A Tool that Enables the Re-use of Experience in Enterprise Architecture Description

Paulo Tomé
Department of Computing Science, Polytechnic Institute of Viseu, Viseu, Portugal

Luís Amaral
Department of Information Systems, University of Minho, Guimarães, Portugal

Ernesto Costa
Department of Computer Science, University of Coimbra, Coimbra, Portugal

Keywords: Enterprise architecture, Enterprise modelling, Case-based reasoning.

Abstract: Enterprise Architecture (EA) is an important organization issue. The EA, resulting from a development process, is an important tool in different situations. It is used as a communication tool between the systems stakeholders. It is an enabler of changes. More importantly, the EA definition allows the build of unifying or coherent forms or structures. There has been growing interest in this topic area in recent years. Several authors have proposed methods, frameworks, languages with the aim of helping organizations in the process of EA definition. This paper proposes a software tool that can be used to re-use experience in EA definition processes. The tool proposed is independent of the framework, method, language or software tool used in the EA description process.

1 INTRODUCTION

The EA plays an important role in organizations in different situations. Nolan and Mulryan (Nolan and Mulryan, 1987) considered that architecture is a key component in the strategic planning process of companies preparing for the 21st century. First it contributes to improve a system’s structure. Moreover, the EA is an important communication tool between system’s stakeholders (Rood, 1994). Rechtin (Rechtin, 1991) considers that the architecture plays an important role in several phases of the system. Besides that, Lankhorst et al. (Lankhorst et al., 2005) considers the EA an important tool in managing a company’s daily operations and McDavid (McDavid, 1999) considers the EA important in future organization changes.

Although there is not a consensual architecture concept definition, it is currently accepted that the EA provides a global and integrated enterprise description (Bennis et al., 1998). The definition given by the Oxford Dictionary (Sykes, 1991) considers architecture:
- the art or science of building;
- thing built, structure.

Since 90’s several aspects of the EA have been greatly researched and given attention. Since then several books have been published and there are many conferences and training sessions about EA. Open Group¹, Zifa² and IRM UK³ are examples of organizations which promote conferences and training sessions around the world. Other organizations, such as IEEE (IEEE, 2000) and Carnegie Mellon University (SEI, 2003) are trying to define the concept of architecture. It is important to mention that after the first use by Amdahl et al. (Amdahl et al., 1964) the term architecture has been widely used in several Information Technology domains. Stecher (Stecher, 1993)

¹www.opengroup.org
²www.zifa.com
³www.irmuk.co.uk
considers the term architecture an umbrella for everything. The Clinger-Cohen Act\(^4\) in 1996 was other remarking decision that increased attention to the EA definition. This decision makes the US public departments to define their EA. Some of these departments, like (CIO, 1999; TEAF, 2000) created their own framework. After eleven years of work, defining the of EA continues and some of the departments are still developing their EA (GAO, 2006). It is also worth mentioning that in the 90’s attention to “systems architecting” began (Rechtin, 1991; Maier, 1996; Rechtin and Maier, 1997; Zwegers, 1998; Rechtin, 1999).

Other authors proposed frameworks, methods, languages and software tools that can be used in EA description. There are also proposals of frameworks, methods, languages and software tools for Software Architecture, Information Architecture (IA), Information Systems Architecture (ISA), Information Technology (IT) and Business Architecture domains. As explained above, the EA is on a higher level of the previously mentioned architecture (Rood, 1994), although some of them make a positive contribution to the EA.

There are several organization which have proposed software tools to develop EA descriptions. The IEAD\(^5\) publishes a list of 17 tools (IFEAD, 2007) on its website. Besides the software tools oriented to develop the EA description, some organizations use other kinds of software tools like PowerDesigner, Racional Rose, Visio, PowerPoint, Word and ErWin. It is important to notice that there is not a single software tool which is widely used. The study of the IEAD (IFEAD, 2005) concluded that Visio is the most widely used tool (33%) in the EA description process. An important characteristic of EA software tools is their data repository (EE, 2005). Most of the software tools, besides enabling the definition of the EA aspects, have a repository with the previous descriptions. However, the use of previous descriptions is no easy. The software tools users must know what they want to use, because the software tools do not have easy mechanisms for retrieving previously developed descriptions.

Architects play an important role in the EA definition process (Gore, 2003; Tandon, 2007). The architect’s experience also plays an important role in the EA quality. In the Information Systems domain it has been demonstrated that experienced analysts produce better models than young ones (Chaiyasut and Shanks, 1994).

This paper shows a software tool that enables the re-use of experience in the EA description process. The proposed tool does not intend to be a competitor to current software tools. Our intention is to propose a tool that complement the current ones. The tool enables EA descriptions to be stored on a single repository and does not depends on the framework, method, software tool or language used. Another important characteristic, the main one in fact, is the fact that the tool enables the experience to be re-used, although this would be a difficult task (Garlan et al., 1995), in the EA description process. The tool was developed based on a theory defined through the formalization of several EA elements.

It is worth mentioning that our tool wants to be similar to KB-Case (Lloyd-Williams, 1994; Tazovich, 1990) tools used in Information Systems Development. Although our tool uses the CBR (Case-Based Reasoning) method because this method it is more appropriate in ill defined domains.

In section 2 the most representative frameworks, methods and languages are presented. It was based on these elements that the theory above mentioned was developed. The theory and the software tool structure is shown in section 3, where the results obtained with the software tool application are also presented.

2 THE ENTERPRISE ARCHITECTURE

As previously mentioned, in the EA domain several authors created frameworks, methods and languages. The most representative frameworks, methods and languages are analyzed in the next tree subsections.

2.1 Enterprise Architecture Frameworks

It is generally accepted, that system’s architecture is too big and complex to consider building all at once (Nolan and Mulryan, 1987; Shah and Kourdi, 2007). For that reason, several authors have proposed EA frameworks that divide the architecture into several parts. In the EA domain there are two main types of frameworks. There are frameworks that define an enterprise reference architecture. The other kind of frameworks enable the EA description. This section will look into to the latter type of framework. The frameworks considered in this section will be those that formally described the EA aspects and as well as others that informally identified the EA aspects.


\(^5\)www.enterprise-architecture.info
As previously mentioned, several authors proposed frameworks that can be used to describe the EA. Each proposal defines the set of aspects that must be described in an EA. Besides that, some of the frameworks define the set of languages that should be used and the relationships that EA elements can have.

Zachman (Zachman, 1987) proposed his first framework in 1987. The framework is organized as a matrix of three columns and five rows. The columns represent the following aspects: data, processes and network which must be described in the EA, while the rows represent the different views that must be addressed in the EA. Zachman considers that the following views must be addressed: ballpark, owners, designers, builders and out-of-context. In 1992 Zachman and Sowa (Zachman and Sowa, 1992) proposed a framework extension. This last framework has the same rows but the new columns were added. The new columns correspond to these aspects: people and motivations and time. Both papers say that EA elements can have relationships, but they are not identified.

The Capgemini Enterprise created the IAF (Integrated Architecture Framework) (Capgemini, 2006). The IAF consists of five “aspect areas”: Business, Information, Information Systems, Technology Infrastructure, Security and Governance. Each of these aspects is described according to three levels of abstraction: conceptual, logical and physical. The IAF does not state the need to use specific languages in the EA description. The existence of relationships between EA elements are mentioned but the relationship types are not specified.

Although the main aim of the project was not to propose a framework, the proposal by Lankhorst et al. (Lankhorst et al., 2004; Lankhorst et al., 2005), which is service oriented, considers a three layer EA: Business layer; Application layer; and Technology layer.

For each layer Lankhorst et al. proposed a language which is described in 2.3.

Besides the software tool (EA WebModeler), Agilence, Inc. (Agilense, 2007) created an EA framework that comprises the following aspects: Business, Application, Information, Technical and Methods. In each of the previous aspects it can be defined more than a model.

The AMOS (Isaac and Leroy, 1994) framework, intended to model the ISA, is supported by the AMIS method. This framework comprises four elements: Functional, Logical, Technical and Organizational. In this framework informal languages are used. In the AMIS meta-model the relationships that architecture elements can have are identified.

The BSP method (IBM, 1984) has a phase that defines an Information Architecture. The Information Architecture is a matrix where the columns are Data classes and the rows are Processes. The Processes are identified by name as is the case with the Data classes. If one Process creates a Data classe in the cross cell a ”C” is written. If a Process uses a Data classes in the cross cell a ”U” is written.

In 1989 Earl (Earl, 1989) proposed a framework for describing IT architecture. The framework comprises four aspects: Computing, Communications, Data and Applications. Earl proposes a matrix to describing IT architecture in which the rows are these aspects and the columns are: Parameters, Schemas, Policies and Plans. Earl proposes an informal notation to describe each aspect and does not identify the relationships that each element can have.

The Gartner framework (Handler, 2007) comprises three aspects: Business, Information and Technical. Each aspect is described according to three detail levels: Conceptual, Logical and Implementation. The framework is organized as a matrix where the columns are the aspects and the rows are the detail levels. Some languages are used in the Gartner framework. For example, the lists of policy’s and rules, messaging and brokers are two examples of descriptions used in the Gartner framework.

IBM also created the IFW (IBM, 1995) for application in Financial Institutions. The framework is divided into three main parts: Organization view, Business view and Technical view. All the parts are described in a five row-matrix.

The Index framework (Boar, 1999) is organized as a 4X4 matrix. The columns of the matrix are: Inventory, Principles, Models and Standards. The rows of the matrix are the aspects: Infrastructure, Data, Applications and Organization. In the Boar’s neither the languages nor the relationships that can be used are specified.

Kim and Everest (Kim and Everest, 1994) created a framework for ISA description based on Zachman’s first framework. The framework comprises the aspects: Processes, Data, Control and Technology. For each of the aspects the authors proposed some languages. The paper also mentions that there can be relationships between the pairs of elements: (Processes, Data), (Data, Control) and (Control, Technology). The elements are related using an XREF matrix.

Rood (Rood, 1994) considered that the enterprise components are: Strategy, Corporate culture, People, Organization structure, Technology, Information, Processes and Tasks. According to Rood each of these components must be described in an EA. The author does not specify any modelling languages for the EA description components. Although the paper men-
tions that the EA elements can have relationships, the author does not identify the relationship types.

Microsoft proposed a framework in 1999 called *Microsoft Solutions Framework* (Microsoft, 1999). This framework enables the description of the following aspects: Business, Applications, Information and Technology. An informal language is used to describe each aspect.

Opdahl (Opdahl, 1996) proposed an ISA framework that consists of five dimensions: Resources, Guidelines, Agents, Activities and Responsibilities. The author does not propose any modelling language nor does he identify the relationship types.

The framework proposed by Tapscott and Caston (Tapscott and Caston, 1993) is for the IT architecture definition. These authors consider that five views must be defined in an IT architecture: Business view, Information view, Technical view, Applications view and Work view. No modelling language or relationship type is identified in this framework.

The Treasury Enterprise Architecture Framework (TEAF) (TEAF, 2000) was created by the Department of the Treasury Chief Information Officer Council. The framework is organized as a 4X4 matrix, where the columns are: Functional view, Information view, Organizational view and Infrastructure view and the rows are: Planner perspective, Owner perspective, Designer perspective and Builder perspective. Some modelling are mentioned in the report. The existence of relationships are also mentioned but they are not identified.

The NIH (National Institutes of Health) Enterprise Architecture framework (NIH, 2007) is based on the FEAF framework. The NIH framework comprises the definition of: Business architecture, Information architecture and Technology architecture. In the Business architecture it is addressed the following aspects: What do they do?, Who does it?, Which Information? and Where is it done?. The Information architecture described: Data, Integration and Applications. The Technology architecture comprises the aspects: Security, Data technology, Collaboration Platforms, Applications technology, Integration technology, Networks and Systems management.

The ARIS framework (Scheer, 1999) comprises five aspects: Organization, Data, Control, Function and Output. For each aspect some formal and informal languages were proposed (Scheer, 1998). Scher proposed a ARIS repository to store all of the models. In the Control view the relationships between views are defined.

The ISO/IEC (ISO/IEC, 1998) created a framework for ODP (Open Distributed Processing) systems. The viewpoints considered in the RM-ODP are: Enterprise, Information, Computational, Engineering and Technology. For each viewpoint a modelling language was proposed. The concerns of consistency between viewpoints are defined in the ISO/IEC document.

It is important to mention in this section the *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* (IEEE, 2000). Although this IEEE recommended practice is oriented for software systems, it can be considered that the framework can be used in other domains. The framework defined in the recommended practice does not have any specific aspect. In the framework it is proposed that an architecture is defined in a set of Views. Each view is defined according to a point-of-view.

### 2.2 Enterprise Architecture Methods

The work of Nolan and Mulryan (Nolan and Mulryan, 1987) is one of the oldest methods. The Nolan and Mulryan’s proposal can be considered an EA method, because it defines the phases of an EA definition process.

The AMIS method (Isaac and Leroy, 1995) intends to be used in ISA modelling. The method has two main phases: analysis-decomposition and design-composition. In the analysis-decomposition phase the functional and the organizational aspects of the AMOS framework (presented above) are described, whilst in the design-composition phase the technical and logical aspects are described.

The Institute of Standards and Technology (NIST) defined an EA model (NIST, 1989) in 1989. According to this model an EA is defined through the following ordered phases: Business architecture, Information architecture, Information systems architecture, Data architecture and Delivery systems architecture.

It is important to mention that some of the United States Public Departments EA frameworks are based on this model.

Ryan and Santucci (Ryan and Santucci, 1993) proposed a method for the development of the Enterprise Information Architecture (EIA). According to these two authors, an EIA must be developed in seven layers and in this order: environment, business requirements, data architecture, application architecture, infrastructure, system software and hardware. The authors do not purpose any specific modelling languages nor specifies any kind relationship type.

The CIO Council created the FEAF (CIO, 1999). Although the authors considered FEAF a framework, we call it a method because the FEAF defines the EA description steps. Four steps are proposed in the
FEAF to define an EA, called Level I to Level IV. The CIO Council identified eight EA components: Architecture drivers, Strategic direction, Current architecture, Target architecture, Transitional process, Architectural segments, Architectural models, Standards. In the FEAF there are no references to the languages that must be used nor to the relationship types of an EA.

The Los Alamos National Laboratory (Alamos, 1994) created a method called Information Architecture for the planning of the laboratory-wide computing, information and communication activities. The method comprises the phases: *Grounding in principles*, *Design and implementation planning* and *Implementation*.

The Open Group proposed a method called TOGAF Architecture Development method (Group, 2002). The method has eight main phases. Each phase has sub-phases. In the TOGAF report several modelling languages it were mentioned. Activity models, Use Cases and Class models are examples of languages that can be used in the TOGAF Architecture Development Model.

Targowski and Rienzo (Targowski, 1996) proposed a method for EA definition that consists of five steps:

- Translate an organization’s mission, goals and strategy into an information mission, goals, and strategies through business planning;
- Identify the information needs of the business functions through the Enterprise Processive Model;
- Create a Federate Information Systems architecture of the Information Management Complex, utilizing the Bill of Systems Processor technique, incorporating functional information needs of the business as well as information mission and goals. Systems planning should recognize independent computerized applications already in place.
- Develop Data/Information/Knowledge, Company Software, Computer Systems, and Communication Networks architectures.
- Design the architecture of the overall Information Management Environment showing system connectivity and integration through graphic design.

In this method informal languages for modelling each EA aspect are used.

Spewak and Hill (Spewak and Hill, 1995) created EAP (Enterprise Architecture Planning). This method creates the top two layers of Zachman framework (first framework). In this proposal it were considered five layers. In the third layer it is defined: *Data architecture, Applications architecture and Technology architecture*.

### 2.3 Architecture Description Languages

Architects use languages to describe the several EA aspects. Few languages were created with the main purpose of describing the EA (Architecture Description Languages (ADLs)) aspects. This kind of situation does not happen in Software Architecture domain, where, for example, there are several ADLs (Medvidovic and Taylor (Medvidovic and Taylor, 2000) describe thirteen ADLs).

The languages used in EA description aspects are:

- Languages with a low level of formality;
- Languages created in other domains;
- ADLs;

In the following paragraphs the main features of each type of language are described.

As previously mentioned, some EA aspects are described through informal languages. For example, the organization’s computer networks are generally described with an informal language where each graph node is a computer or a switch or a router or other kind of device and the edges are links between devices.

The flowchart (Chapin, 1970) is perhaps the oldest language used to describe processes. This language has four main constructors: *Process, Input-output, Flow and Decision*.

Another older notation is the Data Flow Diagram (Gane and Sarson, 1979). This language, which is widely used in the Information Systems (IS) domain, has four types of constructors: *Process, Data store, Flow and External entity*.

The National Institute of Standards and Technology created the IDEF0 (Integration Definition for Function Modeling) (Technology, 1993) and the IDEF1X (Integration Definition for Information Modeling) (FIPS, 1993) languages. The IDEF0 language consists of the following constructors: *Activity, Input, Control, Output, Call* and *Mechanisms*. The IDEF1X language can be used to model the data aspect. This language has the following constructors: *Entity, Attribute, Relationship and Relation categorizationship*.

In the Archimate project (Lankhorst et al., 2004; Lankhorst et al., 2005) three ADLs were created. These ADLs were created for the three layers considered in the project: Application, Business and Technology.

Eertink et al. (Eertink et al., 1999) proposed a language, called AMBER, which allows processes, data and the organization and people involved to be modelled. The AMBER language recognizes three aspect
domains: Actor domain, Behaviour domain and Item domain.

Chen (Chen, 1976) created the most popular Entity-Relation (ER) language. An ER model consists of Entities, Attributes and Relationships. It is important to notice that in the Chen notation the attributes are drawn as nodes and are not placed inside entities like in other ER notations.


The Object Management Group created UML (Unified Modeling Language) (OMG, 2007). UML comprises thirteen languages that can be grouped into structure, behaviour and implementation groups.

BPMN (OMG, 2004) was also created by the Object Management Group. This language comprises several components and connectors constructors. Event, Activity and Gateway are examples of components constructors while sequence flow and message flow are connectors constructors.

The Merise/2 (Panet and Letouche, 1994) is a complete set of tasks, models and techniques for Information Systems Development. The authors of Merise/2 proposed a set of nineteen languages divided into three groups: conceptual models, organizational models and logical models.

In the ISO/IEC 10746-1 (ISO/IEC, 1998) five modelling languages are proposed: Enterprise language, Information language, Computational language, Engineering language and Technology language. The aim of an Enterprise specification is to express the objectives and policy constraints on the system of interest. Through the Information language the relationship and behavior of ODP information objects are specified. The Computational language enables the distribution of processing to be modelled. The Engineering language focuses on the way object interaction is achieved and on the resources needed to do so. Through the Technology language the implementation of the ODP system in terms of a configuration of technology objects representing the hardware and software components of the implementation is described. The basic modelling concepts, common to all languages, are: Objects, Interfaces and Interaction points.

3 THE RE-USING OF EXPERIENCE IN ENTERPRISE ARCHITECTURE

In this section we describe the conceptual framework that supports the TREEAD development. Afterwards, we deal with the TREEAD’s structure and results. The conceptual framework was based on the frameworks, methods and languages presented in sections 2.1 to 2.3. The proposed conceptual framework is built using Grammar Attribute (GA) formalism (Wilhelm and Maurer, 1996). The GA is a rigorous formalism that simultaneously has synthesizing and inheriting mechanisms enabling the identification of object’s attributes.

Our proposal is compatible with all frameworks, methods and languages available in the EA domain. As mentioned in the sections 2.1 to 2.3, we consider, as shown in Figure 1, that the EA has three types of objects: View, Model and Constructor. Each type of object has different kinds of attributes: proper attributes, synthesized and inherited attributes. The latter comes from the objects in an upper level and lower level, respectively.

As explained in sections 2.1 and 2.2, an EA comprises the definition of more than one aspect. Following the IEEE recommended practice (IEEE, 2000), we consider that an EA aspect is described as a view, where each level of detail corresponds to a view. In Table 1 several frameworks and their corresponding views are described.

Specification 1 describes the EA as a set of views. Besides that, there are also attributes: method (me), framework (fr), scope (sc), organization type (org), terms that characterize the EA (Ea terminology) and keywords. The method attribute registers the method used in the EA description process. The framework used in the EA description is registered in the attribute framework. The scope attribute registers the type of EA, which can be the entire organization, a department or a section description. The organization class is registered in the type of organization org attribute.
Table 1: Some Examples of framework’s views.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMOS (Isaac and Leroy, 1994)</td>
<td>Functional; Logical; Technical; Organizational</td>
</tr>
<tr>
<td>BSP Information Architecture (IBM, 1984)</td>
<td>Data; Process</td>
</tr>
<tr>
<td>Earl (Earl, 1989)</td>
<td>Each element of the 4X4 matrix</td>
</tr>
<tr>
<td>Gartner Group (Handler, 2007)</td>
<td>Each element of the 3X3 matrix</td>
</tr>
<tr>
<td>Rood (Rood, 1994)</td>
<td>Strategy; Corporate culture; People; Organization structure; Technology; Information; Processes; Tasks</td>
</tr>
<tr>
<td>IFW (IBM, 1995)</td>
<td>Each element of all matrices</td>
</tr>
<tr>
<td>Index (Boar, 1999)</td>
<td>Each element of 4X4 matrix</td>
</tr>
<tr>
<td>Microsoft Solutions Framework (Microsoft, 1999)</td>
<td>Business; Applications; Information; Technology; Processes; Tasks</td>
</tr>
<tr>
<td>Opdahl (Opdahl, 1996)</td>
<td>Resources; Guidelines; Agents; Activities; Responsibilities</td>
</tr>
<tr>
<td>Tapscott and Caston (Tapscott and Caston, 1993)</td>
<td>Business view; Information view; Technical view; Applications view; Work view</td>
</tr>
<tr>
<td>Targowski and Rienzo (Targowski, 1996)</td>
<td>Data/Information; Software; Communications networks; Computer</td>
</tr>
<tr>
<td>TEAF (TEAF, 2000)</td>
<td>Each element of 4X4 matrix</td>
</tr>
<tr>
<td>Zachman (Zachman and Sowa, 1992)</td>
<td>Each element of the 6X5 matrix</td>
</tr>
</tbody>
</table>

In the terms that characterizes the EA attribute a list of terms that characterizes the organization to which the EA is developed is registered. The keywords attribute is synthesized from the view object and will be described later on.

An EA aspect is described through one or more view objects. Therefore, view objects can comprise a set of models. The Specification 2 describes a View object. This object has one proper attribute (aspect) and the inherited attributes: method (me), framework (fr), scope (sc), organization type (org.t), terms that characterize the EA. The keywords attribute is synthesized from the Model object.

An architect builds models to specify an EA aspect. The Model object, described in Specification 3, has the proper level and aspect (asp) attributes. For example, for a Data Model the level attribute can have one of the following values: conceptual, logical or physical; the aspect attribute has the value data. Furthermore, the Model object inherits the method (me), framework (fr), scope (sc), organization type (org.t), terms that characterize the EA attributes. The keywords and number of components (ncomp) attributes are synthesized from the component constructor object.

As explained in section 2.3, a constructor, described in Specification 4 and 5, can be divided in two types: component or connector. Generally, a constructor has a name (name) and some characteristics (Chs). For example, in the ER notation an attribute has the following characteristics: data type, length and type of attribute (normal, foreign key or primary key). Furthermore, we associated to the Constructor object an attribute for storing keywords (Ks).

These keywords are used to contextualize the application of the constructor. The constructor can belong to another component or can aggregate other components. In the supkeywords attribute and subkeywords stores the keywords of the owner and owned constructors, respectively. The relation attribute stores information that concerns rules used in the modelling task. For example, in modelling tasks decomposing rules are applied. Keywords of linked constructors are stored in the likedkeywords attribute. The relation attribute stores data about relations with other constructors in a different view.

After the grammar specification, we created the TREEAD whose structure is shown in figure 2. The tool has two major components: the client and the server. The architect uses the client components: a
browser and a software design tool. The browser is used to communicate with the server component. It is through the browser that the architect requests for solutions to problems and stores EA descriptions. Besides that, it is through the browser that the architect configures the tools (method, framework and languages) in the server. The server part is implemented using Oracle technologies (Oracle, 2007) and Microsoft ASPX technology (Ahmed et al., 2002).

View \rightarrow aspect S_m.
View.method = View.method;
View.framework = View.framework;
View.scope = View.scope;
View.org_type = View.org_type;
View.description = View.description;
View.aspect = aspect Value;
View.keywords = View.keywords;

S_m \rightarrow Model.
Model.method = S_m.method;
Model.framework = S_m.framework;
Model.scope = S_m.scope;
Model.org_type = S_m.org_type;
Model.description = S_m.description;
Model.aspect = S_m.aspect;
S_m.keywords = Model.keywords;

S_m \rightarrow Model S_m.
Model.method = S_m.method;
Model.framework = S_m.framework;
Model.scope = S_m.scope;
Model.org_type = S_m.org_type;
Model.description = S_m.description;
Model.aspect = S_m.aspect;
S_m.keywords = concat(S_m.keywords, Model.keywords);

Specification 2: View object description.

The 4Rs cycle proposed by Aamodt and Plaza (Aamodt and Plaza, 1994) was implemented in the server. The following modules were also implemented: Software design tools library, XML parser, Knowledge manager, Modelling manager, EA tools manager and Views and modelling languages library. The software design tools library has the knowledge about each specific view and modelling languages as well as the conversation rules between different views and modelling languages.

Model \rightarrow asp level S_a.
Model.aspmodel = asp value;
Model.level = level Value;
S_a.method = Model.method;
S_a.framework = Model.framework;
S_a.scope = Model.scope;
S_a.org_type = Model.org_type;
S_a.description = Model.description;
S_a.aspect = Model.aspect;
S_a.level = Model level;
S_a.aspmodel = asp value;
S_a.ncomp = 1;
Model.keywords = S_a.keywords;

S_a \rightarrow Co.
Co.method = S_a.method;
Co.framework = S_a.framework;
Co.scope = S_a.scope;
Co.org_type = S_a.org_type;
Co.description = S_a.description;
Co.aspect = S_a.aspect;
Co.level = S_a.level;
Co.asplevel = S_a.asplevel;
Co.ncomp = S_a.ncomp;
S_a keywords = Co keywords;

S_a \rightarrow Co S_m.
Co.method = S_a method;
Co.framework = S_a framework;
Co.scope = S_a scope;
Co.org_type = S_a org_type;
Co.description = S_a description;
Co.aspect = S_a aspect;
Co.level = S_a level;
Co.asplevel = S_a.asplevel;
Co.ncomp = S_a.ncomp;
S_m.keywords = concat(S_a keywords, Co keywords);

Specification 3: Model object description.

It is important to notice that in the retrieval and retention phases clustering techniques (Tan et al., 2006) are used. These techniques were used to enable a faster case retrieval. Access to the case-memory is achieved by using Groups of cluster links as shown in figure 2. This strategy significantly reduces the number of case assessments and consequently retrieval time.

Considering the specifications illustrated in 1 to 5, the TREED comprises four types of cases: EA, View, Model and Constructor. For each type of case, the case’s characteristics are the object’s attributes (proper, inherited and synthesized). The case’s solution part is the description of each object. The description of each object is done in XML, a widely
were created by different IT professionals. Despite the models belong to different departments types and to the order specified in the table, each example was introduced sequentially according in table 2, were used to evaluate the TREEAD tool.

A total of thirty data models, as shown developed in example 2) were defined by only one IT professional. A total of thirty data models, as shown defined by different IT professionals while the models were created by different IT professionals. Despite

Figure 2: TREEAD’s structure.

used language.

We used TREEAD for the specification of Zachman’s cell (Owner, Data) to create three examples in different contexts:

- **Example 1**: fifteen data models for different organization departments (such as sales, accounting, manufacturing and so on);
- **Example 2**: six data models of six hospital medical services;
- **Example 3**: nine data models of the same department (sales department).

All the data models were specified using the IDEF1X in the PowerDesigner (Sybase, 2006) software tool. The models developed in examples 1) and 3) were defined by different IT professionals while the models developed in example 2) were defined by only one IT professional. A total of thirty data models, as shown in table 2, were used to evaluate the TREEAD tool.

In this experimental study we want to measure the re-utilization of the tool. Then each model within each example was introduced sequentially according to the order specified in the table.

Figure 3 presents the results of Example 1) where the models belong to different departments types and were created by different IT professionals. Despite the diversity of the data models, there is a mean percentage of re-use, around 44% and 23% for connectors and components respectively. It also has to be noticed that the percentage does not decrease with an increasing number of models in the case memory.

```plaintext
Co → 'connector' name Chs Ks Sup Sub Re Li Ore.
Co type = 'connector';
Co name = name value;
Co characteristics = Chs characteristics;
Co keywords = Ks characteristics;
Co supkeywords = Sup keywords;
Co subkeywords = Sub keywords;
Co relationship = Re relationship;
Co linkkeywords = Li keywords;
Co relation = Ore number;
Co likekeywords = Li keywords;

Chs → name val.
Chs characteristics = (name value, val value);
Chs → name val Chs.
Chs characteristics = concat (Chs, (name value, val value));

Ks → val.
Ks keywords = val value;
Ks → val Ks.
Ks keywords = concat (Ks, val value);

Sup → Ks.
Sup keywords = Ks value;
Sub → Ks.
Sub keywords = Ks value;

Re → name Ks.
Re relationship = (name value, Ks value);

Li → Ks.
Li keywords = Ks value;
Li → Ks Li.
Li keywords = concat (Li, Ks value, Ks value);

Ore → ℵ.
Ore → name.
Ore relation = name value;
Ore → name Ore.
Ore relation = concat (Ore, relation, name value);
```

Specification 5: Constructor object description.

Figure 4 shows the results of Example 2) where the models belong to departments with identical purposes and were defined by the same IT professional. We can see that the percentage of re-utilization is high, above 52% and 83% for connectors and components respectively. This result has to be expected considering the similarities of the departments and the consistency of the modelling phase. And it is shown that the percentage of re-use increases with the number of cases in memory.

Figure 5 presents the results of Example 3) where
Table 2: Number of constructors of each data model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Example</th>
<th>N. of comp.</th>
<th>N. of connect.</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_1</td>
<td>1</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>M_2</td>
<td>1</td>
<td>135</td>
<td>63</td>
</tr>
<tr>
<td>M_3</td>
<td>1</td>
<td>176</td>
<td>23</td>
</tr>
<tr>
<td>M_4</td>
<td>1</td>
<td>80</td>
<td>19</td>
</tr>
<tr>
<td>M_5</td>
<td>1</td>
<td>70</td>
<td>17</td>
</tr>
<tr>
<td>M_6</td>
<td>1</td>
<td>81</td>
<td>6</td>
</tr>
<tr>
<td>M_7</td>
<td>1</td>
<td>161</td>
<td>23</td>
</tr>
<tr>
<td>M_8</td>
<td>1</td>
<td>106</td>
<td>25</td>
</tr>
<tr>
<td>M_9</td>
<td>1</td>
<td>41</td>
<td>5</td>
</tr>
<tr>
<td>M_10</td>
<td>1</td>
<td>103</td>
<td>13</td>
</tr>
<tr>
<td>M_11</td>
<td>1</td>
<td>36</td>
<td>5</td>
</tr>
<tr>
<td>M_12</td>
<td>1</td>
<td>31</td>
<td>5</td>
</tr>
<tr>
<td>M_13</td>
<td>1</td>
<td>210</td>
<td>87</td>
</tr>
<tr>
<td>M_14</td>
<td>1</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>M_15</td>
<td>1</td>
<td>88</td>
<td>5</td>
</tr>
<tr>
<td>M_16</td>
<td>2</td>
<td>281</td>
<td>69</td>
</tr>
<tr>
<td>M_17</td>
<td>2</td>
<td>406</td>
<td>93</td>
</tr>
<tr>
<td>M_18</td>
<td>2</td>
<td>367</td>
<td>78</td>
</tr>
<tr>
<td>M_19</td>
<td>2</td>
<td>391</td>
<td>88</td>
</tr>
<tr>
<td>M_20</td>
<td>2</td>
<td>230</td>
<td>61</td>
</tr>
<tr>
<td>M_21</td>
<td>2</td>
<td>89</td>
<td>394</td>
</tr>
<tr>
<td>M_22</td>
<td>3</td>
<td>73</td>
<td>19</td>
</tr>
<tr>
<td>M_23</td>
<td>3</td>
<td>112</td>
<td>33</td>
</tr>
<tr>
<td>M_24</td>
<td>3</td>
<td>112</td>
<td>29</td>
</tr>
<tr>
<td>M_25</td>
<td>3</td>
<td>58</td>
<td>12</td>
</tr>
<tr>
<td>M_26</td>
<td>3</td>
<td>40</td>
<td>9</td>
</tr>
<tr>
<td>M_27</td>
<td>3</td>
<td>78</td>
<td>17</td>
</tr>
<tr>
<td>M_28</td>
<td>3</td>
<td>55</td>
<td>12</td>
</tr>
<tr>
<td>M_29</td>
<td>3</td>
<td>51</td>
<td>11</td>
</tr>
<tr>
<td>M_30</td>
<td>3</td>
<td>61</td>
<td>15</td>
</tr>
</tbody>
</table>

the models belong to the same department and were created by different IT professionals. We can see that the percentage of re-utilization is above 37% and 43% respectively for connectors and components respectively. This result illustrates that the consistency in the modelling phase influences the outcome. The number cases in the memory is not always related to a high percentage of re-use, at least in terms of the re-use of connectors.

4 CONCLUSIONS

In this paper we proposed a software tool - TREEAD, based on CBR techniques, which enables the re-use of experience in EA description. This work also presents a careful synthesis of methods, frameworks and modelling languages used in EA description. Considering the reviewed frameworks, methods and languages the structure of four EA objects in GA formalism is presented.

The proposed TREEAD supports the use of several EA frameworks, methods, modelling languages and software modelling tools. Furthermore, TREEAD applies Clustering techniques to improve retrieval case time.

Finally we presented the results of the application of TREEAD in three different examples. In spite of
the diversity of the models due to different origins or different strategies of modelling, the percentage of re-use was always above 33% and 15% for connectors and components respectively. The results obtained lead us to conclude that the use of experience in EA description can contribute to a better system development.

REFERENCES


