TREEAD
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 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 importante 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 (Bernus 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 Group[1], Zifa[2] and IRM UK[3] 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)

[1]www.opengroup.org
[2]www.zifa.com
[3]www.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; Tauzovich, 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.

[4]www.cio.gov/Documents/it_management_reform_act_Feb_1996.html

[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 be 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 it 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 mentions 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 are 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). Scheer 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 languages 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 Oracle Corporation created two modelling languages in the CASE*Method: Entity Relationship Modelling (Barker, 1995) and Function and Process Modelling (Barker and Longman, 1992). The Entity Relationship Language has two major constructors: *Entity* and *Relationship*. The Function and Process Modelling lanugage comprises the following constructors: *Function*, *Event*, *Relation*, *Objective*, *Dependency* and *Actor*.

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.

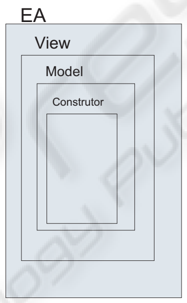

Figure 1: Hierarchy of the EA Objects.

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_t*), *terms that characterize the EA* (Ea_d) 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_type* attribute.

Table 1: Some Examples of framework's views.

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

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 $relationship$ 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 $likeywords$ attribute. The $relation$ attribute stores data about relations with other constructors in a different View.

$EA \rightarrow me\ fra\ sc\ org\_t\ Ea\_d\ S\_v.$
$EA.method = me.Value;$
$EA.framework = fra.Value;$
$EA.scope = sc.Value;$
$EA.org\_type = org\_t.Value;$
$EA.description = Ea\_d.description;$
$S\_v.method = me.Value;$
$S\_v.framework = fra.Value;$
$S\_v.scope = sc.Value;$
$S\_v.org\_type = org\_t.Value;$
$S\_v.description = Ea\_d.description;$
$EA.keywords = S\_v.keywords;$

$Ea\_d \rightarrow des.$
$Ea\_d.description = des.value;$
$Ea\_d \rightarrow des\ Ea\_d.$
$Ea\_d_0.description = concat(des.value, Ea\_d_1.description);$

$S\_v \rightarrow View.$
$View.method = S\_v.method;$
$View.framework = S\_v.framework;$
$View.scope = S\_v.scope;$
$View.org\_type = S\_v.org\_type;$
$View.description = S\_v.description;$
$S\_v.keywords = View.keywords;$
$S\_v \rightarrow View\ S\_v.$
$View.method = S\_v.method;$
$View.framework = S\_v.framework;$
$View.scope = S\_v.scope;$
$View.org\_type = S\_v.org\_type;$
$View.description = S\_v.description;$
$S\_v_0.keywords = concat(S\_v_0.keywords, View.keywords);$

Specification 1: EA object description.

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).

$View \rightarrow aspect\ S\_m.$
$S\_m.method = View.method;$
$S\_m.framework = View.framework;$
$S\_m.scope = View.scope;$
$S\_m.org\_type = View.org\_type;$
$S\_m.description = View.description;$
$View.aspect = aspect.Value;$
$S\_m.aspect = aspect.Value;$
$View.keywords = S\_m.keywords;$

$S\_m \rightarrow Model.$
$Model.method = S\_m.method;$
$Model.framework = S\_m.framework;$
$Model.scope = S\_m.scope;$
$Model.org\_type = S\_m.org\_type;$
$Model.description = S\_m.description;$
$Model.aspect = S\_m.aspect;$
$S\_m.keywords = Model.keywords;$

$S\_m \rightarrow Model\ S\_m.$
$Model.method = S\_m.method;$
$Model.framework = S\_m.framework;$
$Model.scope = S\_m.scope;$
$Model.org\_type = S\_m.org\_type;$
$Model.description = S\_m.description;$
$Model.aspect = S\_m.aspect;$
$S\_m_0.keywords = concat(S\_m_0.keywords, Model.keywords);$

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 parsing rules that enable the parsing of XML files by *XML parser*. The TREEAD's users manage the contents of the Case memory through *Knowledge manager*. In this module the user can correct and disable previously resolved problems. The *Modelling manager* is responsible the entire process of developing an EA description and it allows solutions to problems to be requested and stores new EA descriptions. The *EA tools manager* enables of new methods, views and languages to be configured. This module uses the meta-case definition to specify the meaning for a view and for a modelling language. The *Views and modelling languages library* has the knowledge about each specific view and modelling languages as well as the conversation rules between different views and modelling languages.

$Model \rightarrow asp\ level\ S\_c.$
$Model.aspmodel = asp.Value;$
$Model.level = level.Value;$
$S\_c.method = Model.method;$
$S\_c.framework = Model.framework;$
$S\_c.scope = Model.scope;$
$S\_c.org\_type = Model.org\_type;$
$S\_c.description = Model.description;$
$S\_c.aspect = Model.aspect;$
$S\_c.level = Model.level;$
$S\_c.aspmodel = asp.value;$
$S\_c.ncomp = 1;$
$Model.keywords = S\_c.keywords;$

$S\_c \rightarrow Co.$
$Co.method = S\_c.method;$
$Co.framework = S\_c.framework;$
$Co.scope = S\_c.scope;$
$Co.org\_type = S\_c.org\_type;$
$Co.description = S\_c.description;$
$Co.aspect = S\_c.aspect;$
$Co.level = S\_c.level;$
$Co.asplevel = S\_c.asplevel;$
$Co.ncomp = S\_c.ncomp;$
$S\_c.keywords = Co.keywords;$

$S\_c \rightarrow Co\ S\_c.$
$Co.method = S\_c.method;$
$Co.framework = S\_c.framework;$
$Co.scope = S\_c.scope;$
$Co.org\_type = S\_c.org\_type;$
$Co.description = S\_c.description;$
$Co.aspect = S\_c.aspect;$
$Co.level = S\_c.level;$
$Co.asplevel = S\_c.asplevel;$
$Co.ncomp = S\_c.ncomp;$
$S\_c_0.keywords = concat(S\_c_1.keywords, Co.keywords);$

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 TREEAD 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

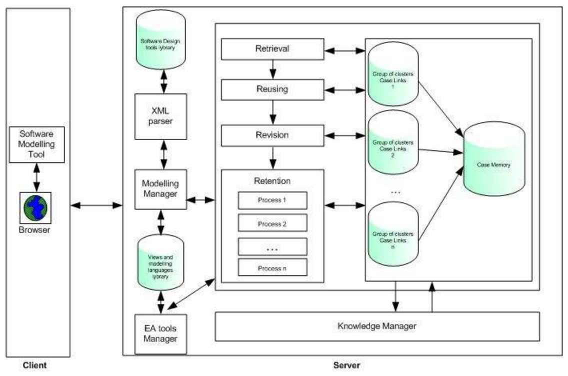

Figure 2: TREEAD's structure.

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).

$Co \rightarrow' component' \; name \; Chs \; Ks \; Sup \; Sub \; Re \; Li \; Ore.$
$Co.type =' component';$
$Co.name = name.value;$
$Co.characts = Chs.characts;$
$Co.keywords = Ks.keywords;$
$Co.supkeywords = Sup.keywords;$
$Co.subkeywords = Sub.keywords;$
$Co.relationship = Re.relationship;$
$Co.likeywords = Li.keywords;$
$Co.noflinks = Li.number;$
$Co.relation = Ore.number;$
$Co.ncomp = Co.ncomp + 1;$

Specification 4: Constructor object description.

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.

$Co \rightarrow' connector' \; name \; Chs \; Ks \; Sup \; Sub \; Re \; Li \; Ore.$
$Co.type =' connector';$
$Co.name = name.value;$
$Co.characteristics = Chs.characteristics;$
$Co.keywords = Ks.characteristics;$
$Co.supkeywords = Sup.keywords;$
$Co.subkeywords = Sub.keywords;$
$Co.relationship = Re.relationship;$
$Co.linkeywords = Li.keywords;$
$Co.relation = Ore.number;$
$Co.likeywords = Lid.keywords;$

$Chs \rightarrow name \; val.$
$Chs.characts = (name.value, val.value);$
$Chs \rightarrow name \; val \; Chs.$
$Chs_0.characts = concat(Chs_1, (name.value, val.value));$

$Ks \rightarrow val.$
$Ks.keywords = val.value;$
$Ks \rightarrow val \; Ks.$
$Ks_0.keywords = concat(Ks_1, val.value);$

$Sup \rightarrow Ks.$
$Sup.keywords = Ks.value;$
$Sub \rightarrow Ks$
$Sub.keywords = Ks.value;$

$Re \rightarrow name \; Ks.$
$Re.relationship = (name.value, Ks.value);$

$Li \rightarrow Ks.$
$Li.keywords = Ks.value;$
$Li \rightarrow Ks \; Li.$
$Li_0.keywords = concat(Li_1.keywords, Ks.value);$

$Ore \rightarrow \varepsilon.$
$Ore \rightarrow name.$
$Ore.relation = name.value;$
$Ore \rightarrow name \; Ore.$
$Ore_0.relation = concat(Ore_1.relation, name.value);$

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.

Model	Example	N. of comp.	N. of connect.
M_1	1	14	7
M_2	1	135	63
M_3	1	176	23
M_4	1	80	19
M_5	1	70	17
M_6	1	81	6
M_7	1	161	23
M_8	1	106	25
M_9	1	41	5
M_{10}	1	103	13
M_{11}	1	36	5
M_{12}	1	31	5
M_{13}	1	210	87
M_{14}	1	20	7
M_{15}	1	88	5
M_{16}	2	281	69
M_{17}	2	406	93
M_{18}	2	367	78
M_{19}	2	391	88
M_{20}	2	230	61
M_{21}	2	89	394
M_{22}	3	73	19
M_{23}	3	112	33
M_{24}	3	112	29
M_{25}	3	58	12
M_{26}	3	40	9
M_{27}	3	78	17
M_{28}	3	55	12
M_{29}	3	51	11
M_{30}	3	61	15

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.

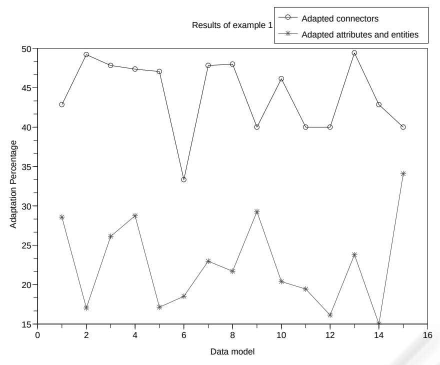

Figure 3: Results of example 1.

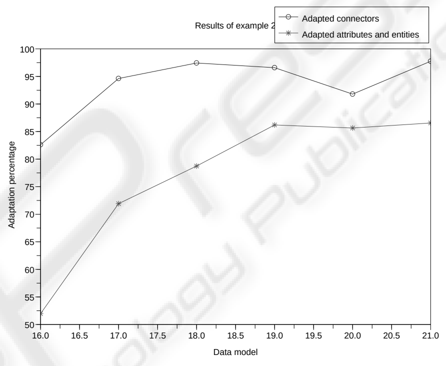

Figure 4: Results of example 2.

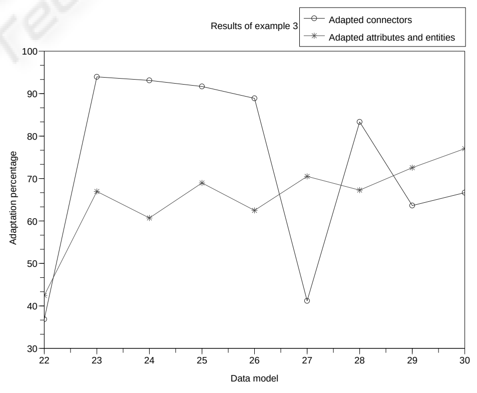

Figure 5: Results of example 3.

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

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations and systems approaches. *AI-Communications*, 7(1):39–52.

Agilense (2007). Agilense enterprise architecture frameworks.

Ahmed, M., Garret, C., FairCloth, J., and Payne, C. (2002). *ASP.NET Web Developers's Guide*. Syngress Publishing.

Alamos, L. (1994). Information architecture: The foundation. Technical report, Los Alamos National Laboratory.

Amdahl, G. M., Blaauw, G. A., and Brooks, F. P. (1964). Architecture of the ibm system/360. *IBM Journal*, 8(2):87–101.

Barker, R. (1995). *Case*Method - Entity Relationship Modelling*. Addison-Wesley.

Barker, R. and Longman, C. (1992). *Case*Method - Function and Process Modelling*. Addison-Wesley.

Bernus, P., Mertins, G., and Schmidt, G. (1998). *Handbook On Architectures of Information Systems*. Springer-Verlag.

Boar, B. H. (1999). *Constructing Blueprints for Enterprise IT Architectures*. John Wiley & Sons, Inc.

Capgemini (2006). Architecture and the integrated architecture framework. Technical report, Capgemini - Consulting, Technology, Outsourcing. www.capgemini.com, Consulted in October 2007.

Chaiyasut, P. and Shanks, G. (1994). Conceptual data modelling process: A study of novice and expert data modellers. In Halpin, T. and Meersman, R., editors, *First International Conference on Object-Role Modelling*, pages 310–323, Brisbane - Queensland.

Chapin, N. (1970). Flowcharting with the ansi standard: A tutorial. *Computing Surveys*, 2(2).

Chen, P. P.-S. (1976). The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

CIO (1999). Enterprise architecture conceptual framework. Technical report, CIO Council. Federal Conceptual Model Subgroup.

Earl, M. J. (1989). *Management Strategies For Information Technology*. Prentice Hall.

EE (2005). Repository-centric enterprise architecture. white paper.

Eertink, H., Janssen, W., and Luttighuis, P. O. (1999). A business process design language. In *World Congress on Formal Methods in Development of Computing Systems*, pages 708–727, Toulose - France. Springer.

FIPS (1993). *Integration Definition for Information Modeling (IDEF1X)*. Federal Information Processing Standards Publications.

Gane, C. and Sarson, T. (1979). *Structured Systems Analysis: Tools and Techniques*. Prentice-Hall.

GAO (2006). Enterprise architecture: Leadership remains key to establishing and levearing architectures for organizational transformation. Technical report, Chairman, Committee on Governmentm Reform, House of Representatives.

Garlan, D., Allen, R., and Ockerbloom, J. (1995). Architectural mismatch: Why resuse is so hard. *IEEE Software*, 12(6):17–26.

Gore, M. (2003). Thoughs on the information system architect role. In *ITC'03*, pages 706–710. IEEE - Computer Society.

Group, O. (2002). The open group architectural framework. Technical report, The Open Group. Verso 8.

Handler, R. (2007). Closing the gap between architecture and model-driven development. In *Gartner Enterprise Architecture Summit*, London, UK. Gartner.

IBM (1984). *Business Systems Planning: Information Systems Planning Guide*. IBM Corperation.

IBM (1995). *Information Framework (IFW) Description*. IBM, 1.02 edition.

IEEE (2000). *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE.

IFEAD (2005). Trends in enterprise architecture 2005: How are organizations progressing.

IFEAD (2007). Enterprise architecture tools review. www.enterprise-architecture.info/EA_tools.htm, Consulted at 27-12-2007.

Isaac, J. C. and Leroy, D. (1994). The amos study: Toward a user validation of the cis architecture design. In *BRG Symposium on C2 Research*, California.

Isaac, J. C. and Leroy, D. (1995). Architecture modeling for better requirements and system specification. In *First International Symposium on Command and Control Research and Technology*, Washington D. C.

ISO/IEC (1998). Information technology - open distributed processing - reference model: Overview.

Kim, Y. G. and Everest, G. C. (1994). Building an is architecture collective wisdom from the field. *Information & Management*, 26:1–11.

Lankhorst, M., Iacob, M., Jonkers, H., Torre, L., Proper, H., Arbab, F., Boer, F., Bonsangue, M., Hoppenbrouwers, S., Zanten, G., Groenewegen, L., Buuren, R., Slagter, R., Campschoer, J., Arbab, F., Steen, M., Stam, A., Wieringa, R., Eck, P., Krukkert, D., Doest, H., Leeuwen, D., Fennema, P., Bosma, H., Cuvelier, M., Penders, P., Bekius, S., and Janssen, W. (2005). *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer.

Lankhorst, M., Jonkers, H., Buuren, R., Leeuwen, D., and Doest, H. (2004). Enterprise architecture modelling - the issue of integration. *Advanced Engineering Informatics*, 18(4):205–216.

Lloyd-Williams, M. (1994). Expert system support for object-oriented database design. *International Journal of Applied Expert Systems*, 1(3).

Maier, M. W. (1996). Systems architecting: An emergent discipline? In *IEEE Aerospace Applications*, volume 3, pages 231–246.

McDavid, D. W. (1999). A standard for business architecture description. *IBM Systems Journal*, 38(1):12–31.

Medvidovic, N. and Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1):70–93.

Microsoft (1999). Microsoft solutions framework. White Paper.

NIH (2007). Nih enterprise architecture framework. Technical report, National Institutes of Health.

NIST (1989). Information management directions: The integration challenge. Technical report, Institute of Standards and Technology.

Nolan, R. L. and Mulryan, D. W. (1987). Undertaking an architecture program. *Stage By Stage*, 7(2).

OMG (2004). Business process modeling notation specification. Technical report, Object Management Group.

OMG (2007). Unified modeling language: Superstructure. Technical report, Object Management Group.

Opdahl, A. L. (1996). A model of the is-architecture alignment problem. In Lind, M., Axelsson, K., Goldkuhl, G., and Hedberg, P., editors, *VITS Autumn*.

Oracle (2007). www.oracle.com.

Panet, G. and Letouche, R. (1994). *Merise / 2: Modles et Techniques Merise Avancs*. Les ditions D' Organisation.

Rechtin, E. (1991). *Systems Architecting: Creating & Building Complex Systems*. Prentice Hall.

Rechtin, E. (1999). *Systems Architecting Of Organizations: Why Eagles Can't Swim*. CRC Press.

Rechtin, E. and Maier, M. W. (1997). *The Art of Systems Architecting*. CRC Press.

Rood, M. A. (1994). Enterprise architecture: Definition, content, and utility. In *Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 106–111, Morgantown, USA. IEEE.

Ryan, H. and Santucci, J. (1993). Building an enterprise information architecture. *Infoworld*, pages 57–58.

Scheer, A.-W. (1998). *ARIS - Business Process Frameworks*. Springer-Verlag, 2ł edition.

Scheer, A.-W. (1999). *ARIS - Business Process Modeling*. Springer-Verlag, 2ł edition.

SEI (2003). How do you define software architecture - www.sei.cmu.edu/architecture/definitions.html. Consulted in 1-07-2003.

Shah, H. and Kourdi, M. E. (2007). Frameworks for enterprise architecture. *IT PrO*, September — October:36–41.

Spewak, S. H. and Hill, S. C. (1995). *Enterprise Architecture Planning, Developing a Blueprint for Data, Applications and Technology*. John Wiley & Sons, Inc.

Stecher, P. (1993). Building business and application systems with retail application architecture. *IBM Systems Journal*, 32(2):278–306.

Sybase (2006). www.sybase.com/products/.

Sykes, J. B. (1991). *The Concise Oxford Dictionary*. Oxford University Press.

Tan, P. N., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining*. Pearson Education.

Tandon, R. (2007). The role of solution architects in systems integrations. *IT PrO*, March, April:26–33.

Tapscott, D. and Caston, A. (1993). *Paradigm Shift: The New Promise of Information Technology*. McGraw-Hill.

Targowski, A. S. (1996). *Global Information Infrastructure: the Birth, Vision and Architecture*. Idea Group Publishing.

Tauzovich, B. (1990). An expert system for conceptual data modelling. In *8th International Conference on the Entity-Relationship Approach*, Toronto, Canada.

TEAF (2000). Treasury enterprise architecture framework. Technical report, Department of Treasury.

Technology, N. I. S. (1993). *Integration Definition for Function Modeling (IDEF0)*. Federal Information Processing Standards Publications.

Wilhelm, R. and Maurer, D. (1996). *Compiler Design*. Addison-Wesley.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3):276–292.

Zachman, J. A. and Sowa, J. F. (1992). Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616.

Zwegers, A. (1998). *On Systems Architecting - a study in shop floor control to determine architecting concepts and principles*. Ph.d thesis, Technische Universiteit Eindhoven. ISBN 90-386-0699-4.