we will present several measures that we propose for evaluating the obtained results from the restructuring point of view and from the adaptive process point of view, as well.

4.1.1. *Quality measures for restructuring.* In order to capture the similarity of two class structures (the one obtained by a clustering algorithm and the original one) we will use three measures.

Each measure evaluates how similar is a partition of the software system S determined after applying a clustering algorithm (as *kRED* or *CBAR*) with a good partition of the software system (as the actual partition of JHotDraw is considered to be).

In the following, let us consider a software system $\mathcal{S}$ consisting of the application classes set $\{C_1, C_2, \ldots, C_l\}$. We assume that the software system S has a good design (as JHotDraw has) and $\mathcal{K} = \{K_1, \ldots K_l\}$ is a partition reported by a clustering algorithm (as *kRED* or *CBAR*).

Definition 1. [1] ***Accuracy of classes recovery - ACC.***

The accuracy of partition $\mathcal{K}$ with respect to the software system $\mathcal{S}$, denoted by $ACC(\mathcal{S}, \mathcal{K})$, is defined as:

$$ACC(\mathcal{S}, \mathcal{K}) = \frac{1}{l} \sum_{i=1}^{l} acc(C_i, \mathcal{K}), \tag{1}$$

$$\sum_{k \in \mathcal{M}_{C_i}} \frac{|C_i \cap k|}{|C_i \cup k|}$$

where $acc(C_i, \mathcal{K}) = \frac{}{|\mathcal{M}_{C_i}|}$. $\mathcal{M}_{C_i}$ *is the set of clusters from* $\mathcal{K}$ *that contain elements from the application class* C_i*, is the accuracy of* $\mathcal{K}$ *with respect to application class* C_i*, i.e.* $\mathcal{M}_{C_i} = \{K_j |\ 1 \leq j \leq l,\ |C_i \cap K_j| \neq 0\}$.

ACC defines the degree to which the partition $\mathcal{K}$ is similar to $\mathcal{S}$. For a given application class C_i, $acc(C_i, \mathcal{K})$ defines the degree to which application class C_i, all its methods and all its attributes were discovered in a single cluster. Based on Definition 1, it can be proved that $ACC(\mathcal{S}, \mathcal{K}) \in [0, 1]$. $ACC(\mathcal{S}, \mathcal{K}) = 1$ iff $acc(C_i, \mathcal{K}) = 1, \forall\ C_i\ 1 \leq i \leq l$, i.e., each application class was discovered in a single cluster. In all other situations, $ACC(\mathcal{S}, \mathcal{K}) < 1$.

Larger values for ACC indicate better partitions with respect to $\mathcal{S}$, meaning that ACC has to be maximized.

Definition 2. ***PRECision of Methods discovery - PRECM.***

The precision of methods discovered in $\mathcal{K}$ *with respect to the software system* $\mathcal{S}$*, denoted by* $PRECM(\mathcal{S}, \mathcal{K})$*, is defined as:*

$$PRECM(\mathcal{S}, \mathcal{K}) = \frac{1}{|nm|} \sum_{m\ is\ a\ method\ from\ \mathcal{S}} precm(m, \mathcal{K}),$$

where $precm(m, \mathcal{K})$ *is the precision of* $\mathcal{K}$ *with respect to the method* m *and* nm *is the number of methods from all the application classes from* $\mathcal{S}$*, i.e.* $precm(m, \mathcal{K}) = \begin{cases} 1 & \text{if } m \text{ was placed in the same class as in } \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$

$PRECM(\mathcal{S}, \mathcal{K})$ defines the percentage of methods from $\mathcal{S}$ that were correctly discovered in $\mathcal{K}$ (we say that a method is correctly discovered if it is placed in its original application class). Based on Definition 2, it can be proved that $PRECM(\mathcal{S}, \mathcal{K}) \in [0, 1]$. $PRECM(\mathcal{S}, \mathcal{K}) = 1$ iff $precm(m, \mathcal{K}) = 1$ for all m, i.e., each method was discovered in its original application class. In all other situations, $PRECM(\mathcal{S}, \mathcal{K}) < 1$.

Larger values for $PRECM$ indicate better partitions with respect to $\mathcal{S}$, meaning that $PRECM$ has to be maximized.

Definition 3. ***PRECision of Attributes discovery - PRECA.***

The precision of attributes discovery in partition $\mathcal{K}$ *with respect to the software system* $\mathcal{S}$*, denoted by* $PRECA(\mathcal{S}, \mathcal{K})$*, is defined as:*

$$PRECA(\mathcal{S}, \mathcal{K}) = \frac{1}{na} \sum_{a\ is\ an\ attribute\ from\ \mathcal{S}} preca(a, \mathcal{K}),$$

where $preca(a, \mathcal{K})$ *is the precision of* $\mathcal{K}$ *with respect to the attribute* a *and* na *is the number of attributes from all the application classes from* $\mathcal{S}$*, i.e.* $preca(a, \mathcal{K}) = \begin{cases} 1 & \text{if } a \text{ was placed in the same class as in } \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$

$PRECA(\mathcal{S}, \mathcal{K})$ defines the percentage of attributes from $\mathcal{S}$ that were correctly discovered in $\mathcal{K}$ (we say that an attribute is correctly discovered if it is placed in its original application class). Based on Definition 3, it can be proved that $PRECA(\mathcal{S}, \mathcal{K}) \in [0, 1]$. $PRECA(\mathcal{S}, \mathcal{K}) = 1$ iff $preca(a, \mathcal{K}) = 1$ for all a, i.e., each attribute was discovered in its original application class. In all other situations, $PRECA(\mathcal{S}, \mathcal{K}) < 1$.

Larger values for $PRECA$ indicate better partitions with respect to $\mathcal{S}$, meaning that $PRECA$ has to be maximized.

4.1.2. *Quality measures for the adaptive process.* As quality measures for *CBAR* algorithm we will consider the number of iterations and the cohesion of the core entities. In other words, we measure how the entities in $Core_j$ $(1 \leq j \leq l)$ remain together in clusters after *CBAR* algorithm ends.

As expected, more stable the core entities are and more they remain together with respect to the initial sets $Core_j$, better was the decision to choose them as seed for the adaptive clustering process.

We express the *cohesion* of the set of cores $CORE = \{Core_1, \ldots, Core_l\}$ as:

$$Coh(CORE) = \frac{\sum_{j=1}^{l} \frac{1}{no\ of\ clusters\ where\ the\ entities\ in\ Core_j\ ended}}{l} \tag{2}$$

The worst case is when each entity from each $Core_j \in CORE$ ends in a different final cluster. The best case is when each $Core_j$ remains compact and it is found in a single final cluster. So, the limits between which $Coh(CORE)$ varies are given below, where the higher the value of $Coh(CORE)$ is, better was the cores choice.

Based on the definition of $Coh(CORE)$, it can be proved that

$$\frac{1}{l} \leq Coh(CORE) \leq 1. \tag{3}$$

4.2. **Experimental results.** Let us consider JHotDraw system from which we have removed 2 application classes: **StorableInput** and **ColorMap**. We denote the resulting system by $\mathcal{S}$. Therefore, $\mathcal{S}$ consists of 171 application classes ($l = 171$). After applying *kRED* algorithm on $\mathcal{S}$ we have obtained a partition $\mathcal{K}$ in which there were 3 misplaced methods. The names of the

methods that were proposed to be moved is shown in the first column of Table 2. The suggested target class is shown in the second column.

Element	Type	Target class
PertFigure.writeTasks	Method	**StorableOutput**
PolygonFigure.distanceFromLine	Method	**Geom**
StandardDrawingView. drawingInvalidated	Method	**DrawingChangeEvent**

TABLE 2. The misplaced elements.

Let now extend $\mathcal{S}$ with the 2 application classes that were initially removed from JHotDraw, **StorableInput** and **ColorMap**. We denote by $\mathcal{S}'$ the extended software system, which, in fact, is the entire JHotDraw system. Consequently, the number of application classes from $\mathcal{S}'$ is 173 ($s = 2$).

There are two possibilities to obtain the restructured partition $\mathcal{K}'$ of the extended system $\mathcal{S}'$.

A. To apply $kRED$ algorithm from scratch on the entire JHotDraw system.
B. To adapt, using $CBAR$ algorithm, the partition $\mathcal{K}$ obtained after applying $kRED$ algorithm before the system's extension.

In the following we will briefly detail variants **A** and **B**.

A. After applying $kRED$ algorithm for JHotDraw case study ($\mathcal{S}'$), we have obtained a partition $\mathcal{K}'$ characterized by the following quality measures:

- $ACC = 0.9829$.
- $PRECM = 0.997$.
- $PRECA = 0.9957$.

In the partition $\mathcal{K}'$ there were 4 methods and 2 attributes that were misplaced in the partition obtained after applying $kRED$ algorithm. The names of the elements (methods, attributes) that were proposed to be moved is shown in the first column of Table 3. The suggested target class is shown in the second column.

B. We have adapted, using $CBAR$ algorithm, the partition $\mathcal{K}$ obtained after applying $kRED$ algorithm before the system's extension. The partition $\mathcal{K}'$ obtained this way is characterized by the following quality measures:

- $ACC = 0.9721$.
- $PRECM = 0.9949$.
- $PRECA = 0.9957$.

Element	Type	Target class
PertFigure.writeTasks	Method	**StorableOutput**
PertFigure.readTasks	Method	**StorableInput**
PolygonFigure.distanceFromLine	Method	**Geom**
StandardDrawingView. drawingInvalidated	Method	**DrawingChangeEvent**
ColorEntry.fName	Attribute	**ColorMap**
ColorEntry.fColor	Attribute	**ColorMap**

TABLE 3. The misplaced elements.

In the partition $\mathcal{K}'$ there were 7 methods and 2 attributes that were misplaced in the obtained partition. The names of the elements (methods, attributes) that were proposed to be moved is shown in the first column of Table 4. The suggested target class is shown in the second column.

Element	Type	Target class
PertFigure.writeTasks	Method	**StorableOutput**
PertFigure.write	Method	**StorableOutput**
CompositeFigure.write	Method	**StorableOutput**
PertFigure.readTasks	Method	**StorableInput**
PertFigure.read	Method	**StorableInput**
PolygonFigure.distanceFromLine	Method	**Geom**
StandardDrawingView. drawingInvalidated	Method	**DrawingChangeEvent**
ColorEntry.fName	Attribute	**ColorMap**
ColorEntry.fColor	Attribute	**ColorMap**

TABLE 4. The misplaced elements.

From Table 4 we can observe that $CBAR$ algorithm determines 3 more refactorings than $kRED$ algorithm:

(i) The *Move Method* refactoring *PertFigure.write* to class *StorableOutput*.

(ii) The *Move Method* refactoring *CompositeFigure.write* to class *StorableOutput*.

(iii) The *Move Method* refactoring *PertFigure.read* to class *StorableInput*.

From our perspective, all these refactorings can be justified. We give below the justification for these refactorings.

(i),(ii) *PertFigure.write* method is responsible with the persistence of an *PertFigure* instance. *PertFigure* class has the following attributes: an instance of *java.awt.Rectangle* and two lists of *Storable* objects. These

attributes need to be persisted when a *PertFigure* instance is written. The class *StorableOutput* is responsible for persisting primitive data types (int, boolean) but also contains method for storing *java.awt.Color* and *Storable* objects. Creating a method in class *StorableOutput* that stores a *java.awt.Rectangle* object or a list of *Storable* objects is justified by the fact that the *StorableOutput* class already contains similar methods. Moving these functionalities to the *StorableOutput* class can help to avoid code duplication.

(iii) *StorableInput* class provides generic functionality for reading primitive and user defined data. The instances stored using *StorableOutput* class can be retrieved using *StorableInput* class. The identified refactoring suggests the need for an additional method in *StorableInput* class to handle the retrieval of *java.awt.Rectangle*, list of *Storable* instances. As in the case of *StorableOutput* class, adding this functionality into the *StorableInput* will avoid code duplication.

In our opinion, applying the *Move Method* refactorings suggested at (i), (ii) and (iii) does not extend the responsibilities of the modified classes, but in some situations we may obtain classes with multiple responsibilities. Further improvements will deal with these possible problems.

We comparatively present in Table 5 the results obtained after applying $kRED$ and $CBAR$ algorithms for restructuring the extended system $\mathcal{S}'$.

Table 5. The results

Quality measure	$kRED$ for 173 classes	$CBAR$ for 173 classes
No. of iterations	6	4
ACC	0.9829	0.9721
PRECM	0.997	0.9949
PRECA	0.9957	0.9957
Coh	-	0.8912

From Table 5 we observe the following:

- $CBAR$ algorithm finds the solution in a smaller number of iterations than $kRED$ algorithm. This confirms that the time needed by $CBAR$ to obtain the results is reduced, and this leads to an increased efficiency of the adaptive process.
- The accuracy of the results provided by $CBAR$ are preserved (the additional refactorings identified by $CBAR$ were justified below).
- The choice of the cluster cores in the adaptive process was good enough (the cohesion is close to 1).

5. Conclusions and Future Work

We have extended in this paper the evaluation of *CBAR* algorithm previously introduced in [2] on the open source case study JHotDraw. *CBAR* is a clustering based method for adapting the class structure of a software system when new application classes are added to the system. The considered experiment shows the potential of our approach.

Further work will be done in order to isolate conditions to decide when it is more effective to adapt (using *CBAR*) the partitioning of the extended software system than to recalculate it from scratch using *kRED* algorithm. We also plan to improve the method for choosing the cluster cores in the adaptive process and to apply the adaptive algorithm *CBAR* on other open source case studies and real software systems.

ACKNOWLEDGEMENT. This work was supported by CNCSIS – UEFISCSU, project number PNII - IDEI 2286/2008.

References

[1] I.G. Czibula and G. Serban. Improving systems design using a clustering approach. *IJCSNS International Journal of Computer Science and Network Security*, 6(12):40–49, 2006.

[2] Gabriela Czibula, Istvan Gergely Czibula. Adaptive Refactoring Using a Core-Based Clustering Approach. In *Proceedings of SEPADS'2010*, 2010, to be published.

[3] Gabriela Czibula, Istvan Gergely Czibula. Clustering Based Adaptive Refactoring. *Wseas Transactions on Computers*, 2010, to be published.

[4] A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok. Refactoring test code. pages 92–95, 2001.

[5] T. Dudzikan and J. Wlodka. Tool-supported dicovery and refactoring of structural weakness. Master's thesis, TU Berlin, Germany, 2002.

[6] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[7] E. Gamma. JHotDraw Project. http://sourceforge.net/projects/jhotdraw.

[8] Chung-Horng Lung. Software architecture recovery and restructuring through clustering techniques. In *ISAW '98: Proceedings of the third international workshop on Software architecture*, pages 101–104, New York, NY, USA, 1998. ACM Press.

[9] Olaf Seng, Johannes Stammel, and David Burkhart. Search-based determination of refactorings for improving the class structure of object-oriented systems. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1909–1916, New York, NY, USA, 2006. ACM Press.

[10] Frank Simon, Silvio Loffler, and Claus Lewerentz. Distance based cohesion measuring. In *proceedings of the 2nd European Software Measurement Conference (FESMA)*, pages 69–83, Technologisch Instituut Amsterdam, 1999.

[11] Frank Simon, Frank Steinbruckner, and Claus Lewerentz. Metrics based refactoring. In *CSMR '01: Proceedings of the Fifth European Conference on Software Maintenance and Reengineering*, pages 30–38, Washington, DC, USA, 2001. IEEE Computer Society.

[12] Ladan Tahvildari and Kostas Kontogiannis. A metric-based approach to enhance design quality through meta-pattern transformations. In *CSMR '03: Proceedings of the Seventh European Conference on Software Maintenance and Reengineering*, pages 183–192, Washington, DC, USA, 2003. IEEE Computer Society.

[13] Zhenchang Xing and Eleni Stroulia. Refactoring detection based on UMLDiff change-facts queries. *WCRE*, pages 263–274, 2006.

[14] Xia Xu, Chung-Horng Lung, Marzia Zaman, and Anand Srinivasan. Program restructuring through clustering techniques. In *SCAM '04: Proceedings of the Source Code Analysis and Manipulation, Fourth IEEE International Workshop on (SCAM'04)*. pages 75–84. Washington, DC, USA, 2004. IEEE Computer Society.

Babeş-Bolyai University, Department of Computer Science, 1, M. Kogalniceanu Street, Cluj-Napoca, Romania

E-mail address: {istvanc, gabis}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 1, 2010

COALGEBRAIC APPROACH FOR PROGRAM BEHAVIOR IN COMONADS OVER TOPOSES

VILIAM SLODIČÁK, VALERIE NOVITZKÁ

Abstract. The practical goal of program behavior studies is to enhance program and system performance. A behavior of running program can be described by evaluating the coalgebraic structure over a collection of algebraic terms on state space. Coalgebras are defined by polynomial endofunctors. We formulate the coalgebras in categories. Toposes are special kind of categories defined by axioms saying roughly that certain constructions one can make with sets can be done in a category. We use approach via toposes and comonads as dual structures to monads. In our paper we introduce the main ideas of our approach for describing the behavior of systems by coalgebras and we illustrate it on the simple examples of data structure queue.

1. OVERVIEW

The aim of programming is to force the computer to execute some actions and to generate expected behavior. The notion of observation and behavior have played an important *rôle* in computer science. This behavior can be positive, e.g. desired behavior, or negative, e.g. side effect that must be excluded from the system. The description of the behavior of a computer system is non-trivial matter. But some formal description of such complex systems is needed if we wish to reason formally about their behavior. This reasoning should achieve the correctness or security of these systems. The dynamical features of formal structures involve a state of affairs which can be possibly observed and modified. We can consider the computer state as the combined contents of all memory cells. A user is able to observe only a part of this state and he can modify this state by typing commands. As a reaction, the computer displays certain behavior.

Received by the editors: April 10, 2010.

2010 *Mathematics Subject Classification.* 18B25, 18C15.

1998 *CR Categories and Descriptors.* G.2.0 [**Mathematics of computing**]: Discrete mathematics – *General*; F.3.2 [**Logics and meanings of programs**]: Semantics of Programming Languages – *Algebraic approaches to semantics.*

Key words and phrases. Program behavior, Category, Coalgebra, Comonad, Topos.

2. BASIC ASPECTS ABOUT PROGRAM BEHAVIOR

The practical objective of program behavior studies is to enhance program and system performance. The knowledge resulting from these studies may be useful in designing new programs and also may be employed to increase the performance of existing programs and systems [4]. The basic idea of the behavioral theory is to determine the relation between internal states and their observable properties. The internal states are often hidden. Computer scientist have introduced many formal structures to capture the state-based dynamics, e.g. automata, transition systems, Petri nets, or the behavior discovery and verification problem as a graph grammar induction [20]. Horst Reichel firstly introduced the notion of behavior in the algebraic specifications [14]. The basic idea was to divide types in a specification into visible and hidden ones. Hidden types capture states and they are not directly accessible. The execution of a computer program causes generation of some behavior that can be observed typically as computer's input and output [6]. The observation of program behavior can be formularized using coalgebras. A program can be considered as an element of the initial algebra arising from the used programming language. In other words it is an inductively defined set P of terms [11, 16]. This set forms a suitable algebra $F(P) \to P$ where F is an endofunctor constructed over the signature of the operations appointed to execution by a program. Each language construct corresponds to certain dynamics captured in coalgebras. The behavior of programs is described by the final coalgebra $P \to G(P)$ where the functor G captures the kind of behavior that can be observed. Shortly, generated computer behavior amounts to the repeated evaluation of a (coinductively defined) coalgebraic structure on an algebra of terms. Thus coalgebraic behavior is generated by an algebraic program. Therefore the algebras are used for constructing basic structures used in computer programs and coalgebras act on the state space of computer describing what can be observed externally. In our research we are interested into coalgebras because the theory of coalgebras is one of the most promising candidates for a mathematical foundation for computer systems, equipped with both ample applicability and mathematical simplicity. Applicability we conceptualize as mathematical representation of computer system that should be able to serve multifarious theoretical problems in computer systems and their solutions. Mathematical simplicity is a source of abstraction which brings wide applicability.

3. COALGEBRAIC APPROACH

The starting notion in coalgebraic approach is the *signature* used in the theory of algebraic specifications [3]. A signature is a couple $\Sigma = (T, \mathcal{F})$

consisting of a set of types T and a set of function symbols $\mathcal{F}$. In a signature we distinguish:

- *constructor operations* which tell us how to generate (algebraic) data elements;
- *destructor operations*, also called observers or transition functions, that tell us what we can observe about our data elements;
- *derived operations* that can be defined inductively or coinductively.

If we define a derived operation f inductively, we define the value of f on all constructors. In a coinductive definition of f we define the values of all destructors on each outcome $f(x)$.

Example. Let σ be an arbitrary type of queue's elements. We define the signature for the data structure queue as parameterized signature $Queue(\sigma)$:

$$
\begin{array}{ll}
Queue(\sigma): & \\
new: & \to Queue(\sigma) \\
error: & \to \sigma \\
addq: & Queue(\sigma), \sigma \to Queue(\sigma) \\
front: & Queue(\sigma) \to \sigma \\
remove: & Queue(\sigma) \to Queue(\sigma) \\
isEmpty: & Queue(\sigma) \to Bool \\
length: & Queue(\sigma) \to nat \\
if-then-else: & Bool, Queue(\sigma), Queue(\sigma) \to Queue(\sigma)
\end{array}
$$

The operations *new* and *addq* are constructors. Operations *remove* and *front* are destructors. Operations *isEmpty* and *length* are derived operations and they can be defined inductively [11]. □

Operations in a signature determine polynomial endofunctor F that can be constructed inductively from using constants, identities, products, coproducts and exponents. Let $\mathbf{C}$ be a category. An F-algebra

$$a = [cons_1, \ldots, cons_n] : F(\mathbf{C}) \to \mathbf{C}$$

describes internal structure of a program system and consists of co-tuple of constructors [11]. For the program behavior the dual notion of algebra, that is *the coalgebra*, is necessary. A G-coalgeba

$$c = \langle destr_1, \ldots, destr_n \rangle : \mathbf{C} \to G(\mathbf{C}).$$

provides observable properties of a system [7, 11].

For a polynomial endofunctor G a *G-coalgebra* is a pair (U, c), where carrier-set U is a set called *state space* and $c : U \to G(U)$ is the *coalgebraic structure* or

operation of the coalgebra (U, c). It is also known as *coalgebra dynamics*. The difference between F-algebra and G-coalgebra is the same as between construction and observation [11]. While F-algebra tells us how to construct elements in the carrier set by the algebraic structure $a : F(A) \to A$ going *into* A, in the case of coalgebra, the coalgebraic operation $c : U \to G(U)$ goes *out* of U. Usually a program can be considered as an element of the many-typed algebra that arises from used programming language. Each language construct also corresponds to certain dynamics which can be described *via* coalgebras. The program's behavior is thus described by suitable coalgebra acting on the state space of the mathematical machine, i.e. computer [9].

In coalgebras we do not know how to form the elements of U; but we have only the operations working on U, which may give some information about U. Therefore, states can be imagined as a black box and we have only limited access to the state space U. G-coalgebras are also models of the corresponding signatures, but instead of the case of F-algebras, these coalgebras are based on destructor operations.

4. CATEGORIES FOR COALGEBRAS

Coalgebraic concept is based on category theory. The notion of coalgebra is the categorical dual of algebras [2, 5]. Program can be considered as an element of the many-typed algebra that arises from used programming language. Behavior of the program is thus described by suitable coalgebra acting on the state space of the mathematical machine, i.e. computer [9].

4.1. **Coalgebras in category.** Now consider the category of carrier-sets denoted $\mathbf{C}$ with carrier-sets as objects and maps between sets as morphisms. It is a category of sets; according the definition this category is a *topos* [1, 16]. We denote by **Coalg** the category of coalgebras, which we are able construct for any polynomial functor $G : \mathbf{C} \to \mathbf{C}$. For this category we define:

- objects of this category are coalgebras over F, e.g. (U, c), (V, d), ...;
- morphisms are coalgebra homomorphisms between coalgebras, e.g. $f : (U, c) \to (V, d)$, ...;
- every object has identity morphism and coalgebra homomorphisms are composable.

It is well known that in the category **Coalg** of G-coalgebras for a given endofunctor $G : \mathbf{C} \to \mathbf{C}$ the terminal object can be constructed as a limit of a certain descending chain [15]. If this category has terminal object then from every object there is just one morphism to the terminal object. This object is final G-coalgebra that is unique up to isomorphism. A final G-coalgebra

can be obtain from the pure observations. Finality provides a tool for defining functions into final G-coalgebra. A function $f : U \to V$ we define so that we describe direct observations with simple next steps as G-coalgebra on V. A function f arises by steps repeating this process [11].

Example. We can define constructors of signature $Queue(\sigma)$ coalgebraically as follows. Let

$$G(Q) = 1 + (Q \times I)$$

be the polynomial functor. We claim that the final G-coalgebra defined over G is the pair $(I^{\mathbb{N}}, next)$ where map

$$next : I^{\mathbb{N}} \to 1 + (I^{\mathbb{N}} \times I)$$

is defined as

$$next(q) = \begin{cases} \kappa_1(*) & \text{if } q \text{ is empty} \\ \kappa_2(q', i) & \text{if } q = addq(q', i) \end{cases}$$

where κ_1, κ_2 are the first and the second *injections* (co-projections) of the coproduct. The first value of this operation expresses the observation when the queue is empty, i.e. we cannot get any element of the queue. The second value expresses that the operation *next* returns the element (observable value) i of the queue q and the another queue q' without this element, where $q = addq(q', i)$. It holds that

$$q' = remove(q) \qquad\qquad i = front(q)$$

so we obtain

$$q = addq(q', i) = addq\left(remove\left(q\right), front\left(q\right)\right)$$

Then the constructor *new* can be defined as the unique operation $new :\to Queue(\sigma)$ in the commutative diagram at the Fig. 1.

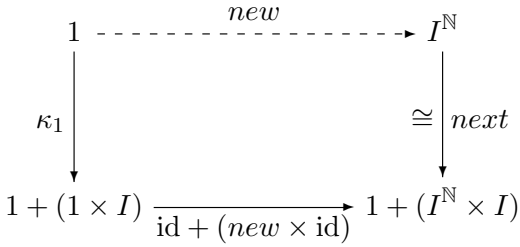


FIGURE 1. Coalgebraic definition of the *new* operator

We are able to define analogously another operations as it was derived in [11, 16].

□

As it was introduced, we can construct category of coalgebras **Coalg** for any polynomial functor $G : \mathbf{C} \to \mathbf{C}$. For a carrier-sets homomorphisms in the category $\mathbf{C}$ we define morphisms between appropriate coalgebras in the category **Coalg**

$$f : (U, c) \to (V, d)$$

such that the diagram at Fig. 2 commutes.

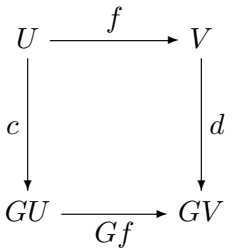


FIGURE 2. Homomorphism of coalgebras and carrier-sets

It follows from the diagram, that there holds the equality

$$Gf \circ c = d \circ f$$

It is clear that the coalgebraic approach is based on categories. The polynomial functors induced by signatures are endofunctors, i.e. their domain and codomain is the same category. The power and expressibility of coalgebraic approach will depend on the concrete categorical structure used for particular programming paradigms [7]. It seems be reasonable that suitable categories will be models of type theories [10, 16]. Their structure and properties ensure the correct treating of data type structures used in all programming paradigms.

It was proved in [2] that the functors

$$\mathcal{U}_G : \mathbf{Coalg} \to \mathbf{C} \qquad\qquad \mathcal{F}_G : \mathbf{C} \to \mathbf{Coalg}$$

form a pair of adjoint functors.

Functor $\mathcal{U}_G$ is the forgetful functor and it assigns to each coalgebra its carrier-set and to each coalgebra homomorphism the function between carrier-sets:

$$\mathcal{U}_G\left(\left(U, U \xrightarrow{c} GU\right)\right) = U \qquad\qquad \mathcal{U}_G(f) = f$$

Analogously we define generating functor $\mathcal{F}_G$. It assigns to each carrier-set the appropriate coalgebra and to each function between carrier-sets it assigns

coalgebra homomorphism:

$$\mathcal{F}_G(U) = \left(GU, GU \xrightarrow{\delta U} GGU\right) \qquad \mathcal{F}_G(f) = Gf$$

where δ is comonadic operation *co-multiplication* and GGU can be shortly redrawn as G^2U.

Functors $\mathcal{U}_G$ and $\mathcal{F}_G$ form an adjoint pair of functors $\mathcal{F}_G \dashv \mathcal{U}_G$

$$\mathcal{F}_G \dashv \mathcal{U}_G$$

which means that their composition is polynomial functor G

$$\mathcal{U}_G \circ \mathcal{F}_G = G$$

This functor is considered as a functor for comonad over category of carrier-sets [1, 16]. As we defined the pair of adjoint functors $\mathcal{F}_G$ and $\mathcal{U}_G$, we consider the co-unit of comonad ε as the co-unit of adjunction [18, 19].

5. COMONADIC APPROACH AND COALGEBRAS

From one point of view, a monad is an abstraction of certain properties of algebraic structures. From another point of view, it is an abstraction of certain properties of adjoint functors. Theory of monads has turned out to be an important tool for studying toposes [2, 16]. We are interested to explanation of categorical formulation of coalgebras for program behavior. Comonads allow us to utilize the properties of coalgebras in categories. While comonads have very important connection to toposes, we will be concerned about the representation of coalgebras via comonads in toposes.

5.1. **Definition of comonad.** Comonad (or cotriple) is dual construction to monad. The comonad over category $\mathbf{C}$ is the monad over opposite category $\mathbf{C}^{op}$.

Comonad is a mathematical structure $\mathcal{G} = (G, \varepsilon, \delta)$ which consists of

- endofunctor $G : \mathbf{C} \to \mathbf{C}$;
- co-unit natural transformation $\varepsilon : G \to \mathrm{id}_{\mathbf{C}}$;
- co-multiplication natural transformation $\delta : G \to G^2$.

Morphisms ε and δ are natural transformations of the functor G. They are subject to the condition that the diagrams at Fig. 3 and Fig. 4 commute.
Coherence triangle at Fig. 4 is the combination of two commutative triangles. It defines left and right identity according to definition of the category. The component of εG at some object X is the component of ε at GX, whereas the component of $G\varepsilon$ at X is $G(\varepsilon X)$ (as at Fig. 6); similar descriptions apply to

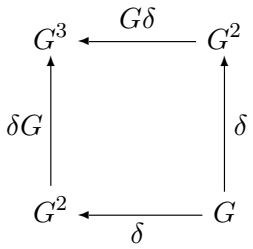


FIGURE 3. Coherence square for comonad

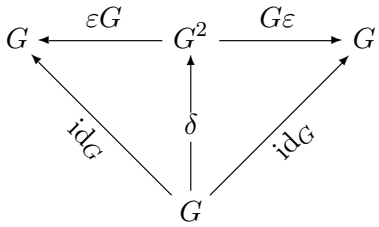


FIGURE 4. Coherence triangle for comonad

δ at Fig. 5.

We say that the comonad is *left-exact* if the functor G is left exact. Functor G is left exact, when it preserves binary products, terminal object and equalizers [2]. For coalgebras we define commutative coherence diagrams over comonad (Fig. 5 and Fig. 6).

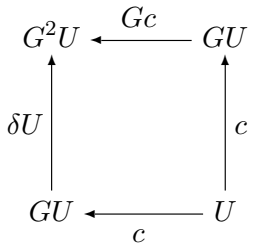


FIGURE 5. Coherence square for coalgebra

Coherence square at Fig. 5 is the basic diagram for generating functor $\mathcal{F}_G$. We are able to construct coalgebra dynamics with that relations for functors (Fig.

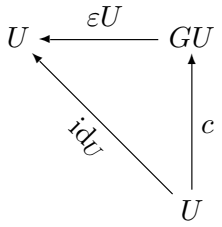


FIGURE 6. Coherence triangle for coalgebra

6). For coalgebras and their state spaces we define relation by the commutative diagram at Fig. 5.
It follows from the diagram at Fig. 5 that the following equation holds:

$$Gf \circ c = d \circ f$$

It means that all paths from the object U into the object GV in diagram constructed as compositions of corresponding arrows are equal in the commutative diagram at Fig. 5.

The diagrams at Fig. 5 and Fig. 6 satisfy the definition of comonadic natural transformations and express how to define coalgebra dynamics and construction of coalgebra dynamics by the generating functor $\mathcal{F}_G$. They also show the definition of comonad by coalgebras.

5.2. **The definition of topos.** Now we formulate basic aspects about toposes. Toposes are important in theoretical informatics for the purpose of modeling computations [1, 13, 17]. A topos is a special kind of category defined by axioms saying roughly that certain constructions one can make with sets can be done in a category [12]. If we want a topos to be a generalized mathematical theory, we suppose that a set of hypotheses or axioms are formulated in predicate logic. They implicitly define some kind of structure of objects and some properties of morphisms in the category $\mathcal{E}$. A topos is really a structure of a general theory defined by axioms formulated possibly in higher-order logic [16]. An elementary topos is such one whose axioms are formulated in the first-order logic, *i.e.* an elementary topos is the generalized axiomatic set theory.

A topos is a special category satisfying important conditions. There are many definitions of topos. In the following text we follow this definition:
A *topos* is a category $\mathcal{E}$ which satisfies the following properties:

- it is cartesian closed category;
- it has finite limits;

- it has representable subobject functor.

5.3. **Coalgebras in toposes.** In [11, 16] we formulated algebras in monadic terms. Analogously we formulated here coalgebras in monads. Kleisli categories are powerful tool for formulating properties of coalgebras over monads. We are able to formulate coalgebras also in categorical terms over comonad. In [16] we showed that toposes as special categories are able to formulate the properties of coalgebras in categorical terms. Category of carrier-sets consists of sets as objects and functions as morphisms between them. While category of sets is a topos [1, 8], the category of carrier-sets is also topos.

We defined coalgebraic operation *new* in chap. 4.1. Now we will consider the codomain of *next* - namely $1 + \left(I^{\mathbb{N}} \times I\right)$ as the subobject classifier Ω. This classifier together with defined morphism *true* will classify whether the subobjects of a given object 1 according to which elements belong to the subobject. We redraw the diagram in Fig. 1 as the diagram in Fig. 7.

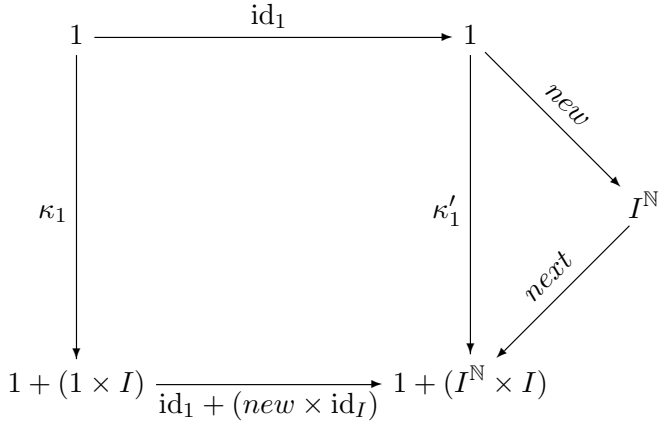


FIGURE 7. Coalgebraic definition of the *new* operator in toposes

The morphism $true : 1 \to \Omega$ is here defined as a pair of morphisms into coproduct:

$$true: \quad 1 \underset{next \circ new}{\overset{\kappa_1'}{\rightrightarrows}} \Omega$$

where $\Omega = 1 + (I^{\mathbb{N}} \times I)$. It satisfies the assumption about elements of queue that

$$next(q) = \begin{cases} \kappa_1(*) & \text{if } q \text{ is empty} \\ \kappa_2(q', i) & \text{if } q = addq(q', i) \end{cases}$$

so we are able with subobject classifier to construct coalgebra operations in topos.

Comonad $\mathcal{G} = (G, \varepsilon, \delta)$ in which G is left-exact functor is called *left-exact comonad*. It holds that for topos $\mathcal{E}$ and for the left-exact comonad $\mathcal{G}$ in $\mathcal{E}$ the category **Coalg** of coalgebras of $\mathcal{G}$ is also a topos [2]. From this property we are able to define previous constructions in toposes - we can define the properties of coalgebras in terms of toposes. For example the notion of *subcoalgebra* and the bisimilarity relation [6] we formulate as a subobject [16].

6. Conclusion

In our paper we presented the main ideas of our approach to the behavioral theory. We explained polynomial endofuctor that is the basic concept in algebraic and coalgebraic approach. The concepts we illustrated on the data structure queue that is simple enough for examples but quite rich for explaining behavioral concepts. We sketched out the comonadic approach based on the toposes and we showed how to construct coalgebraic operation in topos. The topos theory is a powerful tool for defining the behavior of program systems. Our future research will extend this approach with the Kleisli categories and monads and with the correct proving of the bisimilarity of the states. We also want to extend this approach to some paradigms of programming including object-oriented programming and logic programming.

Acknowledgment

This work was supported by VEGA Grant No.1/0175/08: Behavioral categorical models for complex program systems.

References

[1] Barr, M., and Wells, C. *Category Theory for Computing Science*. Prentice Hall International, 1990. ISBN 0-13-120486-6.

[2] Barr, M., and Wells, C. *Toposes, Triples and Theories*. Springer-Verlag, 2002.

[3] Ehrig, H., and Mahr, B. *Fundamentals of Algebraic Specification I.: Equations and Initial Semantics*. No. 6. EATCS, 1985. Monographs on Theoretical Computer Science.

[4] Ferrari, D. The improvement of program behavior. *Computer 9* (1976), 39–47.

[5] Hasuo, I. *Tracing Anonymity with Coalgebras*. PhD thesis, Radboud University Nijmegen, 2008.

[6] Jacobs, B. Introduction to coalgebra. *Towards Mathematics of States and Observations (draft)* (2005).

[7] Jacobs, B., and Rutten, J. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, No. 62 (1997), 222–259.

[8] Johnstone, P. *Topos Theory*. Academic Press Inc., London, 1977.

[9] Mihályi, D. Behaviour of algebraic term sequences. In *Pietriková A., Slodičák V., Főző L. editors: 7th PhD Student Conference and Scientific and Technical Competition of Students of Faculty of Electrical Engineering and Informatics Technical University of Košice, Slovakia* (2007), FEI TU Košice, pp. 153–154. ISBN 978-80-8073-803-7.

[10] Novitzká, V., Mihályi, D., and Slodičák, V. Categorical models of logical systems in the mathematical theory of programming. In *MaCS'06 6th Joint Conference on Mathematics and Computer Science, Book of Abstracts* (2006), University of Pécs, Hungary, pp. 13–14.

[11] Novitzká, V., Mihályi, D., and Verbová, A. Coalgebras as models of systems behaviour. In *International Conference on Applied Electrical Engineering and Informatics, Greece, Athens* (2008), pp. 31–36.

[12] Novitzká, V. Logical reasoning about programming of mathematical machines. In *Acta Electrotechnica et Informatica* (March 2005), Košice, pp. 50–55.

[13] Phoa, W. *An introduction to fibrations, topos theory, the effective topos and modest sets.* The University of Edinburgh, 2006.

[14] Reichel, H. Behavioural equivalence - a unifying concept for initial and final specification methods. In *3rd Hungarian Computer Science Conference* (1981), no. 3, Akadémia kiadó, pp. 27–39.

[15] Schubert, C., and Dzierzon, C. Terminal coalgebras and tree-structures. In *18th Conference for Young Algebraists* (2003), University of Potsdam, Potsdam, Germany.

[16] Slodičák, V. *The Rôle of Toposes in the Informatics.* PhD thesis, Technical University of Košice, Slovakia, 2008. (in slovak).

[17] Slodičák, V. The toposes and their application in categorical and linear logic. In *6th PhD Student Conference and Scientific and Technical Competition of Students of Faculty of Electrical Engineering and Informatics Technical University of Košice* (2006), elfa s.r.o., pp. 121–122.

[18] Turi, D. *Category Theory Lecture Notes.* Laboratory for Foundations of Computer Science, University of Edinburgh, 2001.

[19] Wisbauer, R. *Algebras versus coalgebras.* University of Düsseldorf, Germany, 2007.

[20] Zhao, C., Kong, J., and Zhang, K. Program behavior discovery and verification: A graph grammar approach. *IEEE Transactions on Software Engineering 99*, PrePrints (2010).

Faculty of Electrical Engineering and Informatics, Technical University of Košice

E-mail address: viliam.slodicak@tuke.sk, valerie.novitzka@tuke.sk

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

n-QUASIGROUP CRYPTOGRAPHIC PRIMITIVES: STREAM CIPHERS

ADRIAN PETRESCU

Abstract. In this paper we present two new n-quasigroup stream ciphers based on new n-quasigroup encryption scheme. Also, we present a practical implementation of these ciphers that has very good cryptographic properties. The implementation is based on a design concept of mixing two "incompatible" group operations on the set $\mathbb{Z}_{2^8}$.

1. Introduction

Computationally simple but cryptographically strong cryptographic systems have an important role for efficient digital communication tasks. There is a need for simple cryptographic primitives to implement security in an environment having limited storage and processing power.

Quasigroups based ciphers lead to particular simple yet efficient ciphers.

Almost all results obtained in the application of binary quasigroups in cryptology and coding theory to the end of eighties years of the XX-th century are described in [2] and [3]. A short survey of the known results related to the applications of binary quasigroups for constructing authentication codes, ciphers, and one-way functions is presented in [4].

As far as we know, the only attempts to construct n-quasigroup ciphers are our proposals [8] and [9].

In this paper, we propose two n-quasigroup symmetric-key stream ciphers: a self-synchronized stream cipher and a new type of stream cipher, a totally asynchronous stream cipher.

A totally asynchronous stream cipher is a cipher that cannot recover from an error introduced in the process of communication.

Received by the editors: December 15, 2009.

2010 *Mathematics Subject Classification.* 20N15, 94A60, 68P25.

1998 *CR Categories and Descriptors.* C.2.0 [**Computer System Organization**]: Computer-Communication Networks – *General*; E3 [**Data**]: Data encryption.

Key words and phrases. Cryptography, Security, Stream ciphers.

This paper has been presented at the International Conference Interdisciplinarity in Engineering (INTER-ENG 2009), Târgu-Mureş, Romania, November 12–13, 2009.

Although this property can be seen as a disadvantageous one, there are in fact several useful applications of such ciphers provable secure stream cipher that can guarantee data integrity authentication without using Message Authentication Code or Secure Hash Function.

The implementation of these new ciphers is based on a design concept of mixing two "incompatible" group operation on the same set.

This paper is organized as follows. Section 2 presents a short overview of n-quasigroups. In Section 3 we show the cryptographic properties of n-quasigroup string functions. Section 4 describes a 3-quasigroup encrypting scheme. In Section 5 we present an implementation of a 3-quasigroup self-synchronizing stream cipher and a 3-quasigroup totally asynchronous stream cipher. Conclusions are drawn in Section 6.

2. n-quasigroup definitions

Recall several notions and results which will be used in what follows.

We shall denote the sequence $x_m, x_{m+1}, \ldots, x_n$ by $\{x_i\}_{i=m}^n$ or x_m^n. If $m > n$, then x_m^n will be considered empty.

A non-empty set A together with an n-ary operation $\alpha : A^n \to A$, $n \geq 2$ is called **n-groupoid** and is denoted by (A, α). For $n = 2$ we have a **binary groupoid**.

An n-groupoid (A, α) is called an **n-quasigroup** [1] if the equation

$$\alpha(a_1^{i-1}, x, a_{i+1}^n) = b \tag{1}$$

has a unique solution x for any a_1^n, $b \in A$ and every $i \in \mathbb{N}_n = \{1, \ldots, n\}$.

An equivalent definition, known as *combinatorial definition* is: an n-quasigroup is an n-groupoid such that in the equation

$$\alpha(x_1^n) = x_{n+1} \tag{2}$$

knowledge of any n of the arguments x_1^{n+1} specifies the $(n+1)$-th uniquely.

A **primitive n-quasigroup** [8] is an algebra (A, α, α_1^n), $\alpha, \alpha_i : A^n \to A$, $i \in \mathbb{N}_n$ such that the identities

$$\alpha(x_1^{i-1}, \alpha_i(x_1^n), x_{i+1}^n) = x_i \tag{3}$$

$$\alpha_i(x_1^{i-1}, \alpha(x_1^n), x_{i+1}^n) = x_i \tag{4}$$

$i \in \mathbb{N}_n$, are satisfied.

We note that the operations $\alpha, \alpha_1, \ldots, \alpha_n$ are mutually defined:

$$\alpha(x_1^n) = x_{n+1} \Leftrightarrow \alpha_i(x_1^{i-1}, x_{n+1}, x_{i+1}^n) = x_i, \tag{5}$$

$i \in \mathbb{N}_n$.

An n-quasigroup (A, α) yields a primitive n-quasigroup (A, α, α_1^n) called the **corresponding primitive n-quasigroup**: define $\alpha_i : A^n \to A$, $\alpha_i(a_1^{i-1}, b, a_{i+1}^n) = x$, the unique solution of equation (1).

In turn, a primitive n-quasigroup (A, α, α_1^n) yields n-quasigroups (A, α), (A, α_i), $i \in \mathbb{N}_n$.

Let (A, α) be an n-quasigroup and $[f_1^n; f]$ an ordered system of permutations of the set A. We define a new quasigroup operation β on A as follows:

$$\beta(x_1^n) = f^{-1}(\alpha\{f_i(x_i)\}_{i=1}^n) \tag{6}$$

The n-quasigroups (A, α) and (A, β) are called **isotopic** and $[f_1^n; f]$ an **isotopy of** (A, β) **to** (A, α).

The isotopism of n-quasigroups gives us the power to generate a large number of isotopic n-quasigroups.

3. n-QUASIGROUP STRING FUNCTIONS

In this section we show the cryptographic potentials of n-quasigroup string functions, as a new paradigm in cryptography.

To simplify the notation, we shall consider $n = 3$. The generality of results is not affected.

Let $(A, \alpha, \alpha_1, \alpha_2, \alpha_3)$ be a 3-quasigroup and denote by A^+ the set of all nonempty words formed by the elements of A. For each $a_1a_2a_3a_4 \in A^+$ we define six maps $F_i, G_i : A^+ \to A^+$, $i = 1, 2, 3$, as follows:

$$\begin{aligned} &F_1(x_1 \dots x_n) = y_1 \dots y_n, \\ &y_1 = \alpha(x_1, a_1, a_2), \\ &y_2 = \alpha(x_2, a_3, a_4), \\ &y_j = \alpha(x_j, y_{j-2}, y_{j-1}), \text{ if } j > 2; \end{aligned} \tag{7}$$

$$\begin{aligned} &G_1(x_1 \dots x_n) = y_1 \dots y_n \\ &y_1 = \alpha_1(x_1, a_1, a_2), \\ &y_2 = \alpha_1(x_2, a_3, a_4), \\ &y_j = \alpha_1(x_j, x_{j-2}, x_{j-1}), \text{ if } j > 2; \end{aligned} \tag{8}$$

$$\begin{aligned} &F_2(x_1 \dots x_n) = y_1 \dots y_n \\ &y_1 = \alpha(a_1, x_1, a_2), \\ &y_2 = \alpha(a_3, x_2, a_4), \\ &y_j = \alpha(y_{j-2}, x_j, y_{j-1}), \text{ if } j > 2; \end{aligned} \tag{9}$$

$$\begin{aligned} &G_2(x_1 \dots x_n) = y_1 \dots y_n \\ &y_1 = \alpha_2(a_1, x_1, a_2), \\ &y_2 = \alpha_2(a_3, x_2, a_4), \\ &y_j = \alpha_2(x_{j-2}, x_j, x_{j-1}), \text{ if } j > 2; \end{aligned} \tag{10}$$

$$\begin{aligned} &F_3(x_1 \dots x_n) = y_1 \dots y_n \\ &y_1 = \alpha(a_1, a_2, x_1), \\ &y_2 = \alpha(a_3, a_4, x_2), \\ &y_j = \alpha(y_{j-2}, y_{j-1}, x_j), \text{ if } j > 2; \end{aligned} \tag{11}$$

$$
\begin{aligned}
&G_3(x_1 \dots x_n) = y_1 \dots y_n \\
&y_1 = \alpha_3(a_1, a_2, x_1), \\
&y_2 = \alpha_3(a_3, a_4, x_2), \\
&y_j = \alpha_3(x_{j-2}, x_{j-1}, x_j), \text{ if } j > 2.
\end{aligned} \tag{12}
$$

We call these maps 3-**quasigroup string functions with initial value** (IV) $a_1a_2a_3a_4$.

The maps F_3 and G_3 are generalizations for n-quasigroups of Markovski's binary quasigroup transformations e and d, respectively [6].

The maps F_i and G_i, $i = 1, 2, 3$ have several useful properties for cryptographical purposes.

1. The maps F_i and G_i are permutations on $A^+ : F_iG_i = G_iF_i = 1_{A^+}$ as a consequence of (3) and (4).

2. Each map F_i can lead to a self-synchronizing stream cipher.

For example, let $m = m_1 \dots m_n \in A^+$ be a plaintext $c = F_3(m) = c_1 \dots c_n$ its ciphertext and $c' = c_1 \dots c_{j-1}c'_jc_{j+1} \dots c_n$, $c'_j \in A$ the received text. Then $G_3(c') = m_1 \dots m_{j-1}m'_jm'_{j+1}m'_{j+2}m_{j+3} \dots m_n$ for some $m'_j, m'_{j+1}, m'_{j+2} \in A$. This result follows directly from the definition of G_3.

3. Each map G_i can leads to a totally asynchronous stream cipher.

For example, if we use G_3 as encrypting function and F_3 as decrypting function, then the rest of message after a ciphertext value error is garbled:

$$
\begin{aligned}
&m'_j = \alpha(m_{j-2}, m_{j-1}, c'_j), \\
&m'_{j+1} = \alpha(m_{j-1}, m'_j, c_{j+1}), \\
&m'_{j+2} = \alpha(m'_j, m'_{j+1}, c_{j+2}), \\
&m'_{j+3} = \alpha(m'_{j+1}, m'_{j+2}, c_{j+3}), \dots
\end{aligned}
$$

4. Each map $F_i(G_i)$ can lead to a stream cipher resistive on the brute force attack.

For example, suppose that an intruder knows a cipher text $c = c_1, \dots, c_n = F_1(x_1 \dots x_n)$, where $x_1 \dots x_n$ represents the unknown plaintext. Then, for recovering the quasigroup operation α which is the key of the encrypting method, it should solve a system of equations of the form (7). Taking into account (5), the following statement is true.

Let $c_1 \dots c_n \in A^+$ be a given string. For any 3-quasigroup operation β on A and any elements $a_1, a_2, a_3, a_4 \in A$, there are uniquely determined elements $x_1, \dots, x_n \in A$ such that the equality $F_i(x_1 \dots x_n) = c_1 \dots c_n$ ($G_i(x_1 \dots x_n) = c_1 \dots c_n$) holds.

Indeed, for example, if $i = 1$ we have

$$
\begin{aligned}
&c_1 = \beta(x_1, a_1, a_2) \Leftrightarrow x_1 = \beta_1(c_1, a_1, a_2) \\
&c_2 = \beta(x_2, a_3, a_4) \Leftrightarrow x_2 = \beta_1(c_2, a_3, a_4) \\
&c_j = \beta(x_j, c_{j-2}, c_{j-1}) \Leftrightarrow x_j = \beta_1(c_j, c_{j-2}, c_{j-1})
\end{aligned}
$$

if $j > 2$.

So, the system $F_i(x_1 \dots x_n) = c_1 \dots c_n$ has as many solutions as there are 3-quasigroup operations on the set A.

If $|A| = m$ (cardinality of A), then there are at least $m!(m-1)! \dots 2!1!$ binary quasigroup operations on A. From each binary quasigroup $(A, \cdot)$ we can derive two 3-quasigroups, $\alpha(x_1, x_2, x_3) = (x_1 \cdot x_2) \cdot x_3$ and $\beta(x_1, x_2, x_3) = x_1 \cdot (x_2 \cdot x_3)$.

Such 3-quasigroups are called **reducible**. But there exist irreducible 3-quasigroups with carrier A. Hence the number of 3-quasigroups (A, α) is very large.

5. If an intruder knows both the plaintext and the corresponding ciphertext, in some cases it can't recover quasigroup operation α (see Section 5).

6. Each map F_i has a nice scrambling property. The following is true.

Let $m = m_1 \dots m_n \in A^+$ be an arbitrary string and let $c = c_1 \dots c_n = F_i(m)$. If n is large enough, then the distribution of elements $c_j, j \in \mathbb{N}_n$ is uniform.

4. A 3-quasigroup encryption scheme

Let $(A, \alpha, \alpha_1, \alpha_2, \alpha_3)$ be a 3-quasigroup called the **seed quasigroup**. Denote by $\mathcal{M}$ the **message space** and $\mathcal{C}$ denotes the **ciphertext space**. We put $\mathcal{M} = \mathcal{C} = A^+$. For each element $a \in A$, let f_a be a permutation of A. $K = A^8 \times \{1, 2, 3\}$ is called the **key space**. An element $k = a_1 a_2 \dots a_8 i$ is called a **key**.

From section 3, it follows that the quasigroup operation α must be kept secret. But is not a good idea to use all the time the same quasigroup. The isotopism of quasigroups gives us the power to use a large number of isotopic quasigroups to seed quasigroup.

To simplify the notation we put $f_j = f_{a_j}$. Using the subkey $a_1 a_2 a_3 a_4$, we define a new quasigroup operation β on A as follows:

$$\beta(x_1, x_2, x_3) = f_4^{-1}(\alpha(f_1(x_1), f_2(x_2), f_3(x_3)).$$

For the 3-quasigroup $(A, \beta, \beta_1, \beta_2, \beta_3)$, consider the quasigroup string function F_i and G_i, $i = 1, 2, 3$, with initial value $a_5 a_6 a_7 a_8$.

Finally, we get two stream ciphers:

- a self-synchronizing stream cipher if for each $i = 1, 2, 3$, F_i is the encryption function and G_i the decryption function.

- a totally asynchronous stream cipher if for each $i = 1, 2, 3$, G_i is the encryption function and F_i the decryption function.

The seed quasigroup $(A, \alpha, \alpha_1, \alpha_2, \alpha_3)$, the key space, the set $\{f_a \mid a \in A\}$ and the definitions of string functions F_i and G_i are public knowledge.

The security of our ciphers lies solely on the key, not on the encryption algorithm. Perfect secrecy in the sense of Shanon is obtained if a "one-time" key is used.

For other arity values of quasigroup operations, the encryption scheme is similar.

5. A practical implementation

This section describes a very fast, strong and small 3-quasigroup self-synchronizing stream cipher and a 3-quasigroup totally asynchronous stream cipher.

From the practical viewpoint, the most important quasigroups are of order 2^8-byte encoding and 2^{16}-word encoding. The usage of a general 3-quasigroup in computation requires to store its Cayley table. For a quasigroup of order n, this table has n^3 elements. In particular $(2^8)^3 = 16$MB. In order to overcome the storage requirements for the Cayley table we consider as seed quasigroup $(\mathbb{Z}_{256}, \alpha, \alpha_1, \alpha_2, \alpha_3)$, $\alpha(x, y, z) = x - y - z \pmod{256}$.

To define permutations f_a, we consider a new group operation $\circ$ on $\mathbb{Z}_{256}$ - multiplication modulo 257. This kind of multiplication was first used in IDEA cipher [5].

To generalize the discussion beyond the case of byte encoding [5], let n be one of the integers 1, 2, 4, 8, 16. As of April 2009 the only know Fermat primes are $2^n + 1$.

Let $(\mathbb{Z}^*_{2^n+1}, \cdot)$ denote the multiplicative group of the field $\mathbb{Z}_{2^n+1}$ and let $(\mathbb{Z}_{2^n}, +)$ denote the additive group of the ring $\mathbb{Z}_{2^n}$. Define the direct map

$$d : \mathbb{Z}_{2^n} \to \mathbb{Z}^*_{2^n+1},\ d(x) = \begin{cases} x, \text{if } x \neq 0 \\ 2^n, \text{if } x = 0 \end{cases},$$

and via d and its inverse d^{-1} define a new binary operation on $\mathbb{Z}_{2^n}$,

$$x \circ y = d^{-1}(d(x) \cdot d(y)).$$

Then $(\mathbb{Z}_{2^n}, \circ)$ is a cyclic group isomorphic to $(\mathbb{Z}^*_{2^n+1}, \cdot)$.

On the set $\mathbb{Z}_{2^n}$ we have two group operations $(\mathbb{Z}_{2^n}, +, \circ)$. These operations are "incompatible" in the sense that:

- no distributive law is satisfied:

$$x \circ (y + z) \neq (x \circ y) + (x \circ z),$$
$$x + (y \circ z) \neq (x + y) \circ (x + z);$$

- no generalized associative law is satisfied

$$x \circ (y + z) \neq (x \circ y) + z,$$
$$x + (y \circ z) \neq (x + y) \circ z.$$

Meier and Zimmerman [7] proposed a good performance algorithm and implementation for multiplication modulo $2^n + 1$. This algorithm requires a total of six addition and subtractions, one 8 (16) bit multiplication and one comparison and is based on the following result [5]. Let a, b be two n bit non-zero integers in Z_{2^n+1}. Then

$$ab(\mathrm{mod}(2^n + 1)) = \begin{cases} ab(\mathrm{mod}2^n) - ab\,\mathrm{div}2^n, \text{ if } ab(\mathrm{mod}2^n) \geq ab\,\mathrm{div}2^n \\ ab(\mathrm{mod}2^n) - ab\,\mathrm{div}2^n + 2^n + 1, \text{ otherwise} \end{cases}$$

where $ab\,\mathrm{div}2^n$ denotes the quotient when ab is divided by 2^n.

Now, for each $a \in \mathbb{Z}_{256}$, define the permutation f_a to be $f_a(x) = x \circ a$.

We define the key space $K = \mathbb{Z}_{256}^9$. For each key $k = a_1 \dots a_8 a_9$, we have

$$\begin{aligned} \beta(x_1, x_2, x_3) &= (x_1 \circ a_1 - x_2 \circ a_2 - x_3 \circ a_3) \circ a_4^{-1} \\ \beta_1(x_1, x_2, x_3) &= (x_1 \circ a_4 + x_2 \circ a_2 + x_3 \circ a_3) \circ a_1^{-1} \\ \beta_2(x_1, x_2, x_3) &= (x_1 \circ a_1 - x_2 \circ a_4 - x_3 \circ a_3) \circ a_2^{-1} \\ \beta_3(x_1, x_2, x_3) &= (x_1 \circ a_1 - x_2 \circ a_2 - x_3 \circ a_4) \circ a_3^{-1} \end{aligned}$$

Hence the decryption is essentially the same process as encryption.

We set $i = 3$ if $a_9 \equiv 0 \pmod 3$ and $i = a_9 \pmod 3$ otherwise.

Therefore, we have 2^{32} 3-quasigroups on $\mathbb{Z}_{2^8}$ and $3 \cdot 2^{32}$ pairs of encryption and decryption functions.

The "incompatibility" of the operations $+$ and $\circ$ implies a strong resistance on known plaintext attack. If an intruder already knows both the plaintext $m = m_1 \dots m_n$ and the associated ciphertext $c = c_1 \dots c_n$, as far as we know, brute force is the only method to recover the key from equations of the from

$$c_j = (a_1 \circ c_{j-2} - a_2 \circ c_{j-1} - a_3 \circ m_j) \circ a_4^{-1}$$

for encryption function F_3, for example.

The security of the proposed cipher needs further investigations. The author hereby invite interested parties to attack this proposed cipher and will be grateful to receive the results of any such attacks.

We note that an uniform distribution of the characters of the ciphertext occurred in every of more than 50 experiments, even for short plaintexts.

The cipher was implemented in programming languages $C++$ and Java. In assembly language the obtained code is tiny.

Finally, we present a simple speed test for a $C++$ implementation of this cipher. We compared the average elapsed times in seconds to encrypt and decrypt a file with that to copy the same file one byte at a time.

In a similar way we get a 3-quasigroup totally asynchronous stream cipher. We interchange the maps F_i and G_i.

TABLE 1. Speed test

File size	File copy	Encrypt	Decrypt
489 KB	0.062	0.094	0.094
1.22 MB	0.170	0.219	0.219
2.11 MB	0.234	0.391	0.391
6.01 MB	0.672	1.141	1.141

6. Conclusions

These ciphers are appropriate for a fast online digital communication.

The ciphers structure facilitate a hardware implementation. The similarity of encryption and decryption makes it possible to use the same device in both encryption and decryption.

An extension to n-quasigroups of the encryption scheme (Section 4) is obvious.

In order to improve the security of the proposed cipher, 3-quasigroups can be replaced by n-quasigroups ($n = 4, 5, \dots$) and/or $\mathbb{Z}_{2^8}$ can be replaced by $\mathbb{Z}_{2^{16}}$.

References

[1] V.D. Belousov, *n-ary quasigroups*, Stiintca, Kishinev, 1972 (in Russian).
[2] J. Dénes, A.D. Keedwell, *Latin Squares. New Developments in the Theory and Applications*, North-Holland Publ. Co., Amsterdam, 1991.
[3] J. Déned, A.D. Keedwell, *Some applications of non-associative algebraic systems in cryptology*, PU.M.A., 12, No.2, 2002, 147-195.
[4] M.M. Glukhov, *Some applications of quasigroups in cryptography*, *Prikl. Diskr. Math.*, No. 2 (2), 2008, pp. 28-32 (in Russian).
[5] X. Lai and J.L. Massey, *A proposal for a new block encryption standard*, *Proc. of EUROCRYPT'90*, 1990, pp. 389-404.
[6] S. Markovski, *Quasigroup string processing and application in cryptography*, Invited talk, Proc 1st International Conference on Mathematics and Informatics for Industry, Thessaloniki, Greece, 2003.
[7] C. Meier and R. Zimmerman, *A multiplier modulo* $(2^n + 1)$, Diploma Thesis, Institut für Integrierte Systems, ETH Zürich, Switzerland, February 1991.
[8] A. Petrescu, *Applications of quasigroups in cryptography*, *Proc. Inter-Eng 2007*, Univ. "Petru Maior" of Tg. Mures, Romania, 2007.
[9] A. Petrescu, *A 3-quasigroup stream cipher*, Proc. Inter-Eng. 2009, Univ. "Petru Maior" of Tg. Mures, Romania, 2009, 264-267.

Department of Mathematics and Informatics, Faculty of Sciences and Letters, "Petru Maior" University of Tg. Mureş, 1 Nicolae Iorga Street, 540088 Târgu-Mureş, Romania

E-mail address: apetrescu@upm.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

A NEW VIDEO SURVEILLANCE SYSTEM USING AUTOMATED SECURE SERVICE COMPOSITION

GENGE BÉLA AND HALLER PIROSKA

Abstract. We propose a new video surveillance system based on a new service-oriented middleware. The proposed system includes video capture services and saving services that are composed together in order to ensure that frames are saved for future replay. The main novelty of the proposed middleware and video surveillance system lies in the automated composition of security protocols implemented by system services and in the automated execution of security protocol specifications. This novelty allows us to implement a heterogenous system, where services implement heterogeneous security protocols, are capable of downloading new security protocol specifications and automatically execute them.

1. Introduction

Over the last decade, because of increasing security concerns, video surveillance systems have received significant attention from both the research and industry communities. The use of surveillance systems has become a necessity in airports, traffic, subways, stores and even homes. The requirements for developing such systems, formulated by Ostheimer et al [1], include affordable hardware requirements, low bandwidth consumption and access control procedures.

The use of the Internet to transfer video images between distributed nodes has made it possible to expand the applicability of these systems to remote surveillance [2], allowing the distribution of nodes across multiple countries and

Received by the editors: December 7, 2009.

2010 *Mathematics Subject Classification.* 68M14, 68Q55, 94A62.

1998 *CR Categories and Descriptors.* H.3.5 [**Information Systems**]: Information Storage and Retrieval – *Online Information Services* K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems – *Security and Protection*;

Key words and phrases. Video surveillance, Web services, Security protocols.

This paper has been presented at the International Conference Interdisciplinarity in Engineering (INTER-ENG 2009), Târgu-Mureş, Romania, November 12–13, 2009.

continents. By using the Internet, surveillance systems also adopted modern security protocols such as TLS or VPN to secure data transfers.

However, using existing transport or network layer security protocols comes with a price. First, networks are usually closed environments protected by firewalls or NATs (i.e. Network Address Translation), where access is granted only to HTTP-based protocols. By implementing remote access, surveillance nodes can be placed behind multiple firewalls that can not be controlled by system developers. Second, the use of these protocols lacks the flexibility and extensibility required by applications running in heterogeneous environments. The heterogeneous property of video surveillance systems comes from the variety of protocols used by system nodes that must run for a long period of time and must face protocol changes without interrupting operation.

In this paper we propose a Web service-based approach for implementing video surveillance systems. Web services have been widely used to implement open systems running heterogenous protocols. In the proposed system, capture services (i.e. CAP-S) are used to capture frames and send them to client applications. Frames can be saved and replayed later if capture services transmit frames to saving services (i.e. SAVE-S). Replay is guaranteed by replay services (i.e. REPL-S). Connecting capture services to saving services is not a trivial task. Because each service can use different communication and different security protocols, interconnecting these services becomes a real challenge. Thus, coordinating and supervising this procedure becomes the task of the authorization and composition service (i.e. AUT-S). In order to automatically compose services, protocol specifications must be already available. These are stored and managed by the specification service (i.e. SPEC-S). Also, in order to automatically locate services, we use a name service (i.e. NAME-S).

Existing approaches [1, 2, 3, 4] assume a static interconnection of nodes, where nodes do not dynamically change the protocols or security protocols that are executed. The proposed system introduces many advantages over the mentioned existing systems. First of all, services are implemented as Web services that provide several advantages such as open and standardized protocols, technologies for service discovery and automated service composition. Second, the proposed system allows running different security protocols that provide secure mechanisms for accessing resources. Third, service capabilities are automatically composed together with security protocols, allowing automated service interconnection.

In order to enable the composition of Web services that implement heterogenous security protocols, first, we present a new middleware. The proposed middleware makes sure that both Web service capabilities and security protocols are composed. In our previous work we developed a new automated composition method of security protocols that has been implemented in the proposed middleware [6].

The main novelty of the proposed middleware and video surveillance system lies in the automated composition of security protocols implemented by system services and in the automated execution of security protocol specifications. This novelty allows us to implement a heterogenous system, where services implement heterogeneous security protocols, are capable of downloading new security protocol specifications and automatically execute them.

2. Security Protocol Composition and Specification

2.1. **Composition.** The role of the composition process is to create new, composed services, with an accumulated set of properties. These properties include service and security capabilities.

The composition of service capabilities ensures that the capabilities provided by services can be combined with the capabilities of other services. For each service, we define a set of preconditions and effects. Let ε be the set of preconditions of a service that denotes the type of data that can be received. Also, let ε' be the set of effects corresponding to a composed service, denoting the type of data generated by the service. Then, for a given service S_R, preconditions and effects are represented as $\varepsilon S_R \varepsilon'$. For N services, the composition of service capabilities reduces to finding the sequence for which $\varepsilon'_i = \varepsilon_{i+1}$, where $i = \overline{1, N-1}, N \geq 2$ and $\varepsilon_1 = \phi$. This means that the data corresponding to service preconditions must be available for each service and the set of preconditions corresponding to the first service must be empty, denoted by ϕ in the previous equations.

The composition of security protocols must ensure an accumulated set of security properties and must have a non-destructive effect on the security properties of the original protocols. This is not a trivial task, even when there is a human factor available. However, for the composition of security protocols in an on-line environment, the human factor can not be present, meaning that the entire composition process must be automatic. Such a composition method was developed in our previous work [5, 6], where we also proved that if security protocols are independent then these can be securely composed without losing their security properties. The key to eliminate the human factor is using an

enriched protocol model, with the drawback that modifying security protocols in case of unsatisfied conditions is not possible.

2.2. **Specification.** Constructing a new specification is not a trivial task, if we just look at the information required to be included: preconditions, effects, types, message sequences, security properties, protocol roles, etc. This task becomes even more complex when we realize that by automatically composing security protocols we get new protocols that must be automatically implemented by services.

In our previous work [7], we have chosen WSDL-S [8] and OWL [9] to be the components from which specifications are constructed. Each protocol participant is described by a pair of WSDL-S and OWL files. The role of the WSDL-S component is to describe the message sequences and directions that must be executed by protocol participants. Each WSDL-S component contains the participant's role in executing the protocol (i.e. initiator, respondent or third-party), protocol preconditions, protocol effects and message sequences. The role of the OWL component is to provide semantic information such as the construction, processing and implementation of cryptographic operations (e.g. encryption algorithm, encryption mode, key). This is connected to the WSDL-S component via annotations.

3. Middleware architecture

3.1. **Service oriented architecture.** We define four types of services: name services, specification services, authorization and composition services and resource services. Name services (i.e. NAME-S) are implemented through UDDI [10] registries and are used to register, identify or locate existing services. Specifications, consisting of WSDL-S [8], OWL [9] and SAWSDL [12] files, are stored and managed by specification services (i.e. SPEC-S). Authorization and composition services (i.e. AUT-S) implement the verification mechanisms of client credentials and ensure the composition of service capabilities and security protocols. Finally, the resource services (i.e. RES-S) implement a set of capabilities provided for client applications.

The general middleware architecture is given in Figure 1. Each service contains several modules, a special attention was given to security aspects. Namely, each service contains an M-KEY module that handles cryptographic keys and certificates. These are used by security protocols in order to provide a secure communication between services. Another module that is present in all services except RES-S is M-SPEC, that implements all communication with

the specification service. The RES-S receives the specifications from AUT-S when the composition process is executed.

In order to access NAME-S and SPEC-S without any previous connections with SPEC-S, the mentioned services provide the M-SPPR module for automatically downloading their own specifications. This module implements a standard HTTP/GET protocol for downloading own specifications.

The other modules are specific to each service. Thus, the M-DATA module implements communication with a database; M-CCAP, M-CPREF and M-CTERM are used in the composition of service capabilities and security protocols; M-CPERF is used to evaluate the performance of security protocols after the composition process is ended (based on this evaluation, the most performant protocol is chosen); M-NAME is used to implement the communication with the NAME-S; M-ROUT and M-RIN are used when resources are connected with each other in the composition process; M-CL is used to implement communication with client terminals; M-IOSPEC is used to store specifications; M-AUT-COMP implements composition-specific operations and M-RES implements resource-specific protocols (for AUT-S) and resource capabilities (for RES-S).

The theoretical aspects behind the implementation of these modules have been covered in our previous work [5, 6, 11]. The composition process [5, 6] ensures that the resulting protocols maintain their security properties. This has been implemented in three modules: M-CCAP, M-CPREF and M-CTERM. However, there can be multiple sub-protocols satisfying the same security requirements (i.e. mutual authentication, key exchange). In these cases we used a comparative performance evaluation process that we developed in our previous work [11], implemented in the M-CPERF module.

Resource access by client applications is done in several steps. In order to access a resource services, client applications must be authenticated, their rights must be verified and, if necessary, resources must be composed. This is done by accessing the AUT-S service, for which the specifications must be first downloaded. Clients request the location of AUT-S and SPEC-S from NAME-S, followed by the request of the specifications for AUT-S. The request for accessing RES-S is sent to AUT-S, containing the user credentials. These are verified and a security token is generated by AUT-S that is sent to RES-S. The client receives the generated token and sends it to RES-S, after which it is able to access the capabilities provided by the composed resource. All these steps are executed securely using an underlying security layer presented in the next section.

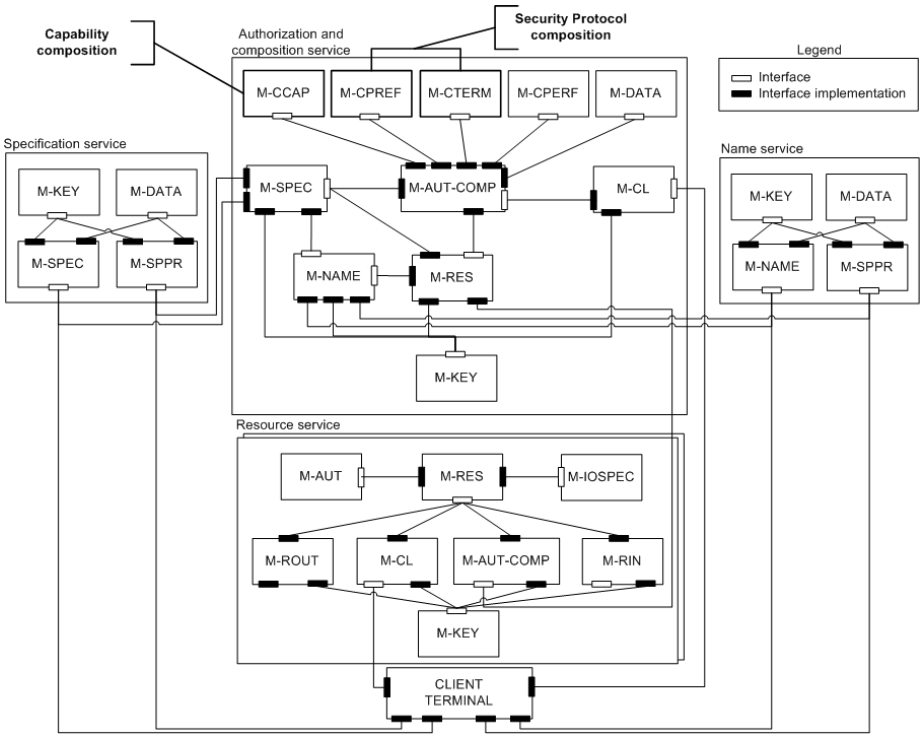


FIGURE 1. Middleware architecture

3.2. **Software architecture.** The architecture of the software stack is given in Figure 2. We identified two main layers: the communication layer and the service layer, given in Figure 2.

The *communication layer* implements service and security protocols. It is built on existing network and XML message-based protocols. Security protocols are implemented using extensions of the WS-Security standard, provided in our previous work [7]. The extensions consist of XML constructions for user names and binary keys required by key exchange and authentication protocols. The automated execution of security protocols is based on specifications developed using existing Web service technologies such as WSDL-S [8] and OWL [9]. Service protocols are described by SAWSDL [12] specifications and are specific to each service. Name services define a set of messages for interrogating service registry, while specification services define messages for downloading specifications. Authorization services define messages for requesting access to services and to send security tokens. Resource services provide two types of

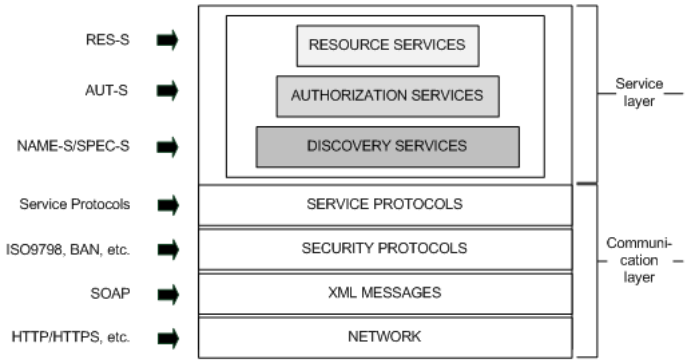


FIGURE 2. Software stack

specifications, for each external entity accessing them: AUT-S specifications provide messages for setting user security tokens, while client specifications provide client access messages.

The *service layer* provides the implementation of service capabilities. The capabilities of NAME-S, SPEC-S and AUT-S have already been discussed before. The capabilities of RES-S are specific to each implementation, ranging from video image capture capabilities to data storage capabilities. In order to implement new resource services, the only components that require change are the capabilities and the service protocol specifications. The security properties of communications with new resources are ensured by the underlying communication layer that remains unchanged.

4. Implementing a new video surveillance system

4.1. **System architecture.** In this section we describe our proposal for a new secure Web service-based video surveillance system. The proposed system is based on the platform presented in the previous section and comes with multiple advantages over existing video surveillance systems.

The proposed video surveillance system defines three types of resource services: capture services, saving services and replay services. Capture services (i.e. CAP-S) are used for capturing video images from physical devices. Frames can be sent directly to user applications or they can be saved for future replay. Frames are stored by saving services (i.e. SAVE-S) that also deal with the distribution of captured frames to storage hardware. Saved frames can be viewed by using replay services (i.e. REPL-S). The three service types are implemented as resource services using the proposed middleware.

To store the captured frames, we use the composition service provided by our middleware. Composition is used to compose SAVE-S with CAP-S services in order to save captured frames. Thus, frames can be replayed later in case a detailed analysis is required for a certain event from the past.

Clients can issue composition requests and the system automatically changes its connections, frames can be automatically redirected to new services. The main advantage of the proposed system is that composition is also possible with services implementing heterogeneous security protocols, because of the special security protocol composition and specification execution modules.

4.2. **Experimental results.** The experiments we conducted focussed on measuring the time required to access single and composed resources and on the time required to transfer data between services and client applications. For each resource type we used a different station with the same hardware configuration: Intel Dual Core CPU, 1.8GHz, 1GB of RAM, Windows XP OS. Our measurements were focused on capturing the processing performances of services, so stations were connected to a Local Area Network using several switches.

First, we measured the accessing time of single resources (i.e. that do not require composition). As shown in Figure 3, operations such as locating and downloading specifications clearly affect system performance. If specifications are not saved, accessing time of single resources is on average 580ms. If specifications are saved, the accessing time drops approx. 200ms. The peaks from Figure 3 denote the cases when the authorization service also needs to download specifications for the requested services.

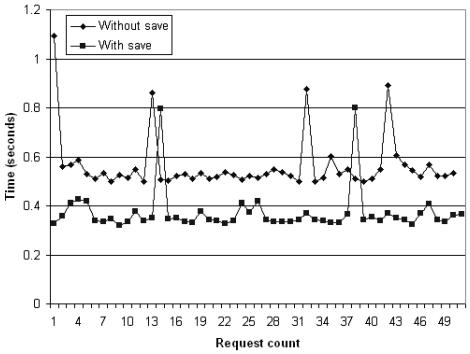
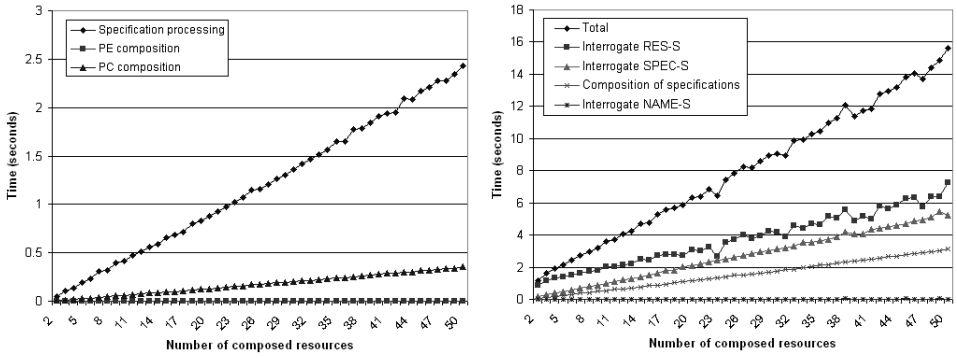


FIGURE 3. Accessing time for single resources



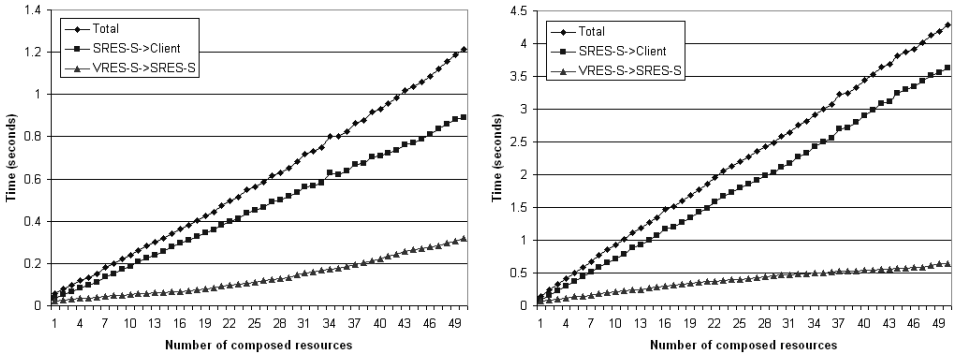
(A) (B)

FIGURE 4. Composition time: (A) Specification composition time (B) Accessing time of composed resources

Next, we measured the actual time required for the composition of specifications. Measurement results are shown in Figure 4 (A). The performance of the composition of preconditions and effects (i.e. PE-composition) and of the composition of protocol chains (i.e. PC-composition) is relatively good, considering that the composition takes around 450ms for 50 resources. However, the composition time is also influenced by the size and number of specifications, as shown in the same figure. Thus, the actual processing of specifications can reach almost 2.5 seconds for 50 resources.

While accessing composed resources, users do not only have to wait for the composition of specifications, but they also have to interrogate multiple services, such as the name service or specification service. Also, in the composition process the authorization service must connect to all resource services involved, must download their specifications and must compose them. These operations all contribute to the total accessing time for composed resources, as shown in Figure 4 (B).

Another aspect we measured was the time needed for frames to reach client applications in case of composed resources. We considered the two types of resources mentioned at the beginning of this section: SAVE (i.e. SRES-S) and VIDEO (i.e. VRES-S). We composed 2 to 50 VIDEO resources with a single SAVE resource and we measured the average time for frames to reach the SAVE service (i.e. from VRES-S to SRES-S) and the average time for frames to reach the client application from the SAVE resource (i.e. SAVE-Client).



(A) (B)

FIGURE 5. Transfer time: (A) 5fps, 10KB/frame (B) 10fps, 20KB/frame

The actual capturing of frames from hardware has only been simulated by the M-RES module using random bytes.

Because the transfer time is affected by the size of the frames and by the number of frames/second, we considered two cases. First, we considered a 5fps frame-rate, with the size of 10KB for each frame. For this case, results are shown in Figure 5 (A). For another case, we considered a frame-rate of 10fps with the size of 20KB for each frame. Results for this later case are shown in Figure 5 (B). By inspecting the measurement results from Figure 5 (A) and Figure 5 (B), we realize that the execution time increases almost 4 times in the later case. This is because we did not only increase the number of frames/second, but we also doubled the frame size, leading to an even more increased processing overhead consisting of cryptographic and XML processing operations.

5. CONCLUSIONS

We presented a new middleware for the automated composition of Web services that user heterogeneous security protocols. The approach used for the composition of security protocols was developed in our previous work and was implemented in the proposed middleware as a composition module.

The novelty introduced by the paper is the video surveillance system that can be used to create new, composed video resources. In this paper we composed video capture with video save resources in order to provide saving capabilities for frames that must be replayed in the future. The performances of the implemented system are highly dependent on the frame-rate and the size of the capture frames, which can be improved by using dedicated hardware for cryptographic operations or XML parsing.

References

[1] D. Ostheimer, S. Lemay, D. Mayisela, P. Dagba, M. Ghazal, A. Amer, *A modular distributed video surveillance system over IP*, in Proc. IEEE Canadian Conference on Electrical and Computer Engineering, Ottawa, Ontario, Canada, 2006, pp. 1001–1004.

[2] X. Yuan, Z. Sun, Y. Varol, G. Bebis, *A distributed visual surveillance system*, IEEE international conference on advanced video and signal based surveillance proceedings, 2003, pp. 199–204.

[3] J. Sun, S. A. Velastin, B. Lo, M.A. Vicentio-Silva, L. Khoudour, *A Distributed Surveillance System to Improve Personal Security in Public Transport*, Proceedings of the European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT-04), 2004, pp. 7–14.

[4] G. Gualdi, A. Prati, R. Cucchiara, E. Ardizzone, M. La Cascia, L. Lo Presti, M. Morana, *Enabling technologies on hybrid camera networks for behavioral analysis of unattended indoor environments and their surroundings*, Proceeding of the 1st ACM workshop on Vision networks for behavior analysis, 2008, pp. 101–108.

[5] B. Genge, I. Ignat, *Verifying the Independence of Security Protocols*, IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 2007, pp. 155–163.

[6] B. Genge, I. Ignat, P. Haller, *Automated Composition of Security Protocols*, IEEE International Conference on Intelligent Communication and Processing, Cluj-Napoca, Romania, 2009, pp. 251–258.

[7] B. Genge, P. Haller, *Towards Automated Secure Web Service Execution*, Networking 2009, Aachen, Germany, May 11-15, Lecture Notes in Computer Science (LNCS 5550), L. Fratta et al. (Eds.), Springer-Verlag, 2009, pp. 943–954.

[8] World Wide Web Consortium, *Web Service Semantics - WSDL-S*, W3C Member Submission, 7 November, 2005.

[9] World Wide Web Consortium, *OWL Web Ontology Language Reference*, W3C Recommendation, 10 February, 2004.

[10] Organization for the Advancement of Structured Information Standards, *Universal Description Discovery and Integration*, available at http://www.uddi.org/pubs/uddi_v3.htm, 2004.

[11] B. Genge, P. Haller, I. Ignat, O. Ratoi, *Informal specification-based performance evaluation of security protocols*, IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 2008, pp. 193–200.

[12] D. Martin, M. Paolucci, M. Wagner, *Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective*, OWL-S: Experiences and Directions - workshop at 4th European Semantic Web Conf, 2007.

"Petru Maior" University of Târgu Mureş, Department of Electrical Engineering, Nicolaie Iorga St., No. 1, Târgu Mureş, 540088, Romania

E-mail address: {bgenge,phaller}@engineering.upm.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

A NEW MULTIMEDIA STREAMING PLATFORM BASED ON XPCOM COMPONENTS

OVIDIU RATOI, HALLER PIROSKA, GENGE BÉLA

Abstract. We propose a platform for distributed multimedia systems. The proposed platform is implemented using the Netscape Portable Runtime (NSPR) and the Cross-Platform Component Object Model (XPCOM). This ensures system portability, flexibility and performance. The platform is equipped with bandwidth management components that control the transfer rates between components, allowing real-time streaming across multiple networks.

1. Introduction

Recent years have shown an increased interest towards multimedia rich applications. Multimedia content ranges from text or simple images to audio and video data or even animations. The increased availability of broadband Internet connections leads the way towards applications that offer high quality multimedia streaming over wide area networks. In this context there is a need for solutions that enable application developers to quickly and effortlessly develop this kind of applications.

In the same time software components technologies and component based software engineering are maturing, in fact the use of components is a sign of maturity in any field of engineering. The usage of software components offers a lot of advantages the most important of them being reusability, a component once developed may be reused in any number of applications, depending of how generic are the services it offers. Also the task of applications developers changes from development of new software to composition of existing pieces. Another characteristic property of software components is encapsulation. This property hides the internal structure and exposes a well defined

Received by the editors: December 7, 2009.

2010 *Mathematics Subject Classification.* 68M10, 68M14.

1998 *CR Categories and Descriptors.* H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Audio input/output, Video*;

Key words and phrases. Multimedia platform, XPCOM, NSPR.

This paper has been presented at the International Conference Interdisciplinarity in Engineering (INTER-ENG 2009), Târgu-Mureş, Romania, November 12–13, 2009.

interface through which the services are accessed. Encapsulation confers high flexibility to component based software as individual components can be easily replaced with improved ones as long as their interface remains identical.

Network communication was always an important issue when handling multimedia content especially when real time streaming was involved. A research team tackled this problem and proposed a multimedia applications middleware which regulated network traffic, optimized resource usage and offered a high degree of portability to applications [6].

Our goal was to create an interactive Web application based on components that allows bi-directional, real time communication between the resources and the user. This paper describes the component based platform proposed and implemented by us for multimedia transfer.

The paper is structured as follows. In section 2 we provide a short description of the Mozilla platform. In section 3 we provide a detailed description of the proposed platform and we describe the interface exposed by the platform and used by applications. A case study for the Multimedia Platform is presented in section 4, where beside the test results made for different type of applications the proposed platform provides an adaptive stream control component. We end the paper with a conclusion and future work in section 5.

2. Mozilla Platform architecture

Mozilla is an open source portable platform, developed and maintained by the Mozilla Foundation, best suited for rapid development of highly interactive visual applications [4].

Mozilla based applications have three possibilities to access the Operating System: using the JVM (Java Virtual Machine), using plugins or through an API called NSPR (Netscape Portable Runtime). NSPR is a portable API designed to provide operating system level services like threads and synchronization support, file and network I/O, memory management, time management, atomic operations or process creation.

XPCOM (Cross Platform Component Object Model) is Mozilla's object management and discovery system very similar to Microsoft COM and remotely to CORBA (Common Object Request Broker Architecture). It was designed to provide greater flexibility to the platform and to applications developed on top of it. These components can be created in a variety of languages ranging from C, C++ to JavaScript or Python and are accessed through a set of interfaces they implement [7]. In order to provide greater portability and implementation language independence, interfaces are described in a special language called XPIDL (Cross Platform Interface Definition Language), a variant of CORBA IDL. Object lifetime management and interface discovery

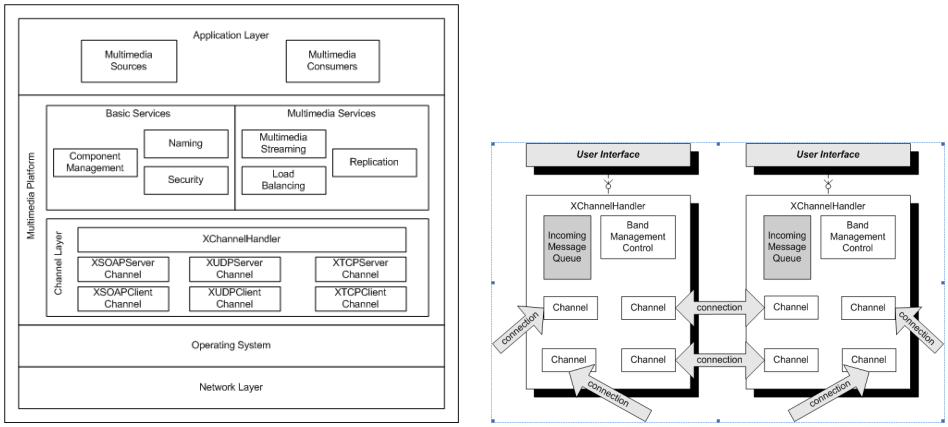
are implemented in a Microsoft COM style, by reference counting and special methods for querying all implemented interfaces.

Using the Mozilla platform, we focused on the development of streaming components, capable of receiving multimedia data from different sources, decode it, and deliver it to the user interface. The modular architecture of the Mozilla platform enables developers to add or remove modules with little effort, fitting the software to the available hardware and adjusting functionality to match product requirements. Our components adjust the transfer rate continuously, monitoring the devices and network capabilities. The need of the self-managing components was recently introduced in Web technologies [5], but not in implementations.

Another novelty of our approach is that the platform supports not only binary streaming (through binary channels), but also a collection of channels communicating through the use of messages specific to web services.

3. A Platform for Distributed Multimedia Applications

3.1. **Platform model description.** A well known method for providing a high degree of transparency and portability to distributed applications is positioning an intermediate layer called by us Multimedia Platform between the operating system and the application. In Figure. 1a the multilayer structure of a multimedia applications platform is presented.



(A) Multimedia Platform Architecture

(B) Platform Connection Model

Figure 1. Multimedia platform design

The bottom layer in this architecture is represented by the network which provides host computer interconnection and basic data transmission services. On the following level we have the operating system which provides services ranging from process management or memory organization to communication and synchronization. Above the operating system we can find the Multimedia Platform which is divided into two sections: one channel layer (described later in this paper) and a service layer. On top of the Multimedia Platform there is the application layer which contains multimedia sources or multimedia consumers.

The Multimedia Platform offers a service for data streaming between multimedia sources and consumers. Whenever a consumer needs data from a source a stream between them has to be established. The communication is based on the concept of channels. A channel is a logical communication link between two software entities, as presented in Figure 1b. The communication requires the existence of a connection between each pair of communicating applications. Channels embody communication protocols, while access is provided through one single interface.

3.2. **Platform interfaces description.** In the current form the platform provides six types of channels, but the Multimedia platform architecture offers an easy way to add more channels. Each type of channel represents one component, which is loaded by the platform. The interface implemented by the channels is the same for all of them. A general view of the channel architecture is presented in Figure 2.

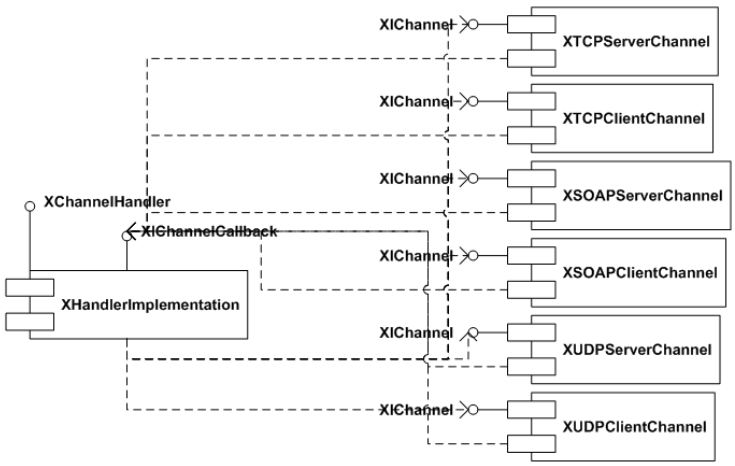


FIGURE 2. Multimedia Platform Architecture

For the moment applications can chose from the following channel types: XSOAPServerChannel, XSOAPClientChannel, XUDPServerChannel, XUDPClientChannel, XTCPServerChannel and XTCPClientChannel. The reason for introducing the concept of the UDP server and client channels was to create a similarity between the concept of a TCP connection and a UDP one. By using these channel types, applications that make use of TCP channels can be very easily switched to UPD channels.

The SOAP channels are created for developing applications based on WEB Services architecture. It provides SOAP communication protocol for creating both server and client applications. The main problem with today's distributed systems is interoperability. Web Services intend to solve this problem by introducing software components that are capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP [2].

SOAP (i.e. Simple Object Access Protocol) provides a simple mechanism for exchanging structured and typed information through the form of XML messages [1]. In order for our platform to support a standard web service interface, our platform has a SOAP-based channel capable of sending and receiving standard SOAP messages. For the implementation of the SOAP transport we used the open source gSOAP library [3].

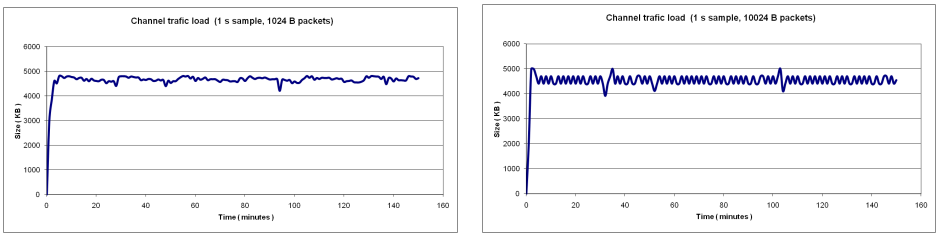
The SOAP standard does not only provide a means for exchanging XML data, but also binary data through the use of base64 or hex encodings. Because of this, integrating streaming data into SOAP messages becomes a straightforward process.

4. Multimedia platform case study

4.1. **Channel traffic load.** Once the platform was operational we tested it for maximum traffic load. For this purpose, one client and one server application was designed based on the proposed platform. Both applications were developed as stand-alone applications and neither of them had a graphical interface. The purpose of those two applications was to exchange messages at maximum speed. Once the messages ware received, they ware extracted from the platform and erased as quickly as possible, with no other processing made.

The tests were made over the internet using two Windows machines on a period of 160 minutes with a 1 minute sampling time. For testing purposes we used the TCP type of channels. The results are shown in Figure 3a and Figure 3b.

The first test used 1024 B packets and the second one used packets 10 times bigger. The spikes on the graph appeared when the internal queue size reached the maximum value and, as a failsafe measure, the platform stopped reading data from the channels. In this case the channel traffic load showed



(A) 1024 B packets (B) 10024 B packets

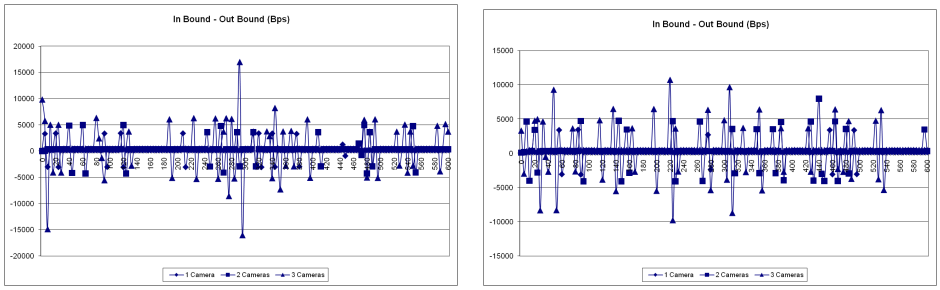
FIGURE 3. Test results at 1 minute time interval

a decrease until some of the incoming messages were processed. As we can see from the graphics the maximum traffic load over the internet using the current architecture of the platform is in the range of 4000 to 5000 Kb of data. This value is a satisfying one for a client-site application used in real time multimedia streaming, and especially for a web-based client application.

4.2. **Video streaming application test results.** For testing the platform in a multimedia environment we had created a stream server running in XULRunner and a web-based client application running in Mozilla Firefox 2.0.0.20. Both of them used the proposed multimedia platform. For the client application, the interaction between the browser and the platform was made using Java Script. For testing purposes we opened several instances of the client application that were connected to the stream server. Several sources were also connected to the stream server and the client applications received frames from them.

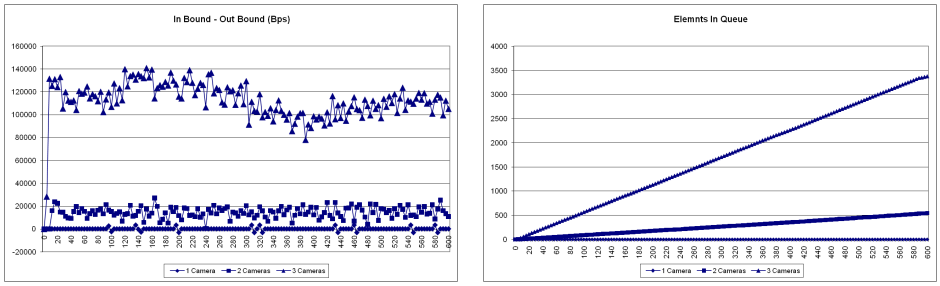
Using the model presented above, the video streaming application was tested on several platforms with variable number of cameras. Three parameters were measured: Incoming Bandwidth resulting from data received on the communication channel established with the stream server, Outgoing Bandwidth resulting from the total size of the video frames transmitted to and displayed by the user interface and Queue Size representing the number of video frames stored in the object waiting queue. All tests were conducted with the same application, stream server and cameras on a period of 10 minutes with a 4 seconds sampling interval. Once again the TCP based channels were used again. The results are presented in the following figures.

Test results show that the application performs well on both Windows and Mac OS environments and although there are oscillations in bandwidth, the waiting queue never grows bigger than two frames which, considering a data rate of 7-10 fps from each camera, translates into a very small delay, even when receiving stream from three different cameras. The performance of



(A) Windows XP (B) Mac OS X

FIGURE 4. Mesured Bandwidth



(A) Bandwidth (B) Queue size

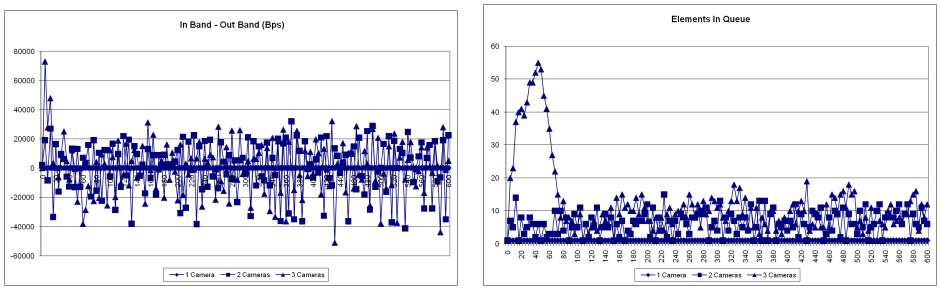
FIGURE 5. Mozilla on Linux

the same application is significantly worst in the Linux environment especially when video stream is received from more than one camera. The operating systems are not responsible for the different performances of the platform. As a mater of fact, the Incoming Bandwidth on all of the tested platforms was in the same range. The differences ware because of the Outgoing Bandwidth. Mozilla Firefox has a different behavior on those environments when rendering frames. Figure 5a shows that when displaying images from three different cameras the difference between incoming and outgoing bandwidth is quite high, which produces an abrupt accumulation of frames in the waiting queue, as it can be seen from Figure 5b. From this increase of the waiting queue size results an unacceptable delay in the video stream.

4.3. **Adaptive stream control.** For using an adaptive stream control, the stream servers need to have a mechanism for setting the prescribed value for

client bandwidth. Exploiting this facility could improve applications performance on some platforms by reducing delays in stream, especially when a large number of devices are observed. Because the internet bandwidth can vary in time and the application is an interactive one where the number of devices from which stream is received can vary in time, an adaptive control mechanism has to be implemented. A possible solution would be to introduce a new XPCOM component responsible for gathering parameter values measured by the channels and taking control decisions according to them.

In the proposed model the Band Management Control component has a passive role. Every channel reports periodically to it the Incoming Bandwidth. The Outgoing Bandwidth is also computed periodically. The adaptive control algorithm is using those two values along with the Queue Size for computing the maximum Incoming Bandwidth for each channel. This value is send to the streaming server, which in turn will set the prescribed value for client bandwidth. Because out stream server accomplishes this by dropping some frames video quality will decrease but there will not be any delays, thus maintaining the real-time quality of the stream. Test results are presented in Figure 6a and Figure 6b.



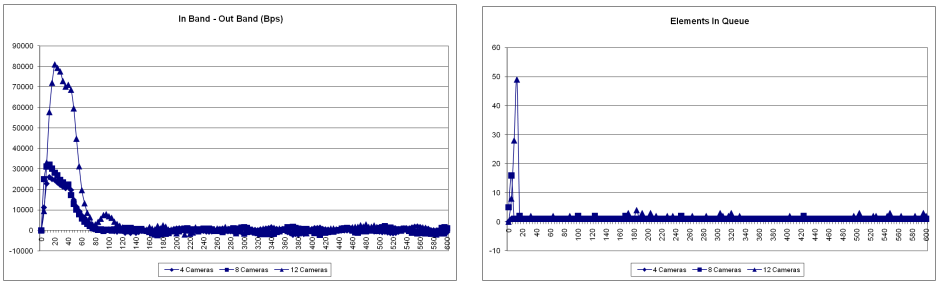
(A) Controlled bandwidth (B) Queue size

FIGURE 6. Adaptive stream control

Figure 6b clearly shows that when bandwidth control is present, even if images from 3 cameras are received, the size of the waiting queue decreases and then maintains its value in a relatively small interval, under 20 frames. Taking into account that the number of frames captured from a camera in one second ranges between 7 and 10 frames this produces only a small, acceptable, delay in the video stream.

4.4. **Scalability issues.** Further tests have been made in order to demonstrate that the model we proposed is scalable. Four separate channels were used each handling up to 3 different streams summing up to a total of 12

simultaneous video streams. The system performance can be seen in Figure 7a and Figure 7b. These are the results only for one operating system but the others behave in a very similar manner. Figure 8 shows the response in case of a variable number of streams.



(A) Bandwidth (B) Queue size

FIGURE 7. Scalability response

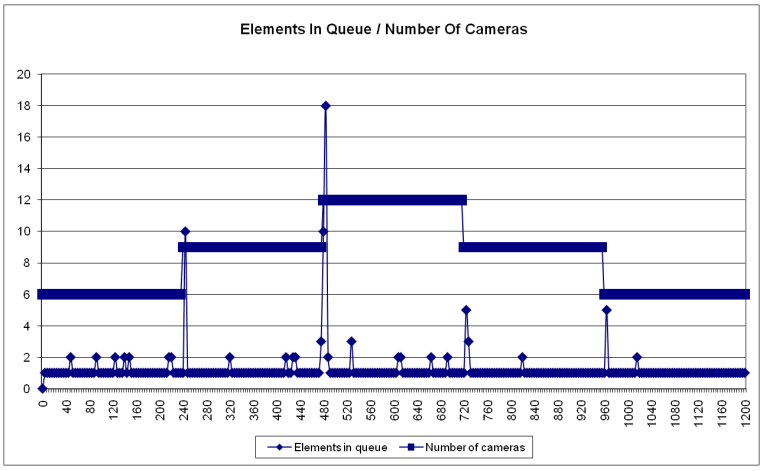


FIGURE 8. Response to variable number of streams

5. Conclusions and future work

In this paper we proposed a component based, real time streaming, portable, Web platform for distributed multimedia applications, developed on the Mozilla Platform.

Based on this model we implemented a set of components that can be used in any kind of multimedia streaming application. Furthermore a monitoring and control mechanism was presented, which allows the application to dynamically change transfer rates in order to reduce delays in the stream caused by slow presentation rates.

This approach also simplifies the development of multimedia centered applications and ensures their transparency, portability and performance. By providing a unique interface for all supported channel types, application developers can easily change the underlying transport and channel type.

As future work we intend to extend the adaptive stream control mechanism and to prepare the multimedia platform for usage in a dynamic QoS environment. This means that applications could fine tune the adaptive control mechanism in such a way that different types of multimedia content should be treated different. What this means is the fact that audio streams could be preferred over video ones or even the other way around if necessary.

References

[1] Don Box et all, *Simple Object Access Protocol*, http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[2] Jeffrey C. Broberg, *Glossary for the OASIS WebService Interactive Applications*, http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm

[3] Robert A. van Engelen and Kyle Gallivan, *The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks*, in the proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), pp. 128-135, May 21-24, 2002.

[4] Alan Grosskurth, Ali Echihabi, *Concrete Architecture of Mozilla.*

[5] H. Liu and M. Parashar, *Rule-based Monitoring and Steering of Distributed Scientific Applications*, in International Journal of High Performance Computing and Networking (IJHPCN), issue 1, 2005.

[6] M. Lohse, M. Repplinger, P. Slusallek, *An Open Middleware Architecture for Network-Integrated Multimedia*, in Proceedings of the Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems: Protocols and Systems for Interactive Distributed Multimedia, Portugal, pp. 327-338, 2002.

[7] Doug Turner, Ian Oeschgeri, *Creating XPCOM Components*,in Brownhen Publishing, 2003.

"Petru Maior" University of Târgu Mureş, Department of Electrical Engineering, Nicolaie Iorga St., No. 1, Târgu Mureş, 540088, Romania

E-mail address: oratoi@engineering.upm.ro, phaller@upm.ro, bgenge@upm.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

REASONING INTROSPECTION AND VISUALISATION FRAMEWORK FOR ONTOLOGY MAPPING ON THE SEMANTIC WEB

MIKLOS NAGY, MARIA VARGAS-VERA

Abstract. Cognitive support for ontology mapping systems become more and more important because the size and complexity of the results increases due to the availability of more and more ontologies on the Semantic Web. This support is especially required for reasoning and result introspection since the results need to be presented in a way that users can easily understand them. User understanding is crucial because the end users are the only one who can actually judge if a certain reasoning process is flawed or not. As such the quality and usability of the system is directly dependent on these kind of supports. In this paper we present a representation framework that can be used by different systems in order to store and visualise the reasoning behind ontology mapping.

1. Introduction

To date the quality of the ontology mapping was considered to be an important factor for systems that need to produce mappings between different ontologies. However, evaluation of ontology mapping systems has demonstrated that even if systems use a wide variety techniques, it is difficult to push the mapping quality beyond certain limits. It has also been recognised [1] that in order to gain better user acceptance, systems need to introduce cognitive support for the users i.e. reduce the difficulty of understanding the presented mappings. Further in order to improve the quality of the mapping systems these intermediary details need to be exposed to the users who can actually judge if the certain reasoning process is flawed or not. This important feedback or the ability to introspect can then be exploited by the system designers or

Received by the editors: December 6, 2009.

2010 *Mathematics Subject Classification.* 68T30, 68U35.

1998 *CR Categories and Descriptors.* I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *Representation languages*;

Key words and phrases. Artificial intelligence, Semantic Web, Ontology mapping.

This paper has been presented at the International Conference Interdisciplinarity in Engineering (INTER-ENG 2009), Târgu-Mureş, Romania, November 12–13, 2009.

ultimately the system itself through improving the reasoning processes, which is carried out behind the scenes in order to produce the end results. This ability to introspect the internal reasoning steps is a fundamental component of how human beings reason, learn and adapt. However, many existing ontology mapping systems that use different forms of reasoning exclude the possibility of introspection because their design does not allow a representation of their own reasoning procedures as data. Using a model of reasoning based on observable effect it is possible to test the ability of any given data structure to represent reasoning. Through such a model we present a minimal data structure necessary to record a computable reasoning process and define the operations that can be performed on this representation to facilitate computer reasoning. This model facilitates the introduction and development of basic operations, which perform reasoning tasks using data recorded in this format. It is necessary that we define a formal description of the structures and operations to facilitate reasoning on the application of stored reasoning procedures. By the help of such framework provable assertions about the nature and the limits of numerical reasoning can be made.

Our main objective is to establish a standard framework for ontology mapping systems that allows mapping systems to detect reasoning failures and to refine the function of reasoning mechanisms in order to improve system performance and avoid future reasoning problems. To achieve this goal, the users have the possibility to introspectively monitor the end result of the reasoning process and determine the possible causes of its failures and possibly perform actions that affect future reasoning processes. This is important aspect of the need to visualise the mapping stems from the fact that even though ontology mapping tools converge towards automatic mapping processes, users always play an important role for validating these results. Therefore, it is important that mappings are presented in a way that can easily be understood by end users and not just by specialised domain experts.

The main contribution of this paper is a reasoning introspection and visual mapping representation framework for ontology mapping that goes beyond the end results and includes the intermediary reasoning steps, which can be exploited by both end users, system designers or ontology engineers.

The paper is organised as follows. In section 2 we describe the human reasoning process, the modelled reasoning workflow and the reasoning steps that is supported by our introspection framework. In section 3 we present the visualisation components for both mapping and reasoning and discuss the level of cognitive support that can be achieved by 3D modelling. In section 4 we present our experiments with the OAEI benchmarks and in section 5 we summarise the advantages and current limitations of our framework. Section 6 presents the related work and in section 7 we draw our conclusions.

2. Representation model

2.1. **Human reasoning process.** When humans create mappings between two ontologies they rely heavily on their past experiences or existing knowledge about the domain. Experts can follow different processes due to their personal preferences, however we have considered a generalised mapping process in our scenario. First they select an initial set of terms from both ontologies that they believe can correspond to each other. At this step the candidate mappings are selected from the whole results. Candidate mappings are not more than hypothesises for the mapping that needs to be proved correct. Naturally the selected hypothesis is associated with a great deal of uncertainty, which stems from the lack of information about the context of these terms. Ideally more experts are involved in this process at the same time from probably different domains. Each expert with its own knowledge and experience selects candidate mappings from both ontologies. Once the candidate mappings are selected based on evidences that support their hypothesis they need to combine their subjective opinions into a more coherent view. This procedure ideally results in a consensus where the best mappings are selected. The process can be summed up in 5 steps:

(1) Select candidate mappings.
(2) Build hypotheses for possible mappings.
(3) Find evidence for proving that our hypothesis is true.
(4) Eliminate the terms that do not support our initial belief.
(5) Combine different beliefs into a more coherent ones i.e. reach consensus over the selected mappings.

The before mentioned process is perfectly modelled by existing evidential reasoning approaches e.g. the Dempster-Shafer theory (DS theory) of evidence. This model includes all levels of sub attributes, with the possibility of different frame of discernment. Further it is possible to derive the expected utility values directly from the combined experts' belief distributions.

When human experts create mappings across different domains they usually base the end result on some sort of consensus between different experts. Each expert examines a subset of the terms from both ontologies and using their background knowledge and experience they gradually eliminate terms from the subset till they reach the final result. Each expert goes through the same reasoning process and finally they discuss the results explaining why they have selected a particular mapping. Our proposed reasoning inspection framework intends to support ontology mapping systems that model the above-mentioned human reasoning process.

2.2. **Modelled reasoning workflow for ontology mapping.** Ontology mapping systems carry out several iterative steps (Fig. 1) to select the final mappings from a number of candidate mappings. During this process background knowledge is consulted in order to extend the original variables with ones that can possible describe the concepts that need to be matched. The overall process can be described as follows:

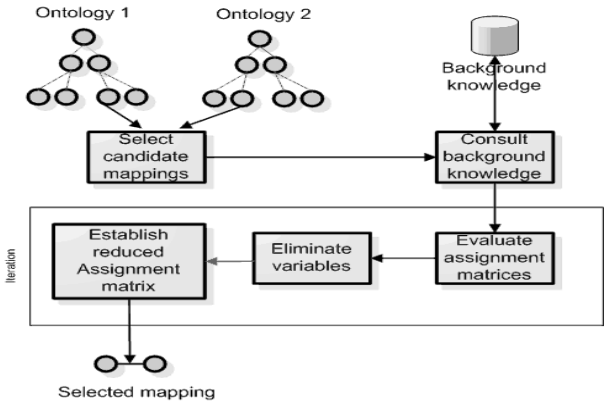


FIGURE 1. Reasoning process

(1) Select mapping candidates: In this step the system takes candidate mappings from both ontologies. The main objective of this step is to create an initial set of concepts that need to be compared to each other. Different systems can use various methods to select these candidates e.g. string similarities can be used to pre-filter concepts or a certain number of concepts are selected from both source and target ontologies.
(2) Consult background knowledge: The main objective of this step is to determine the possible meaning of the selected terms from the ontologies. Different background knowledge can be used e.g. WordNet or the Semantic Web itself. In this step the system selects a pre-defined number of additional terms that will be added to the candidate mapping sets. The system creates the preliminary assignment matrix.
(3) Evaluate assignment matrices: Using a wide variety of methods e.g. string similarity, graph structure or probabilistic information the system evaluates the initial similarities between the source and target ontologies. Different systems can use a single method or different methods, which need to be combined into a more coherent view.

(4) Eliminate variables: In this step the system selects the terms, which either above a pre-defined threshold in terms of similarity or most likely to be a match based on probabilistic information.
(5) Establish reduced assignment matrix: The system reduces the number of variables and if no clear matching is found it consults the background knowledge and starts the assignments again from step 3.

The above mentioned process is a general one. Different systems can implement differently each step however the main characteristics of these steps remain the same across different systems. Therefore in our representation framework we foresee to support any system that implements the before mentioned process.

3. Visualisation of the mapping

Visualising ontology mappings involves situations where the number of items that are concurrently displayed on the screen increases, which in turn worsens the graphical perception of the scene and complicates spotting details. If the amount of visualization space needed to represent all the mappings within the result set outnumbers the space available on the screen, a few options remain available: to scale down the whole image to the detriment of readability, to present on the screen just a portion of it and allow its navigation or to summarize the information in a condensed graph and provide means for exploration and expansion. As the effectiveness of these options depends on the task the users need to carry out, a combined usage of them offers a suitable approach. However combining these approaches using 3D space can considerably enhance[2] the productivity of the users.

Therefore in order to visualise the mappings we propose two different panes (each ontology has different pane) in 3D where the selected concepts are visible but their connections initially are not. The idea is to hide initially the network of connections as this potentially distracts the user from the details. Once the user selects a concept the system can reveal the connections to the selected concepts from the second ontology. In case the user wants to proceed for the reasoning the system reveals the reasoning states. During 3D visualization, the mappings are graphically encoded and displayed together with the different term's relationships, such as the belief in their similarities and differences. The strengths of the similarities can be represented differently using positions and several retinal variables: colour and orientation. For example once the user selects a term from the first ontology the mapped terms can be displayed where the most probable correspondences are displayed opposite to the selected term and the least probably terms are farther from the centre. Because of the data can be projected onto topological spaces with 3 orthogonal axis one can

transpose these relations into a perspective that spontaneously highlights the magnitude of the semantic relations involved between the terms. As a result the position of a term on the side of each representation can directly give its amplitude of similarity in comparison to the other ones in the group.

Recreating and visualising the behaviour of complex reasoning procedures can be accomplished via two tasks. Primarily the amount of data available to a reasoning system, which records its own actions for further use is overwhelmingly large. Thus, some process must select data that needs to be considered for the visualisation process. The second portion of this process of visualising reasoning is the formation of a mapping between the chosen data and the recent history of events. These processes are complex, however, through the development of a formal data model we hope to facilitate the development of algorithms, which accomplish these tasks as well as establishing limitations on their operations. Further the task of managing and accessing large information spaces like reasoning is a problem of large scale cognition. This is because it is hard to visualise the large number of terms and how these terms are related during the reasoning process. As we have discussed in the previous sections the gradually decreasing size of the reasoning space can be perceived as hierarchical (pyramid like structure in 3D) spaces. Our basic idea is to visualise the reasoning based on conetree [4, 5]. A cone tree is a 3D representation of a tree structure, i.e. a standard $G = (V, E)$ graph with vertices and edges. In a cone tree representation, the root of a tree (represented by a cube, a sphere or some other appropriate object) is located at the tip of a transparent cone. The children of the root node are arranged around the base of the cone. Each child can be the root node of a subtree, which is represented in a recursive fashion by a cone whose tip is located at the object representing the child. Cone tree visualisation can be improved, particularly for very large datasets, by techniques such as usage-based filtering, animated zooming, hand-coupled rotation, coalescing of distant nodes, texturing, effective use of colour for depth cueing. For representing the reasoning space for term comparisons like ontology mapping the top of the hierarchy represents the result mapping pair of terms and situated in the apex of the cone with its children placed evenly spaced along its base. The next layer of nodes is drawn below the first, with their children in cones. Cone base level diameters are reduced at each level, which ensures that the shape will form a pyramid like structure(Fig. 2). This representation allows the user to easily navigate between the layers as directional movements ensure that nothing blocks the view of cones behind the user's point of view.

Our reasoning representation model is converted into a 3D model by our visualisation framework. This model can be viewed by a virtual reality viewer

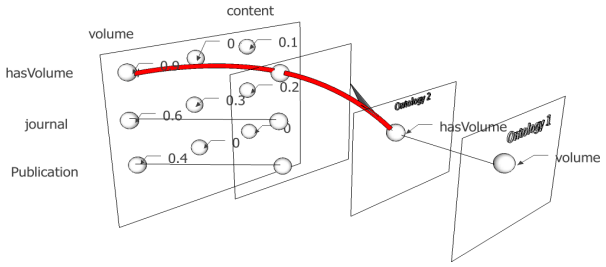


FIGURE 2. Reasoning with two steps

such as jReality [1] or different 3D modellers like Google SketchUp [2]. Therefore the user has the possibility to zoom in, out or move around the mapping model in order to discover the mappings and the reasoning model.

3.1. **Pluggable framework overview.** One of our main objective when designing the introspection and visualisation framework was that it should easily be integrated into other ontology mapping systems or 3D modellers. The conceptual overview of the systems is depicted on Fig. 3.

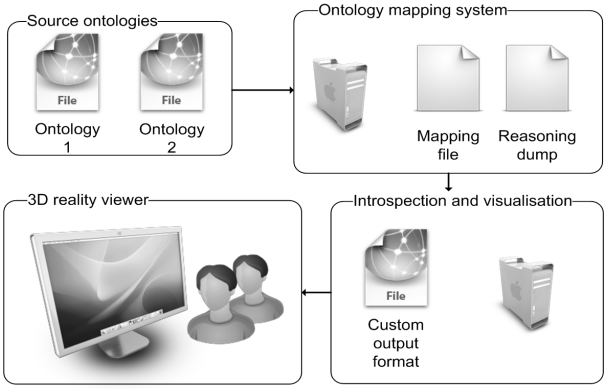


FIGURE 3. Visualisation framework overview

[1] http://www3.math.tu-berlin.de/jreality/
[2] http://sketchup.google.com/

Ontology mapping systems work with source ontologies and produce

(1) Intermediary dump files that contain the reasoning steps for the retained mappings.
(2) The result mapping file in e.g. in OAEI format.

Without visualisation this is the only output that the systems produce to the users. This approach works fine with small mapping files, however quickly become unmanageable for the users once the size of these files increase. Our introspection and visualisation framework takes these files as its input and convert it to custom 3D modells based on the representation described in section 2 . These model files can be opened with different 3D modelling tools by the user, which provides the functionality for zooming and moving around the mappings. An example interface using Google SketchUp is depicted on Fig 4.

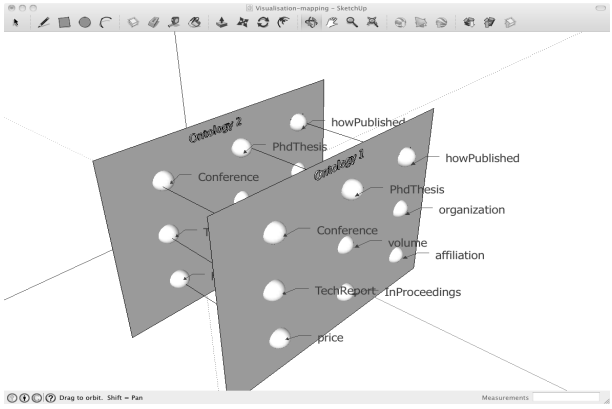


FIGURE 4. Example Google SketchUp interface

4. Experiments with the OAEI benchmarks

We have carried out experiments with the benchmark ontologies of the Ontology Alignment Evaluation Initiative(OAEI) [3], which is an international initiative that has been set up for evaluating ontology matching algorithms. The experiments were carried out in order to determine the average level of layers, terms that need to be stored and visualised to the user. This is important as the more reasoning steps we need to visualise and store the more difficult is to manage in terms of computational complexity. Our main objective was to evaluate the correlation between concepts, properties in the

[3] http://oaei.ontologymatching.org/

ontology and the number of steps(iteration) necessary to select a candidate mapping to the result set. The OAEI benchmark contains tests, which were systematically generated starting from some reference ontology and discarding a number of information in order to evaluate how the algorithm behave when this information is lacking. The bibliographic reference ontology (different classifications of publications) contained 33 named classes, 24 object properties, 40 data properties. Our results are depicted in Table 1.

TABLE 1. Number of iterations to be presented

Reasoning steps	1	2	3	4	5	6	7	8
Classes	1972	58	79	68	82	113	91	689
Properties	2722	22	46	53	55	35	30	1532

We have measured the number of cases that belong to a particular iteration number for the whole dataset. It is worth to note that the used ontology mapping system [7] limits the number of iterations per mapping selection to a maximum of 8. Therefore table 1 contains all the possible number of iterations i.e. steps that need to be presented to the user from 1 to 8. Experiments have shown that the majority of cases(1972 for classes, 2722 for properties) belong to the single step reasoning i.e. the system selects the best mapping from the first assigned similarity assignment matrix. On the second place the maximum number of iterations 8 needs to be displayed. The rest of the cases, which constitutes only a small proportion of the total cases are divided between 2-7 reasoning steps. The initial results are encouraging as it demonstrates that the reasoning and mapping visualisation needs to present a one step reasoning for the majority of the cases. However these results also show that there is a need to investigate how to give the possibility to the user to navigate through complex reasoning steps.

5. Advantages and limitations

Our proposed framework has several advantages. First of all our framework has been designed as a pluggable component that can be fitted to both ontology matching systems and industry standard 3D visualisation framework. This is an important aspect as the usability of our visualisation framework does not depend on certain tools. Therefore the users can use a wide variety of mapping tools and virtual reality viewers. Secondly the structure, concepts and relations between concepts of mapping or reasoning can be presented to the user in a graphical form to facilitate a better understanding of the results. This 3D view can then be represented in a space where shapes, sizes and locations are governed by the sizes, overlaps and other properties of the

different shapes, giving the user an intuitive feel of where the bulk of the information in the space exists. Additionally, one can manipulate the mapped view to show only those parts that appear to be the interest of the user. This view should provide a clearer picture of the relations between the resulting concepts, properties and instances.

The existing limitation of our proposed framework can be grouped into two distinct categories i.e. limitations on the storage of reasoning steps and the visualisation components. Concerning how the reasoning steps are stored our proposed framework fits well to most ontology mapping systems that use internal similarity matrixes for producing mappings. However not all mapping systems can be integrated directly with our framework. In these cases conversion needs to be done first that might not always be feasible. Concerning the visualisation components there is considerable room for improvements. Currently using different 3D modelling frameworks users can zoom and move around the 3D graph structures but the framework does not allow the interaction with the 3D structure itself i.e. users cannot change the presented structure. This can pose difficulties if the users wants to investigate a particular part of the reasoning space. Our primary research goal for the future is to find appropriate visualisation framework that could accommodate this need. Further currently it is not possible to use different filters or sorting for the mapping result set therefore out initial prototype is more suitable for mappings where the result mapping set is not too large. Nevertheless it is our future objective to investigate how different filtering and sorting possibilities can improve the visualisation of the mappings.

6. Related work

Several ontology editing tools [8, 9] provide visual interface for visualising ontologies including various plug-ins that aim to support different aspects of the ontology management lifecycle including, creation, checking and visualisation. Different tools and plugins usually implement different approached for editing or visualising these ontologies. However the most common visualisation model is 2D graph.

It has been acknowledged that interfaces will play an important role on the Semantic Web in order to gain better user acceptance. However most of the initial visualisation work [10] was carried out on representing ontologies for knowledge engineers. These techniques include tree [8] , network [11] or probably the most commonly used graph [9] representations. Most techniques display information in 2D nevertheless 3D representations [12] become popular as large scale ontologies can be presented more comprehensively for the

users. Popular 3D ontology representation technique is to project the ontology network on the sphere [12] where the most relevant element appear bigger than the irrelevant ones on the peripheries of the sphere. Other 3D technique for information visualisation is called cone or cam trees [3] where hierarchical information is arranged in circles on different levels. Levels in the tree corresponds to visual depth. This technique was proposed well before the semantic web has been conceived. Based on the ontology visualisation techniques further research has been carried out on representing ontology mappings as well. CogZ [1] is a tree based technique, which proposes various cognitive support for the decision making processes(interaction, analysis, representation) used in the mapping task. Its main objective is to reduce the cognitive load experienced by users. Other solution employs [2] a three dimensional tree based visualisation that allows for the selection of multiple class, exclusion of classes, and saving the merged classes to an OWL ontology.

Several techniques have been proposed to visualise Semantic Web Data using both 2D and 3D. Nevertheless the dominant approaches are based on graph visualisation[13, 14]. 2D solutions in particular involve certain limitations when visualizing complex networks therefore many researchers have studied different graph visualization in 3D.

7. Conclusion

The process of developing algorithms to support numerical and introspective reasoning and visualisation for ontology mapping systems requires a great deal of understanding of these domains. By specifying standard data types and approaches to these methods, it is our hope that the development of these algorithms can be further investigated. Further introspective reasoning requires a domain independent approach to reasoning technique. The model presented here demonstrates one method of achieving this domain independence in a way, which is designed to allow the development of future algorithms. Based on our proposed framework we hope that future introspective mapping systems can be developed, which contributes to a better acceptance of these systems. Our initial visualisation framework can be improved in several ways. Firstly the visual interface does not give the possibility to the users to filter the result set. These aspects of our work need to be investigated further. Secondly the users cannot change the layout of the presented mappings. In order to achieve higher level of interaction our future objective is to investigate how the possibility of changing the layout can contribute to the objective to provide a better an easier understanding of the mappings. From the contribution point of view our framework proposes a standard way of storing the states of the system during the reasoning process. This is an important aspect of the

mapping, which was not investigated so for in the context of ontology mapping. Our 3D visualisation framework build that presents the reasoning steps and the mapping itself can help the end users to navigate easily between the mappings without being overwhelmed by the complexities inherent to any 2D graph representation model.

References

[1] S.M. Falconer, M.A.D. Storey, *A Cognitive Support Framework for Ontology Mapping*, Proceedings of 6th International Semantic Web Conference (ISWC2007), 2007, pp. 114–127.

[2] D. Brokenshire, P. Lougheed, *3D Visualization – Ontology Visualization for Mapping*, Technical Report, School of Interactive Arts and Technology, Simon Fraser University, 2008.

[3] G.G. Robertson, J. D Mackinlay, S. K. Card, *Cone trees: Animated Cone trees: 3D visualizations of hierarchical information*, Proceedings of ACM SIGCHI '91 Conference on Human Factors in Computing Systems, 1991, pp. 189–194.

[4] G.G. Robertson, J. D Mackinlay, S. K. Card, *Cone Trees: animated 3D visualizations of hierarchical information*, CHI 91: Proceedings of the SIGCHI conference on Human factors in computing systems, 1991, pp. 189–194.

[5] A. P. Calitz, D. Munro, *Representation of hierarchical structures in 3D space*, AFRIGRAPH 01: Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation, 2001, pp. 59–64.

[6] B. Shneiderman, *Tree visualization with tree-maps: 2-d space-filling approach*, ACM Trans. Graph., 1/11 (1992), pp. 92–99.

[7] M. Nagy, M. Vargas-Vera, P. Stolarski, *DSSim results for OAEI 2008*, The 3rd International Workshop on Ontology Matching, 2008.

[8] Stanford Center Biomedical Informatics Research, *Protege : http://protege.stanford.edu/*, 2009.

[9] Michael Sintek, *OntoViz : http://protegewiki.stanford.edu/index.php/OntoViz*, 2007.

[10] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, E. Giannopoulou, *Ontology visualization methods-a survey*, ACM Comput. Surv., 4 (2007).

[11] P. W. Eklund, N. Roberts, S. P. Green, *OntoRama: Browsing an RDF Ontology using a Hyperbolic-like Browser*, The First International Symposium on CyberWorlds (CW2002), 2002, pp. 405–411.

[12] A. Bosca, D. Bonino, P. Pellegrino, *OntoSphere: more than a 3D ontology visualization tool*, CEUR Workshop Proceedings, 166 (2005).

[13] P. Mutton, J. Golbeck, *Visualization of Semantic Metadata and Ontologies*, IV '03: Proceedings of the Seventh International Conference on Information Visualization, 2003.

[14] I. Herman, G. Melanon, M. S. Marshall, *Graph Visualization and Navigation in Information Visualization: a Survey*, IEEE Transactions on Visualization and Computer Graphics, 6 (2000), pp. 24–43.

The Open University, United Kingdom
E-mail address: M.Nagy@open.ac.uk, M.Vargas-Vera@open.ac.uk

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

ACTION RECOGNITION USING DTW AND PETRI NETS

TAMÁS VAJDA

Abstract. This paper proposes a new approach for recognition in monocular video the human behavior sequence. We use a simple to complex approach in action recognition by decomposing it to its basic elements. The human body parts motions are tracked and classified individually. The body parts motions are matched using an adapted Dynamic Time Warping (DTW) method, an approximation of DTW algorithm that has linear time and space complexity. The adapted DTW uses a three step approach and in the second step we may eliminate the most of the incorrect template which reduces the time for comparing the entire template database. The results of the DTW matching are used to activate hierarchical Petri Nets used to classify the behavior.

1. Introduction

Recognizing human behavior from monocular video sequences is one of the most promising application of computer vision. The behavior recognition has two big issues: the first one is the human tracking which represents the measurement stage and the second is the recognition stage which is the measurement processing stage. We will focus here mostly on the second issue. The behavior recognition is challenging because of the high degree of motions, the coarsest human model is represented by 28 dimensions, and missing or erroneous measurement. Due to the high degree of motion, the actions can be often classified into several categories simultaneously. Some activities have a natural compositional structure. Behavior is composed mostly from basic action units (run and hand-wave, walk and shake hands). Even the transition between simple activities naturally has temporal segments of ambiguity and overlap. The research devoted to human motion recognition is extensive, we refer to [5, 11, 8] for comprehensive surveys. A common approach to recognize or model sequential data like human motion is the use of Hidden Markov Model (HMM) on both 2D observations [9, 12] and 3D observations. In HMM

Received by the editors: December 7, 2009.

2010 *Mathematics Subject Classification.* 68T45.

1998 *CR Categories and Descriptors.* Code [**I.4.8**]: Subtopic – *Motion*.

Key words and phrases. Behavior recognition, DTW, Petri Nets.

This paper has been presented at the International Conference Interdisciplinarity in Engineering (INTER-ENG 2009), Târgu-Mureş, Romania, November 12–13, 2009.

[1] sequential data is modeled using a Markov model that has finite states. We must choose and determine the number of states in advance for a motion, but the motion can have different time length. Therefore, it is difficult to set the optimal number of state corresponding to each motion. Recently, there has been increasing interest in using conditional random field (CRF) [2, 3] for learning of sequences. The advantage of CRF over HMM is its conditional nature, resulting in relaxation of the independence assumption, which is required by HMM to ensure tractable inference [4]. But all these methods assume that we know the number of states for every motion. Other approaches make use of templates or global trajectories of motion. Using global trajectories is highly dependent from the environment where the system is built, and can separate the composed action which introduces high interclass variation making it hard to classify the motion [6, 7]. Another problem of using global trajectories in action recognition is that it is very difficult to find the silence point which mark the possible beginning of a new action. The main contribution of this paper is the introduction of novel action representation. Using the decomposition method we can create an environment independent representation of different actions by representing every body part motion relative to its parent. The second contribution is DTW adaptation for human motion recognition. In this paper we will also demonstrate that the Petri Nets are suitable for behavior recognition.

2. Pictorial structure based human detection and posture estimation

The first step for behavior recognition is the measurement. In our case we want to know the current configuration of the human body, its relation to other moving objects, and its relation to its environment. To achieve this goal we used the Pictorial structure method introduced by Felzenszwalb [13]. In this approach the human body is modeled by a collection of parts in a deformable configuration, with "spring-like" connections between pairs of parts. These connections are modeling spatial relations between parts. Appearances and spatial relationships of individual parts can be used to detect an object. Best match of the pictorial structures depends on how well each part matches its location and how well the locations agree with the deformable model. Matching a pictorial structure does not involve making any decisions about location of individual parts; more work is to find a global minimum of energy function without any initialization.

The pictorial structure model can be represented as an undirected graph $G = \{V, E\}$, where V represents the body parts set and E represents the relation between parts. An instance of the object is given by its configuration $L = \{l_1 \dots l_n\}$ where l_i is the location of part v_i.

To find the best match of a body configuration within an image we find the L^* that minimizes the sum

$$\arg\max_{L}(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\epsilon E} d_{ij}(l_i, l_j)) \quad (1)$$

where $m_i(l_i)$ measures the cost of mismatching part v_i. with location l_i and $d_{ij}(l_i, l_j)$ measures the cost of deforming the model when placing v_i at location l_i and v_j at l_j. To measure this deformation we use Mahalanobis distance. By using the statistical framework proposed by Felzenszwalb the Pictorial structures can be viewed as an energy minimization problem in terms of statistical estimation. In this framework we need the model parameters $M = (u, E, c)$ where $u = u_1 \dots u_n$ are appearance parameters, E indicates connections between parts and $c = \{c_{ij}|(v_i, v_j)\epsilon E\}$ represents connection parameters. These parameters are learned from training examples.

Using the Posterior Sampling method, we get for every frame the optimal location and configuration of body model.We used the pictorial structure on a background-subtracted image and a feature image. As result of detection we get the body's relative configuration, the absolute position of body parts.

3. Action decomposition

Human motion can be represented in many ways: silhouette, volumetric representation of motion, temporal templates or global trajectories. We have two ways to represent motion: global trajectories, or decomposing the motion to its basic elements. Because the human motion can be compositional or concurrent, the global trajectories are not the best choice. Some actions need only legs for example walk, run, jump, and some only the hand: handshaking, waving. For this reason we decomposed the action to its basic elements - to body part motions. To make the recognition easier, we track every body part individually and relative to its parents body part. Using this approach we can use only those basic motions (body part motions) in the classification which are relevant so we can easily recognize composed motion too. The first and most significant motion is the torso motion. Here we look at two elements, the motion relative to the image (global motion) and the angular motion.

The torso represents the root of body parts in the pictorial structure. The upper legs, and upper arms are connected to the torso and we analyze only their angular motion between -270 and +270 grades. The absolute motion is tracked between -180 and +180. The 180 and 270 values represent a buffer zone. If the motion angle is above 180 or below -180, we will have two possible time series. Three events can reset one of the time series: the angular motion returns quickly between -180 and 180 degrees; the DTW matching for one of them has a strong result or the angle is increased above 270 or decreased below

FIGURE 1. Relative motion of the upper leg relative to the torso

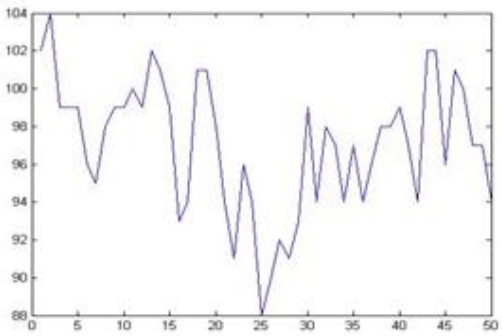


FIGURE 2. Full resolution time series of waving - upper arm

-270. (Figure 1). The lower legs are connected and their angular motions are relative to the upper legs. Also the lower arm angular motions are tracked relatively to the upper arm. We do not track the motion of the head. In Figure 2. a time series of motion is presented for the upper arm representing the waving action.

The most important point in the motion series are the peaks and the still (constant) points, because they mark a change in the motion direction. Knowing that the same action can be done at different speed the time between two direction changes in a body part motion is not so relevant.

4. Dynamic-Time Warping method for behavior recognition

Body part motions are time series in which every measurement is an element from the series with constant time periods between them. The best

way to compare saved template motion and measurement, which can be time series of different length, is the dynamic time warping (DTW) algorithm. The DTW compares two time series where we note the template series with T and the measurement with X, of lengths $|T|$ and $|X|$,

$$X = x_1, x_2, ..., x_i, ...x_{|X|} \quad T = t_1, t_2, ..., t_j, ...x_{|T|} \tag{2}$$

construct a warping path W

$$W = w_1, w_2, ..., w_|k| \quad \max(|X|, |T|) \leq k \leq |X| + |T| \tag{3}$$

where k is the length of the warping path at the n^{th} element of the warping path is

$$w_n = (i, j) \tag{4}$$

where i, j are an index from time series X, and T. Every index of both time series is to be used in the warping path and need to increase monotonically in the warp path. The minimum-distance warping path is optimal, where the distance of a warping path W is

$$Dist(W) = \sum_{n=1}^{n=k} Dist(w_{ni}, w_{nj}) \tag{5}$$

$Dist(W)$ is the distance of warping path W, and $Dist(w_{ni}, w_{nj})$ is the distance between the two data point indexes in the n^{th} element of the warp path. A two-dimensional $|X|$ by $|T|$ cost matrix D is constructed, where the value at $D(i, j)$ is the minimum distance warping path that can be constructed from the two time series $X' = x_1, x_2, \ldots, x_i$ and $T' = t_1, t_2, \ldots, t_j$. The value at $D(|X|, |T|)$ will contain the minimum-distance warping path between time series X and T. To find the minimum-distance warp path, every cell of the cost matrix must be filled.

$$D(i, j) = Dist(i, j) + \min(D(i-1, j), D((i, j-1), d(i-1, j-1)) \tag{6}$$

$D(i, j)$ is the minimum warp distance of two time series of lengths i and j, if the minimum warp distances are already known for all slightly smaller portions of that time series that are a single data point away from lengths i and j. After the entire matrix is filled, the warping path is actually calculated in reverse order performing a greedy search that evaluates cells to the left, down, and diagonally to the bottom-left starting at $D(|X|, |T|)$ to $D(1, 1)$.

Whichever of left, down, and diagonally adjacent cells has the smallest value is added to the beginning of the warping path found so far, and the search continues from that cell. The search stops when $D(1,1)$ is reached. To speed up and to avoid the marginalized warping path a slop constrain is introduced by Sakoe-Chiba [14]. Our adapted DTW version uses a merged version of Fats DTW introduced by Stan Salvador and Philip Chan [12] and the Sakoe-Chiba band constrain[14]. The adapted version of DTW algorithm uses a multilevel approach with following key operations:

(1) Shrink — Shrinks a time series into a smaller time series, that represents only the peak or constant values from the time series;
(2) Coarse DTW — Finds a minimum-distance warping path for the shrunk series and uses that warping path as an initial guess for the full "resolution's" minimum-distance warp path;
(3) Final DTW — Refines the warping path projected from a lower resolution through local adjustments of the warping path using Sakoe-Chiba constrain.

There are some major enhancement compared to the Stan Salvador and Philip Chan's FastDTW algorithm. The first is in the coarsening step. The FastDTW only computes an average of the neighborhood values and run several times to produce many different resolutions of the time series. Using this method to shrink the time series we may lose important information and get a low compression of the data. Compared to other time series in the series representing human body part motions the most significant moments are the direction changes. The shrinking average operation smooths the time series causing loss of information. In our approach instead of averaging the time series we use a heuristic selection of the data keeping only the peaks and constant values from the series. This is done by keeping only those x_i elements from X if the one of the next two conditions is true:

$$((x_i \leq x_{i-1}) \wedge (x_i \geq x_{i+1}))||((x_i \geq x_{i-1}) \wedge (x_i \leq x_{i+1})) \tag{7}$$

The resulting time series is a smaller one or equal to the original time series and we lose very low amount of information.

Figure 3 represents the original time series, of waving upper arm and the shrunken series. The second step we make a classical DTW comparison of the shrunken templates and the shrunken input. Using this comparison we can eliminate the majority of the template and only few template need to be compared at higher resolution.

Figure 4 shows the shrunk time series cost matrix and the projection of this to the original resolution cost matrix. Projection takes a warping path calculated at a lower resolution and determines what cells in the next higher resolution time series the warping path passes through. This projected path

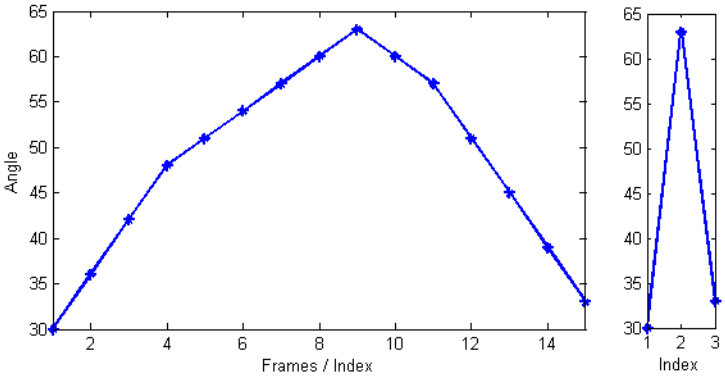


FIGURE 3. The original and the shrink time series of waving - upper arm

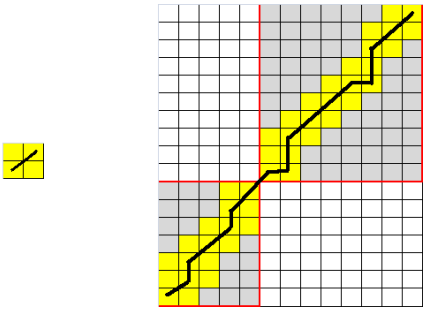


FIGURE 4. The coarse and the full resolution cost matrix with warping path

is then used as heuristic during solution refinement to find a warping path at a higher resolution. To make it faster we use Sakoe-Chiba band constrain. The Final DTW step is a refinement, finds the optimal warping path in the neighborhood of the projected path, where the size of the neighborhood is determined locally by the distance between two consecutive points in shrunk series and the difference between the length of the template series and the input series. This will find the optimal warping path through the area of the warping path that was projected from the lower resolution.

The motion templates are computed using a dataset of labeled motion. For these motions the shrunken time series is computed. We choose as starting point the median shrunken time series of a motion. Using this series we compute a mean of all time series. The resulted template will be of the same length as the median series.

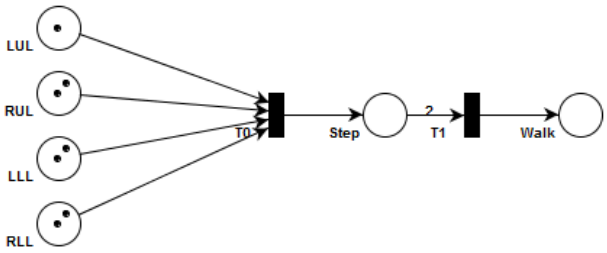


FIGURE 5. Output of the system single person

5. Behavior Recognition using Petri Nets

Human motion has two types: short timescales and basic motion (sustained (running, walking, jogging, etc.- typically periodic), punctuate (jump, punch, kick), parametric (reach, etc.)), and long timescale and complex composite motion (walking and waving, reading a book, etc.) The motion structure is a hierarchical one. The complex behavior is composed of simple motion, and the motion is composed of simple movements — "basic action". The use of Petri Net was proposed by many researchers but they applied it only to complex behavior based on the recognized basic action. We proposed the extension of this Petri Networks to recognize the basic action too, and the input for this network is represented the output of the DTW. The DTW comparison only categorize the body part motion into the classes. Each class has an associated place in the network. If the DTW categorizes a body part motion in a class the associated place get a token. Using the Petri nets synchronization procedure we can decide about the actual basic activity.

To exemplify this we model a simple walking behavior. The step state is activated by the adapted DTW and the repeated activation of the step state activates the walking state in Figure 5. In the Petri Net the states may or may not represent a behavior. By adding new labeled states we may extend the Petri net to recognize new motion.

6. Experiments

We used the detected position and the configuration of the pictorial structure to measure the speed of torso and to track the relative motion of the body parts relative to their parents. These parameters are compared to the saved templates using the adapted FastDTW and are eliminated at an early stage if the distance between the coarse variant of the series is bigger than a threshold. To construct the templates database we have annotated and saved 4 different actions from 10 different videos. For every body part we compared the saved

FIGURE 6. Output of the system single person

FIGURE 7. Output of the system two person

motion series with the adapted DTW. If the difference between them is too large they are dropped. If they are similar we choose the median series from them. To recognize the behavior a hierarchical Petri Net was used. The net was designed manually and the parameters were tuned by experiment. For experiments indoor scenes were used, with simple and composed actions. In Figures 6 and 7 we present an output of the system.

7. CONCLUSIONS

Two improvements of human action recognition have been presented: an efficient representation of motion by decomposing it to its basic elements and a FastDTW algorithm adapted for human motion recognition purpose. The angular motion representation introduced by the paper is efficient by reducing the matching problem to a 1D matching problem. By using a domain bigger than 360 degrees with the two hypothesis approach we eliminate the error introduced by the angle between +180 and -180. Using the adapted DTW the recognition is two time faster than with the FastDTW because we can eliminate many of templates at the first step at coarse comparison. Using

the motion decomposition and the hierarchical Petri Nets we were able to recognize also the composite actions such as standing and handshaking.

References

[1] L. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition. Proc. IEEE, 77 (2), 1989, pp. 257-286.
[2] J. Laffey, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", Pmc. 18th ICML, 2001, pp. 282-289.
[3] Sminchisescu C., Kanaujia A., Zhiguo Li, Metaxas, D., "Conditional models for contextual human motion" ICCV 2005. Tenth IEEE International Conference on recognition Computer Vision, 17-21 Oct 2005, vol. 2, pp. 1808-1815.
[4] Okada, S., Hasegawa, O., Motion Recognition based on Dynamic-Time Warping Method with Self-Organizing Incremental Neural Network, ICPR 2008. 19th International Conference on Pattern Recognition, 2008, pp. 1-4.
[5] J. Aggarwal and Q. Cai, Human Motion Analysis: A Review. CVIU, 73 (3), 1999, pp. 428-440.
[6] M. Black and A. Jepson, A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In ECCV, 1998, pp. 909-923.
[7] A. Blake, B. North, and M. Isard, Learning Multi-Class Dynamics. NIPS, 1999, pp. 389-395.
[8] A. Bobick and J. Davis, The recognition of human movement using temporal templates. In PAMI, 2001, pp. 257 - 267.
[9] M. Brand, N. Oliver, and A. Pentland, Coupled Hidded Markov models for complex action recognition. In CVPR, 1996, pp.994-1000.
[10] D. Gavrila, The Visual Analysis of Human Movement: A Survey. CVIU, 73 (1),1999, pp. 82-98.
[11] S. Gong and T. Xing, Recognition of group activities using dynamic probabilistic networks. In ICCV, vol. 2, 2003, pp. 742-750.
[12] S. Salvador and P. Chan, "FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space" KDD Workshop on Mining Temporal and Sequential Data, 2004, pp. 70-80.
[13] Pedro F. Felzenszwalb, Daniel P. Huttenlocher. s.l, Pictorial Structures for Object Recognition Intl. Journal of Computer Vision, 2005, pp.55-79.
[14] Sakoe H. and S. Chiba, Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoustics, Speech, and Signal Proc., ASSP-26, 1978, pp. 43 - 49.

Electrical Engineering Department, Faculty of Technical and Human Sciences, Sapientia University, Târgu Mureş, Romania

E-mail address: vajdat@ms.sapientia.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

FUZZY CLUSTERING IN AN INTELLIGENT AGENT FOR DIAGNOSIS ESTABLISHMENT

VLAD ZDRENGHEA, DIANA OFELIA MAN, AND MARIA TOSA-ABRUDAN

Abstract. In this paper we present a way to use fuzzy clustering for generating fuzzy rule bases in the implementation of an intelligent agent that interacts with human for diagnosis establishment: The Medical Diagnostics System. The system is intended to be a software learning application mainly destined to orientate the resident doctors in the process of establishing a diagnostic for the patients they are examining.

We used fuzzy c-means clustering to assign symptoms to the different types of aphasia categories. The results were compared with the results in some subtests of the Aachen Aphasia Test (AAT).

1. Introduction

Intelligent Agents Systems and Multi-agent systems are the common words that largely supplant for Distributed Artificial Intelligence (DAI) Systems. Distributed Artificial Intelligence (DAI) systems can be defined as cooperative systems where a set of agents act together to solve a given problem. These agents are often heterogeneous, for example in Decision Support System the interaction takes place between a human and an artificial problem solver. In the Medical Decision System the human operator - a resident medical doctor - is interacting with the software in order to achieve the patient diagnostic.

In DAI, there is no universal definition of "agent", but Ferber's definition is quite appropriate for drawing a clear image of an agent: "An agent is a real or virtual entity which is emerged in an environment where it can take some actions, which is able to perceive and represent partially this environment, which is able to communicate with the other agents and which possesses an

Received by the editors: December 7, 2009.

2010 *Mathematics Subject Classification.* 03B52, 93C42.

1998 *CR Categories and Descriptors.* I.2.1 [**Computing Methodologies**]: Artificial Intelligence – *Applications and Expert Systems*; I.2.3 [**Computing Methodologies**]: Artificial Intelligence – *Deduction and Theorem Proving*.

Key words and phrases. Fuzzy clustering, Fuzzy systems, Expert Systems, Uncertainty.

This paper has been presented at the International Conference Interdisciplinarity in Engineering (INTER-ENG 2009), Târgu-Mureş, Romania, November 12–13, 2009.

autonomous behavior that is a consequence of its observations, its knowledge and its interactions with the other agents" [1].

The Medical Diagnosis System is an agent kind software program that is taking somehow the role of an experienced medical person, which benefits of a vast medical knowledge regarding symptoms and diseases and haze the role to orientate the young resident doctors in the process of diagnosis establishment. The action this agent system is taken is to generate at each iteration the next more appropriate question whose answer will bring the diagnosis process closer to its end: the diagnostic of the patient. The environment the agent is able to represent is given by a knowledge database that contains general symptoms like temperature, symptoms values (e.g. 38 degrees is a symptom value for the symptom temperature) , diseases hierarchical structure, the symptom values that are associated to a disease (38 degree temperature is associated to a flue) and the relation between this disease symptom values, some may be mandatory, some may be optional and each symptom value has a weight meaning its "importance" for a specific disease. The medical diagnostic systems is not communicating with some other software intelligent agents at this time at least, but is interacting with a human operator, re-evaluates the situation and gives an new suggestion question at each iteration/ after each answer.

Medical diagnosis is an excellent field where fuzzy sets theory can be applied with success, due to the high prominence of sources of uncertainty that should be taken into account when the diagnosis of a disease must be formulated.

The medical diagnosis problem is inherently a classification problem, where for each vector of symptoms measurements one or a set of possible diagnoses are associated [2].

Fuzzy system can be designed based on expert knowledge. Several approach have been proposed to built fuzzy system from numerical data, including fuzzy clustering-based algorithms, neoro-fuzzy systems and genetic fuzzy rules generation. First, in order to obtain a good initial fuzzy system, a fuzzy clustering algorithm is used, to identify the antecedents of fuzzy system, while the consequents are designed separately to reduce computational burden. Second, the precision performance, the number of fuzzy rules and the number of fuzzy sets are taken into account. Among the different fuzzy modeling techniques, the Takagi-Sugeno (TS) model has attracted most attention.

This paper is concerned with rule extraction from data by means of fuzzy clustering in the product space of inputs and outputs where each cluster coresponds to a fuzzy IF-THEN rule.

The rest of paper is organized as follows. In Section II, the TS fuzzy model is presented, next we describe a variety of fuzzy clustering methods and present some examples. The last subsection concludes the paper.

2. Fuzzy modeling and fuzzy clustering

2.1. **Takagi-Sugeno (TS) fuzzy model.** The Takagi-Sugeno fuzzy model is a fuzzy rule-based model suitable for the approximation of many systems and function. The construction of a TS fuzzy model is usually done in two steps. In the first step, the fuzzy sets (membership function) in the rule antecedents are determined. In the second step, the parameters of the consequent functions are estimated [3].

In the TS fuzzy model, the rule consequents are typically taken to be either crisp numbers or linear functions of the inputs: R_i: IF x is A_i THEN $y_i = a_i^T x + b_i, i = 1, 2...M$, where $x \in R^n$ is the input variable (antecedent) and $y \in R$ is the output (consequent) of the i^{th} rule R_i. The number of rules is denoted by M and A_i is the (multivariante) antecedent fuzzy set of the i^{th} rule:

$$A_i(x) : R^n \to [0,1], A_i(x) = \prod_{i=1}^{n} u_{ij}(x_j) \tag{1}$$

where $u_{ij}^{(x_j)}$ is the univariate membership functions. For the k^{th} input x_k, the total output y(k) of the model is computed as follows:

$$y(k) = \sum_{i=1}^{n} u_{ki} y_i(k) \tag{2}$$

where u_{ki} is the normalised degree of the fulfilment of the antecedent clause of rule R_i:

$$u_{ik} = \frac{A_i(x_k)}{\sum_{j=1}^{M} A_j(x_k)} \tag{3}$$

2.2. **Fuzzy clustering algorithm. Fuzzy C-Means algorithm**

The most popular fuzzy clustering algorithm is *Fuzzy c-Means* (Bezdek, 1981). It is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade. It tooks several names before FCM such as: Fuzzy ISODATA, Fuzzy K-Means.

Given a set $X = \{x_1, x_2, ..., x_n\} \subset \Re^p$ of sample data, the aim of the algorithm is to determine the prototypes in such a way that the objective function is minimized.

The objective function is:

$$J(M, p_1, p_2, ..., p_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} (\sum_{k=1}^{n} u_{ik}^q d_{ik}^2)) \tag{4}$$

subject to:

$$\sum_{k=1}^{n} u_{ik} > 0, \forall i \in \{1, 2...c\}, \sum_{i=1}^{c} u_{ik} = 1, \forall k \tag{5}$$

where u_{ik} stands for the membership degree of datum x_k to cluster i, d_{ik} is the distance of datum x_k to cluster i, represented by the prototype p_i and c is the number of clusters. The parameter q ($q \in [1, \infty)$) is a weighting exponent (fuzziness exponent). Usually q=2 is chosen. At q=1, FCM collapses to HCM algorithm.

The first constraint guarantees that no cluster is empty and the second condition ensures that the sum of the membership degrees for each datum equals 1.

The output of FCM algorithm is not a partition, thus: $C_i \bigcap C_j \neq \emptyset, i \neq j$.

There are two necessary conditions for J to reach a minimum:

$$p_i = \frac{\sum_{k=1}^{n} u_{ik}^q x_k}{\sum_{k=1}^{n} u_{ik}^q} \tag{6}$$

$$u_{ik} = \frac{(\frac{1}{d_{ik}})^{1/(q-1)}}{\sum_{j=1}^{c} (\frac{1}{d_{jk}})^{1/(q-1)}} \tag{7}$$

where d_{ik} is the distance between object x_k and the center of cluster C_i [4].

FCM Algorithm

1. Initialize the membership matrix U with random values between 0 and 1 within the constraints of (2).
2. Calculate c cluster centres $p_i, i = 1..c$ using (3).
3. Compute the objective function according to (1). Stop if either it is below a certain threshold level or its improvement over the previous iteration is below a certain tolerance.
4. Compute a new U using (4).
5. Go to step 2.

Advantages:

1. Compared with the HCM, it is quite insensitive to its initialization.
2. FCM algorithm generalised the notion of membership to emulate the fuzzy clustering structures found in real-world.
3. It employs an intuitive objective function.
4. It performs robustly, thus it always converges to a solution.
5. It is simply in terms of programming implementation.
6. It minimizes intra cluster variance as well.

Disadvantages:

1. We must know the number of clusters a priori.

2. It is sensitive to initialisation.
3. It is sensitive to noise and outlier points.
4. It finds clusters of the same shape.
5. It does not use a good clustering criterion when clusters are close to one another but are not equal in size or population [5].

2.3. **Fuzzy Diagnosis.** The problem of medical diagnosis can be formalized as a classification problem, where a set of c diagnoses are defined for a certain medical problem and formalized as class labels:

$$C = \{C_1, C_2, ..., C_c\} \tag{8}$$

In order to assign a diagnosis to a patient, a set of symptoms are measured and formalized as a n-dimensional real vector $x = (x_1, x_2, , x_n)$. To perform diagnosis, a classifier is needed to perform a mapping:

$$D : X \subseteq \Re^n \to C \tag{9}$$

The domain X defines the range of possible values that each component of x can hold [2].

Our dataset consists of some cases with several attributes. Each symptoms are associated the symptoms values. For instance: Daily Disposition (Sad, Happy, Normal), Weight Change (Growth, Drop, No), Insomnia (Yes, No), Attempted Suicide (Yes, No), Social Life (Isolation, Normal), Personal outfit (Damaged, Good), Language (Vague, Normal), Delusion (Yes, No), Hallucinations (Yes, No), Thinking (Magical, Normal). We can consider four classes of diagnoses: schizophrenia, mood disorders, personality disorder, disorders due to substance use psychoactive.

2.4. **Identification by Fuzzy Clustering and Cluster Reduction.** A clustering method that has proven suitable for the identification of TS fuzzy model is the Gustafson-Kessel algorithm. Compared with Fuzzy C-Means algorithm, it employs an adaptive distance norm in order to detect clusters of different geometric shapes in the data set.

Each cluster in the product space of the input/output data, represents a rule in the rule base. The goal is to establish the fuzzy antecedents A_i in the rule (1) and these are defined by the fuzzy clusters found in the data. Univariate membership function u_{ij} can be obtained by projections onto the various input variables x_j spanning the cluster space.

2.5. **Experiments and Results. Aachen Aphasia Test (AAT).**

Aachen Aphasia Test (AAT) is publicly available at the following web address: http://fuzzy.iau.dtu.dk/aphasia.nsf/PatLight . Aphasia is the loss or impairment of the ability to use or comprehend words often a result of stroke

or head injury. Data of 256 aphasic patients, treated in the Department of Neurology at the RWTH Aachen, were collected in a database since 1986.

In aphasiology, there are many inconsistencies concerning the definition and interpretation of aphasic syndromes. In a clinical setting, the following aphasic syndromes are distinguished. These syndromes are strictly empirical and based on a statistically reliable co-occurrence of a set of symptoms: Broca's Aphasi, Wernicke's Aphasia, Global Aphasia, Anomic Aphasia, Conduction Aphasia.

Factor analysis was applied on a correlation matrix of 26 symptoms of language disorders and led to five factors (Keyserlingk et al., 2000). These factors displayed meaningful indication of the disease.

Factor-No.	Meaning
I	severity of disturbance
II	expressive vs. comprehensive
III	granularity of phonetic mistakes
IV	awareness of disease
V	deficits in communication

Table 1-Factors derived from the factor analysis

After the factors have been gained they are usually transformed into 'simple structure' to render easier interpretation of their significance. The principle of the 'simple structure' is to work out from all possible feature configurations - how scattered they may be - the ideal configuration, in which the variable possesses the simplest complexity, i.e., it can be described by only one single factor. We treated the factors with the so-called varimax method (Weber, 1980).

Fuzzy c-mean clustering (Bezdek, 1981) was then used to advise the symptoms to the different entities, because of polarization of the five factors results in at least 10 categories.

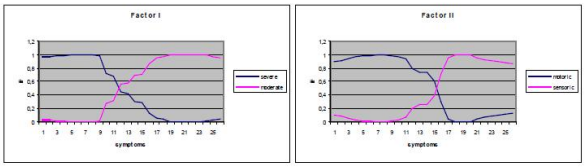


Figure 1: Graphical presentation of the results of the c-means clustering. Factor I (left) represents the overall severity of disturbance whereas factor II (right) indicates the more expressive or more comprehensive character of the language disorder

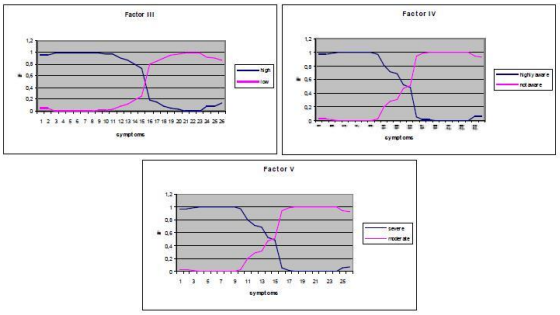


Figure 2: Graphical presentation of the results of the c-means clustering. Factor III and IV (upper row) represent the granularity of the phonetic language disorders and the patients awareness of the disease, factor V (below) exposes the deficits in communication.

The resulting classes of the clustering method are presented in the Figure . For graphical interpretation the different symptoms were put in order according to their membership to the respective feature, severe or moderate overall severity of disturbance. The clustering procedure leads to clearly distinguishable classes of symptoms. The clusters can be separated simply, as it is indicated in the small areas of overlap between the respective features. Furthermore, the description of language failures by Fuzzy C-Mean classification of analyzed factors correspond in many but not in all cases to the traditional diagnostic scheme [6].

2.6. **Conclusion.** The application of fuzzy clustering to the identification of Takagi-Sugeno (TS) fuzzy models has been addressed.

We used c-mean fuzzy clustering for classification after feature extraction from an aphasia database. The additional feature extraction allows to ensure the statistical validity of the factors. The clustering method seems to be insufficient to distinguish the granularity of the phonetic mistakes correctly. However, overall severity of the disease and the character of the language disorder can be distinguished much better.

As further work we want to use these techniques for contructing an intelligent agent that interacts with human for diagnosis establishment.

References

[1] J. Ferber, *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999.

[2] G. Castellano, A.M. Fanelli, C. Mencar *A Fuzzy Clustering Approach for Mining Diagnostic Rules*, Conference on Systems, Man and Cybernetics, 2003. IEEE International, 2, 2003, 2007–2012.

[3] M. Setnes, *Supervised Fuzzy Clustering for Rule Extraction*,IEEE Transactions on Fuzzy Systems, 8, 4, 2000, 416–424.
[4] F. Hoppner, F. Klowonn, R. Kruse, T. Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*, John Wiley and Sons, 1999.
[5] R. Kruse, C. Doring, MJ. Lesot, *Advances in Fuzzy Clustering and its Applications*, Hardcover, 2007
[6] G. Berks, D.G. von Keyserlingk, J. Jantzen, M. Dotoli, H. Axer, *Fuzzy clustering - A versatile mean to explore medical databases*, ESIT 2000, 2000, 453–457.

Iuliu Haţieganu University of Medicine and Pharmacy, Cluj-Napoca, Romania

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: vladzdrenghea@yahoo.com, {mandiana,maria}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

A PROPOSED APPROACH FOR PLATFORM INTEROPERABILITY

PAUL HORAŢIU STAN AND CAMELIA ŞERBAN

Abstract. This paper presents a new approach regarding interoperability. The novelty of the proposed solution resides in using a proxy object for each parameter object in order to avoid serialization. The main guidelines and the architecture related to developing a framework in order to automatically generate the source code for communication between platforms are proposed. Finally, there are summarized the contributions of this work and future improvements.

1. Introduction

An actual software engineering problem is the improvement of the software development process and the final product quality. In order to realize this, the source code should be reusable [8]. Obviously, it is a good news for a developer if an old library developed in Java can be used in .NET without being necessary to rewrite the code. In this case, it saves time and improves the quality of the final software system, because the Java library was tested in the past and it works fine.

There are many frameworks written for both Java and .NET, for instance: Hibernate [9] respectively NHibernate, JUnit [10] respectively NUnit etc. These frameworks could be written for a platform and reused from another platform, for instance instead of rewriting Hibernate for .NET it can be reused directly.

The current paper presents the architecture and the main guidelines related to developing a framework in order to automatically generate the source code for communication between platforms.

The proposed approach is discussed as follows: Section 2 highlights the current interoperability approaches, and the proposed solution together with the motivation of the problem and the disadvantages of actual interoperability

Received by the editors: February 5, 2010.

2010 *Mathematics Subject Classification.* 68N15.

1998 *CR Categories and Descriptors.* D.2.12 [**Software**]: SOFTWARE ENGINEERING – *Interoperability*; D.2.13 [**Software**]: SOFTWARE ENGINEERING – *Reusable Software*;

Key words and phrases. Proxy, Automatic Source Code Generation, Interoperability.

approaches. Section 3 describes in details the technical details, the theoretical concepts used in order to implement the interoperability mechanism and an implementation solution for .NET-Java. Finally Section 4 summarizes the contributions of this work and presents future research directions.

2. The Problem

The general context of this article is focused on how to access remote objects developed for different programming languages in a transparent way, like they were written for client programming language. The problem is how the parameters can be passed to the remote methods without being necessary to serialize them on client platform and restored on remote platform in order to be used there. The main idea presented in this article is that each object should be executed in the address space of the process which has created itself.

This section is composed by two parts, first presents the current interoperability approaches and the second describes the proposed solution. The proposed solution is presented as follows: the motivation of the problem, the disadvantages of current approaches and a short description of the technical solution.

2.1. **Current interoperability approaches.**

Nowadays there are many software applications that communicate and solve business problems. These applications are developed using different platforms, for instance Java, .NET, PHP, Perl, Python, Pascal etc. In order to realize the communication these applications should use the same protocol. There are many techniques used for implementing the communication between applications, for instance: COM (Microsoft Component Object Model), CORBA (Common Object Request Broker Architecture), Java RMI (Remote Method Invocation) [11], .NET Remoting [12], WEB Services based applications [2], SCA (Software Component Architecture) [3] etc.

The Component Object Model (COM) lets an object expose its functionality to other components and to host applications [16]. COM is Microsoft's initial component object model. A binary standard for the efficient interoperation across component boundaries. A COM component can implement several COM classes, each uniquely identified by a class ID (CLSID). Each COM class can implement several COM interfaces. A COM interface provides a set of operations and is uniquely identified by an interface ID (IID). A COM object is an instance of a COM class, but does not necessarily constitute a single object (split object). Clients use COM objects solely via the interfaces provided by that object. Each interface has a QueryInterface operation that can be used to ask for any of the other interfaces of the COM object based on

IIDs. COM object servers execute in processes that can be partitioned into COM apartments [1].

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together, i.e. it supports multiple platforms [15].

First and foremost, CORBA covers the specification of the interfaces of object request brokers (ORBs). An ORB accepts requests for method invocations and relays them to the addressed object, bridging platform and language gaps. An ORB also provides interface and implementation repositories that make the system fully self-describing (reflection). CORBA also covers the binding of programming languages to ORB-understood interfaces. Such interfaces are described in a standardized interface definition language (IDL). CORBA 3 added the CORBA component model (CCM), which is designed to be a superset of EJB [1].

RMI-IIOP provides interoperability with other CORBA objects implemented in various languages - but only if all the remote interfaces are originally defined as Java RMI interfaces. It is of particular interest to programmers using Enterprise JavaBeans (EJB), since the remote object model for EJB components is based on the RMI API [17].

The .NET remoting support combines context and reflection infrastructure with flexible support for proxies, channels, and messages to provide building blocks for a wide variety of communication styles and patterns [1].

Web services are similar to deployed components from a client point of view as they offer features via standard interfaces. However, services do not reveal their implementations platform and infrastructure requirements. Web services are far more self-contained than typical components or even applications. By being almost completely self-contained, services cannot be reused in different contexts, but are merely good to be used as is. To enable reuse, services would have to have explicit context dependencies and offer reconfiguration by binding their required interfaces against selected provided interfaces of other services [1].

In the Simple Object Access Protocol (SOAP) model each object is serialized on the source platform, sent to the destination platform and deserialized there. If the serialized object is very large then the overall performance decreases.

Service Component Architecture (SCA) defines a way to create components and a mechanism for describing how those components work together. SCA was originally created by a group of vendors, including BEA, IBM, Oracle, SAP, and others [4].

Every SCA application is built from one or more components. The application might contain a few components implemented as Java classes, others written in C++, and still others defined using BPEL, all spread across a group of machines [4].

2.2. **Proposed Approach.** This paper intends to presents a technique for implementing the platform interoperability and the source code reusability between programming languages.

2.2.1. *Problem Motivation.* The platform interoperability approach allows the source code written for a given platform to be called from another platform, this will determine a more reusable source code. The software development process will be improved because the code rewriting tasks are skipped, the development time is minimized and the final deliverable quality is improved.

On the other hand, the libraries reusability instead of redesigning and rewriting them implies a better software systems functionality. In many cases when a functionality is rewritten, bugs appear in project, then in the software industry it's recommended to reuse modules that works fine instead of rewriting them [3].

2.2.2. *Drawbacks of current interoperability approaches.* Some current interoperability approaches are based on the COM technology, the Java and/or .NET classes are converted to COM objects and after that are imported using native methods for Java and using COM objects for .NET. There are business solutions that use this technique for interoperability. The COM technology disadvantage is that it is operating system dependent, so it does not allow platform interoperability.

In CORBA, the server objects are managed by an object broker that can be accessed by clients in order to send messages to server objects. The limitation of CORBA is that the server objects methods could not return and could not have parameters of types defined on client side.

Unlike COM and CORBA, proposed approach is platform independent and allows remote object's methods to return and to have parameters of client side defined types.

A web services based approach limitation is the objects passed from client to server and vice-versa should be serialized. The communication process is composed of the following steps:

- *Serialize the object in an XML format*
- *Send the serialized object*
- *Deserialize the object from the XML format*

Proposed solution solves this limitation, the objects should not be serializable because proxy objects are generated and it manage the communication

with real object instead of serializing the real objects and deserializing them on the destination platform. This technique involves only few data exchanges between platforms used for creating and managing proxy objects. This solution is applicable for systems that does not involve data intensive processing but involves calling remote procedures and functions.

Unlike Web Services approaches, in the proposed solution for each remote object a proxy object is generated on the client side. This allows to avoid serialization/deserialization of huge objects. In some scenarios the overall performance is increased and in other ones the performance is decreased.

The Service Component Architecture model defines a top-down approach, in contrast with the proposed technique that is focused on reusing old software routines, that involves a bottom-up development process.

Presented solution can be used if parts of a distributed object should be used by many applications. For this use case our solution is better than SOAP model that implies serialization and deserialization of the remote object, in contrast, the proposed solution will exchange only short messages in order to manipulate the needed remote object parts.

2.2.3. *Conceptual view of the proposed solution.* The main solution idea is that each object is executed in the address space of the process which has created itself. This means these objects are not serialized, sent to remote platform and deserialized there in order to be used.

In the proposed solution for each remote object a proxy object is generated on the client side. There are three use cases for a proxy object: create itself, erase itself and invoke one of their methods, for all of the above situations the request is redirected to the remote object, when a proxy object is created a remote object is created too, when the proxy is erased from memory the remote object is erased too and when a proxy method is invoked the request is forwarded to the remote object. These three situations are presented in the figure 1.

3. Technical details

This section presents the high level architecture of the proposed approach and the implementation solution. This solution is applicable for .NET and Java, but it can be extended to support other development platforms.

We discuss the proposed approach as follows: Sections 3.1. presents the main architecture and introduces the terms and definitions used later in this article, Section 3.2 shows how the proposed solution works, finally, the main architecture of the developed interoperability framework for Java and .NET is presented in Section 3.3.

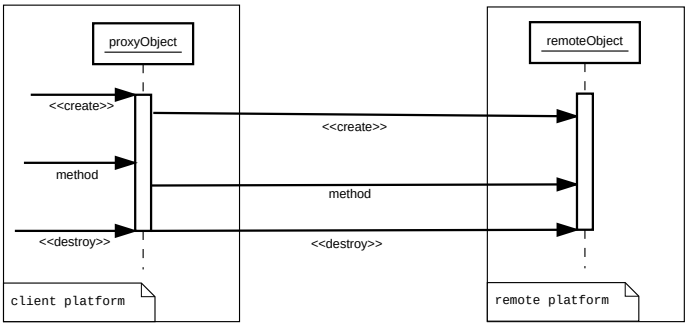


FIGURE 1. Proposed Architecture

3.1. **Architecture.** The proposed solution is based on the Proxy design pattern presented in [7].

In the proposed solution for each remote object a proxy object is generated on the client side. When the client platform creates the proxy then the server creates the real object and store them in a hash table based on an automatically generated identifier. When the client side platform erase the proxy object, for instance the garbage collector decides to remove from memory the unreferenced objects then the server side platform will remove from the remote objects hash table the real object. When a client object invokes a method on the proxy object the request is redirected to the remote object. These three situations are presented in the figure 2.

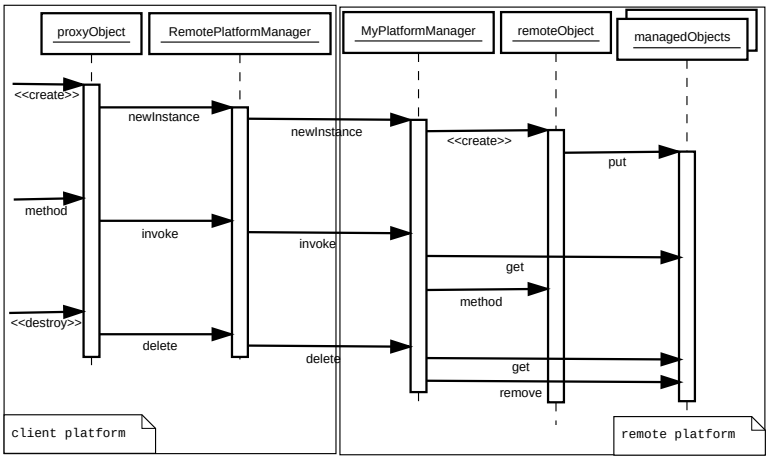


FIGURE 2. Communication Architecture

Definition 1. *RemotePlatform is the development environment and the programming language used in order to develop routines that should be imported.*

Definition 2. *ClientPlatform is the development environment and the programming language in which should be imported the remote routines.*

Definition 3. *RemoteType is a data type defined on the RemotePlatform that should be accessed from ClientPlatform.*

Definition 4. *ProxyType is a data type defined on the ClientPlatform. An instance of this type represents a proxy to an instance of RemoteType. This type is automatically generated.*

Definition 5. *RemotePlatformManager is a data type that manage the communication with RemotePlatform. This data type has been written for ClientPlatform and is used by ProxyType in order to redirect the methods requests. It is included in the proposed framework.*

Definition 6. *MyPlatformManager is a data type that manages the communication with the ClientPlatform. This data type has been written for RemotePlatform, it receives commands from RemotePlatformManager and redirects them to instances of RemoteType. It is included in the proposed framework.*

3.2. **How it works.** The general context is: there is a data type *RemoteType* on the developing platform *RemotePlatform* and it should be used in a class *C* on developing platform *ClientPlatform*. Based on the *RemoteType* an XML file is generated that specifies how the type can be used. Using this XML file the source code for the *ProxyType* is automatically generated. The *ProxyType* is written in the *ClientPlatform* programming language. This process is exposed in the figure 3.

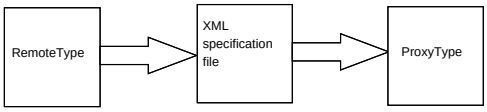


FIGURE 3. Automatically Generating the Proxy Type

After the *ProxyType* is generated it can be used on the *ClientPlatform*. The remote types in binary format should be available, that means .jar files if the remote objects are written in Java or .dll files if the remote objects are written in .NET.

On the *ClientPlatform* there is the class *RemotePlatformManager*, this class manages the communication with the *RemotePlatform*. On the *RemotePlatform* there is the class *MyPlatformManager* that manage the communication with the clients. *MyPlatformManager* stores all remote objects in

a hash table using unique identifier for each object. When a method is invoked on the proxy object the *RemotePlatformManager* delegates the responsibility to the *MyPlatformManager* witch invokes the method on the remote object. When the proxy object is disposed from client side then the remote object is deleted from hash table.

A parameter type of a proxy object's method can be in one of the following situations:

- A standard type: *String*, *byte*, *short*, *int*, *long*, *char*, *float*, *double*, *boolean*. In this case the parameter is transmitted by value.
- A proxy type. In this situation the object identifier is sent to the *RemotePlatform*, in order to identify the real object.
- A client type. In this case the parameter is stored in a hash table on the *ClientPlatform* and an automatically generated identifier is passed to the *RemotePlatform*. The *RemotePlatform* will use the identifier in order to create a proxy object for the real client side object.

Figure 4 describes the passing parameter mechanism for the situations presented above.

The return type of a proxy object's method can be in one of the following situations:

- A standard type: *String*, *byte*, *short*, *int*, *long*, *char*, *float*, *double*, *boolean*. In this case the parameter is transmitted by value.
- A proxy type. In this situation the object identifier is sent from the *RemotePlatform* to the *ClientPlatform*, on the client side a proxy object is created that represents the real remote object.
- A client type. In this case on the *RemotePlatform* there is a proxy object to a real client object. The parameter object identifier is sent from *RemotePlatform* to the *ClientPlatform* in order to identify the client real object and this client object is returned by the proxy method.

RemotePlatformManager and *MyPlatformManager* are responsible for communication between platforms. The communication can be implemented using different protocols such as: TCP/IP, shared memory, HTTP etc. In this research the TCP/IP has been chosen for sending XML data in order to realize the communication.

3.3. **The developed framework for interoperability.** During this research a framework for interoperability between .NET and Java has been developed. Current version allows only Java classes to be used in .NET. On the following three listings the request XML files for the next three scenarios are presented:

- create a remote object
- invoke a method on a remote object
- delete a remote object

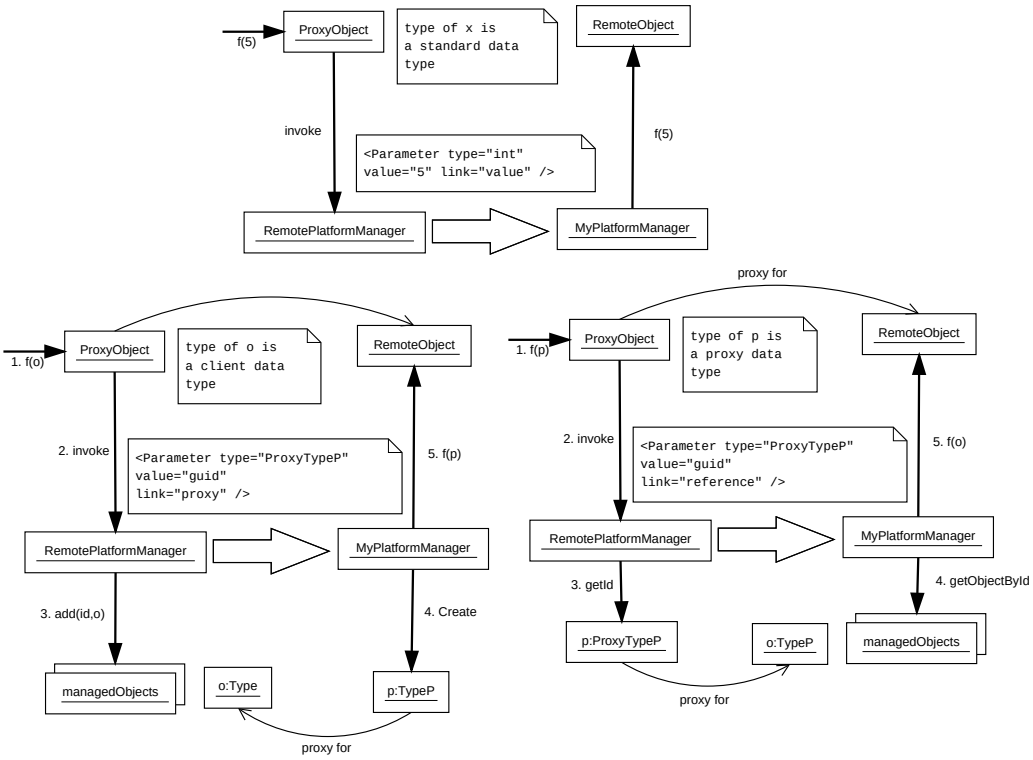


FIGURE 4. Passing parameter by value or by reference

Listing 1. Create Instance XML Request

```
<?xml version="1.0" encoding="utf-16"?>
<Request type="createInstance">
  <Class>cars.Car</Class>
  <Guid />
  <MethodName />
</Request>
```

Listing 2. Invoke XML Request

```
<?xml version="1.0" encoding="utf-16"?>
<Request type="invokeOnInstance">
  <Class />
  <Guid>63eba091-4ec1-4be1-bc9c-0f656efddc54</Guid>
  <MethodName>setPrice</MethodName>
  <Parameter type="int" value="12" link="value"/>
```

```
7 </Request>
```

Listing 3. Delete Instance XML Request

```
1 <?xml version="1.0" encoding="utf-16"?>
2 <Request type="delete">
3   <Class />
4   <Guid>63eba091-4ec1-4be1-bc9c-0f656efddc54</Guid>
5   <MethodName />
6 </Request>
```

The type attribute of the Request element can have the following values:

- *createInstance* in order to create a remote object. In this situation the element Class is used in order to identify the remote object class. A constraint on the remote object is to have a parameter less constructor. Using reflection the remote object is instantiated and an object identifier is generated and returned.
- *invokeOnInstance* in order to invoke a method on a remote object. In this case the Guid element is used to identify the remote object. The method name and the list of parameters is used to identify the remote object's method. The method is invoked using reflection and the result is returned. The response XML will be presented later.
- *delete* in order to delete an instance of an object. In this case the Guid is used to identify the remote object and to remove it from the remote objects hash table.

In the following paragraph we discuss the *Parameter* XML element. The attribute *link* can have the following two values:

- *value* in this case the *value* attribute represents the parameter value
- *reference* in this case the *value* attribute represents the object identifier for a remote object.

The *type* attribute represents the parameter type, it can be one of the following standard type *String*, *byte*, *short*, *int*, *long*, *char*, *float*, *double*, *boolean* or a full remote type name.

The following three listings present the XML responses for the three request types:

Listing 4. Create Instance XML Response

```
1 <?xml version="1.0"?>
2 <Response type="Done">
3   <ErrorMessage/>
4   <Guid>85eec14b-f6ba-4783-af87-0d69fda884c2</Guid>
```

```
5   <Value type="" value="" link="value"/>
6 </Response>
```

Listing 5. Invoke XML Response

```
1 <?xml version="1.0"?>
2 <Response type="Done">
3   <ErrorMessage/>
4   <Guid/>
5   <Value type="void" value="null" link="value"/>
6 </Response>
```

Listing 6. Delete Instance XML Response

```
1 <?xml version="1.0"?>
2 <Response type="Done">
3   <ErrorMessage/>
4   <Guid/>
5   <Value type="" value="" link="value"/>
6 </Response>
```

The *type* attribute can have the following two values:

- *Done* if the request was successfully executed. In this case the *Guid* is the object identifier if a create instance request was made and the *Value* element is the method return parameter if an invoke request was made. The *Value* has the same attributes like the *Parameter* attribute.
- *Error* if some error occurred on the *RemotePlatform*. In this case the *ErrorMessage* element contains the full message error.

4. Conclusions and Future Work

The main contributions of this paper are: a new approach of the interoperability issues between different platforms, regarding data types and objects, and a new development interoperability framework between Java and .NET, this can be easily extended to other platforms.

The novelty of the proposed solution resides in: using a proxy object in order to avoid serialization for each remote object and the parameter objects are executed in their own address space, that is the address space of the process which has created themselves.

A disadvantage of the proposed technique is the increasing of the execution time because there should be executed open connection operations and parameter values movements, but the main algorithms complexity is not affected. The proof of the proposed concept can be extended into a commercial

application server that should allow the deploy/access process for the routines written on different platforms.

5. Acknowledgment

The authors wish to thank for the financial support provided from programs co-financed by the SECTORAL OPERATIONAL PROGRAMME HUMAN RESOURCES DEVELOPMENT, Contract **POSDRU 6/1.5/S/3** "Doctoral studies: through science towards society". (Paul Horaţiu Stan)

This research has been supported by the the Romanian CNCSIS through the PNII-IDEI research grant ID_550/2007. (Camelia Şerban)

The authors would like to thank professor Bazil Pârv and professor Ioan Lazăr for their precious help.

References

[1] Clemens Szyperski, *Beyond Object-Oriented Programming*, second edition, ACM Press New York 2002
[2] Ethan Cerami, *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI and WSDL*, O'Reilly Media, Inc. February 2002.
[3] Schach, Stephen R., *Object-Oriented Software Engineering* McGraw-Hill Science/Engineering September 2007.
[4] David Chappell. *Introducing SCA*, July 2007.
[5] *SCA Service Component Architecture, Assembly Model Specification* 2007.
[6] B Nolan, B Brown, L Balmelli, T Bohn, U Wahli *Model Driven Systems Development with Rational Products* IBM 2008.
[7] Erich Gamma, Richard Helm, and Ralph Johnson, and John Vlissides, *Design Patterns: 50 specific ways to improve your use of the standard template library.* Pearson Education, Inc 1995.
[8] Beck Kent. *Test Driven Development: By Example.* Addison-Wesley Professional, 2003.
[9] Christian Bauer, Gavin King, *Hibernate in Action: Practical Object/Relational Mapping*, Manning Publications August 2004.
[10] Vincent Massol, *JUnit in Action*, Manning Publications November 2003.
[11] William Grosso, *Java RMI (Java Series)*, O'Reilly Media, Inc. Octomber 2001.
[12] Ingo Szpuszta, Mario Rammer, *Advanced .NET Remoting 2nd Edition*, APRESS February 2005.
[13] James Snell *Programming Web Services with SOAP*
[14] http://www.w3.org/TR/soap, accessed on April 06, 2010
[15] http://en.wikipedia.org/wiki/CORBA, accessed on April 06, 2010
[16] http://msdn.microsoft.com/en-us/library/kew41ycz.aspx, accessed on April 06, 2010
[17] http://java.sun.com/j2se/1.4.2/docs/guide/rmi-iiop, accessed on April 06, 2010

Babeş-Bolyai University, Department of Computer Science,, 1 Kogalniceanu St., 400084, Cluj-Napoca, Romania,
E-mail address: horatiu@cs.ubbcluj.ro, camelia@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 2, 2010

SMART SENSORS FOR SOFTWARE TELEMETRY SYSTEMS

ADRIAN ROMAN

Abstract. Software telemetry is a software project management approach that uses sensors attached to development tools to monitor and control the development process and the resulted products. Software telemetry proposes solutions for metrics collection cost problem and metrics decision-making problem, but it also comes with several downsides. This article identifies the problems of software telemetry and proposes the concept of smart sensors as possible solution for improving the approach.

1. Introduction

1.1. **Software measurements.** Software development is known as a slow and expensive process, often resulting in low quality products with serious reliability, usability, and performance problems. For decades, software development researchers have been attempting to control and improve software development processes and increase the quality of the final products guided by DeMarco's saying[1], "*You can neither predict nor control what you cannot measure*".

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules [2]. Software measurements refers to the characteristics of a software product or the software process and make the project management more effective. However, practitioners face various barriers in applying metrics. Due to the high cost associated metrics collection and difficulties in using them in the decision-making process, measurements are not widely used within the software organizations.

Software telemetry, a new approach in the software measurement field, is a step towards solving both the metrics collection cost problem and metrics

Received by the editors: 2010/4/25.

2010 *Mathematics Subject Classification.* 68N30, 68U35.

1998 *CR Categories and Descriptors.* K.6.3 [**Management of Computing and Information Systems**]: Software Management – *Software development*; K.6.3 [**Management of Computing and Information Systems**]: Software Management – *Software process*; D.2.8 [**Software**]: Metrics – *Process metrics*.

Key words and phrases. Software telemetry, software project telemetry, software sensor, smart sensor.

decision-making problem. Software telemetry also comes with several downsides, related to the way sensors are implemented and also related to the set of metrics collected.

This article identifies the existing problems of the software telemetry and introduces the smart sensors as possible solution.

1.2. **Software telemetry.** Software project telemetry is an approach to software project management that uses sensors to collect software metrics automatically and unobtrusively. The following important properties define software telemetry as a style of software metrics definition, collection, and analysis[3]:

- *Automatic collection of the data by sensors attached to tools which measure various characteristics of the project environment on constant basis.*
- *The data is a stream of events. Each event is time-stamped which is significant for analysis.* Data analysis in software project telemetry is focused on the changes over time of various parameters.
- *The data is accessible to both developers and the managers.* Project members have to be able to transparently access the collected data.
- *Telemetry analyses exhibit graceful degradation.* The analyses should provide value even if complete data over the entire project's lifespan is not available.
- *Analysis of data includes monitoring and control of processes as well as short-term prediction.* Telemetry analysis represents the project state at a given time and how it changes at various timescales.

Software telemetry is used in projects where the software developers work at a location which is inaccessible for manual metrics collection activities. This approach is in contrast with the software metrics data which is required to be collected by the developer. Furthermore, the software telemetry data has its focus on the changes over time in the measurements of various processes and product during development. In addition, the access to telemetry data is not restricted to software quality improvement groups, all developers and managers of a project can view the data in order to take a decision according to the future requirement of the project. Also, the telemetry analysis shows the current state of the project and the changes that occurs in it over time, the scale of which can be decided. The parallel display of more than one project state values and the pattern of their change allows for opportunistic analysis, which determine how different variable co-vary with each other.

Software telemetry addresses the *metrics collection cost problem* by automated data collection - sensors replace the manual metrics collection activities, thus, reducing operational costs. It addresses the *metrics decision-making*

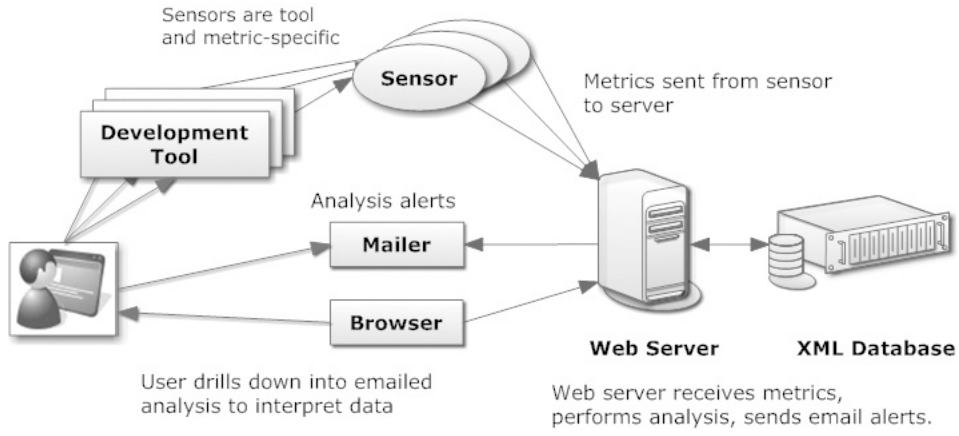


FIGURE 1. The architecture of a software telemetry system[4]

problem through high-level visual perspectives on software development process. Unlike traditional metrics approaches which are primarily based on historical project databases and focused on model-based project comparison, software project telemetry emphasizes project dynamics and in-process control.

1.3. **Software telemetry system architecture.** A software project telemetry system consists of three components: the sensor, the collector (server) application and the data visualization interface. Figure 1 illustrates the architecture of a software telemetry system[4].

Sensors are an essential part of any telemetry system. In software project telemetry systems data is collected automatically by tools / sensors that regularly measure various characteristics of the development environment. Data consists of a stream of time stamped events and developers and managers can immediately access the data.

Sensors are attached to development environments and can be implemented as plug-ins (in certain cases called *add-ins* or *add-ons*). Plug-ins are optional components which can be used to enable the dynamic construction of flexible and complex systems, passing as much of the configuration management effort as possible to the system rather than the user, allowing graceful upgrading of systems over time without stopping and restarting.

The basic function of a sensor is to collect data from the development environment and send it to a centralized collector application. In order to obtain usable and valuable data, more requirements have to be fulfilled by a sensor[6]:

- *It has to be completely integrated to the development environment.* The sensor should not interfere with the development process.
- *It has to be able to send data a centralized warehouse (server application).* A connection between the development environment and the server has to be established and the sensor has to be able to collect the data and send them to the database.
- *It has to be able to send data automatically.* The process of data collection and sending has to be automatic, with no human intervention or effort.
- *It has to be able to timestamp events.* As the data analyses are based on evolution over time, the data has to be time stamped.
- *It has to gather significant project management data (e.g. count the code lines for the current open projects).* The sensor have to be able to collect data useful for analysis, not just any data.
- *A setup has to be provided (easy deployment).* The deployment of a sensor has to be easy to perform.

Some examples of sensors used in software telemetry systems are listed below:

- *A plug-in for an IDE (integrated development environment) such as Visual Studio, and Eclipse.* It can record individual developer activities automatically, such as code editing effort, compilation attempts, and results, etc.
- *A plug-in for a version control system, such as CVS or SVN.* It can monitor code check-in and check-out activities, and compute or show the differences between revisions.
- *A plug-in for a bug tracking or issue management system, such as Bugzilla, and Jira.* Whenever an issue is reported or its status is updated, the sensor can detect such activities and record the relevant information.

1.4. **Downsides of software telemetry.** As with any other approach, the use of software telemetry has several disadvantages attached to it as well. For example, it is very much possible to misinterpret or misuse the software project telemetry data. Furthermore, the adoption of software telemetry approach for decision making and measurement purposes emphasize an increased use of tools to manage process and products which can incur additional cost in the project[3].

The main downsides of software telemetry are sum up below:

- **Sensors are tool and metric-specific. A sensor must be developed for each type of tool to be monitored.** Although, this is one time cost, every time a new development tool has to be monitored, a sensor has to be developed. Moreover, every change in the metrics required by the analysis component or in the development tool architecture results in changes to the sensor level.

- **Some metrics are not suitable for automated collection.** For example, the software development effort. It is almost impossible to construct a sensor that knows how much effort a developer has contributed to a project. The total effort represents not only the development, programming time, but also the analysis, design, thinking effort that cannot be monitored and collected by sensors attached to the development tools. It is still an open research question whether all important metrics can be captured by sensors or not.

The introduction of smart sensors addresses these problems and proposes solutions for further improvements of software telemetry.

2. Smart sensors and proposed architecture

In order to overcome the downsides previously mentioned, an extension of the existing architecture is possible. Sensors with extended properties can be designed and implemented as part of a telemetry system. A so-called smart sensor (could also be called extended sensor) should have the following properties added to the ones mentioned in section 1.2 of this article:

- *It is machine-specific sensor instead of tool-specific sensor.* This means that a smart sensor is able to monitor several applications in the same time and it is independent of the development tools. Thus, sensors are designed and implemented for every type of machine instead of every type of development tool.
- *It is able to collect an extended range of data. Not metric-specific.* The sensor must be capable of collecting a large number of metrics by design.
- *It is able to receive messages from the centralized application.* The smart sensor should be able to receive and interpret messages from the centralized application. The messages should indicate the metrics to be collected by the sensors.
- *It is able to collect and send only the required data.* In order to reduce the data overload, only required data should be monitored and sent to the centralized application.
- *It allows user interaction. The sensor should have a user interface that allows the user interaction.* The interface should allow the input of metrics that cannot be collected automatically and should not interfere with the sensor functioning process.

2.1. **Proposed architecture for smart sensor integration.** The architecture of software telemetry system [5] described in figure 1 easily integrates such a smart/extended sensor (figure 2). The main components of the system remain the same, only that data flow is a little bit different. Centralized application is responsible for both data collection and sending instructions and

messages to sensors. Management interface is used to sustain the decision process. Project managers are able to tell exactly what metrics they need to be collected for a certain period of time. Smart sensors have all the properties of

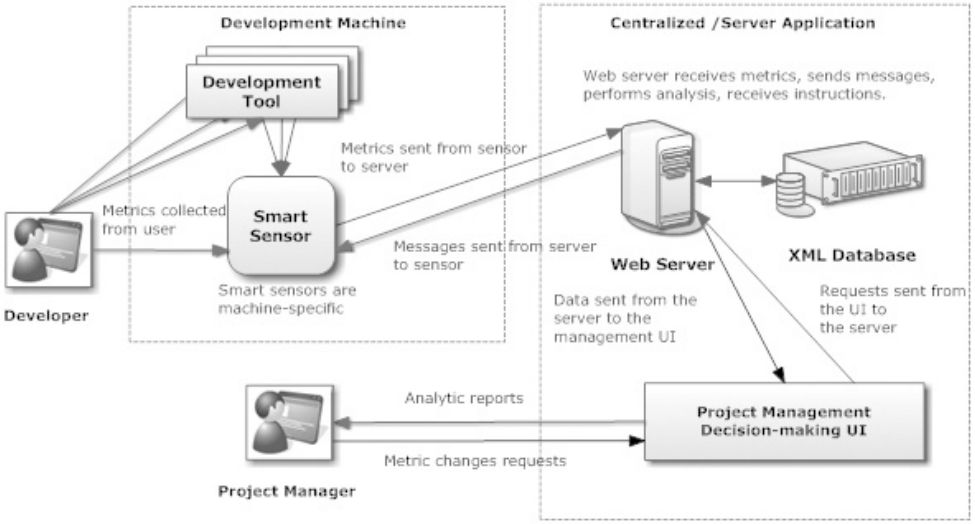


FIGURE 2. Smart sensors in a software project telemetry system

regular software sensors plus several more that are meant to solve both problems identified in Section 1.3. Table 1 shows how each problem is addressed by the smart sensor solution.

Problem	Solution
A sensor must be developed for each type of tool to be monitored	Smart sensors are machine-level. There is no need to develop a sensor for each tool. Smart sensors are able to collect an extended range of data. Not metric-specific. Smart sensors are able to receive messages and change the set of collected metrics dynamically.
Some metrics are not suitable for automated collection	Smart sensors allow user interaction. Thus, metrics that are not suitable for automatic collection can be collected through the user interface and sent to the centralized applications in the same way as data collected automatically.

2.2. **Smart sensor implementation considerations.** Creating software sensors is not easy. The developers need to know a lot about the development environment that the sensor is attached to. Plus, development environments have bugs in their plugin APIs, they are due to frequent changes and most of the time are not well documented.

Smart sensors try to solve these problems by implementing a machine-level application that is able to capture all the necessary information in a generic way, independent from the development environment. Such a sensor should be able to collect metrics like the time spent on a certain application, file or project, the idle time, the number of code lines written for different configuration and development environments. This is the biggest challenge of this approach and further study is needed as it involves high-level programming skills.

Another important feature of a smart sensor is the capability of collecting only specified metrics. The sensors receive messages from the server and collects only certain metrics. A telemetry language for the communication between sensors and the centralized application has to be designed.

The central idea of sensor-based metrics collection is that sensors are designed to collect metrics automatically and unobtrusively. Once they are installed, they work silently in the background. But smart sensor also allow user interaction. Metrics that cannot be collected automatically, such as the time spent for brainstorming or team meetings, can be introduced in the software telemetry system via a thin application that connects the user with the sensor. The data then is forwarded by the sensor to the server application.

Offline storage needs to be also considered for the situations when the development stations fail to connect to the centralized server. When the communication with the server is not possible the metrics are stored locally using XML files, a small database or open source solutions like Google Gear.

3. Conclusions and future work

The use of software project telemetry supports project management decision making. Furthermore, the automated collection of data adds significant value to software telemetry as it makes all metrics more comparable and current. However, there is always a chance that the data obtained through sensors of software telemetry can be misinterpreted or misused and it also increases the dependency of the project team on tools for managing process and products. Hence it can be said that although software telemetry approach does not provide a silver bullet to solve all problems that are associated with metrics-based project management and decision making, however, it does address the inherent problems found in traditional measurement and provides for a new approach to more local, in-process decision making.

Software telemetry presents two main downsides: (1) **sensors are tool and metric-specific** and (2) **some metrics are not suitable for automated collection**. A sensor has to be developed for every type of development tool and every change in the required set of metrics results in changes to the sensor level. Also there are metrics that cannot be collected automatically and require human input.

This article introduces the concept of smart sensors that can be used in software telemetry systems to overcome the above mentioned downsides. A smart sensor is metric-specific, is able to collect a large range of data and allows user input. Thus, sensors are implemented for every type of machine instead of every type of development tool and user can input metrics that are collected, transported and analysed by the telemetry system. Further research is need on the way smart sensors can be implemented to achieve the proposed properties, on the communication protocols to be used and on the architecture of software telemetry systems using this approach.

References

[1] T. DeMarco, *Controlling Software Projects*, Yourdon Press, 1982.

[2] N.E. Fenton, S.L. Pfeeger, *Software Metrics: A rigorous and Practical Approach*, Thomson Computer Press, 1997.

[3] P. M. Johnson, H. Kou, M. Paulding, Q. Zhang, A. Kagawa, T. Yamashita, *Improving software development management through software project telemetry*, Software, IEEE, 22(4), 2005, pp. 76-85.

[4] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, W.E.J. Doane, *Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined*, Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, 2003, pp. 641 - 646.

[5] A. Roman, *Proposed Architecture for Software Telemetry Systems*, Proceedings of the National Symposium ZAC 2008, ISBN: 978-973-610-730-6, 2008, pp.31-37.

[6] A. Roman, *Towards Building Software Project Telemetry Tools*, Proceedings of the International Conference "European Integration Between Tradition and Modernity", ISSN: 1844-2048, 2008, pp.731-734.

"Petru Maior" University of Tirgu-Mures
E-mail address: aroman@science.upm.ro