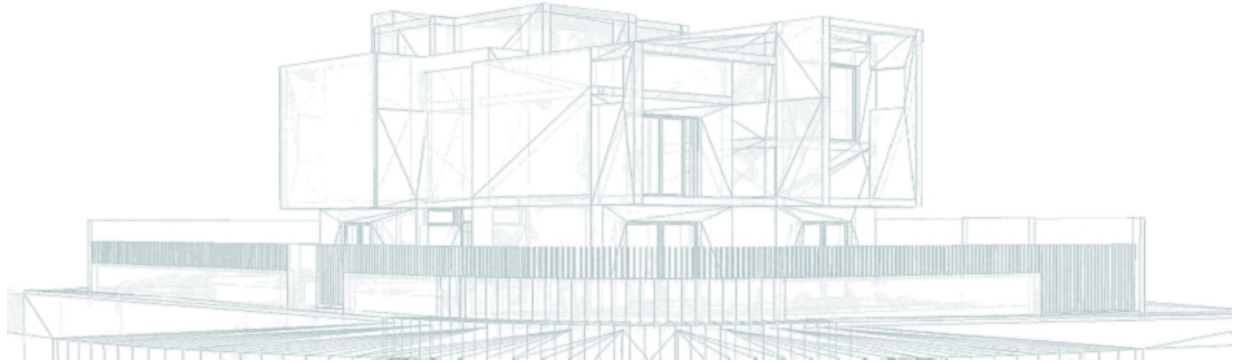




INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



Enterprise Architecture

Information Systems Architecture Visualization

Luís Miguel Jesus Soares

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Júri

Presidente: Prof. Alberto Manuel Rodrigues da Silva

Orientador: Prof. Miguel Leitão Bignolas Mira da Silva

Vogal: Prof. José Carlos Martins Delgado

Setembro de 2007

« In a world full of technicians and politicians all having different levels of understanding, a graphic representation was often the only way to make a point; a single plummeting graphic usually aroused ten times the reaction inspired by volumes of spreadsheets. »

Dan Brown, "Digital Fortress" (1998)

Resumo

O exercício da arquitectura empresarial (AE) comprovou o seu potencial enquanto ferramenta de gestão, na descrição da empresa nos seus vários planos e a vários níveis de abstracção, da mesma forma que a arquitectura de um edifício é descrita num projecto. Assim, com o tempo, as AEs ganharam relevância, tornando-se um objecto de estudo, com normas, *frameworks*, métodos, ferramentas, etc.

Um arquitecto empresarial tem de saber responder a todos os intervenientes da empresa, desde os clientes e utilizadores até aos que implementam e mantêm os sistemas. No entanto, a AE é muitas vezes vista como desnecessária ou desactualizada. Para agravar, os seus modelos são muitas vezes demasiado técnicos, desactualizados e pouco flexíveis. Por estes motivos, a AE dificilmente agrada aos diversos tipos de interveniente, perturbando a tomada de decisões, a consciência empresarial, a comunicação entre o negócio e os gestores das TI, entre outras. Para ultrapassar este problema e para que se façam entender, os arquitectos empresariais têm de envolver os vários *stakeholders* e comunicar de forma clara e na linguagem do negócio.

Uma AE tem pouco valor efectivo se as pessoas não perceberem os seus modelos. Neste contexto, as técnicas de visualização desempenham um papel central na sua partilha e comunicação. É do conhecimento geral que a visualização é um recurso fundamental na comunicação: na resolução de problemas, no suporte à tomada de decisões, justificando argumentos, representando estruturas complexas, entre outros. Por outro lado, a computação gráfica deu um grande salto, o que pode levar a soluções de visualização mais sofisticadas.

A visualização de informação é uma área bem estudada, mas infelizmente não se pode afirmar o mesmo na visualização de AEs. É por isso um desafio propor uma solução nesta área, que atravessa várias outras áreas tão distintas, entre as quais a psicologia, a engenharia organizacional e a computação gráfica. Assim, a nossa proposta é um sistema de visualização de AEs que tira partido de metáforas visuais para representar os modelos da AE. Propomos ainda a separação definitiva entre a modelação e a visualização de AEs.

A metáfora visual, um importante método de visualização, é definida como a representação de um novo sistema com recurso a atributos visuais de um outro sistema, familiar ao utilizador, e que se comporta de modo similar. As metáforas visuais já comprovaram o seu valor no desenho de interfaces (veja-se por exemplo, a metáfora *desktop*) e têm um apoio incontestável em termos teóricos.

Com o uso de metáforas visuais, o sistema proposto beneficia do modo padronizado e simbólico que o cérebro usa para pensar e aprender. Apresentaremos um exemplo dessa vantagem através da metáfora da cidade. A cidade é um ambiente espacial complexo, e mesmo assim as pessoas estão habituadas a orientar-se nesse meio. Sabem como obter informação, como chegar a destinos específicos e como fazer uso das suas infra-estruturas. Demonstraremos essa metáfora através da representação da arquitectura de aplicações.

Palavras-chave: Visualização, arquitectura empresarial, metáfora visual, metáfora da cidade, representação, *stakeholder*, recuperação de informação.

Abstract

Enterprise Architectures have proven their potential as a management tool for describing an enterprise in its various plans and levels of abstraction, the same way blueprints delineate house architecture. Over time, EAs have gained momentum, now being a field of study, with standards, frameworks, methods, tools, etc.

Enterprise architects need to communicate with all the stakeholders of the enterprise, ranging from clients and users to those who build and maintain systems. However, EAs are often seen as overhead or outdated. In addition, its models are usually too technical and inflexible. For those reasons, EAs difficultly satisfy stakeholder diversity and their concerns, thus perturbing decision-making, enterprise consciousness, communication between business and IT, etc. To overcome such problem, architects need to involve stakeholders a lot and communicate in a clear way using terms of the business to make their point.

EAs have little actual value if stakeholders cannot understand its models. In this context, visualization techniques perform a central role in their communicating and sharing. It is nowadays, clear and straightforward to think of visualization as a leading resource for communicating within the organization: solving problems, supporting decision-making, justifying arguments, representing complex structures, among many other endings. In addition, computer graphics has suffered a huge leap forward, which can lead to more sophisticated solutions.

Moreover, information visualization is a well-studied and mature field while EA visualization is not. It is therefore a challenge to propose a solution within this field, crossing so distinct study areas, such as psychology, organizational engineering, and graphics design. Our proposal is an EA visualization system that uses visual metaphors to represent EA models. Furthermore, we propose the definite separation between EA modeling and EA visualization.

The visual metaphor, an important visualization method, is a representation of a new system by means of visual attributes corresponding to a different system, familiar to the user, which behaves in a similar way. Visual metaphors have proven their success in user interface design (for example, with the desktop metaphor), and have an established support from theory.

With the use of metaphors, the system benefits from the brain's highly patterned and symbolic thinking and learning. We put in test that skill through the city metaphor. Cities are very complex spatial environments and yet, people are used to navigating within them. They know how to get information, how to reach particular destinations, and how to make use of the infrastructure. We will demonstrate the city metaphor representing the applications architecture.

Keywords: Visualization, enterprise architectures, visual metaphor, city metaphor, representation, stakeholder, information retrieval.

Acknowledgements

Thanks to my mother Maria de Fátima Soares, for the support and comprehension during the elaboration of this thesis.

I also thank Prof. Miguel Mira da Silva for his advice, guidance, and friendship throughout all period of the thesis, namely in his responses to the thousands of e-mails I sent him.

Thanks to my colleagues and friends, Joana Esteves, Maria do Carmo Cardim, João Stott, Diogo Nesbitt, Cláudia Dias, José Ruivo and Rui Rodrigues for all their advice and support along the stages of my research work, also helping me with our conversations and precious comments and suggestions.

The OutSystems Community also helped me a lot, in what concerns some technical hitches, for what I am thankful.

Enterprise architectures have a lot to learn with traditional architecture, and in this context, I am thankful to the architects Ricardo Meireles and Cristiano Costa.

Finally, a special word to my friends Elisabete Moreira and Paula Antunes for their help, support, and comprehension for my absence and for hearing my reliefs.

Table of Contents

RESUMO	IV
ABSTRACT	VI
ACKNOWLEDGEMENTS	VIII
LIST OF FIGURES	XIV
LIST OF TABLES	XVI
LIST OF DEFINITIONS	XVIII
1.INTRODUCTION.....	1
1.1. Thesis Statement	2
1.2. Thesis Organization.....	2
2.VISUALIZATION	3
2.1. Motivation	3
2.1.1. <i>Communication process</i>	4
2.1.2. <i>Cognitive fit theory</i>	4
2.2. Foundations	6
2.2.1. <i>Classifications</i>	6
2.2.2. <i>Card reference model</i>	7
2.3. Methods.....	8
2.4. Best Practices	8
2.4.1. <i>Human visual system</i>	9
2.4.2. <i>Interface design</i>	10
2.5. Taxonomies	11
2.5.1. <i>Roman taxonomy</i>	11
2.5.2. <i>Price taxonomy</i>	12
2.5.3. <i>Maletic taxonomy</i>	12
2.5.4. <i>Comparison</i>	13
2.6. Summary	13
3.EA VISUALIZATION.....	15
3.1. Foundations	15
3.2. Methods.....	17
3.2.1. <i>Graph layout algorithms</i>	18
3.2.2. <i>Translation rules</i>	18
3.3. Best Practices	20
3.4. Graphical Modeling Languages	21
3.5. Tools.....	23

3.6. Viewpoint Frameworks	24
3.6.1. <i>IEEE Standard 1471-2000</i>	25
3.6.2. <i>Microsoft architectural perspective</i>	26
3.6.3. <i>Enterprise architecture grid</i>	27
3.6.4. <i>TOGAF architecture views</i>	28
3.6.5. <i>Zachman framework</i>	28
3.6.6. <i>Kruchten ‘4+1’ view model</i>	29
3.6.7. <i>RM-ODP</i>	29
3.6.8. <i>Archimate framework</i>	30
3.6.9. <i>Comparison</i>	31
3.7. Summary	31
4.PROBLEM	33
4.1. Definition	33
4.2. Background	33
4.3. Motivation	34
5.PROPOSAL	35
5.1. Definition	35
5.2. Constraints and Requirements.....	35
5.3. Reference Models.....	36
5.3.1. <i>Averbukh framework</i>	37
5.3.2. <i>IEEE 1471-2000</i>	39
5.4. EA Visualization Process	40
5.5. Validation	41
5.6. Summary	43
6.IMPLEMENTATION	45
6.1. “The cITY”.....	45
6.2. “CMDB Web Viewer”	47
6.2.1. <i>Conceptual model</i>	47
6.2.2. <i>Presentation layer</i>	48
6.2.3. <i>Functional analysis</i>	50
6.2.4. <i>Logical architecture</i>	52
6.2.5. <i>Physical architecture</i>	52
6.2.6. <i>API (Application Programming Interface)</i>	53
6.3. Summary	54

7.CASE STUDY	55
7.1. The Project	55
7.2. Chronology.....	56
7.3. Development Process	57
7.4. Results	58
7.5. Summary	59
8.EVALUATION.....	61
8.1. Proposal.....	61
8.2. “The cITy”.....	61
8.3. “CMDB Web Viewer”	62
9.RELATED WORK	63
10. CONCLUSION	67
REFERENCES.....	69
GLOSSARY.....	73
APPENDIXES	77
Appendix A – IEEE 1471-2000	79
Appendix B– Zachman Framework	81
Appendix C – TOGAF Architecture Views	83
Appendix D – Mapping between Zachman and TOGAF	85
Appendix E – EA Tools Listing	87
Appendix F – Periodic Table of Visualization Methods.....	89
Appendix G – Nielsen Usability Heuristics	91
Appendix H – Shneiderman Rules of Interface Design	93
Appendix I – CMDB API.....	95

List of Figures

FIGURE 1 – VISUALIZATION	1
FIGURE 2 – VISUALIZATION STUDIES	3
FIGURE 3 – COMMUNICATION WITHIN THE ENTERPRISE	4
FIGURE 4 – THE COMMUNICATION PROCESS (BAIDA, 2002).....	4
FIGURE 5 – A PICTURE IS WORTH A 1000 WORDS	5
FIGURE 6 – ORIGINAL COGNITIVE FIT MODEL (VESSEY, 1991).....	5
FIGURE 7 – VISUALIZATION REFERENCE MODEL (CARD, ET AL., 1999).....	7
FIGURE 8 – STEPS TO VISUAL EXCELLENCE (OSCAR VISUAL METAPHOR).....	9
FIGURE 9 – GESTALT EFFECTS CAN INFLUENCE HOW WE PERCEIVE IMAGES (TUFTE, 1990).....	10
FIGURE 10 – PRICE TAXONOMY (PART OF THE EXPANDABLE N-ARY TREE) (PRICE, ET AL., 1993)	12
FIGURE 11 – ARCHITECTURE BLUEPRINT VS. ENTERPRISE ARCHITECTURE DIAGRAM	15
FIGURE 12 – EA VISUALIZATION (ARBAB, ET AL., 2003)	16
FIGURE 13 – HOW EA VISUALIZATION FITS IN THE EA PROCESS (LANKHORST, 1998).....	16
FIGURE 14 – THE LIFECYCLE FOR THE ENTERPRISE UNIFIED PROCESS (EUP) v2004 (AMBLER, 2006).....	17
FIGURE 15 – GRAPH EXAMPLE (ARBAB, ET AL., 2003).....	18
FIGURE 16 – APPLICATION OF TRANSLATION RULES (LANKHORST, 1998)	19
FIGURE 17 – PROCESS ILLUSTRATION (LANKHORST, 1998)	19
FIGURE 18 – LANDSCAPE MAP (LANKHORST, 1998).....	20
FIGURE 19 – 3D BUSINESS PROCESS (SCHÖNHAGE, ET AL., 2000)	20
FIGURE 20 – MODELING LANGUAGES BY LEVEL OF ABSTRACTION (STEEN, ET AL., 2004)	22
FIGURE 21 – EXAMPLE OF VIEWS OVER AN EA	25
FIGURE 22 – VIEW AND VIEWPOINT CONCEPTS (CAMERA METAPHOR).....	26
FIGURE 23 – FORMAL ARCHITECTURE VISUALIZATION MODEL (IEEE-SA, 2000)	26
FIGURE 24 – MICROSOFT ARCHITECTURAL PERSPECTIVE (PLATT, 2002)	27
FIGURE 25 – TOGAF ENTERPRISE ARCHITECTURE VIEWS.....	28
FIGURE 26 – SIMPLIFIED ZACHMAN FRAMEWORK (SEE APPENDIX B).....	28
FIGURE 27 – KRUCHTEN ‘4+1’ VIEW MODEL (KRUCHTEN, 1995).....	29
FIGURE 28 – THE FOUR RM-ODP STANDARDS (VALLECILLO, 2001).....	30
FIGURE 29 – CLASSIFICATION OF ENTERPRISE ARCHITECTURE VIEWPOINTS (LANKHORST, 1998)	30
FIGURE 30 – SUMMARY OF EA VISUALIZATION CONCEPTS	32
FIGURE 31 – EA MACRO-PROCESS PART.....	33
FIGURE 32 – METAPHOR DEFINITION	37
FIGURE 33 – CONCEPTUAL MODEL VS. MENTAL MODEL.....	38
FIGURE 34 – VISUAL METAPHOR EFFECT.....	38
FIGURE 35 – CONCEPTUAL PROPOSED SYSTEM	39
FIGURE 36 – “CREATE VIEW” AND “USE VIEW” BUSINESS PROCESSES	40
FIGURE 37 – “CREATE VIEWPOINT” BUSINESS PROCESS.....	41

FIGURE 38 – “CITY” NAVIGATION.....	46
FIGURE 39 – HELICOPTER VIEW OVER “THE CITY”	46
FIGURE 40 – DETAILS ON A <i>SYSTEM</i> OF “THE CITY”.....	46
FIGURE 41 – CMDB ENGINE AND ITS CLIENTS	47
FIGURE 42 – CMDB CONCEPTUAL MODEL.....	48
FIGURE 43 – CMDB WEBSITE CONCEPTUAL MAP.....	48
FIGURE 44 – “WEB VIEWER” SUPPORTING METAPHOR.....	49
FIGURE 45 – CMDB CI DETAILS	49
FIGURE 46 – CMDB “WEB VIEWER” SCREENSHOTS (META-CI LIST, META-CI DETAILS AND CI LIST).....	51
FIGURE 47 – IMPLEMENTATION LOGICAL ARCHITECTURE	52
FIGURE 48 – IMPLEMENTATION PHYSICAL ARCHITECTURE	53
FIGURE 49 – SCOPE OF CASE STUDY: ISA VISUALIZATION	55
FIGURE 50 – PROJECT ORGANIZATION CHART	55
FIGURE 51 – ORGANIZATION OF THE EA.....	56
FIGURE 52 – PROJECT CRHONOLOGY	57
FIGURE 53 – QUANTITY OF USABILITY PROBLEMS VS. NUMBER OF TEST USERS (NIELSEN, 2000)	58
FIGURE 54 – TITANIC DISASTER: DEATHS PER CLASS, SEX, AND AGE (KOSARA, 2007).....	65
FIGURE 55 – CONCEPTUAL MODEL OF ARCHITECTURAL DESCRIPTION OF IEEE 1471-2000 (IEEE-SA, 2000).....	80
FIGURE 56 – PERIODIC TABLE OF VISUALIZATION METHODS (LENGLER, ET AL., 2007)	89

List of Tables

TABLE 1 – TYPES OF VISUALIZATION (TEGARDEN, 1999), (TORY, ET AL., 2004), (STASKO, ET AL., 1998).....	6
TABLE 2 – LENGLER’ CLASSIFICATION SYSTEM (LENGLER, ET AL., 2007).....	6
TABLE 3 – MAPPING BETWEEN NIELSEN AND SHNEIDERMAN INTERFACE GUIDELINES	10
TABLE 4 – ROMAN’S TAXONOMY (ROMAN, ET AL., 1993).....	12
TABLE 5 – MAPPING BETWEEN THREE DIFFERENT VISUALIZATION TAXONOMIES.....	13
TABLE 6 – COMPARISON BETWEEN VISUALIZATION TAXONOMIES	13
TABLE 7 – THE EA GRID (PULKKINEN, 2006)	28
TABLE 8 – COMPARISON BETWEEN ARCHITECTURE VIEWPOINT FRAMEWORKS.....	31
TABLE 9 – REFERENCE MODELS AND BEST PRACTICES	36
TABLE 10 – TYPICAL INTERFACE VISUAL METAPHORS	37
TABLE 11 – INTERFACE EVALUATION’ QUESTIONNAIRES (PERLMAN, 2000)	42
TABLE 12 – CITY METAPHOR TO REPRESENT <i>APPLICATIONS ARCHITECTURE</i>	45
TABLE 13 – “WEB VIEWER” SUPPORTING METAPHOR	48
TABLE 14 – IMPLEMENTATION FORMAL DEFINITION	54
TABLE 15 – COMPARISON BETWEEN INTERFACE EVALUATION TECHNIQUES (JEFFRIES, ET AL., 1991)	58
TABLE 16 – RELATED WORK	63
TABLE 17 – TYPICAL VISUAL METAPHORS	64
TABLE 18 – ZACHMAN FRAMEWORK (ZACHMAN, 1987).....	81
TABLE 19 – TOGAF VIEWS (THE OPEN GROUP, 2006).....	83
TABLE 20 – MAPING BETWEEN ZACHMAN AND TOGAF FRAMEWORKS (THE OPEN GROUP, 2006)	85
TABLE 21 – EA TOOLS LIST (SCHEKKERMAN, 2007).....	87
TABLE 22 – CMDB ENGINE API SPECIFICATION.....	95

List of Definitions

DEFINITION I – VISUAL THINKING AND VISUAL IMAGERY	4
DEFINITION II – COGNITIVE FIT THEORY	5
DEFINITION III – VISUALIZATION.....	6
DEFINITION IV – INFORMATION GRAPHIC	8
DEFINITION V – VISUALIZATION METHOD	8
DEFINITION VI – VISUALIZATION TAXONOMY	11
DEFINITION VII – ENTERPRISE ARCHITECTURE	15
DEFINITION VIII – ENTERPRISE ARCHITECT.....	15
DEFINITION IX – EA VISUALIZATION.....	16
DEFINITION X – VIEWPOINT FRAMEWORK	25
DEFINITION XI – STAKEHOLDER.....	26
DEFINITION XII – (STAKEHOLDER) CONCERN	26
DEFINITION XIII – VIEWPOINT.....	26
DEFINITION XIV – VIEW.....	26
DEFINITION XV – VISUAL METAPHOR	37
DEFINITION XVI – MENTAL MODEL.....	38
DEFINITION XVII – CONCEPTUAL MODEL.....	38
DEFINITION XVIII – BUSINESS METRIC.....	41
DEFINITION XIX – QUESTIONNAIRE	42
DEFINITION XX – CMDB	47
DEFINITION XXI – CI.....	47
DEFINITION XXII – 3-TIER ARCHITECTURE.....	52

1. Introduction

Enterprise architecture is increasingly seen as the best way to maximize existing IT investment while planning for future growth. Only by establishing a blueprint with which to operate can companies comply with corporate and government mandates requiring closer alignment of business processes with IT investment. Further, since change is a constant in IT, the blueprint must be dynamic enough to anticipate and adapt to evolving business models and economic trends (Sherman, 2004). To face this complexity, companies like BP, Intel, and Volkswagen, felt the need to rely on *enterprise architectures* (EAs). Over the time, EAs have become more important, and turned into a field of study, resulting in standards, frameworks, methods, tools, etc.

Enterprise architects need to communicate with all the stakeholders of the system, ranging from clients and users to those who build and maintain the system (Lankhorst, 1998). The final product is a set of artifacts that describe what and how a business operates. The EA process addresses the documentation and understanding of the discrete structural components of the enterprise, typically within the following four categories: business, applications, information, and technology. In conclusion, EAs are *blueprints* for systematically and completely defining an organization's current (*baseline*) or desired (*target*) environment (Schekkerman, 2007).

On the other hand, visual representations are fundamental when talking about enterprise architectures. *Visualization* has accompanied us since we exist as an intelligent life form. Rock paintings are made since the Upper Paleolithic, 40 000 years ago. Nevertheless, the basis objective remains the same: to transform some information or knowledge into some kind of visual representation which is friendly to another human being (Figure 1). In this sense, visualization can serve as a way to inform, share feelings, or persuade someone (Baida, 2002).

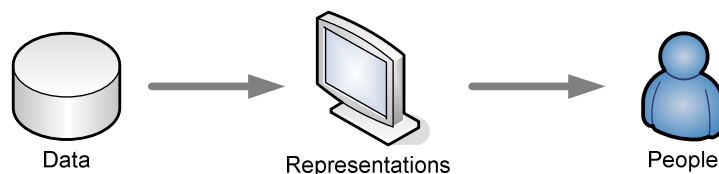


Figure 1 – Visualization

People *need* visualizations in order to solve problems. Numerous *visualization systems* over the years tell us that. Enterprise architectures use visualizations to teach its models to managers, employees and other stakeholders. The need of *communication* and the recent advances in visualization technologies provide the capability to begin to use human visual/spatial abilities to solve the abstract problems found in business (Tegarden, 1999).

1.1. Thesis Statement

Current EA tools are essentially oriented for enterprise modeling, leaving behind visualization. EA models are usually outdated, too technical or inflexible, this way not appropriate for visualization, and interfering with decision-making, enterprise consciousness, solving capabilities, etc.

Our proposal is an EA visualization system that uses visual metaphors to represent EA models. With the use of metaphors, the system benefits from the brain's highly patterned and symbolic thinking and learning. We put in test that skill through the city metaphor. We will demonstrate that metaphor representing the applications architecture.

1.2. Thesis Organization

Current work spins around visualization, and therefore, information representation is always a concern. Definitions are highlighted, tables are presented with a coherent theme, figures are highly expressive, etc. Tables and graphics are used frequently since we believe a picture is really worth a thousand words. Concept definitions are placed contextually. All these resources are used in a way that one could generally understand this work only by analyzing them.

From here on, we will call “chapter”, “section” and “sub-section” to the headings of first, second and third order, respectively. When applied, each chapter has a small introduction that contextualizes it and an end summary that summarizes its main ideas. The list below presents the thesis chapters:

2. *Visualization*: survey of current bibliographic status of general thesis context;
3. *EA Visualization*: survey of current bibliographic status of the specific thesis subject;
4. *Problem*: problem enunciation in detail;
5. *Proposal*: proposed solution for the given problem;
6. *Implementation*: details about the concrete implementation of the proposed solution;
7. *Case Study*: real situation that demonstrates the implementation;
8. *Evaluation*: evaluation of the work as a whole;
9. *Related Work*: academic or business projects related with the proposal;
10. *Conclusion*: work conclusions and future work.

2. Visualization

The discipline of visualization studies is an emergent one and as such represents a so far still highly unstructured domain of research that includes scholars from such distant domains as human-computer interaction, graphic design, management, or architecture (Lengler, et al., 2007). Moreover, visualization it is tightly connected to the people communication needs.

The emerging field of visualization studies examines the use of pictures to improve the access to information, the quality of software, or the communication of knowledge. Prominent research fields in this area are information visualization (a domain of computer science), scientific visualization, or software visualization. Other highly relevant research sectors for the understanding of pictures in communication are design studies and the psychology of perception.

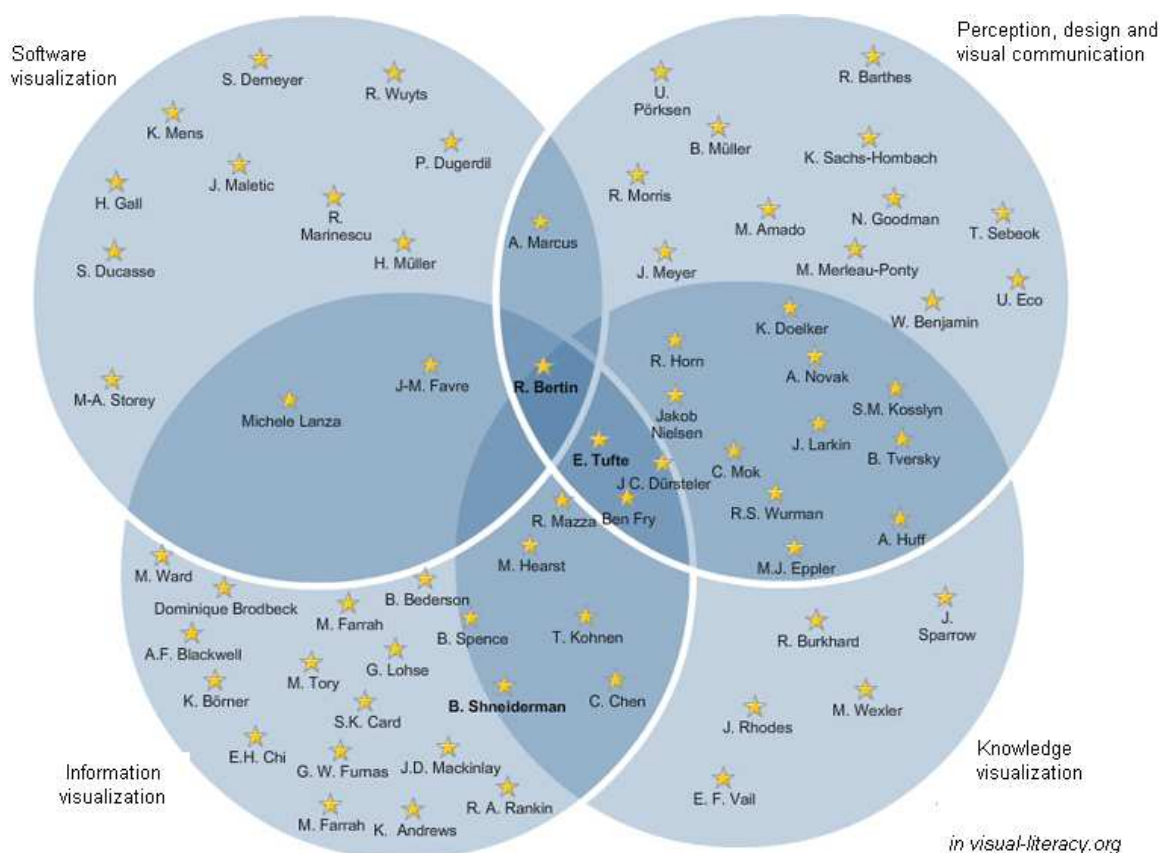


Figure 2 – Visualization studies

2.1. Motivation

Before defining visualization, we need to answer why visualization after all. Much of the research on human cognition shows that visualizations enlarge problem-solving capabilities. Visualization benefits from the facts that humans receive visual information through the *visual system*, which is the human sense with larger bandwidth: Miller stated that a human input channel capacity is greater when visual abilities are used (Miller, 1956). Due to of the benefits of visualization, *visualization systems* can enhance the capability to process information that is either one-dimensional or multidimensional (Tegarden, 1999). On the other hand, humans have a great capacity of symbolic thinking.

2.1.1. Communication process

Baida argues that the need for visualizations stems from the existence of a communication problem between people (e.g. business managers and architects). He adds that visualization (or communication design) serves different types of communication within the enterprise (Figure 3); people communicate in order to reach one or more goals effectively and efficiently. He considers that goal typically *cognitive* or *informative*, in architecture communication (Baida, 2002). In order to that communication to be successful, he defends a communication process for communication (Figure 4).

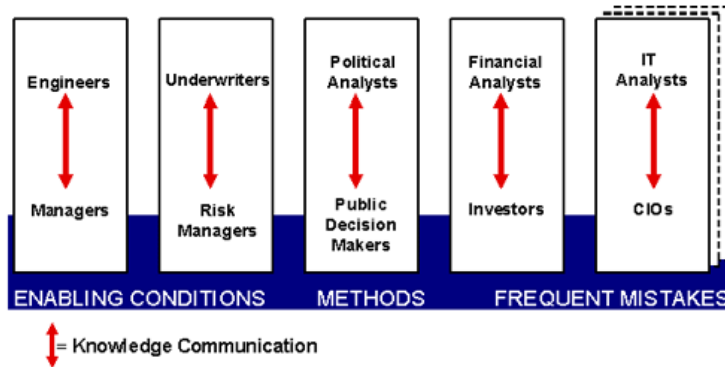


Figure 3 – Communication within the enterprise

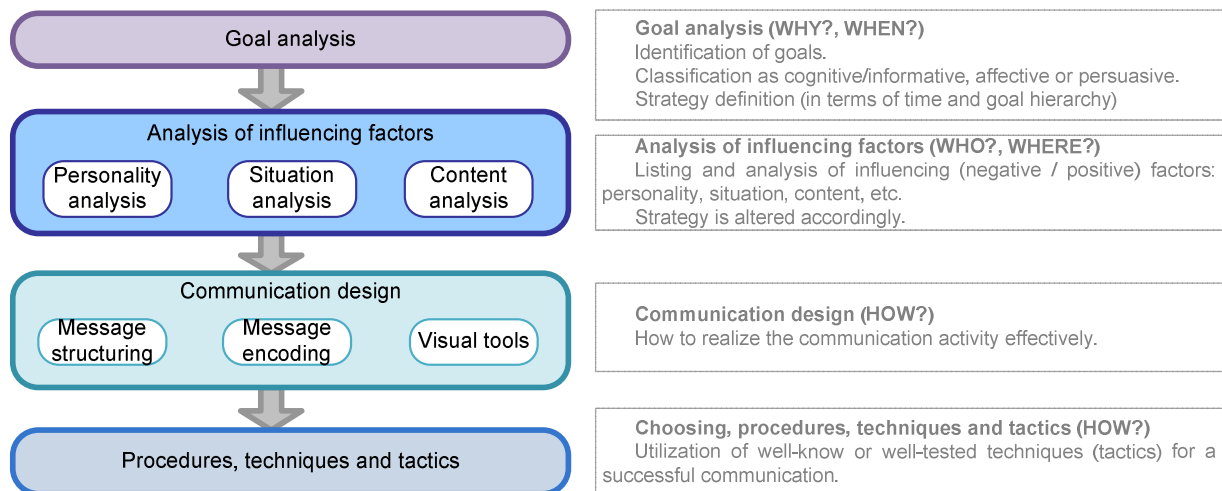


Figure 4 – The communication process (Baida, 2002)

2.1.2. Cognitive fit theory

Visual recall seems to be better than *verbal recall* (Kosslyn, 1986), i.e., a picture really is worth a 1000 words (Figure 5). It is generally accepted that human understanding is improved by *visual representations*. These concepts are studied in detail within *visual imagery* and *visual thinking* fields, which are similar concepts:

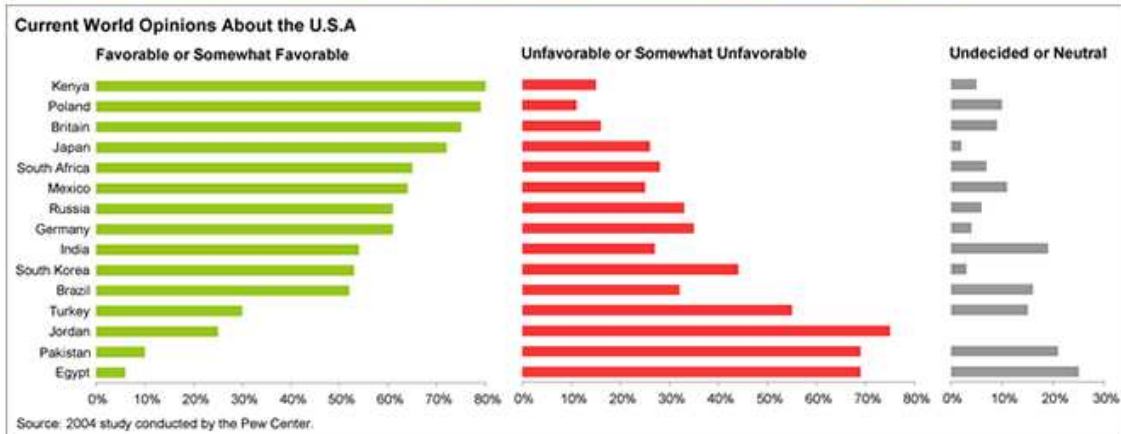
Definition I – Visual thinking and visual imagery

Picture thinking, visual thinking, or visual/spatial learning is the common phenomenon of thinking through visual processing. Visual imagery is the phenomenon of seeing in the absence of a visual stimulus.

Before

Favorable or Unfavorable View of the U.S.	
Brazil: % with somewhat or very favorable opinion of the U.S.:	52%
Brazil: % with somewhat or very unfavorable opinion of the U.S.:	32%
Mexico: % with somewhat or very favorable opinion of the U.S.:	64%
Mexico: % with somewhat or very unfavorable opinion of the U.S.:	25%
Britain: % with somewhat or very favorable opinion of the U.S.:	75%
Britain: % with somewhat or very unfavorable opinion of the U.S.:	16%
Germany: % with somewhat or very favorable opinion of the U.S.:	61%
Germany: % with somewhat or very unfavorable opinion of the U.S.:	35%
Russia: % with somewhat or very favorable opinion of the U.S.:	61%
Russia: % with somewhat or very unfavorable opinion of the U.S.:	33%
Poland: % with somewhat or very favorable opinion of the U.S.:	79%
Poland: % with somewhat or very unfavorable opinion of the U.S.:	11%

After



in perceptualedge.com

Figure 5 – A picture is worth a 1000 words

Cognitive fit theory (Vessey, 1991) is related with the benefits of representation. This theory has been originally used to explain why graphs are sometimes better than tables (Figure 5) for supporting decision-making. In its basic form, it states that:

Definition II – Cognitive fit theory

In problem solving, cognitive fit theory states that a solution to a problem is an outcome of the relationship between the problem representation and problem solving tasks (Vessey, 1991).

According to Vessey, matching representation to tasks leads to the use of similar problem-solving processes, and hence the formulation of a consistent mental representation. In other words, the better the "fit" is between those two constructs, the more effective and efficient the problem solving process (Figure 6). Therefore, problem solving with cognitive fit leads to effective and efficient problem-solving performance. In consequence, we should use the most appropriate visualization mechanism for the given task (Maletic, et al., 2002).

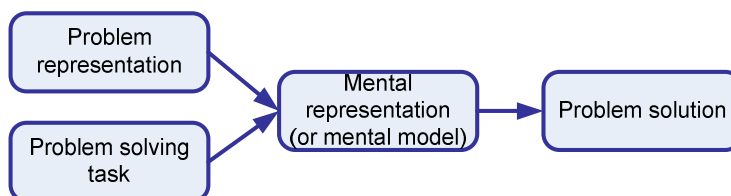


Figure 6 – Original cognitive fit model (Vessey, 1991)

2.2. Foundations

In general, *visualization* (Definition III) is the process of representing data as a visual image; to *visualize* means to put data in a visual form. The underlying domain data can be *concrete* (a car, a house ...) or *abstract* (profit, sales, effort ...) (Tegarden, 1999). According to its nature, visualization can be classified as *specific*, which makes an informed visualization that behaves according to domain semantic, or *generic*, makes a blinded and uninformed representation.









Definition III – Visualization

Visualization is the formation of mental visual images, the act or process of interpreting in visual terms or of putting into visual form (Owen, 1999).

2.2.1. Classifications

Historically, visualization was categorized into two major areas (Tory, et al., 2004): *scientific visualization* and *information visualization*. Tegarden added *virtual reality* to this classification (Tegarden, 1999). *Software visualization* is studied in detail in (Stasko, et al., 1998) (Table 1):

Table 1 – Types of visualization (Tegarden, 1999), (Tory, et al., 2004), (Stasko, et al., 1998)

Type of visualization	Underlying data is...
 Data / Information visualization	Abstract or concrete (e.g. temperature vs. month, web browser or house blueprint)
 Scientific visualization	Usually concrete (e.g. simulation of two mixing fluids)
 Software visualization	Abstract (e.g. algorithm simulation)
 Statistic visualization	Abstract (e.g. analysis of histograms)
 Business visualization	Abstract, discrete, and multi-dimensional, historical or real-time (e.g. dashboard with business performance indicators)
 Geographic visualization	Abstract or concrete (e.g. route viewer like ViaMichelin.com)
 Process visualization	Abstract (e.g. employee workflow)
 Virtual reality	Usually concrete (e.g. a realistic game in a room)

Lengler proposed a more complete approach based on four dimensions (Lengler, et al., 2007):

Table 2 – Lengler’ classification system (Lengler, et al., 2007)

Dimension	Classific.	Description	Examples
Visualization method	<i>Data</i>	Visual representations of quantities data in schematic form (either with or without axes).	Charts, histograms
	<i>Information</i>	The use of interactive visual representations of data to amplify cognition: the data is transformed into an image; it is mapped to screen space. Can have interactivity.	Data map, timeline, radar chart
	<i>Concept</i>	Methods to elaborate (mostly) qualitative concepts, ideas, plans, and analyses.	Mind map, gantt chart
	<i>Strategy</i>	The systematic use of complementary visual representations in the analysis, development, formulation, communication, and implementation of strategies.	Value chain, strategy map
	<i>Metaphor</i>	Position information graphically to organize information and convey an insight about the represented information through the characteristics of the employed metaphor.	Tree, iceberg, bridge, funnel
	<i>Compound</i>	The complementary use of different graphic representation formats in one single schema or frame.	Rich picture, knowledge map
Task and interaction (level of detail)	<i>Overview</i>	Depending on the task, visualization can emphasize certain aspects of the data. These type of visual formats emphasize the global view on information and allow an overall, first glance impression, macro pattern detection and insight.	Timeline
	<i>Detail and overview</i>	Those methods adhere in one way or another to Shneiderman’s visualization mantra Overview first, zoom and filter, then details-on demand.	Table, flow chart
	<i>Detail</i>	These visualization methods highlight individual pieces of information and their characteristics. They make micro patterns apparent.	Nassi-Shneiderman d.
Cognitive process	<i>Convergent thinking</i>	Is a mode of critical thinking in which a person attempts to reduce complexity through analysis and synthesis.	Venn diagram
	<i>Divergent thinking</i>	Is a mode of thinking in which a person generates many unique, creative responses to a question or problem.	Semantic network
Represented information	<i>Structure</i>	Such as hierarchies or networks.	Flow chart
	<i>Process</i>	Either stepwise cyclical in time and/or continuous sequential.	Clustering

Lengler defends that this visualization categorization system (Lengler, et al., 2007):

- Provides a descriptive overview over the visualization domain;
- Functions as an inventory or repository like a structured toolbox;
- Can become a problem solving heuristic that relates possible visualization methods to visualization challenges;
- Helps to recognize the similarities and differences among different types of visualization methods as well as to compare different types of visualization methods along pertinent criteria.

As a result, Lengler classification can assist researchers and practitioners in identifying relevant visualization methods and assess their application parameters. Please refer to Appendix F for the *periodic table of visualization techniques*.

2.2.2. Card reference model

A *reference model for visualization (process)* was introduced by Card (Card, et al., 1999). Card states that visualization is a mapping from data to a visual form that the human perceives (Figure 7). This model has some similarities with MVC (Model-view-controller) and 3-tier model (Definition XXII). At the visual form stage, Card presents the notion of distinct views, which has similarities with IEEE-1471 and most viewpoint frameworks (section 2.6).

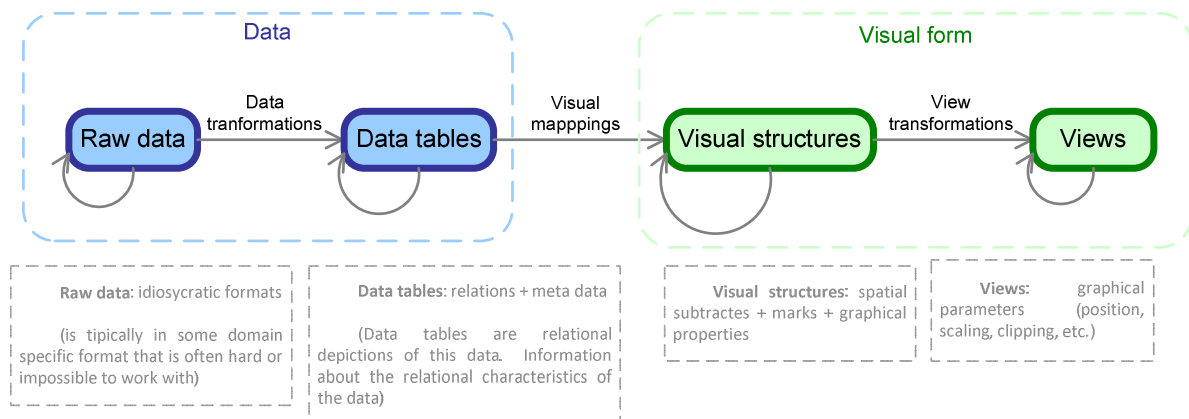


Figure 7 – Visualization reference model (Card, et al., 1999)

Concerning *information retrieval*, Card affirms that each piece of information has a *cost* associated with finding and accessing it. An office, at a particular moment, is characterized by a cost structure over the information in it. What is usually meant by an organized office is one with a cost structure arranged to lower the cost of the information-based work processes performed within it. File cabinets, desks, filing systems, and computer-based information retrieval systems can be thought of abstractly as just means for changing that cost structure of information.

Still with the office metaphor, a small amount of information (either frequently-needed or in immediate use) is kept where the cost of access is low – in an *immediate storage area*, principally the desk. Voluminous, less-used information is kept in a higher-cost, larger-capacity *secondary storage area*. More information is available in the library and other *tertiary storage areas*.

In short, organizing the parts of a system hierarchically often improves the quantity of information processed relative to processing cost. A visualization system must follow that office metaphor in its structure, esthetics, navigation, etc.

2.3. Methods

Visualization is any technique for creating images, diagrams, or animations to communicate a message. Visualization through visual imagery (Definition I) has been an effective way to communicate both abstract and concrete ideas since the dawn of man.

The basic visualization primitives there are *visual abstraction*, 3D, interactivity, animation, color, hue, saturation, relative dimension and position, layout, symbols, style, etc. *Information graphics* (Definition IV) rely on those primitives to serve communication.

Definition IV – Information graphic

An information graphic (or infographic) is a visual representation of information, data, or knowledge. Common information graphics include several types of charts and maps, flowcharts, organization charts, mind maps, etc.

For example, a *chart* or graph is a type of information graphic that represents tabular numeric data and/or functions. Common charts include scatter plot, histogram, bar, pie chart, and line chart.

Visualization methods (Definition V) are a generalization of *information graphics* to any type of visualization. These are high-level methods with an elevated communication potential. Typical methods include concept mapping, evocative knowledge diagrams, argumentation diagrams, or rich *visual metaphors* (see Appendix F). There is not a direct and easy way to know what method one should use. Many times, a combination of methods is the wisest decision. The only key to successful visualizations is common sense.

For example, when domain data is abstract an analogy is desirable (Tegarden, 1999). *Visual metaphors* are the best way to portray that analogy (Definition XV). Note that business information is many times abstract (e.g. business models, profits, processes: intangible things).

Definition V – Visualization method

A visualization method is a systematic, rule-based, external, permanent, and graphic representation that depicts information in a way that is conducive to acquiring insights, developing an elaborate understanding, or communicating experiences (Lengler, et al., 2007).

2.4. Best Practices

Given the verified human visualization characteristics, many visualization systems have been implemented and diverse studies made (for example, see Figure 8). Consequently, some best visualization practices took place. Besides visualization general practices, we need to pay attention to the human visual system since it is the main input of visualizations. On the other hand, interface design has many big practices which visualization design can adopt. These fields will be depicted bellow.

Of course, the designer must also pay attention to non-visualization issues: for example, to design effective visualizations, the designer must first understand the data that will be used as the basis of the visualization. The designer must identify the sources, level of completeness and type of the data (Tegarden, 1999).



Figure 8 – Steps to visual excellence (Oscar visual metaphor)

2.4.1. Human visual system

The visual system allows us to assimilate information from the environment to help guide our actions. A complex set of components give us the capacity to perceive. The visual system has its own specific properties and surely, they influence how we understand things. Psychologists, image researchers, biologists, among others have studied the visual system (Figure 2):

- Humans are only good at working with models that do not include more than 20 elements (Horton, 1991);
- Humans are only good at processing seven (plus or minus two) elements at a time (Miller, 1956);
- It is important to combine some attributes to visualization methods in order to improve humans' ability to process information, including *hue*, *saturation*, *size* and *brightness* (Miller, 1956):
- *Information format* have a significant effect on information acquisition. Component characteristics of displays, such as the form, organization, and sequence of information, influence decision processes (Kleinmuntz, et al., 1993);
- Kosslyn, an imagery researcher, states three maxims to guide graphic display design (Kosslyn, 1994):
 - "The mind is not a camera": our visual system seems to have a number of separate input channels in which to gather visual information. The different channels are sensitive to different types of changes. We should utilize these distinct channels, otherwise, we may "overload" the decision-maker;

- “The mind judges a book by its cover”: design a visual representation that is natural. By natural, we mean that the visualization is capable of being associated with the "real-world" entity at an intuitive level;
- “The spirit is willing, but the mind is weak”: we have a limited amount of information that we can retain in short term memory at one time. The only way to increase this amount is by "chunking" the information being stored.
- The Gestalt Laws of Organization have guided the study of how people perceive visual components as organized patterns or wholes, instead of many different parts. Gestalt is a German word that translates to "configuration or pattern". According to this theory, five (or six) main factors determine how we group things according to visual perception (Figure 9) (Lankhorst, 1998):
 - *Proximity*: people have a tendency to relate objects near to each other (e.g.: visual cluster analysis);
 - *Continuity*: people have a tendency to perceive a line as continuing its establishing direction;
 - *Closure*: people have a tendency to perceive incomplete objects as complete, to close gaps, and to perceive asymmetric objects as symmetric;
 - *Similarity*: people have a tendency to perceive objects that are similar to each other as belonging to a unit (e.g.: same size → equal importance);
 - *Common fate*: people have a tendency to perceive different objects that move or function in a similar manner as a unit (e.g.: different rotation → different domain).



Figure 9 – Gestalt effects can influence how we perceive images (Tufte, 1990)

2.4.2. Interface design

In many ways, the design of visualization is very similar to the design of a user *interface* (Tegarden, 1999). In fact, we can adopt that experience and good practices:

- For example, Nielsen (Nielsen, 1994) proposes 10 heuristics to evaluate an interface (Appendix G). Shneiderman (Shneiderman, 1986) suggests 8 golden rules (Appendix H). These recommendations are similar (Table 3). Note that H6 is explained considering human brain limitations: see “*Human Visual System*” sub-section above.

Table 3 – Mapping between Nielsen and Shneiderman interface guidelines

(Nielsen, 1994)	(Shneiderman, 1986)
H1. Visibility of system status	3°. Offer informative feedback
H3. User control and freedom	7°. Support internal locus of control
H4. Consistency and standards	1°. Strive for consistency
H6. Recognition rather than recall	8°. Reduce short term-memory
H7. Flexibility and efficiency of use	2°. Enable frequent users to use shortcuts 4°. Design dialog to yield closure
H9. Help users recognize, diagnose, and recover from errors	6°. Permit easy reversal of actions 5°. Offer simple error handling

- Additionally, Tufte provides six objectives that any graphic should meet (Tufte, 1983), (Tufte, 1990):

- simply show the data;
- insure that the user is thinking about the substance of the graphic;
- avoid any unnecessary decorations;
- compress as much information into as small a space as possible;
- promote comparisons between different pieces of data, provide views of the data at different levels of detail.
- Finally, Shneiderman adds that an information visualization system should support seven high-level user needs (Shneiderman, 1996). Additionally, from his experience in visualization systems, he recommends that the first four user needs should be present in order to the user (“visualization seeking mantra”):
 1. *Overview*: gain an overview of the entire collection;
 2. *Zoom* : zoom in on items of interest;
 3. *Filter*: filter out uninteresting items;
 4. *Details-on-demand*: select an item or group and get details when needed;
 - *Relate*: view relationships among items;
 - *History*: keep a history of actions to support undo, replay, and progressive refinement;
 - *Extract*: allow extraction of sub-collections and of the query parameters.

2.5. Taxonomies

Software visualizations systems are blossoming nowadays due to the new demanding characteristics of business but also because of more powerful graphical hardware capabilities. In this sense, there is the need to classify and categorize those systems. Therefore, a *visualization taxonomy* is a framework to accomplish that formal task (Definition VI). Most known taxonomies are: Maletic taxonomy (Maletic, et al., 2002), Roman taxonomy (Roman, et al., 1993), Price taxonomy (Price, et al., 1993), Myers taxonomy (Myers, 1990), Shu taxonomy and Stasko taxonomy. Regardless of some of these taxonomies were originally conceived to serve software visualization, they can be straightforwardly used within other visualization types (Table 1).

Definition VI – Visualization taxonomy

A visualization taxonomy is a formal method to categorize and classify a visualization system. It may also introduce a formal language to ease communication, thus helping to solve related problems.

On the other hand, taxonomies can further serious investigation in any field of study. A common language terminology facilitates communication about ideas or discoveries. Taxonomies provide this common language and allow new discoveries to be identified and catalogued. They also show where an apparently new discovery is a refinement or variation of something else (Price, et al., 1993).

2.5.1. Roman taxonomy

The authors of the Roman’ taxonomy analyzed several other taxonomies, concluding that each one has its own rationale and merits, but found them less than satisfactory because they are not based on a well-formulated model or theory of the field (Roman, et al., 1993). The result was a taxonomy, which

they claim to be compatible with previous ones. Roman's taxonomy considers five axes sub-divided in several aspects (Table 4). It takes into account three distinct stakeholders: the programmer, the animator, and the viewer.

Table 4 – Roman's taxonomy (Roman, et al., 1993)

Scope	Abstraction	Specification method	Interface	Presentation
<ul style="list-style-type: none"> ○ Code ○ Data state ○ Control state ○ Behavior 	<ul style="list-style-type: none"> ○ Direct representation ○ Structural representation ○ Synthesized representation 	<ul style="list-style-type: none"> ○ Predefinition ○ Annotation ○ Declaration ○ Annotation 	<ul style="list-style-type: none"> ○ Simple objects ○ Composite objects ○ Visual events ○ World (dimensionality) ○ Multiple worlds ○ Interaction through controls ○ Interaction through image 	<ul style="list-style-type: none"> ○ Analytical presentation ○ Explanatory presentation ○ Orchestration presentation

2.5.2. Price taxonomy

Price taxonomy constitutes an enhancement of several other taxonomies of the moment. One of its main concerns is its expandability: a good taxonomy must be expandable to permit new discoveries to be catalogued and more detailed study in specific areas (Price, et al., 1993). A multi-level tree may describe the entire taxonomy may be described by (Figure 10). First level considers six categories:

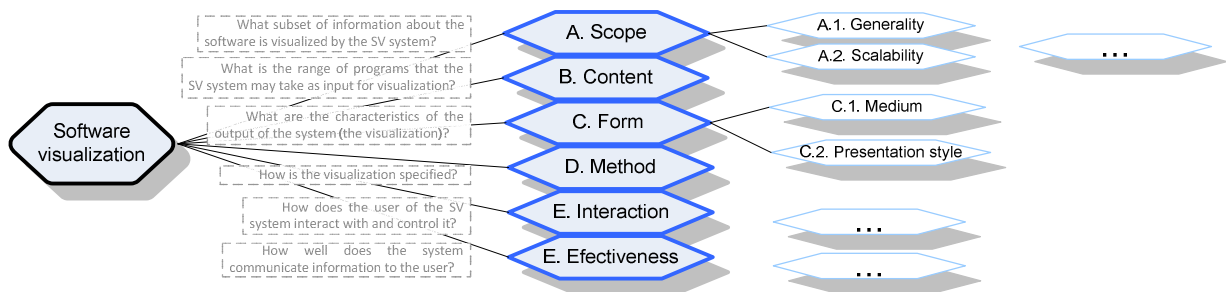


Figure 10 – Price taxonomy (part of the expandable N-ary tree) (Price, et al., 1993)

2.5.3. Maletic taxonomy

No single software visualization tool can address all software engineering tasks simultaneously. While this may be obvious, taxonomies often highlight the lack of functionality in tool rather than focusing on its strength in addressing particular problem. In this context, Maletic taxonomy (Maletic, et al., 2002) tries to grant value to fine tools at a very specific task. The five taxonomy dimensions define a framework capable of accommodating a large spectrum of software visualization systems, including topics outside the taxonomy:

- *Tasks*: why is the visualization needed?
- *Audience*: who will use the visualization?
- *Target*: what is the data source to represent?
- *Representation*: how to represent it?
- *Medium*: where to represent the visualization?

Notice the similarity of this taxonomy and the communication process in Figure 4. An example of a system based on this taxonomy is found at (Marcus, et al., 2003).

2.5.4. Comparison

Visualization taxonomies are different ways to interpret the same thing: a visualization system. Given that, it is usually possible to map one to any other (Table 5). The main conclusion is that each taxonomy focus on certain things than others does not.

Table 5 – Mapping between three different visualization taxonomies

(Roman, et al., 1993) (1 st level)	(Price, et al., 1993) (1 st level)	(Maletic, et al., 2002)
N/A	F.1: Purpose	Task
N/A	F.1: Purpose	Audience
Scope, Abstraction	A: Scope, B: Content	Target
Specification method, Interface, Presentation	C: Form, D: Method, E: Interaction, F: Effectiveness	Representation
N/A	C: Form	Medium

Table 6 gives a clue about some properties of those taxonomies. One should use the most adjusted with the objectives and visualization type (Table 1 and Table 2).

Table 6 – Comparison between visualization taxonomies

Dimension	(Roman, et al., 1993)	(Price, et al., 1993)	(Maletic, et al., 2002)
Expandability	++	+++++	+
Universality	+++	++++	+++++
Simplicity	+++++	++	+++++
Formal definition	+++++	+++	++

2.6. Summary

This chapter gathered the foundations of visualization, visual representation, visualization taxonomies, and their good practices. It is now clear and straightforward to think of *visualization* as a leading method for communicating within the organization: solving problems, supporting decision-making, justifying arguments, assimilating complex structures, among many other endings: human understanding is improved by visual representations. Visualization serves in practice *analysis* and *communication* (Tegarden, 1999) (Baida, 2002) by:

- providing an overview of complex data sets, identifying structure, patterns, trends, anomalies, and relationships in data, supporting decision making;
- exploiting human visual system to extract information from data / solve problems;
- identifying the areas of interest. A major problem for users of modern information systems is the retrieval of new and previously viewed information from the system (Dieberger, et al., 1998);
- reducing the gap between people with different backgrounds.

Although visualization systems differ in purposes and implementation issues, they all have something in common, namely, that they manipulate some visual model of the phenomenon under consideration serving as a basis for translating a computer model into a concrete graphic representation. That model can be of *abstract* or *concrete* data.

Numerous existing general-purpose and special-purpose visualization systems are traditionally divided into three main classes: scientific, information and software visualization systems. Visualization taxonomies help to organize and classify those systems.

More information about visualization and related fields can be found at:

- eagereyes.org: lots of resources on visualization issues and related subjects;
- www.infovis.net: dedicated to information visualization with many resources;
- www.visual-literacy.org: visualization for communication, engineering and business;
- www.visualcomplexity.com: unified resource space on visualization of complex networks;
- services.alphaworks.ibm.com/manyeyes/home: many precious and innovative visualizations.

3. EA Visualization

This chapter represents the connection between *visualization* (studied in earlier sections) and *enterprise architectures*. *Enterprise architecture* study is a young, immature field, with much less experience and best practices than other disciplines. A common definition of the discipline is missing: it seems that any ten architects would have at least eleven different definitions of enterprise architecture (Baida, 2002). If traditional architects have so formal and systematic representations, it would be expectable that enterprise architects follow the same path.

3.1. Foundations

The basic idea of EAs is to compare a diagram of the organization analogous to a house blueprint (Figure 11). Just as an architect produces drawings of different stories, at distinct detail, so an enterprise architect produces several diagrams of different domains, at distinct detail (Definition VIII).

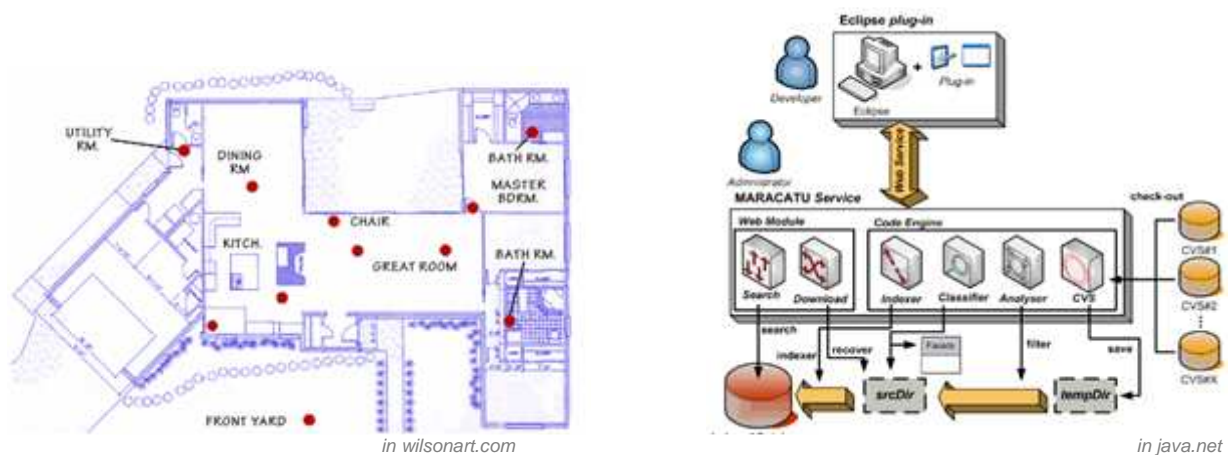


Figure 11 – Architecture blueprint vs. Enterprise Architecture diagram

The general definition states that an *architecture* is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution (IEEE-SA, 2000). In an enterprise context, we obtain the following definitions:

Definition VII – Enterprise Architecture

An enterprise architecture (EA) is a coherent whole of principles, methods and models that are used in the design and realization of an enterprise’s organizational structure, business processes, information systems, and infrastructure (Lankhorst, 1998).

Definition VIII – Enterprise architect

Responsible for EAs, who maps, defines, and standardizes technology, data, and business processes to make IT enables business strategy today and tomorrow (Daniel, 2007).

Having introduced *visualization* and *enterprise architecture*, it is now opportune to cross these two subjects. We therefore, obtain EA visualization, which comprises *visualizing* EAs, applying the best practices and visualization knowledge.

EA visualization (Definition IX) (Figure 13) is part of the *EA process* and serves different stakeholders (e.g., client, developer, vendor, or user) and their concerns. An EA is of no use if people cannot understand it, so everyone should have a good mental model (Definition XVI) about the enterprise, being able to communicate about it, decide through it, and learn from it.

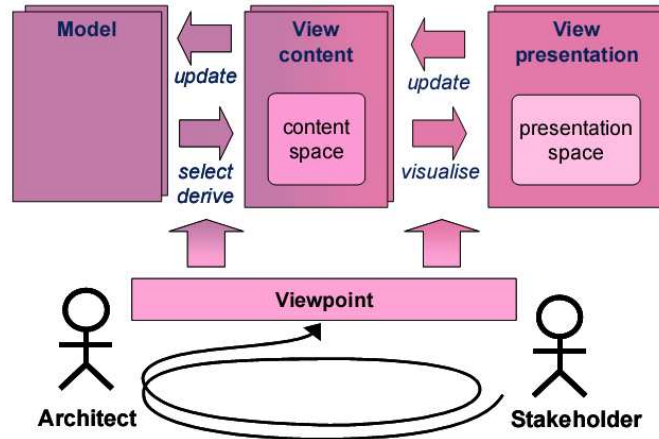


Figure 12 – EA visualization (Arbab, et al., 2003)

Definition IX – EA visualization

EA visualization is the process that delivers understandable and opportune visualizations of EA models to stakeholders.

An EA model is a representation of an EA sub-section, which consists of the various structures and processes of an organization. Therefore, EA modeling means the definition of those models. In other words, EA modeling is the process of defining the enterprise subsets, usually through diagrams. EA modeling precedes EA Visualization and provides its main inputs. EA prepares those models for visualization by different stakeholders with different skills, jobs, concerns (Definition IX). However, usually, EA models are directly presented to people. The following arguments explain why this still happens:

- At first glance, it is cheaper and faster to deliver not altered models to people, instead of investing in a separated process of visualization;
- In most bibliography and business, it is common to mix these two subjects since few people see the importance of EA visualization;
- EA tools are not prepared to deal with EA visualization, and are usually *model* or *repository* oriented, but not *visualization* oriented.

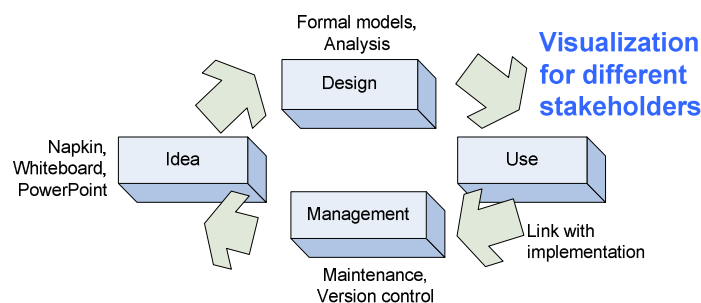


Figure 13 – How EA visualization fits in the EA process (Lankhorst, 1998)

As a result, EA modeling is often seen as EA visualization. Lankhorst and EUP support the separation of these processes (Figure 13 and Figure 14).

The *Enterprise Unified Process* (EUP) is an extension to the RUP (Rational Unified Process is quickly becoming the de facto standard of software development methodology). Figure 14 depicts the EUP lifecycle. Note that “analysis and design” starts after “business modeling”, which also supports the idea of separation between EA modeling and EA visualization.

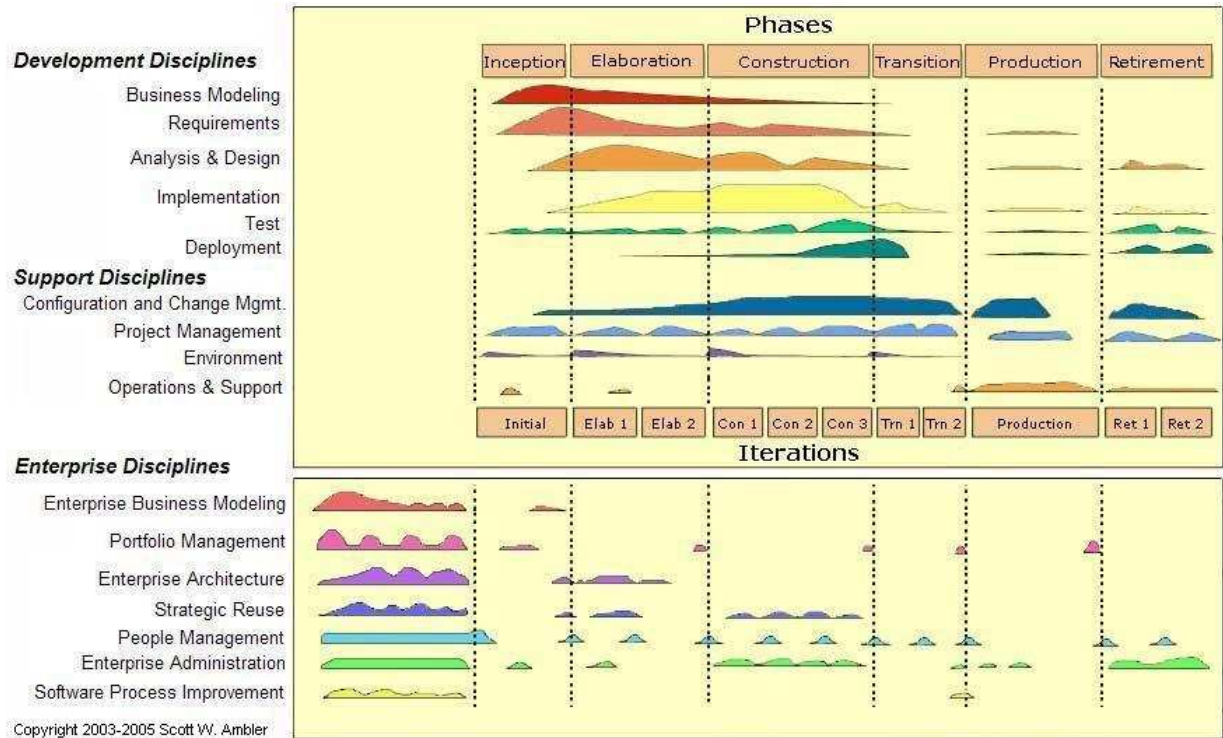


Figure 14 – The lifecycle for the Enterprise Unified Process (EUP) v2004 (Ambler, 2006)

3.2. Methods

Since EAs are a specific domain, some specific methods were developed. However, most methods derive from the classical ones adapted to EAs. Common EA visualization methods include:

- Business process flowcharting;
- Graph layout algorithms (Arbab, et al., 2003);
- Process mapping;
- Critical path method visualization;
- Interaction CRUD matrixes;
- Responsibility matrixes;
- Animation (of business processes, As-is → To-be, etc.);
- Graphical impact analysis;
- *Visual metaphors*;
- Graphical modeling languages (see section 3.4);
- General architecture drawings;
- *Information graphics* (text, tables, charts) (see section 3.3);

Graph layout algorithms and *visual metaphors* and are the most important and common techniques.

3.2.1. Graph layout algorithms

Although very old, the graph drawing (Figure 15) topic has become extremely dynamic in the last twenty years due to the variety of information visualization applications in which they are used. The most common ways of using a graph layout algorithm are the following (Arbab, et al., 2003):

- *Automatic layout*: the layout algorithm works independently of the user intervention. The only exception is the choice of the layout algorithm to be used. Sometimes a set of rules can be coded to choose automatically (and dynamically) the most appropriate layout algorithm for the particular type of graph being laid out;
- *Semi-automatic layout*: the end user can improve the result of the automatic layout procedure by hand. Thus, a certain editing functionality is provided together with the layout algorithm. The user is allowed to move some nodes and/or fix their location and then perform the layout again;
- *Static layout*: the layout algorithm is completely redone ("from scratch") each time the graph is changed;
- *Incremental layout*: when the layout algorithm is performed a second time on a modified graph, the algorithm tries to preserve the stability of the layout as much as possible. The layout is not performed again from scratch. The algorithm uses the previous layout as an initial solution.

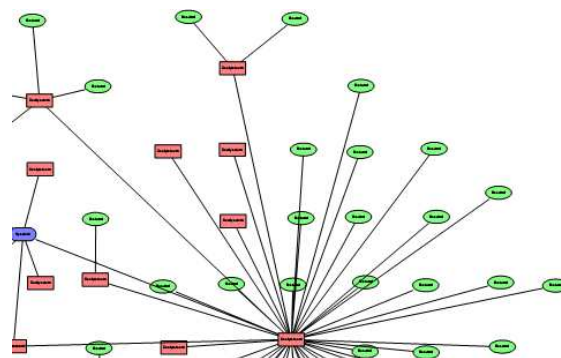


Figure 15 – Graph example (Arbab, et al., 2003)

A state of the art of graphs for EA visualization can be found at (Arbab, et al., 2003).

3.2.2. Translation rules

Visual metaphors (Definition XV) are essential when representing data or building interfaces. In EA visualization, we can call metaphors *translation rules* (Lankhorst, 1998). Translation rules teach how to translate EA models into visual artifacts (Figure 16). For example, consider UML classes, which are usually represented by a rectangle divided in three. It is not expectable that you deliver these “boxes” to stakeholders. However if you have a rule stating that every “box” of type database should be translated to a picture, and if all “boxes” and other artifacts are covered, you will get a representation that speaks the client language. These graphical representations enable architects to use visual properties to enrich representations instead of using “cold” model representations. We will see now three examples how translation rules can be used to create stakeholder’s views: *process illustrations*, *landscape maps* and *3D business processes*.

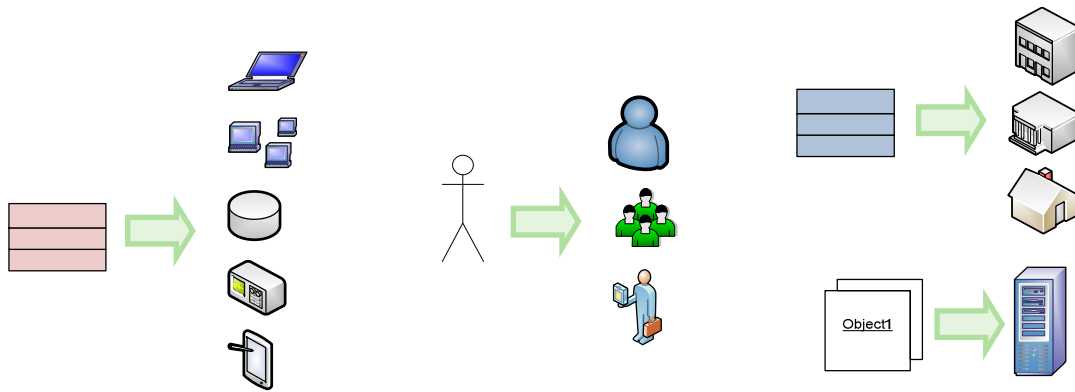


Figure 16 – Application of translation rules (Lankhorst, 1998)

Process illustrations are a perfect example of the utilization of translation rules and are used for presenting business processes to stakeholders like employees and managers. The key idea is to abstract details regarding applications and technology involved and to use recognizable terms and intuitive notation (Figure 17) (Lankhorst, 1998).

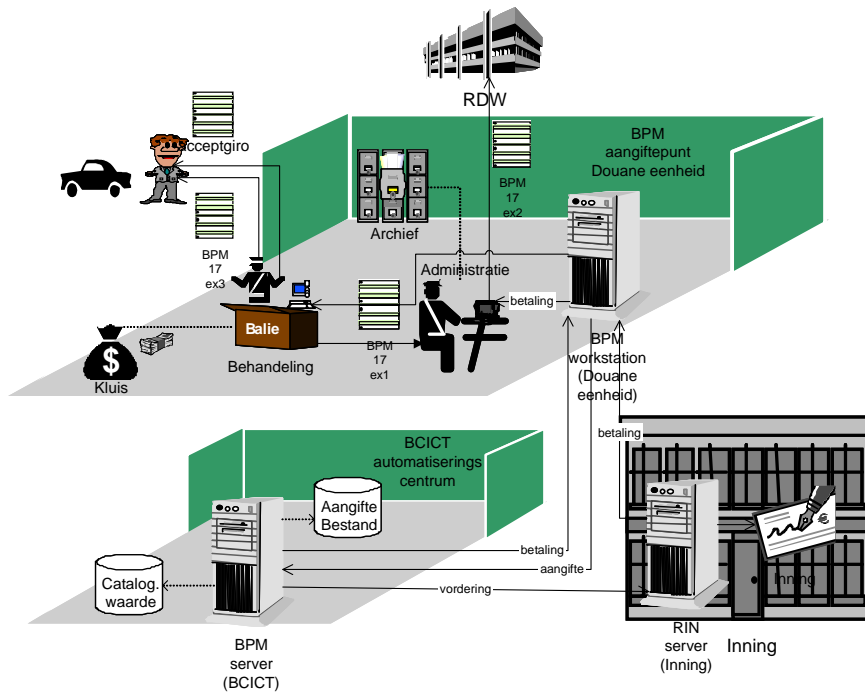


Figure 17 – Process illustration (Lankhorst, 1998)

Another example of visual metaphors is the *landscape maps* (Figure 18). They present architectural elements in the form of an easy to understand 2D map. A landscape map view on architectures provides non-technical stakeholders (such as managers) with a high-level overview, without burdening them with technicalities of architectural drawings (Lankhorst, 1998).

An architect can freely choose the dimensions of a landscape map from the architecture that is being modeled. In most cases, the vertical axis represents behavior such as business processes or functions and the horizontal axis represents “cases” for which those functions or processes must be executed.

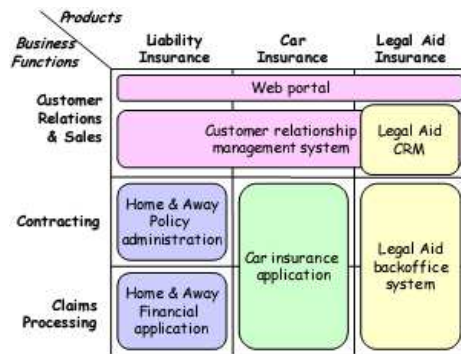


Figure 18 – Landscape map (Lankhorst, 1998)

Finally, the last example we present of the use of translation rules is the 3D business processes. We can see how to use 3D gadgets to represent business processes, and its associated data using 3D histograms (Schönhage, et al., 2000).

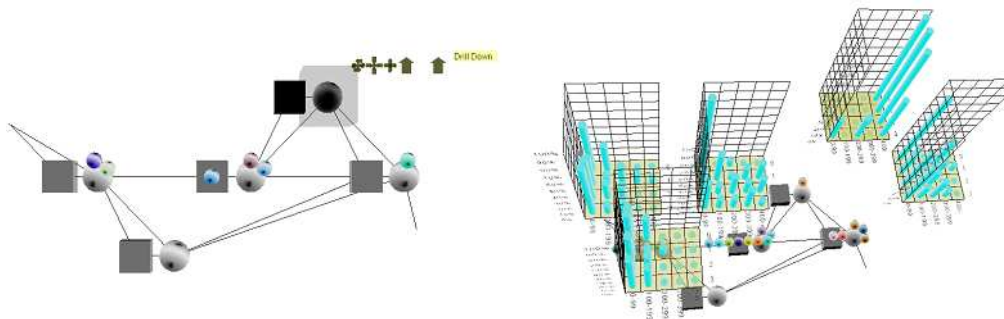


Figure 19 – 3D business process (Schönhage, et al., 2000)

3.3. Best Practices

In EA visualization, it is imperative to follow general visualization and interface design good practices, since they (see Chapter 2, section “Visualization Good Practices”). Apart from that, Baida, at his master thesis, collected a set of guidelines, related do architecture visualization, which were harvested on a practical case study he made within real enterprise situations (Baida, 2002):

- Present hard benefits first, and then soft benefits;
- Refrain from using jargon;
- Visualizations should demonstrate that the architect speaks the business manager’s language;
- Hide complexity, be fast and short;
- The way to relate “As-is” and “To-be” architectures to each other depends on the architect’s goal;
- Architecture visualizations must support switching between the architect’s profession and the business manager’s profession;
- Use verbal communication to present an architecture, and not textual documents;
- A visualization must have one consistent design line through all parts of it;
- By using items that the audience recognizes, the audience identifies itself with what it sees, and is more willing to accept it;
- Use colors to communicate;

- When presenting pictures and text at the same time, present pictures on the *Left Visual Field*, and text on the *right visual field*;
- Managers require powerful mental models of those things they must understand within the organization and its environment, in order to be able to simulate outcomes;
- The visualization of an architecture must be related to the mental model visualization;

Additionally, an important work on EA visualization (Arbab, et al., 2003) presents a list of requisites EA visualization must be compliant to:

- *Representation of concepts*: the visual representation of concepts must be easily adaptable. For instance, the following aspects can be changed: shapes and symbols, background, colors;
- *Printable formats*: visualization must be supported in several two dimensional, printable formats. For instance diagrams, reports, tables;
- *Consistency of presentation*: a *consistent* graphical presentation will be obtained by sharing presentation rules for concepts among all viewpoints that visualize them. *Unambiguous* presentation of concepts is obtained by a careful selection of graphical symbols;
- *Computer supported formats*: visualization must be supported in several computer-supported formats (for instance, 3D presentation, audio, animation).

3.4. Graphical Modeling Languages

Graphical modeling languages play an indirect but important role in EA visualization. It is indirect because EA modeling precedes EA visualization. It is important since, EA representations can use, directly or indirectly, native representation of those languages and are many times the only resource available.

Visual modeling conventions have been subject of lot of research work. For example, Lankhorst gives some tips to apply when visually modeling (Lankhorst, 1998):

- *layout*: use white space; distinguish between normal and exceptional cases; use symmetry to stress similarities; model time dependence from left to right; avoid crossing lines;
- *symbols*: use similar shapes for similar concepts; use line width to stress important relations;
- *color*: use color for emphasis; use color for similarity; use color to convey emotions; limit the number of colors;
- *text*: use domain-specific terminology (stakeholders one); use naming conventions.

Figure 20 shows a classification method of modeling languages according to their abstraction level. At the base of the triangle, we find the architectural concepts used in existing, domain-specific modeling languages and standards; UML is an example of a language in this category. At the top of the triangle, we find the “most general” concepts for system architectures, merely comprising notions such as “object”, “component”, and “relation” (Steen, et al., 2004).

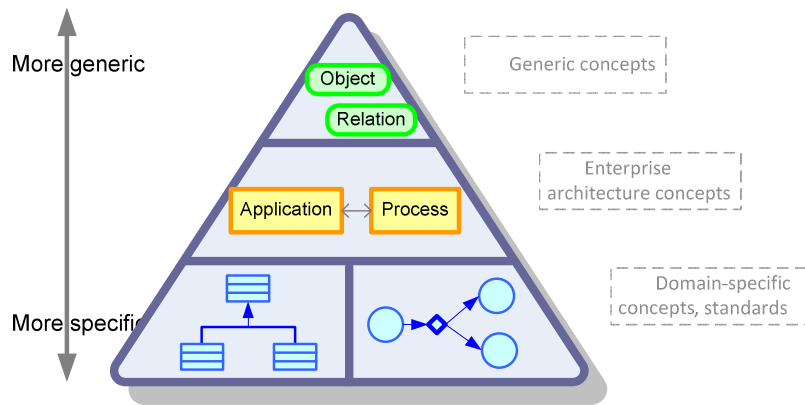


Figure 20 – Modeling languages by level of abstraction (Steen, et al., 2004)

Given that EA modeling is not the central subject of this thesis, we will just make a brief overview over the most known graphical modeling languages (alphabetically sorted):

- *Archimate*: (Lankhorst, 1998) appears to be the most EA oriented language. Its guiding principles are integration between different architectures (business, information, technology, application architectures), to provide a basis for visualization and analysis within EA tools; intelligibility for non-expert stakeholders; use of familiar concepts and notation; high-level and holistic modeling. Several EA tools currently support ArchiMate language and the accompanying framework (see Appendix E);
- *BPMN*: (Business Process Modeling Notation) provides a full set of graphic symbols to represent business processes. BPMN's primary goal is to provide a notation that is understood by all stakeholders, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.
- *EPC*: (Event-driven Process Chain) is the modeling language of ARIS. ARIS (Architecture of Integrated Information Systems) is a method for analyzing processes and for taking a holistic view of process design, management, workflow, and application processes;
- *Express-G*: it is used to identify classes, the data attributes of classes and the relationships that exist between classes. It is a graphical version of its base language Express (ISO 10303-11);
- *FMC*: (Fundamental Modeling Concepts) provide a framework to describe software-intensive systems. It strongly emphasizes the communication about software-intensive systems by using a semi-formal graphical notation. FMC distinguishes three different diagram types: compositional structure, dynamic structure, and value range structure;
- *IDEF*: (Integrated DEFinition Methods) modeling techniques were designed to capture the processes and structure of information in an organization. An enterprise is a complex organism and needs a holistic representation. Therefore, IDEF considers several modeling languages: IDEFØ, IDEF1, IDEF1X, IDEF3, IDEF4;
- *LOVEM*: (Line of Visibility Enterprise Modeling) developed by IBM for process-related projects, from simple process capture to serious Business Process Reengineering (BPR). LOVEM is a graphical, user-friendly approach for business process and workflow design or redesign;

- *Petri nets*: one of several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations;
- *SysML*: (System Modeling Language) domain-specific modeling language for systems engineering applications. It supports the specification, analysis, design, verification, and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities;
- *UML*: (Unified Modeling Language) helps specify, visualize, and document models of software systems, including their structure and design. However, UML can be used for business and other non-software systems. Some of the UML best-known EA usages are *class diagram*, *activity diagram*, *sequence diagram*, and *use case diagram*;

3.5. Tools

According to Schekkerman, EA tools can directly serve not only enterprise architects (Definition VIII), but also strategic planners and enterprise program managers (Schekkerman, 2007). For EAs to be useful and provide business value, their development, maintenance, and implementation should be managed effectively and supported by tools.

According to Giga's Peyret, nowadays modeling tools fall into three categories (Sherman, 2004). We added the following four based on (Lankhorst, 1998):

- *Enterprise architecture repository*: repositories store development and system management models with the goal of providing a global view of the enterprise. Repositories may offer a variety of capabilities for viewing and analyzing model information;
- *IT modeling*: these tools model, store, and extend EA models for sharing among enterprise stakeholders. They go beyond an earlier generation of classical modeling UML tools (for example, Rational), which provided logical and physical views, by providing conceptual views of data or process models. Thus, they are suited for business modeling, not just as development tools;
- *General modeling*: the differentiator here is the ability to store both traditional EA models as well as more generalized or customized models to support domain-specific or discipline-specific styles of modeling;
- *Reporting and publication*: tools that allow the design, either interactively or through confirmation, of reports and viewpoints for specific stakeholders;
- *Storage and retrieval*: metadata repositories that store meta-models, models, and viewpoint specifications;
- *Viewpoint designer*: Using a framework present earlier or a custom one, the enterprise architect can specify the rules that will later lead to the creation and use of views;
- *View presentation tool*: represents the final goal of an EA; with this tool, one can deliver views to different stakeholders. This tool relies on combined representation techniques to deliver

presentations. Nowadays it is essential to use some technologies like the Web to provide an easier presentation.

Automation is key as enterprises look to repositories that can be populated automatically from inventory management, network management, and modeling tools. Beyond saving time, automation also ensures that everyone involved in determining IT priorities is working with the same information – which is especially critical as the information used to set priorities increasingly comes from nontechnical personnel.

EA tools usually support at least one architecture framework and sometimes adopt their own. Framework support is another key aspects when choosing an EA tool (Appendix E). The market has plenty of EA tools and utilities, especially for business process management, and it would be impossible to name all. Some examples are:

- *(Altiris) Service & Asset Management Suite*: combines enterprise asset and service management disciplines into a single web-based architecture, repository, and console which help unite disparate departments and processes;
- *(ASG) Rochade*: provides a streamlined process for centralizing the management of metadata from sources throughout the enterprise;
- *(BiZZdesign) Architect*: it is designed for capturing architecture, and visualizing and analyzing the coherence between different architectural domains. In addition, each domain may be modeled at various levels of abstraction;
- *(Casewise) Corporate Modeler*: enables teams to model, document, analyze and improve the design of organization's EA. This includes business processes, the resources that perform them and the systems and data that support each process;
- *(Mega) Modeling Suite*: has the flexibility to handle a wide variety of modeling-based projects, including business process mapping, EA, enterprise risk management (ERM), corporate compliance, and IS management;
- *(Proforma) ProVision*: enterprise modeling software for understanding and optimizing both business and IT.
- *(Telelogic) System Architect* enables to build a fully integrated collection of models and documents across four key architecture domains: business, information, systems and technology;
- *(Trous) Métis*: offers a range of modeling options, from providing business owners with read-only visual models to support decision-making, through advanced designers and developers who want to create, adapt or extend objects, relationships and search criteria within a meta-model template;

3.6. Viewpoint Frameworks

Since an enterprise has several stakeholders, each one with different concerns and levels of understanding, some natural complexity arises. Shneiderman adds that the designer should be aware of the diversity of the potential users and the tasks that the user interface is to support (Shneiderman, 1986). A *viewpoint framework* is a tool to systematize and deal with this complex issue.

The use of a *viewpoint framework* (Definition X) strongly influences what people “see”, since it is the basis for the creation of viewpoints and views (Figure 22). A viewpoint framework shows how to connect people to views, by describing how to create, select, and use viewpoints. Enterprise architects can choose to adopt one of those, but many times, choose to develop their own.

Definition X – Viewpoint framework

A viewpoint framework is a tool and standard for defining viewpoints (Definition XIII). It comprehends a formal a systematic way to analyze architectures.

A number of (enterprise) architecture frameworks exists: the Zachman Framework (Zachman, 1987), Kruchen’s ‘4+1’ view model (Kruchten, 1995), RM-ODP (Vallecillo, 2001), TOGAF (The Open Group, 2006), DoDAF, MoDAF, CIMOSA, etc. These frameworks are just different ways of looking at the same think; usually it is possible to map one to any other (see Appendix D).

3.6.1. IEEE Standard 1471-2000

Despite being a software architecture standard, IEEE Standard 1471-2000 (IEEE-SA, 2000) (see Appendix A) is generic enough to be considered in any system or enterprise architecture. This will be our launching ramp to the concrete viewpoint frameworks, since it defines foundation concepts like view, viewpoint, stakeholder, concern, and the relations among them that we will use in the rest of the document. In fact, most frameworks adopted this standard given the benefits of standardization.

A complete model is often too complex to be understood and communicated in its most detailed form, showing all the relationships between the various business and technical components (The Open Group, 2006). For this reason, IEEE 1471-2000 recommends customizing views in function of stakeholders (Figure 21). In few words, predefined views reduce visual complexity of the models, since a view only contains aspects relevant to that situation based on a set of concerns per stakeholder.

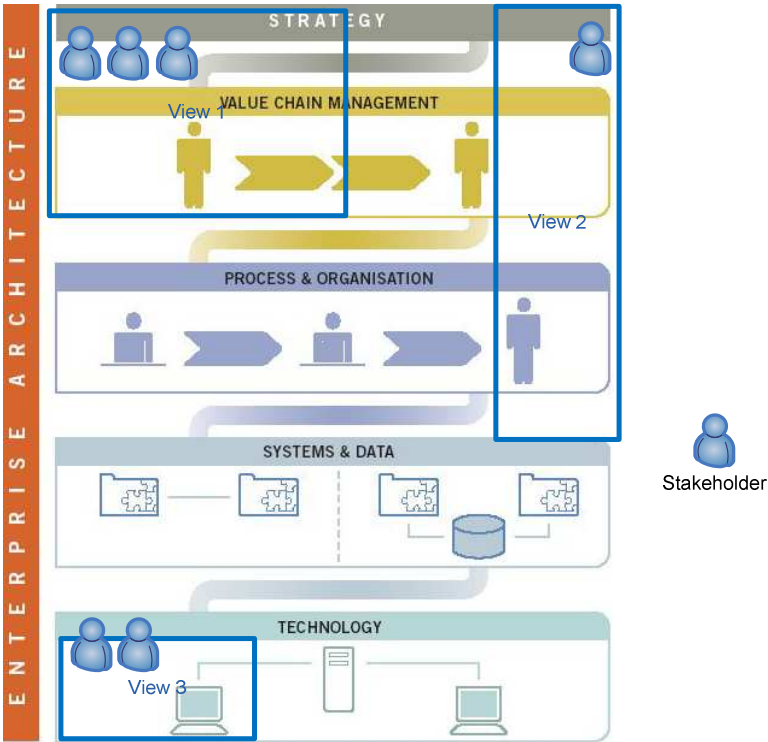


Figure 21 – Example of views over an EA

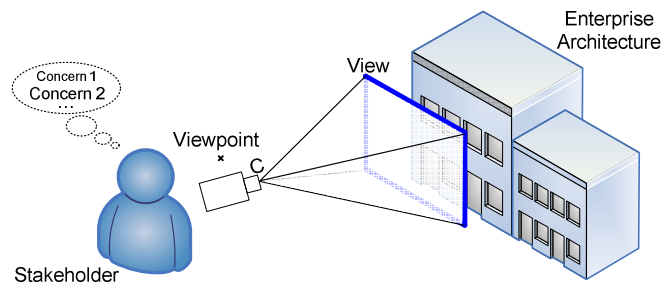


Figure 22 – View and viewpoint concepts (camera metaphor)

Consequently, an enterprise architect must develop multiple views of information system architecture, to enable that architecture to be communicated to, and understood by, the different stakeholders in the system (The Open Group, 2006). IEEE 1471-2000 core concepts are illustrated in Figure 22 and Figure 23 and are defined as (Appendix A):

Definition XI – Stakeholder

A stakeholder is an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.

Definition XII – (Stakeholder) Concern

Concerns are the key interests that are crucially important to the stakeholders in the system, and determine its acceptability.

Definition XIII – Viewpoint

A viewpoint is a specification of the conventions for constructing and using a view, by establishing the purposes and audience for a view and the techniques for its creation and analysis.

Definition XIV – View

A view is a representation of a whole system from the perspective of a set of concerns, in terms meaningful to stakeholders. A view may contain one or more architectural model, allowing it to utilize multiple notations.

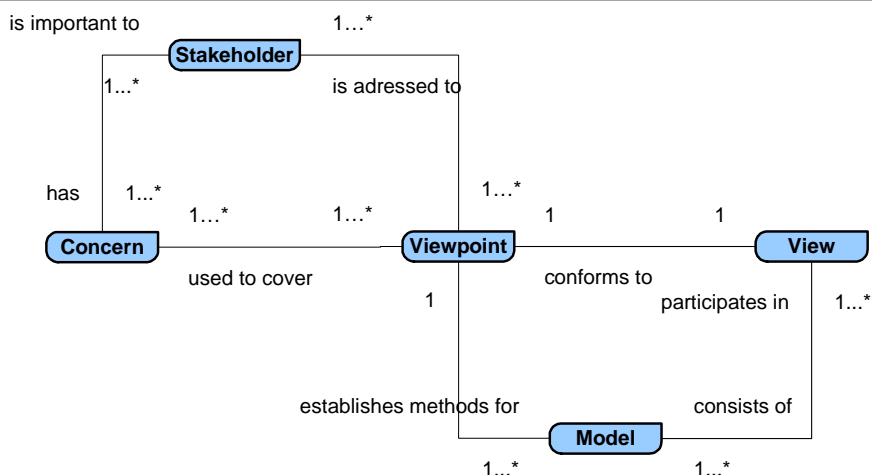


Figure 23 – Formal architecture visualization model (IEEE-SA, 2000)

3.6.2. Microsoft architectural perspective

The Microsoft approach is the most abstract one. It is not actually a real framework, but it is a good way to introduce the genuine ones. The basis for *Microsoft architectural perspective* is the four-domain

architecture (business, applications, information, and technology architectures) (Figure 24). For each kind of architecture, there are four views (Platt, 2002):

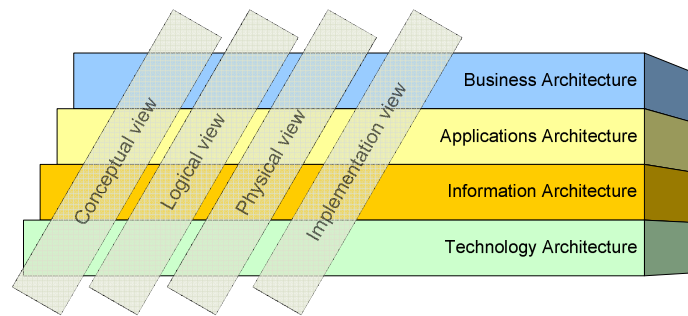


Figure 24 – Microsoft architectural perspective (Platt, 2002)

- *Conceptual*: are the most abstract and tend to be described in terms that are most familiar to the (non-IT professional) users of the system; are used to define functional requirements and the business users' view of the application in order to generate a business model;
- *Logical*: show the main functional components and their relationships within a system, independent of technical details regarding functionality implementation;
- *Physical*: are the least abstract and illustrate specific implementation components and their relationships. Each element in the physical view is implemented, normally by a design and development process, as a software or hardware system;
- *Implementation*: are normally owned by the development or operations organization within the enterprise.

3.6.3. Enterprise architecture grid

The EA grid (Pulkkinen, 2006) is the most common framework in the literature and business world. It is also known as the “four-domain architecture”. Four views (dimensions) to the EA are supported (Table 7). These main dimensions of EA are:

- *Business Architecture (BA)*: depicts the business dimension (Business processes, service structures, organization of activities) ;
- *Information Architecture (IA)*: captures the information dimension of EA; high level structures of business information and, at a more detailed level, the data architecture;
- *Systems Architecture (SA/AA)*: contains the systems dimension, the information systems of the enterprise. Some conventions call it the Applications Architecture or Portfolio, the latter stressing the nature of the information systems as a business asset;
- *Technology Architecture (TA)*: covers the technologies and technological structures used to build the information and communication systems in the enterprise.

Abstraction level differentiation is necessary. For the EA framework, the levels of architectural decision making are adopted with adaptations to planning work:

- *Enterprise level*: the Enterprise Architect’s decision scope is the whole enterprise;
- *Domain level*: the Domain Architect’s decision scope is a domain within the enterprise;
- *System level*: the System Architect’s decision scope is a system he works with.

Table 7 – The EA grid (Pulkkinen, 2006)

View Level	BA	IA	SA / AA	TA
Enterprise	<ul style="list-style-type: none"> Business and management decisions; Portfolio of business; Mission, business strategies and visions. 	<ul style="list-style-type: none"> Strategic information management considerations; Information value chain. 	<ul style="list-style-type: none"> Strategic systems portfolio (application portfolio). 	<ul style="list-style-type: none"> Strategic technology portfolio; Vendor relationships; Enterprise technology guidelines and policies.
Domain	<ul style="list-style-type: none"> Services / products in the domain; Business processes for their production. 	<ul style="list-style-type: none"> Information management of the domain. 	<ul style="list-style-type: none"> Domain systems map; Interoperability. 	<ul style="list-style-type: none"> Technologies infrastructure; Platforms, networks, data communication.
Systems	<ul style="list-style-type: none"> Business requirements for the systems data management 	<ul style="list-style-type: none"> Data architectures; Data harmonization principles; Data storages. 	<ul style="list-style-type: none"> Systems architecture; ISA; Application patterns; Developer guidelines. 	<ul style="list-style-type: none"> System-level technology architecture; Technical implementation guidelines.

3.6.4. TOGAF architecture views

TOGAF (The Open Group Architecture Framework) is an industry standard architecture framework that may be used freely by any organization wishing to develop an IS architecture (The Open Group, 2006). TOGAF is much more than a viewpoint framework, but we are just interested here in the recommended set of views. Like Microsoft and the EA grid, TOGAF also adheres to the four-domain architecture (Figure 25). In addition, it strictly follows IEEE 1471-2000 standard (sub-section 3.6.1).

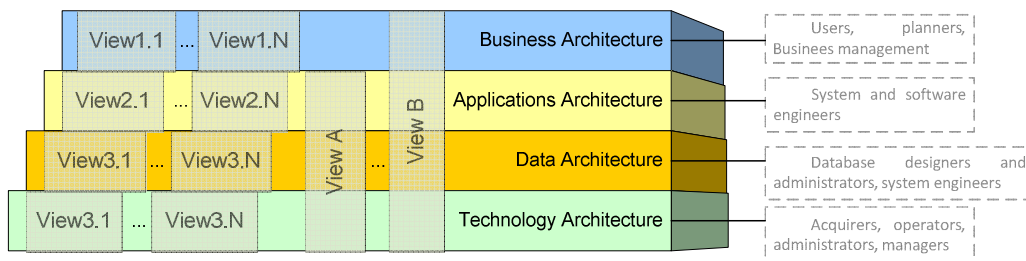


Figure 25 – TOGAF enterprise architecture views

TOGAF recommends the use of well-known patterns to create views. For example, the three-tier model, the client-server model, or the OSI reference models all have some common form of representation that TOGAF recommends to base your views.

3.6.5. Zachman framework

Zachman framework (Zachman, 1987) is designed for the whole enterprise architecture and provides a formal and highly structured way for defining an EA. Zachman Framework uses a grid model based around six questions (What, How, Where, Who, When, and Why) organized by five nominated stakeholder groups (Planner, Owner, Designer, Builder and Subcontractor) (Figure 26). Its main advantage is to give a holistic view of the enterprise being modeled.

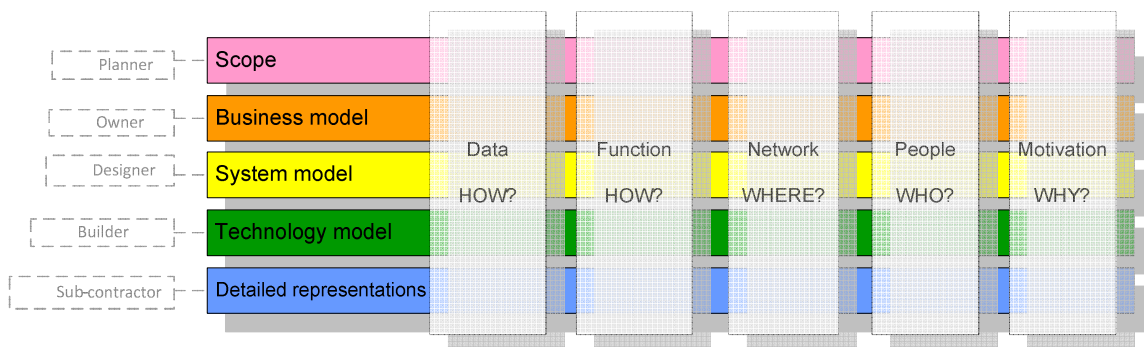


Figure 26 – Simplified Zachman Framework (see Appendix B)

Since the columns map stakeholder's concerns, and the lines have stakeholders associated, this framework can help define a full set of views for the whole enterprise: each cell will result in a viewpoint. Therefore, it is the most complete, but most complex viewpoint framework.

3.6.6. Kruchten '4+1' view model

Kruchten '4+1' View Model (Kruchten, 1995) is the most straightforward viewpoint framework analyzed, since it even recommends the type of diagrams for each viewpoint. The '4+1' view model separates a system into four essential views: *logical*, *process*, *physical*, and *development*. Finally, it considers the *use case* view that describes the functional aspects of the system as a whole (Figure 27), sometimes called the scenarios. Views at Figure 27 can be described as (Kruchten, 1995):

- *Logical view*: describes the (object-oriented system) system in terms of abstractions, such as classes and objects. The logical view typically contains class diagrams, sequence diagrams, and collaboration diagrams. Other types of diagrams can be used where applicable;
- *Development view*: describes the structure of modules, files, and/or packages in the system. The package diagram can be used to describe this view;
- *Process view*: describes the system processes and how they communicate with each other. Activity diagrams are quite often used to describe this view;
- *Physical view*: describes how the system is installed and how it executes in a network of computers. Deployment diagrams are often used to describe this view;
- *Use case view*: describes the functionality of the system. It can be described using case diagrams and use case specifications.

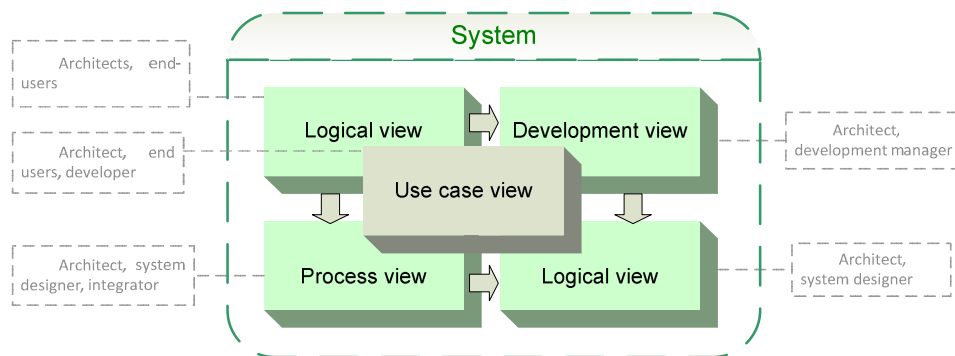


Figure 27 – Kruchten '4+1' view model (Kruchten, 1995)

3.6.7. RM-ODP

RM-ODP (Reference Model for Open Distributed Processing, ISO 10746) (Vallecillo, 2001) defines the semantics of fundamental concepts and constructs of information management used for specification of any system independently of a specific methodology, technology, or tool. All stakeholders could use the same explicitly defined system of concepts, thus providing for traceability and maintainability of business, IT system, and technology specifications. Definitions in RM-ODP are based on the concepts of abstraction (defined as suppression of irrelevant detail) and precision.

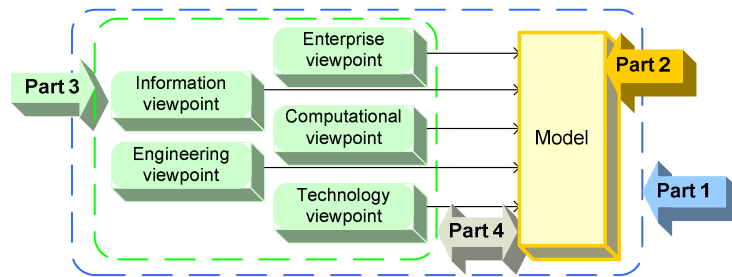


Figure 28 – The four RM-ODP standards (Vallecillo, 2001)

RM-ODP architecture standard (Figure 28) defines *overview*, *foundation*, *architecture*, and *architectural semantics* standards:

- *Enterprise viewpoint*: represents the business model and the business requirements. This view should be understandable by all stakeholders in the business environment. It is the viewpoint used to communicate the business needs to the architecture;
- *Information viewpoint*: is concerned with the semantics of information and information processing;
- *Computational viewpoint*: is concerned with the interactional patterns between the components (services), described through their interfaces;
- *Engineering viewpoint*: is concerned with the design of distributed systems;
- *Technology viewpoint*: is related with the provision of an underlying infrastructure. It focuses on the technologies and the products for implementation.

3.6.8. Archimate framework

To help the architect selecting the right viewpoints, *Lankhorst* introduces a framework as well as a language to express EAs (Lankhorst, 1998), (Steen, et al., 2004). It is based on two dimensions: *purpose* and *content* (both classify viewpoints). Purpose can be of:

- *designing*: supports the design process from initial scratch to detailed design;
- *deciding*: assists managers in the decision making, through cross-domain architecture relations;
- *informing*: inform any stakeholder about the EA, in order to achieve understanding, obtain commitment, and convince adversaries.

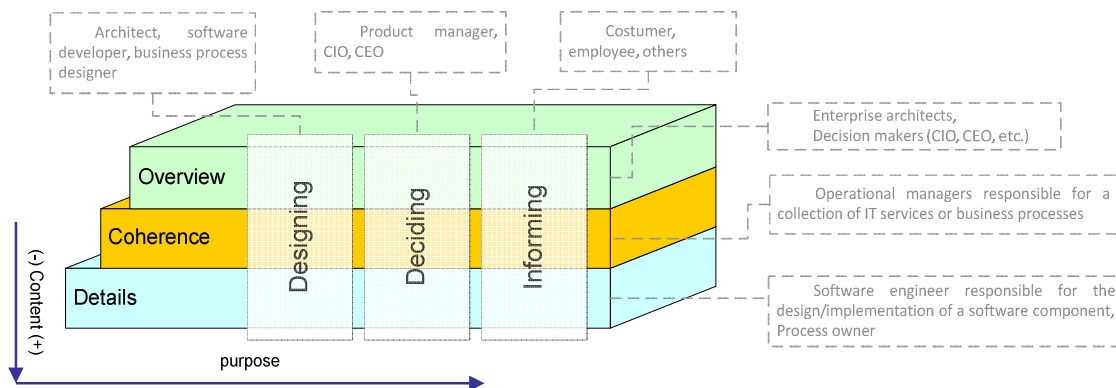


Figure 29 – Classification of enterprise architecture viewpoints (Lankhorst, 1998)

The greatest advantage of using ArchiMate framework is the total support that the accompanying modeling language, ArchiMate language and some tools (see Appendix E) provide. An architect would spend less time integrating different technologies and methods by using the ArchiMate's complete solution.

3.6.9. Comparison

Depending on the enterprise architect objectives and the enterprise needs, one should apply the framework that fits better. Table 8 gives a clue about this choice:

Table 8 – Comparison between architecture viewpoint frameworks

	EA grid	TOGAF	Zachman framework	Kruchten '4+1' view model	RM-ODP	ArchiMate
Practical	+	++	++	+++++	+++	+++
Formality	+	++++	+++++	++	++	+++
Flexibility	++	+++	++	+++	+++	++++
Simplicity	+++++	+	++	+++++	++	+++
Tool support	++++	+++++	+++++	+++	++	+++++

3.7. Summary

In this chapter, we have seen the application of general *visualization* to EAs. The EA modeling process produces a set of models. Through EA visualization, these models will result in final user views. EA visualization (Figure 29) is the process that delivers understandable visualizations of EA models to stakeholders.

When creating views, it is a key aspect to promote user interaction and selection, navigation and control, analysis and reporting. A view must not contain more aspects than the concerns addressed in its respective viewpoint and must use representations familiar to the target stakeholder(s) having its purpose and content well defined. Preferentially, the creation of views should be made systematically, using a viewpoint framework.

A *viewpoint framework* recommends a set of views one can use. A viewpoint framework is a tool and standard for defining viewpoints (Definition XIII). It comprehends a formal a systematic way to analyze architectures. The most known examples of viewpoint frameworks are TOGAF, Zachman, and the EA grid. Most of the frameworks follow directly or indirectly IEEE-1471 standard, which defines a common language for architecture visualization. At least, a viewpoint framework comprises two dimensions:

- *Level of detail*: definition of views with different level of detail (LOD) through visual abstraction;
- *Stakeholder*: definition of views per stakeholder (or groups of);

In what concerns the actual creation of views, *visual metaphors* are a prime technique in achieving representations. In EA visualization, we call them translation rules. Translation rules teach how to translate EA models into visual artifacts, speaking stakeholder language.

Probably, in visualization design, the most important recommendation is to speak the stakeholders' language. The various visualization related fields recommend us many good practices, but that one is the most universal.

The following diagram relates general and EA visualization main concepts. It relates those concepts with the ones on the second chapter. Not all concepts are shown for simplicity reasons.

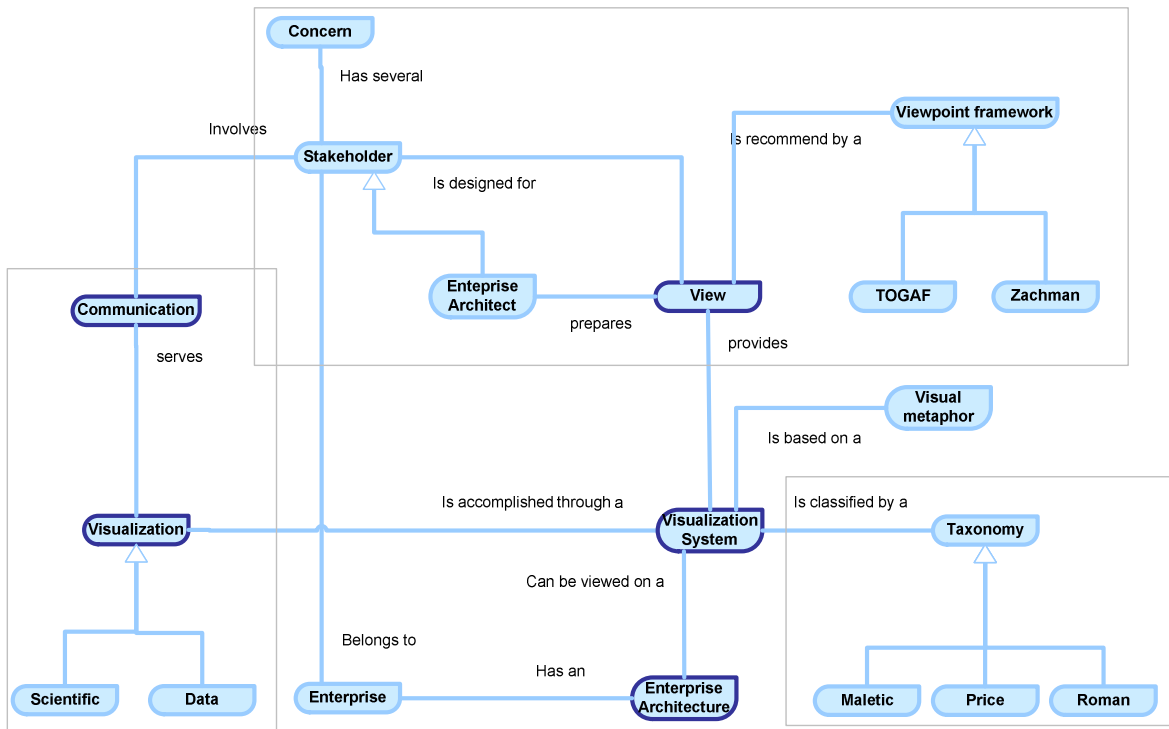


Figure 30 – Summary of EA visualization concepts

The following sites are dedicated to the discussion and resources on EA visualization:

- www.perceptualedge.com: a blog about effective business visualization;
- www.keystonesandrivets.com: another blog dedicated to communication problem in the between business and IT;
- www.theenterpriseearchitect.eu: discussion on several architectures and related fields;
- www.enterprise-architecture.info: the main and most important resource center on EA.

4. Problem

We should formally define the problem before trying to solve it; we should recognize common terminology such as purpose, situation, problem, cause, solvable cause, issue, and solution. *Cognitive fit theory* (Definition III) affirms that a solution depends upon problem representation. For these reasons, we will analyze and define current problem in a formal way.

4.1. Definition

It is opportune to specify in more detail the *object of study*, or in other words, the scope of our problem, among the vast and various EA fields.

The EA visualization process is not concerned only on the prettiest representations, but with the creation, analysis, proper delivering, and usage of views (and viewpoints). One must see only what he is supposed to see and with the exact level of detail and in an opportune moment.

Current EA tools are essentially oriented for enterprise modeling, leaving behind visualization (Figure 31). EA models are usually outdated, too technical or inflexible, this way not appropriate for visualization, and interfering with decision-making, enterprise consciousness, solving capabilities, etc.

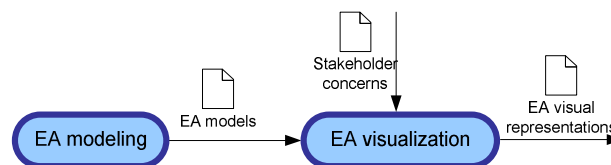


Figure 31 – EA macro-process part

EA visualization assumes the following premises (this is a generic usage of Maletic taxonomy, introduced in “State of the Art”):

- “Who?” *Enterprise architects* provide representations of the enterprise to *stakeholders*;
- “What?” Architecture graphical representations;
- “When?” When a stakeholder asks for or needs an enterprise representation;
- “Where?” In an enterprise environment and/or context;
- “Why?” Essentially, to promote enterprise models to people, ease communication and understating, provide analyses, support change, and support decision making;
- “How?” Using known techniques that support EA visualization (section 2.3), specified by frameworks and implemented by EA tools.

4.2. Background

Visualization can be of data/information, scientific, software, statistic, business, geographic, process, or virtual reality (Table 1). Given that we want to visualize pre-defined EA models, *information visualization* is the most correct classification.

An EA model is a representation of an EA sub-section, which consists of the various structures and processes of an organization. Therefore, EA modeling means the definition of those models. In other

words, EA modeling is the process of defining the enterprise subsets, usually through diagrams. EA modeling is the process that precedes EA Visualization and provides its main inputs.

EA visualization appears subsequently and prepares those models for visualization by different stakeholders with different skills, jobs, concerns (Definition IX).

However, usually EA models are directly presented to people. The following arguments explain why this still happens:

- At first glance, it is cheaper and faster to deliver not altered models to people, instead of investing in a separated process of visualization;
- In most bibliography and business, it is common to mix these two subjects since few people see the importance of EA visualization;
- As a result, usually EA tools are not prepared to deal with EA visualization, and are *model* or *repository* oriented, but not *visualization* oriented.

4.3. Motivation

Architecture is often seen as dusty overhead or outdated modeling. To overcome such feelings, architects need to involve stakeholders a lot. They also have to communicate in a clear way using terms of the business to make their point. Visualization can help a lot in communicating architecture. However, not all ways of modeling are sufficient. Transforming abstract thought into graphic is a complex process (Haan, 2007). In fact, if you search on the web for “enterprise architecture visualization” you will not probably get relevant results. If you ask managers about it, maybe you will hear something about EA modeling. The most important project on the area is the Archimate project. Its goal is provide this integration, by developing an architecture language and visualization techniques that picture several domains and their relations (Arbab, et al., 2003).

If the problem is not well justified, it might not look interesting to solve or may seem already solved. Several reasons justify the importance of the problem in hands:

- The visualization problem it is not new. People always needed visual representations to communicate. In fact, any definition of EA comprises visualization. This way, we need to invest on it;
- We need to solve the communication problems within the enterprise (Figure 3), especially the ones related with business and IT;
- An enterprise has plenty internal and external stakeholders, each one with distinct objectives, concerns and enterprise knowledge. In this context, using only modeling-oriented EA tools, it is obvious that this complexity is not taken into account;
- The original graphical EA models are usually complicated to most people, with many technical details and inopportune information;
- An EA comprises much more than an IT architecture (is should be enterprise holistic), but people often see it that way;
- EA is dynamic and alive, and we tools to deal with that, to assure it do not get outdated.

5. Proposal

Visualization is a well-studied and mature field. However, *EA visualization* is a recent subject. In fact, this concept does not explicitly exist. It is therefore a challenge to propose a solution within this relative new field, which crosses distinct study areas. In addition, computer graphics suffered a huge forward jump, which can lead to a more sophisticated solution. In this context, next sections will fit our proposal within the theory stated in the first chapters as well as a detailed definition of our proposal.

5.1. Definition

Our proposal is an EA visualization system that uses visual metaphors to represent EA models. With the use of metaphors, the system benefits from the highly patterned and symbolic manner that the brain uses to think and learn. The visualization system has a back-office for the enterprise architect prepare viewpoints and views to stakeholders, and a dynamic front office that presents views according to the stakeholder.

Even the concept of EA visualization does not explicitly exist, and therefore it makes part of our proposal. We define EA visualization (Definition IX) not only as set of graphics, but as the business process that deliver understandable, relevant, and opportune enterprise representations to stakeholders, based on their *permanent concerns* but also on *present concerns*. *Permanent concerns* are the ones related with his job and enterprise mission, while *present concerns* are those related with a given task of his daily functions.

With the requisite dynamic, we mean that the system must automatically read the underlying EA models without further modeling or design. The enterprise architect only needs to define viewpoints together with the abstraction level, *visual metaphor*, etc.

The use of metaphors can facilitate the learning of the new system in terms of a known one. In fact, visualization is always a metaphor by its nature, since it associates concepts of the model to visual objects representing the former by means of the latter for proper interpretation by the user (Averbukh, 2001). Additionally, we propose the definition of a conceptual model of that system, with the objective to prepare the system to stakeholders, reducing the opposition to change. Again, the use of *visual metaphors* could foster this exercise.

Finally, we defend the application of a *viewpoint framework* (Definition X) if the enterprise does not yet made it. The resulting views are a vital input of our visualization system.

5.2. Constraints and Requirements

A proposal of this type needs a good understanding of the EA implementation maturity. At an early stage, the proposed system could be used to make people understand the value EAs. In later stages, it could be used in its entire splendor; to systematically deliver views to stakeholders. In any case, it is essential to consider at least one *enterprise architect*, responsible for the EA program. In big and

medium size companies, he belongs to the EA sub-department, but in smaller companies, people from other departments can accomplish the job.

Additionally, the enterprise must have any kind of modeling process that prepares EA models to be used by the EA visualization process. Finally, a good definition of internal and external stakeholders is essential, together with their concerns.

The visualization requirements are thematically categorized as follows (Bosma, et al., 2002)

- General requirements on definition, generation and adaptability of views and visualization;
- Requirements on viewpoints and views ;
- Requirements on presentation.

In the context of our thesis we are interested in the last ones. We will not make a pure requirement analysis, since that encompasses techniques like prototyping, a case study, requirement workshops, etc. We will just overview the indispensable functional requirements of the solution we are proposing:

- This system should offer different levels of detail (abstraction levels);
- Should be compliant to visualization best practices, namely to the visualization seeking mantra: overview first, zoom and filter, then details on demand (Shneiderman, 1996);
- Promote feel to *navigate* with flexibility and swiftness;
- Should be dynamic, in the sense that if the EA changes, the system smoothly adapts to it;
- Must expect to deal with larger data sets (expandability);
- The system interface must respect well-know interface rules (see Appendix G and Appendix H).

5.3. Reference Models

By definition, a *reference model* is an abstract template for the development of more specific models in a given domain. In this sense, we consider fundamental to complement our proposal with some essential models.

Table 9 – Reference models and best practices

Model	Motivation	Main concepts	Resources
Averbukh framework	User interfaces use metaphors, and our system should use them for the same reasons	Visual metaphor, metaphor language	Definition XXII, (FOLDOC, 1998), (Averbukh, 2001), Sub-section 4.3.1
IEEE 1471-2000	It is fundamental to guide projects with standards	Architecture, stakeholder, concern, view, viewpoint, ...	Definition VII, Definition XI, Definition XII, Definition XIV, Definition XIII, Figure 22, (IEEE-SA, 2000), Appendix A
Maletic' taxonomy	A visualization taxonomy defines a common language	Tasks, audience, target, representation, medium	Table 3, (Maletic, et al., 2002)
Communication process	EA process serves the communication process	Communication design	Figure 4, Section 1.1, (Baida, 2002)
Nielsen interface heuristics	The system interface must will be used by many types of user	Usability, consistency, esthetics	Appendix G
Viewpoint framework	Recommends a set of viewpoints, thus assisting the enterprise architect	-	Definition X, Section 2.6

5.3.1. Averbukh framework

We can learn a lot about the utilization of metaphors by analyzing the design of *interfaces*. In fact, a good interface is always supported by a well-known metaphor: indeed, Averbukh says there are no "metaphorless" visualizations of computer models and program entities. For example, a PC movie player conserves the old VCR buttons (Play, Pause, Stop. etc.). Other well-known interface metaphors are the tree (Figure 32) book (Card, et al., 1996), bookshelf, window, theatre, desktop, etc. (Table 10).

Definition XV – Visual metaphor

A visual metaphor is a representation of a new system by means of visual attributes corresponding to a different system, familiar to the user, which behaves in a similar way (Dürsteler, 2002).

Table 10 – Typical interface visual metaphors

Metaphor	Representation
Tree	File browser (Figure 8)
Desktop	Operating system desktop
Book	Electronic document reader
Ticket shop	Online ticket line
Shopping cart	Online shop
Camera	3D Navigator
Office	Operating system
Traditional mail	E-mail

Dieberger reinforces that the use of appropriate navigation metaphors can help to make the structure of modern information systems easier to understand, and therefore, easier to use (Dieberger, et al., 1998). Not only interface design makes use of visual metaphors; an example of a scientific use of a metaphor is Rutherford's model of the atom, which compares the structure of the hydrogen atom to the solar system. From here on, we will use this symbol to represent the action of a metaphor: ➡

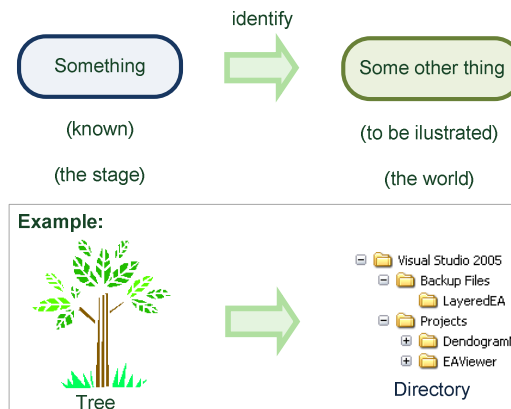


Figure 32 – Metaphor definition

Additionally, a good and well-supported metaphor can help the user create a *mental model* faithful to the original *conceptual model* (Figure 33). *Imagery* based research found that the mental models used in simulating things related to vision or sound tend to draw upon real-world analogies. Furthermore, concrete or "natural" images (those that have a real-world counterpart) are faster to retrieve than abstract images (Tegarden, 1999). User interface *metaphors* mapped from physical concepts and experiences may be considered more "intuitive" (Brockhoff, 2000). This result of course has a bearing on business visualization.

Definition XVI – Mental model

A mental model is a mental representation that people use to understand a system. Its quality depends on the subjacent conceptual model (Figure 33).

Definition XVII – Conceptual model

The conceptual model is a high-level description of the structure and functioning of a system, made by the designers (Figure 33).

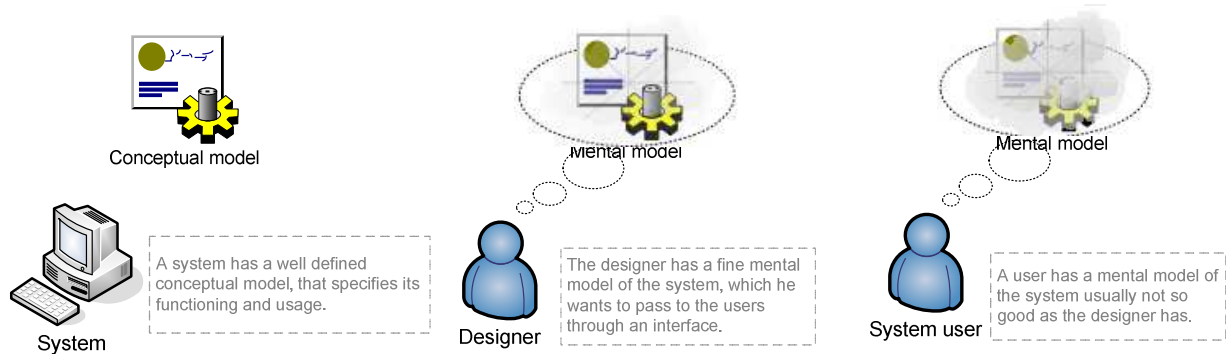


Figure 33 – Conceptual model vs. mental model

The given arguments are enough to justify the use of metaphors in our thesis. However, there is something left to explain. Our proposal considers a system that loads an EA model and dynamically translates it to a visual representation based on a preset metaphor. In this sense, we need to make sure that translation is possible.

We relied on Averbukh framework (Averbukh, 2001) to affirm that a visualization system based on visual metaphors comprises a well-defined language. In this sense, Averbukh affirms that each visualization system contains as its core, the language considering as a unity of the vocabulary, syntax, semantics, and pragmatics. Additionally, visualization languages are built upon some basic idea of similarities between application domain entities with visual objects, i.e., upon a visualization metaphor.

In short, as Figure 34 shows (compare it with Figure 32), the use of a formal visual metaphor makes possible the mapping from domain data values do visual parameters. That visual metaphor defines a formal language that a visualization system can automate. This concept is one of the basis pillars of our proposal.



Figure 34 – Visual metaphor effect

The relationship between data values and visual parameters has to be a univocal relationship; otherwise, if more than one data value is mapped onto the same visual parameter than it will be impossible to distinguish one value's influence from the other. On the other hand, there can always be visual parameters that are not used to map information, as long as there is no need for them to be utilized (Marcus, et al., 2003).

Possible metaphors when representing EAs are the 3D city metaphor, the solar system, process illustrations, the periodic table, a factory, etc. since there are widely known but also due to their specific properties and variable complexity.

5.3.2. IEEE 1471-2000

A *standard* is as a published specification that establishes a common language, and contains a technical specification or other precise criteria and is designed to be used consistently, as a rule, a guideline, or a definition. Bearing this in mind, ISO 1471-2000 (IEEE-SA, 2000) (see Appendix A) works as a guiding principle in our proposal. This standard defines concepts like stakeholder, view, viewpoint, architecture, etc. In order to provide a common language we adopted these concepts (Definition XI, Definition XII, Definition XIII, Definition XIV).

Recall from Definition XIII that a *viewpoint* is a specification of the conventions for constructing and using a view. According to IEEE 1471-2000, each viewpoint shall be specified by:

- 1) A viewpoint name;
- 2) The stakeholders to be addressed by the viewpoint;
- 3) The concerns to be addressed by the viewpoint;
- 4) The language, modeling techniques, or analytical methods to be used in constructing a view based upon the viewpoint.

This definition represents the motivation of our first conceptual module: the viewpoint designer (Figure 35, at top). The enterprise architect will use this module to prepare viewpoints.

In addition, according to that standard, a view is a representation of a whole system from the perspective of a related set of concerns. It defines what appears on the final representation as well as the level of abstraction (*view X viewpoint → graphical representation*).

We propose dynamic views based on the viewpoints/views defined earlier. Each stakeholder would see only his concerns addressed (Figure 35, at bottom left).

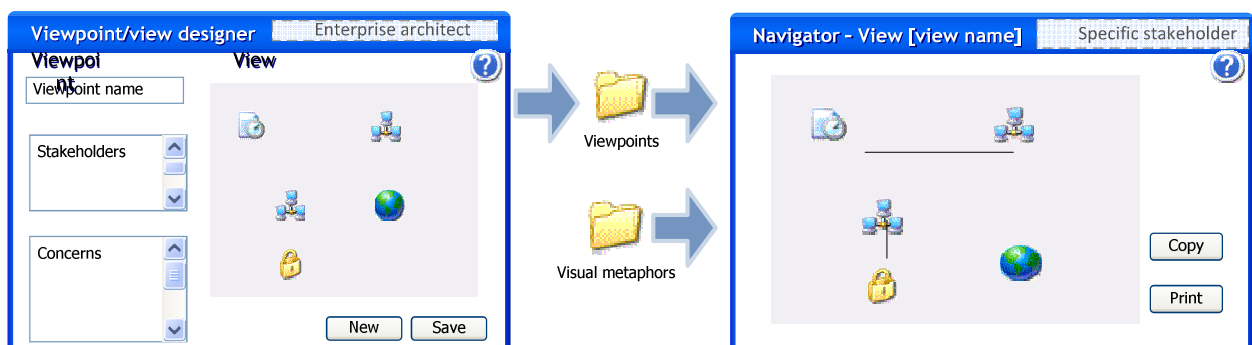


Figure 35 – Conceptual proposed system

Please note that the modules are just conceptual, which means that a real implementation could concretize a different structure of modules. Besides this set of modules, we propose two business processes to handle the creation and usage of views and viewpoints (see section 4.4).

5.4. EA Visualization Process

A pre-requisite of our proposal is the separation between the EA visualization and EA modeling processes (both belong to EA macro-process). The EA visualization process is depicted here, because EA visualization comprises the graphics but also the way to makes them useful. The following BPMN drawings present in detail the creation, analysis, and usage of views and viewpoints. Note that this is just a possible approach, and an enterprise must adapt it to her reality.

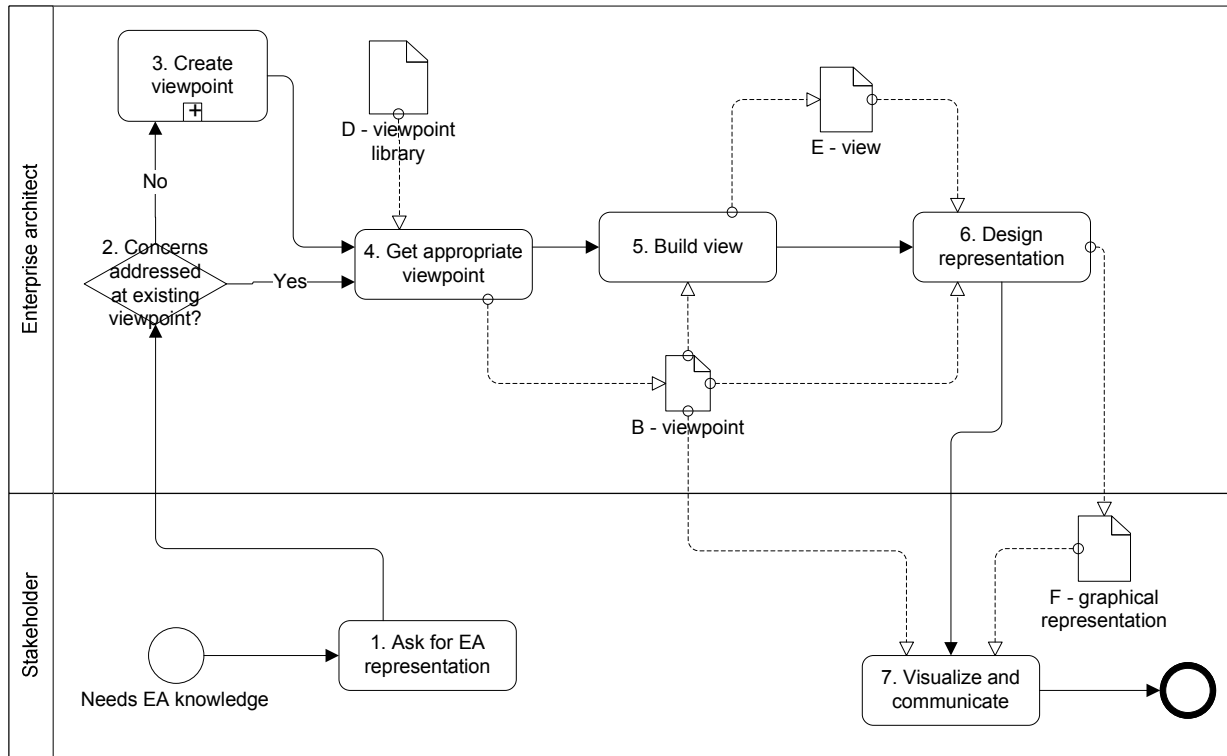


Figure 36 – “Create view” and “Use view” business processes

The process activities are:

1. A stakeholder (e.g. end user, CIO, database engineer, etc.) needs some graphical representation of a EA domain subset, and asks the enterprise architect for it;
2. The enterprise architect checks the viewpoint library, to see if a viewpoint was already created to that goal;
3. If not, it creates a viewpoint (depicted below);
4. The appropriate viewpoint means the one that covers the stakeholder concerns. This one is returned by the viewpoint library;
5. From here on, we have something graphical: according to the viewpoint template the enterprise architect knows what content to put in the view;
6. Having defined the view content, the architect (using the translation rules, defining the layout, colors, etc.);
7. The stakeholder analyses that graphical representation, according to the viewpoint recommendations and mode of use.

The enterprise architect backstage work is to create viewpoints based on the enterprise stakeholders and their concerns. The respective process is the following:

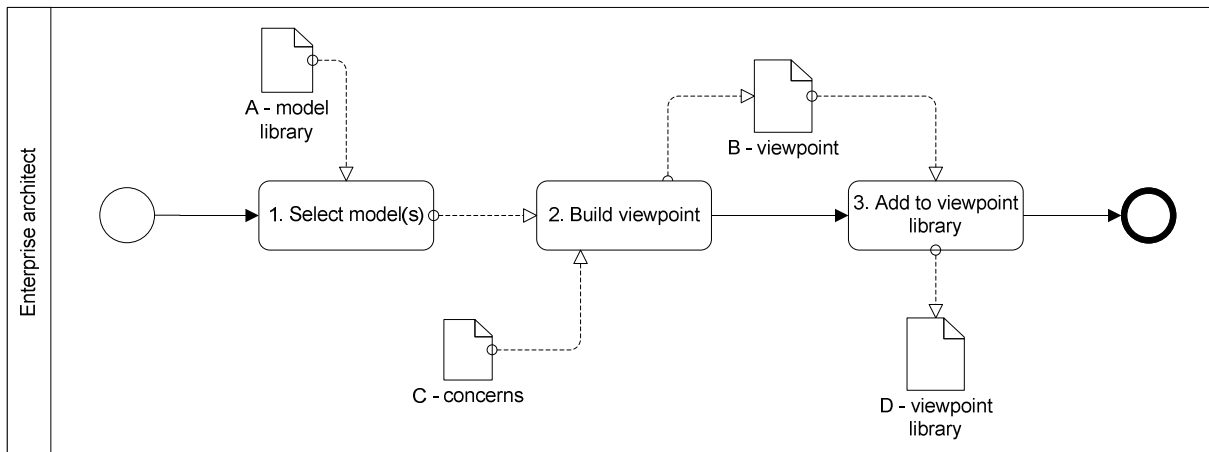


Figure 37 – “Create Viewpoint” business process

The process activities are:

1. The enterprise architect selects among a variety of enterprise models, outputted from the EA modeling process;
2. Having in mind a set of stakeholder concerns, it build a viewpoint, where it specifies: how to build views (e.g. translations rules), how to use them, among other viewpoint related data;
3. The enterprise architect adds this viewpoint to viewpoints library, for later use.

For simplicity reasons, the “obtain stakeholder commitment” decision point was not shown. This activity would be placed between activities 2 and 3. The stakeholder has to agree with the impact of what the viewpoint describes.

5.5. Validation

It is fundamental to define and document ways to validate a possible solution, or else how could we determine its success? In this sense, we use the notion of business metric (Definition XVIII) to introduce our validation method. Informally speaking, what we want to measure is the users’ satisfaction about the system. In this context, that would be their understanding and satisfaction with EA models.

Definition XVIII – Business metric

A business metric is a high-level existing management performance indicator that champions pay special importance (e.g. profitability percentage, customer satisfaction, inventory levels, time to market, yield etc.) (iSixSigma, 2005).

In order to evaluate the mapping of data to a visual metaphor (Definition XV), and thus its success, Mackinlay defined two criteria (Mackinlay, 1986):

- *Expressiveness*: capability of the metaphor to visually represent all the information we desire to visualize. For instance, if the number of visual parameters available in the metaphor for displaying information is fewer than the number of data values we wish to visualize, the metaphor will not be able to meet the expressiveness criterion;

- *Effectiveness*: relationship between data values and visual parameters has to be a univocal relationship; otherwise, if more than one data value is mapped onto the same visual parameter than it will be impossible to distinguish one value's influence from the other. On the other hand, there can always be visual parameters that are not used to map information, as long as there is no need for them to be utilized.

In other words, a visual metaphor should be:

- *consistent*: objects that are similar in the database should also look out similar;
- *easy to understand*: the user should immediately understand what the visual objects represent and how to interact with them;
- *powerful*: they should be able to represent all kind of objects and data contained of the data source.

In conclusion, those measures could be the basis to define some metrics. On another hand, *questionnaires* have long been used to evaluate user interfaces (Perlman, 2000). The biggest single advantage is that a usability questionnaire gives us feedback from the point of view of the user. It also comprises a standardized opinion questionnaire to avoid criticisms of subjectivity.

Definition XIX – Questionnaire

A questionnaire is a method for the elicitation, recording, and collecting of information (Kirakowski, 2000).

A critical question was to find out whether to use a pre-built questionnaire or to build our own. Kirakowski affirms it would take a lot of time, patience, and resources (skills of statistic, psychology, etc.) to build a new questionnaire (Kirakowski, 2000). The sensate option is to use a questionnaire that has already been developed and standardized by someone else (Table 11).

Table 11 – Interface evaluation' questionnaires (Perlman, 2000)

Acr.	Instrument	Reference	Institution	Licence	N.
QUIS	Questionnaire for User Interface Satisfaction	(Chin <i>et al</i> , 1988)	Maryland	Commercial	7
PUEU	Perceived Usefulness and Ease of Use	(Davis, 1989)	IBM	Public	2
NAU	Nielsen's Attributes of Usability	(Nielsen, 1994)	Bellcore	Public	5
NHE	Nielsen's Heuristic Evaluation	(Nielsen, 1994)	Bellcore	Public	10
CSUQ	Computer System Usability Questionnaire	(Lewis, 1995)	IBM	Public	9
ASQ	After Scenario Questionnaire	(Lewis, 1995)	IBM	Public	3
PHUE	Practical Heuristics for Usability Evaluation	(Perlman, 1997)	OSU	Public	3
PUTQ	Purdue Usability Testing Questionnaire	(Lin <i>et al</i> , 1997)	Purdue	Public	100
SUMI	Software Usability Measurement Inventory	N/A	University College Cork	Commercial	50
WAMMI	Website Analysis and MeasureMent Inventory	N/A	WAMMI	Commercial	20
SUS	System Usability Scale	(Brooke, 1996)	John Brooke	Public	10

Other usability related questionnaires include USE, IsoNorm, IsoMetrics, etc.

So, which should we use? We will first eliminate the ones not appropriated for this application.

PUEU measures usefulness and ease of use but it is not what we want to measure (despite being important we had to concentrate on a unique study given the time constraints). NHE is a direct use of Nielsen heuristics and thus not to appropriated for regular users. NAU and ASQ are too general for this study. PUTQ is too long. PHUE and QUIS are useful but maybe in a more advanced stage of system usage.

SUS besides being public it has a measurement scale and broad questions and therefore we will use this. SUS is a reliable, low-cost usability scale that can be used for global assessments of systems usability (Brooke, 1996). SUMI also seems to be adequate but too long for our purposes (we will choose some questions). SUS is composed of the questions (scale 1 to 5):

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn many things before I could get going with this system.

SUMI questionnaire is composed of 50 questions, but we select only eight of those (scale 1 to 4):

1. (2) I would recommend this software to my colleagues.
2. (13) The way that system information is presented is clear and understandable.
3. (19) I feel in command of this software when I am using it.
4. (26) Tasks can be performed in a straightforward manner using this software.
5. (33) The organization of the menus or information lists seems quite logical.
6. (34) The software allows the user to be economic of keystrokes.
7. (41) The software has not always done what I was expecting.
8. (48) It is easy to see at a glance what the options are at each stage.

5.6. Summary

The proposed system would read a set of EA models as input and display it in visual terms, utilizing the best-suited metaphor. In addition, the enterprise architect could prepare viewpoints to utilize later, specifying the target stakeholder(s).

Following the best practices of user interface design and many studies that support metaphors as a fundamental tool to teach users the conceptual model of the system, we also propose that each conceptual module must be sustained by a well-defined metaphor. As a guideline, it is always important to speak the stakeholder language. Quoting William Butler Yeats, «*Think like a wise man, but communicate in the language of the people*». This objective is achieved also with the help of metaphors.

The following facts support the use of visual metaphors in this proposal:

- Users must construct a good system mental model in order to rapidly interact correctly with it;
- The way things are represented has a strong impact on the success of problem solving and communication;

- Visual metaphors are critical when representing complex subjects; software interfaces use them long ago with success;
- Much of the research on human cognition shows that visualizations enlarge problem-solving capabilities (Tegarden, 1999);
- Different stakeholders have different needs, concerns, and levels of understanding.

The most important is to choose the best metaphor to represent the enterprise sub-architecture in hands: the one familiar to the users, representing the maximum variants of the problem with the minimum modifications.

The system should be divided in two parts: preparation of viewpoints and later usage of those viewpoints to construct actual views (in practice, the system should comply IEEE 1471-2000 architecture standard (IEEE-SA, 2000); i.e. to be oriented to the creation/use of views and viewpoints, define clearly its stakeholders and map their concerns, etc) (see section 4.3.2).

6. Implementation

This chapter presents the practical details about the concrete implementation efforts of our proposal. It consists of two proofs-of-concept prototypes: a city metaphor prototype (“The ciTy”) and a CMDB system, which uses “a person and its friends” as its visual metaphor.

6.1. “The ciTy”

An example of a visual metaphor that could be utilized on the proposed system is the *city metaphor*. Cities are very complex spatial environments and yet, people are used to navigating within cities. They know how to get information, how to reach particular destinations, and how to make use of the infrastructure. Furthermore, cities possess a unique set of navigational tools that lend themselves to creating sub-metaphors. A city metaphor makes this existing knowledge about a structured environment available to the user of a computerized information system (Dieberger, et al., 1998).

This city must be prepared for the EA data we want to represent. Enterprise information is multi-dimensional, subject to constant change, highly related, abstract (e.g. business processes, applications) and concrete (e.g. IT components, employees).

The city is composed of blocks, each of one composed of several buildings with several stories. We consider that this city could be used in many EA representation efforts. Therefore, we will demonstrate it with the *applications architecture*. Let us assume that that an *application* is embed in a *system* (of *applications*) and that it contains several *modules*. Each *module* has its own functionality. Additionally, an *application* has several properties we would like to see represented. Table 12 makes the bridge between the city metaphor and the *applications architecture*:

Table 12 – City metaphor to represent *applications architecture*

City metaphor	Possible use
Block	<i>System</i> (or other <i>application</i> aggregator)
Building	<i>Application</i>
Story	<i>Module</i>
Base story	Base <i>module</i> of an <i>application</i>
Story interior	<i>Module</i> details
Block position within the city	Arbitrary <i>system</i> property
Block pavement color	Arbitrary <i>system</i> property
Building position within the block	Arbitrary <i>application</i> property
Building height	Number of <i>modules</i>
Building shape	Arbitrary <i>application</i> property
Building roof shape	Arbitrary <i>application</i> property
Roof color	Arbitrary <i>application</i> property
Story position	Arbitrary <i>module</i> property
Story color	Arbitrary <i>module</i> property

Besides the dimensions obtained with the table above, we can add an arbitrary number of dimensions through interactivity: for example, displaying *application* information, when the user passes the mouse over a building. Additionally, the user could dynamically map the metaphor to the city, according to its preferences of visualization. The only concerns should be that size, position and shape dimensions should be used when representing discrete variables. On the other hand, color (or texture) should represent continuous variables.

Another benefit of this metaphor is that we can explore it to higher levels. For example, we could use a taxi to travel through the blocks (Figure 38), or an elevator to explore an application interior (e.g. their relations with business processes, etc.). He also could use the helicopter view (Figure 39) to overview the city, giving a panorama of the enterprise applications. This way, visualization would stress the roof color / shape and the building shape dimensions. However, if the user zooms into a block, he can analyze the details about the *system* it represents; getting an overview of the *applications* the system contains (Figure 40).



Figure 38 – “cITy” navigation

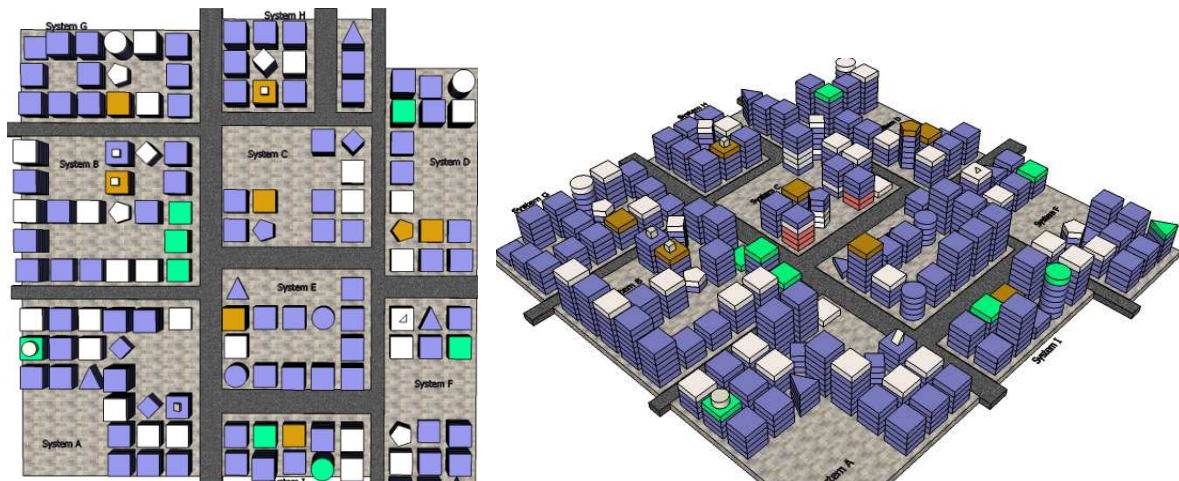


Figure 39 – Helicopter view over “the cITy”



Figure 40 – Details on a *system* of “the cITy”

This city prototype was implemented in a 3D graphical modeling tool: *Google SketchUp*. This is a powerful and easy-to-learn 3D software tool, great for building and modifying 3D models quickly and easily (Google, 2007). In addition, it has the capability to export models to Web formats, like VRML, which we think essential when sharing the prototype.

6.2. “CMDB Web Viewer”

Our second prototype uses a CMDB as its data bank. CMDB is the ITIL recommended data repository. We think this is the best way to store EA data, since it also serves multiple enterprise interests (e.g. several ITIL processes) and can be a highly integrated solution. We underline that a CMDB is not in any way associated with EAs. Only the way you understand that database that can lead it to an EA model data source. In the context of our thesis, only “Web Viewer” will be analyzed; however, this CMDB was built with an open interface and has several clients (Figure 41).

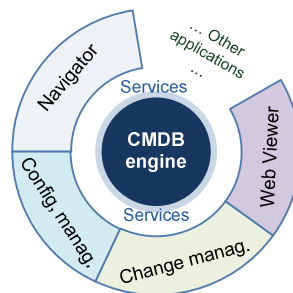


Figure 41 – CMDB engine and its clients

“CMDB Web Viewer”’s subjacent technology is the *OutSystems* platform. It offers the possibility to rapidly create and deploy enterprise applications and then change them at any stage of their life cycle. Moreover, it promotes the development of 3-tier (Definition XXII) compliant applications.

6.2.1. Conceptual model

Before we jump to the actual implementation of the system, it is opportune to introduce its *conceptual model* (Definition XVII). Well-defined conceptual models propitiate faithful mental models (Definition XVI) on stakeholders. If an enterprise architect presents a good conceptual model, people will be more prepared and motivated, and consequently better understand the system, since they previously understood its structure, guidelines, and motivation.

Our implementation follows some ITIL concepts: CMDB and CI:

Definition XX – CMDB

A CMDB (Configuration Management Database) is a logical database which keeps track of CI’s, their versions, status, and the relationships between them (Bon, 2005).

Definition XXI – CI

A CI (Configuration Item) is an IT component and the services this provides. CI’s can include PC hardware, software, network components, servers, central processors, documentation, procedures, services, IT users, IT staff, business units, etc. (Bon, 2005)

These concepts had to be extended, and as a result, our terminology is the following (Figure 42):

- **Metadata:** Meta-CI’s and Meta-Relations: entities and business rules of the organization and IT:
 - *Meta-CI:* type of CI. Ex.: Application, System, Employee ...
 - *Meta-Relation:* relation between two Meta-CI’s. Ex.: Application is used by Employee.
- **Data:** Relies on metadata to instantiate CI’s and relations:
 - *CI:* Configuration Item (Definition XXI). Ex.: SAP, Commercial System, John Doe ...
 - *Relation:* connection between two CI’s. Ex.: SAP is used by John Doe.

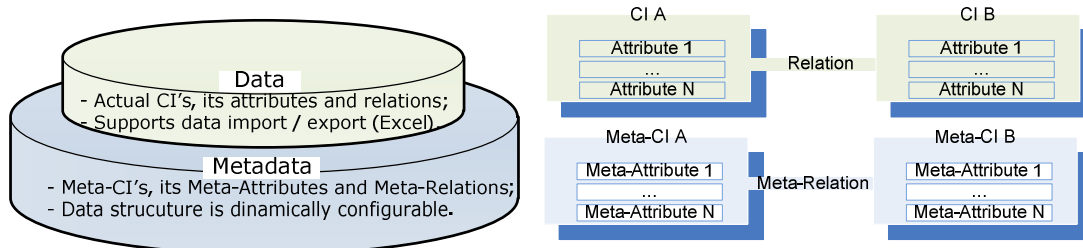


Figure 42 – CMDB conceptual model

In practice, the “metadata part” defines the possibilities and rules that the “data part” can handle. It is like a database at user level: users populate the database according to how administrators had described the meta-level.

Figure 41, Figure 42, Figure 43, and Table 13 could be part of a hypothetical presentation with the objective to pass knowledge about the conceptual model of the system, and thus, helping to persuade people about the importance of EA.

6.2.2. Presentation layer

“Web viewer” works as the CMDB back-office. It is the main *graphical user interface* of the CMDB engine. It consists of a group of web pages designed to view, edit, import, export, and navigate through EA data. Those web pages are divided in two logical parts: the “*metadata*” and “*data*” (Figure 43) which correspond to the system’s *conceptual model* (Figure 42).

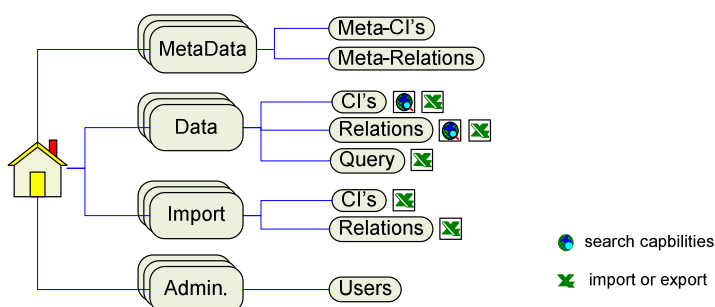


Figure 43 – CMDB website conceptual map

Dürsteler states that the use of appropriate navigation metaphors can help to make the structure of modern information systems easier to understand, and therefore, easier to use (Dürsteler, 2002). In this sense, we found imperative to rely our system on a well-defined metaphor. The supporting *metaphor* of “Web viewer” is the notion of someone’s personal identity and its surrounding friends. The basic concepts of the applied metaphor are the following are stated at Table 13. Additionally, an interesting comparison to make is the one provided by Figure 44 and Figure 45.

Table 13 – “Web Viewer” supporting metaphor

“World”	System
A person has an identifying name (Ex.: John)	A CI has a name (ex.: SRV_HP33)
A picture helps recognizing that person	An icon helps to recognize a CI
That person has a nationality (ex. Portuguese, British, etc.)	A CI has a type (ex.: Server)
A person has characteristics (Ex.: height, birthday, etc.)	CI's have attributes that better describe them
He/she has friends, colleagues, boss, etc.	CI's are related with other CI's (Ex.: SRV_HP33 has Applications installed, is used by Employees, etc.)



Figure 44 – “Web Viewer” supporting metaphor

CI's » Aplicações » GESFORMA Path and CI name



Aplicação "GESFORMA"

Atributos	CI attributes
Nome:	GESFORMA
Meta-CI:	Aplicação
Descrição:	
Referência:	608
Referência original:	a661
Activa:	Yes
Alias:	
Externa:	No
Número de Utilizadores:	
Portfólio:	No
Serviço TI:	No
Sistemas de Suporte:	No
SOX:	No
Terceiros:	No

Operations

Relações	Related CI's
Criar Relação	
Contém Módulos:	
base	
MNP-2	
E-Commerce	
WebSite	
Está abrangido em Âmbitos:	
Departamental	
É desenvolvido por Fornecedores:	
Microsoft	
É mantido por Fornecedores:	
Microsoft	

Figure 45 – CMDB CI details

6.2.3. Functional analysis

This section describes the functional aspect of the solution. The software part of the solution consists of an application with several client modules (Figure 41). Working as a one, these modules deliver a set of functions we think essential to manage a CMDB and in our case, EA models. However, in the present context we are only interested in the functionalities of the "Web Viewer" module:

- *Knowledge repository:*
 - Meta-CI descriptions (ex.: definition of Application) and Meta-Relation descriptions;
 - CI's and Relations description.
- *Metadata management* (Figure 46, top and middle):
 - Create, edit and delete Meta-CI's and Meta-Relations;
 - Manage Types of Relations (ex.: "includes", "depends"...);
 - Navigate through Metadata;
 - Filter capabilities;
 - Meta-CI details;
 - Define Meta-Attributes and an icon per Meta-CI;
- *Data management* (Figure 46, bottom);
 - Create, edit and delete CI's and Relations;
 - Define Attributes of a CI;
 - Search CI by string;
 - Filter CI's and Relations (regular and advanced);
 - Navigate through CI's and Relations;
 - Make a query;
 - Links to documents;
- *Import/Export:*
 - Import CI's, Relations and CI's Attributes through an Excel file;
 - Export CI's (with or without its Attributes);
 - Export Relations;
- *Administration:*
 - User management;
 - Profile management.

Meta-CI's

Em desenvolvimento. Ver auxiliares

[Criar Meta-CI](#) | [Ver CIs](#)

	Aplicação	Suporta um ou vários processos de negócio e tem as seguintes propriedades: - tem um conjunto de fu ...		
	Base de Dados	Sistema que organiza informação com uma estrutura regular. Esta estrutura pode ter a forma de uma ta ...		
	Categoria de Aplicação	Forma de classificação de aplicações. Ex.: Package, À medida.		
	Documento	Documentação relevante para a descrição ou operação da infra-estrutura. Ex.: Especificação de Requi ...		
	Domínio	Área de negócio representativa da cadeia de valor.		
	Empresa	Pode ter relação de detenção com outras empresas. Pode ser empresa cliente, fornecedora directa ou ...		
	Equipamento	Estação de trabalho, cliente ou servidora. (Referido como "máquina" no modelo inicial)		
	Fornecedor			
	Função de Negócio	Caracteriza relação entre domínio e aplicação. Finalidade de uma aplicação no contexto de um domínio ...		
	Funcionalidade	Conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades espe ...		
	Grupo de Funcionalidades	Agregação de funcionalidades tendo em conta um processo de negócio. Ex.: Billing, Gestão Comercial ...		
	Interface	Canal de troca de informação entre duas ou mais aplicações, que podem ser de dois tipos, Ponto-a-Pon ...		
	Macro-Funcionalidade	Agrupa funcionalidades.		
	Macro-Processo	Agregador macro dos principais tipos de processos de negócio		

Meta-CI's » Aplicação

[Criar Meta-CI](#) | [Criar Meta-Relação](#) | [Ver Aplicações](#)

Aplicação

Nome:

Auxiliar?

Em desenvolvimento?

Descrição: Suporta um ou vários processos de negócio e tem as seguintes propriedades:
- tem um conjunto de funcionalidades, de forma a responder aos requisitos de negócio, que se podem traduzir em módulos ou não;
- disponibiliza forma de acesso às funcional ...

Fontes:

Referência:

Meta-Relações

[Criar Meta-Relação](#)

Relação	CI relacionado
Contém	Módulo
Suporta	Função de Negócio
Está abrangido em	Âmbito
Está instalado em	(Ambiente) Configuração
Está incluído em	Sistema
Está incluído em	Grupo de Aplicações
É desenvolvido por	Fornecedor
É mantido por	Fornecedor
É explorado por	Fornecedor
É suportado por	Serviço
É afectado por	(Tipo CRUD) RFC
É utilizado por	Macro-Processo

Meta-Atributos

[Criar Meta-Atributo](#)

Nome	Descrição
Activa	A aplicação está de momento activa?
Alias	Outro nome pelo qual também a aplicação é con...
Externa	
Número de Utilizadores	Quantidade de utilizadores desta aplicação
Portfólio	A aplicação faz parte do portefólio aplicacion...
Serviço TI	A aplicação é provedora de serviços informáti...
Sistemas de Suporte	Faz parte dos sistemas de suporte?
SOX	A aplicação é SOX compliant?
Terceiros	Desenvolvida por terceiros?

CI's » Macro-Funcionalidades

[Criar Macro-Funcionalidade](#) | [Criar Relação](#)

Filtrar:

Filtre por Meta-CI
Ex.: Aplicação, Pessoa, ...

Pesquise por nome, descrição ou referên
Ex.: 5263, DSL, contactos, ...

[Ver Atributos](#) | [Ver Descrição](#)

Nome CI	Descrição
Macro-Funcionalidade Gestor de Versões	
Macro-Funcionalidade Actualização do cadastro	
Macro-Funcionalidade Análise de fiabilidade	
Macro-Funcionalidade Análise de investimentos	
Macro-Funcionalidade Autenticação de utilizadores/equipas	
Macro-Funcionalidade Calculo de curto circuitos	
Macro-Funcionalidade Configuração Normal da Exploração da Rede	
Macro-Funcionalidade Configurações	
Macro-Funcionalidade Consulta de Áreas Afectadas/Grandes Incidentes	
Macro-Funcionalidade Consulta de Comunicações de Avaria	

Figure 46 – CMDB “Web viewer” screenshots (Meta-CI list, Meta-CI details and CI list)

6.2.4. Logical architecture

It is common and indeed a good idea to guide the design and implementation phases of a software project within well-defined patterns. In software engineering, a *design pattern* is a general repeatable solution to a commonly occurring problem in software design (e.g. proxy, client-server, OSI reference model, 3-tier, etc.). 3-tier architecture pattern (Definition XXII) guided the architecture of the CMDB implementation.

Definition XXII – 3-tier architecture

3-tier architecture is a client-server architecture in which the user interface, functional process logic (business rules) and data storage / access are developed and maintained as independent modules, most often on separate platforms (FOLDOC, 1998).

The implemented architecture supports perfectly our solution: *presentation layer* (includes *user interface*) plays a central role in our system, since this thesis regards visualization. “Web viewer” and the other modules (Figure 41) communicate with public services delivered by the *logic layer*.

The *data layer* corresponds to a central CMDB (Definition XX) and its data access features. Note that this CMDB is much more than the data source of our solution; it may represent the data source of many other applications, since it contains arbitrary and multi-dimensional enterprise / IT information (Figure 47).

For example, the implemented CMDB can support the ITIL processes of Change Management, Configuration Management, etc., as stated in “Case Study”.

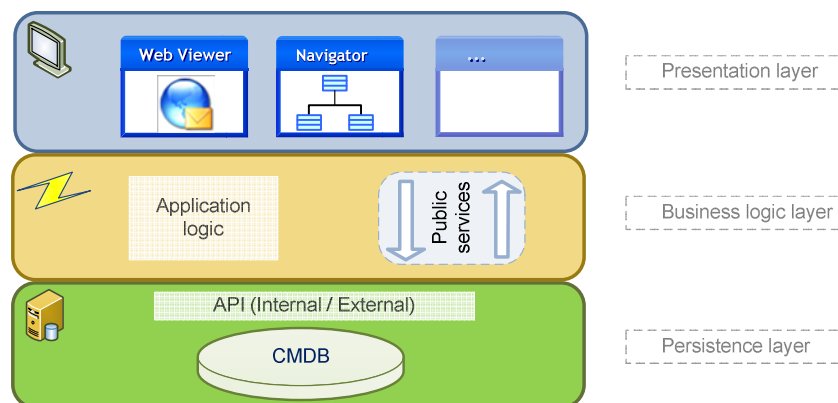


Figure 47 – Implementation logical architecture

6.2.5. Physical architecture

Physical architecture represents the actual deployment of the logical modules (Figure 48). We present a possible solution (Figure 48), but obviously, there exist other deployment possibilities. A central CMDB keeps and operates the EA data and other enterprise data (Figure 41).

“Web viewer” module is deployed on the CMDB physical server. However, it can be accessed anywhere with an internet connection (or intranet, depending on the type of deploy). A SQL database server actually holds the EA data. Navigator accesses public web services to access the CMDB data. Finally, other ITIL related applications get the data using the same public services (Figure 41).

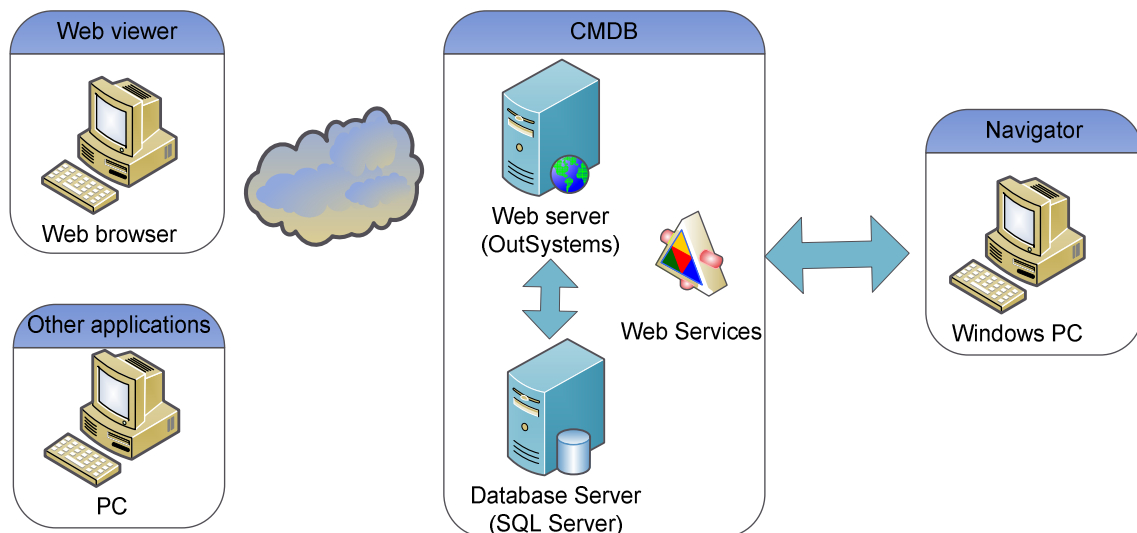


Figure 48 – Implementation physical architecture

6.2.6. API (Application Programming Interface)

It is widely known that an application is early dead if does not offer integration capabilities. We carefully designed a set of services so that other applications could make use of them. In fact, the functionalities presented above rely on them to work. In practice, the CDMDB front office and all client modules (Figure 41) reuse them in its interface so there is no repeated code. Together, these services form the interface (*API*) (Figure 41, Figure 42) of what we call CDMDB engine (see Appendix I):

- 1) Delete CI
- 2) Delete Meta-CI
- 3) Delete Meta-Relation
- 4) Delete User
- 5) List CI's
- 6) List Meta-CIs
- 7) List Meta-Relations
- 8) List Relations
- 9) CI details
- 10) Meta-CI details
- 11) Meta-Relation details
- 12) Relation details
- 13) Get Attribute
- 14) Update Attribute
- 15) Create or update CI
- 16) Create or update Meta-CI
- 17) Create or update Meta-Relation
- 18) Create or update Relation

We had the concern to encapsulate several similar functions in the same service, which is a good practice, according to SOA (Service Oriented Architecture). A good example is that the same function serves to "list", "search", and "filter" CI's.

6.3. Summary

Two aspects have to be highlighted in our solution: it is well sustained by theory and it naturally deals with the subjacent data dynamism.

In interface design, it is a good practice to choose a metaphor following some criteria. We adopted metaphors that everyone is familiar with: “a person and its friends” and a “city”.

The following table outlines the implementation of the two prototypes:

Table 14 – Implementation formal definition

Aspect	“The cITy”	“CMDB Web Viewer”
Metaphor	“City”	“A person and its friends”
Description	3D environment based on a metaphor that people know well.	CMDB front-office, which has the capability to deal EA data.
Represented domain	Applications architecture	Configuration Items and its relations
Underlying technologies	Google SketchUp	OutSystems, WebServices
Interfaces	[n/a]	IN: CMDB API
Reference models	<ul style="list-style-type: none"> • 3-tier architecture; • 10 interface heuristics (Nielsen, 1994); • Visualization seeking mantra: <ol style="list-style-type: none"> 1. Overview, 2. zoom, 3. filter and 4. details-on-demand (Shneiderman, 1996) 	

7. Case Study

There is no better way to prove the value of a solution to a problem than testing it within a real situation. Therefore, we went through a project that puts in practice our proposal. In order to better specify the general scope of this case study we present Figure 49:

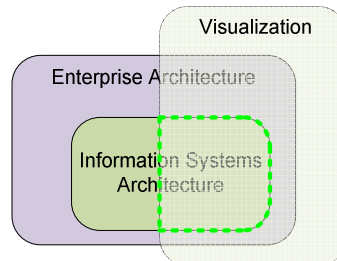


Figure 49 – Scope of case study: ISA visualization

Despite the project had several distinct objectives, the interesting part to this thesis is the one identified: visualization of information systems architecture (ISA). The project we are going to describe represents our case study.

7.1. The Project

This project was born in the biggest Portuguese company, more in particular at the department of IT. The central objective of the project was the implementation of a CMDB. In addition, we should import existing enterprise data and adapt existing functioning processes. Project main stakeholders (and clients) were the IT Architecture sub-department. Target users were the teams that already used the Access Database as an IT database.

For an effective IT management of the company, the IS department needs to keep a configuration description of its infrastructure that supports the services fundamental to the business processes already implemented within the company. Besides the infrastructure information, business processes need other business information, its relations with the IT, with the suppliers and their contracts, etc.

Five finalist students, a senior consultant, and an external consultant composed our team (Figure 50). All these resources worked at the mentioned company three days a week. The senior consultant is an ITIL expert and managed the project and its sub-projects. The external consultant gave precious directions at many points of the project.

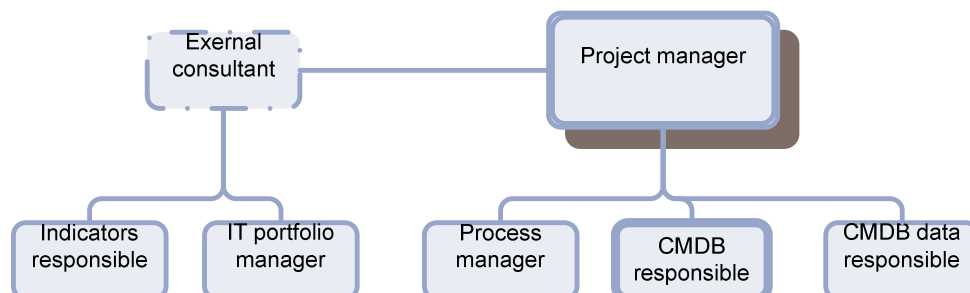


Figure 50 – Project organization chart

It was in this context that the project started. As a guideline and framework, we follow the ITIL best practices and recommendations in the project development process. In few words, we considered essential to implement a CMDB and some processes essential to solve some of their problems. One of the first tasks was to find the information sources and from here sketch the subjacent meta-model. Main data sources were MS Excel sheets and an MS Access database. Since the result was a big set of entities and their relations, we organize these them by architecture (The Open Group, 2006):

- *Business architecture*: company organization and business processes;
- *Applications architecture*: mapping of applications and their relations with processes;
- *Technology architecture*: software and hardware that supports applications;
- *Information architecture*: logic structure of the company data entities;
- *Services architecture*: definition of orchestrated services that support business and IT.

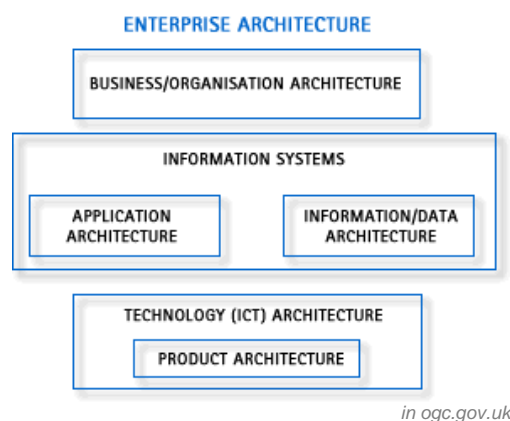


Figure 51 – Organization of the EA

It was fundamental to prepare the CMDB to handle the needs of information. Business needs, TI services, relation with suppliers and partners, among others are examples of things that change frequently. To support this need, with the minimum effort and no programming required, the CMDB offers two distinct configurable levels (Figure 42):

- *Metadata*: conceptual definition of entities (Meta-CI's) and their relations;
- *Data*: instantiation of those entities and relations with concrete data.

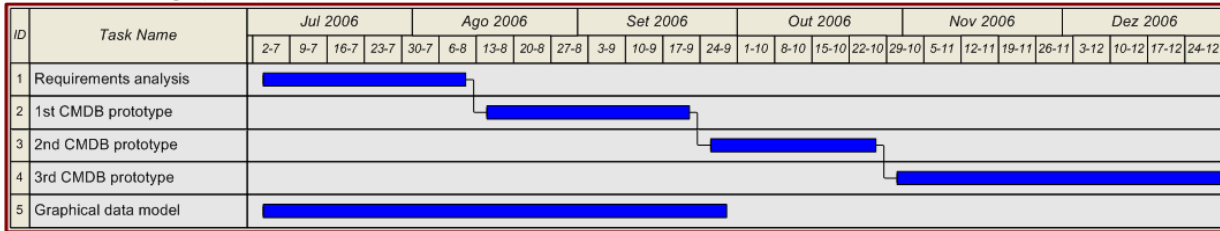
These levels offer the capability to change the subjacent data model at any time without unexpected consequences. For example, if we desired to add an attribute to the Meta-CI "Application", we could do it at any time at through the same interface we use to manage concrete data, even if we already have that table populated. The back-office offers a friendly and graphical user-interface, through which the user can manage the levels described.

We prepared other modules that directly support the processes with the same name: "Change Management" and "Release Management", inspired in some ITIL best practices.

7.2. Chronology

Planning is the process of thinking about the activities required to create a desired future on some scale. This thought process is essential to the creation and refinement of a plan, or integration of it with other plans. In the context of our project, the following plan was followed:

First stage (2006):



Second stage (2007):

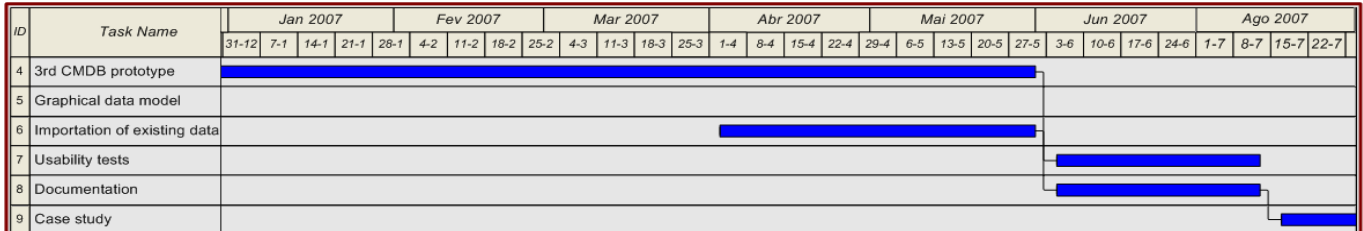


Figure 52 – Project crhronology

Through the first stage of the project (2006), we essentially understood data sources, current situation analysis, data entities, and relations, among others. In parallel, we developed the first prototypes and the graphical data model that illustrated the EA entities.

At the second stage, we developed the final prototype of the CMDB engine and its client modules. Additionally, we tested the application by several ways. Finally, we produced the documentation for the client enterprise, as well as the case study.

7.3. Development Process

The prototyping model is a software development process that begins with requirements collection, followed by prototyping and user evaluation. *Prototyping* is the process of quickly putting together a working model (a *prototype*) in order to test various aspects of a design, illustrate ideas or features, and gather early user feedback. We invested in an iterative prototyping as the system design process. Several prototypes were produced in order to evaluate the acceptability, usability, and feasibility of the system. In fact, prototyping is one of the most used techniques of requirements elicitation, in requirements engineering.

We evaluated the prototype by evaluating its *usability*. *Usability* is a general term that encompasses everything having to do with “ease of use”. *Usability testing* involves measuring the ease with which users can complete common *tasks* on an *interface* of any system. The *user interface* is the boundary between users and the system. Users interact with the system through it. For this reason, the user interface must be first evaluated in order to validate it and release the system. To accomplish it, there are four well-known techniques (Jeffries, et al., 1991):

Table 15 – Comparison between interface evaluation techniques (Jeffries, et al., 1991)

	Advantages	Disadvantages
Heuristic evaluation	<ul style="list-style-type: none"> Identifies many more problems; Identifies more serious problems; Low cost. 	<ul style="list-style-type: none"> Requires UI expertise; Requires several evaluators.
Usability testing	<ul style="list-style-type: none"> Identifies serious and recurring problems; Avoids low-priority problems. 	<ul style="list-style-type: none"> Requires UI expertise; High cost¹; Misses consistency problems.
Guidelines	<ul style="list-style-type: none"> Identifies recurring and general problems; Can be used by software developers. 	<ul style="list-style-type: none"> Misses some severe problems.
Cognitive walkthrough	<ul style="list-style-type: none"> Helps define users' goals and assumptions; Can be used by software developers. 	<ul style="list-style-type: none"> Needs task definition methodology; Tedious; Misses general and recurring problems;

We felt that *usability testing* was the most adequate technique, so the system was highly tested during its various prototypes. In fact, given the lack of time, the system was tested as it was being developed, and consequently many flaws were corrected almost on the fly.

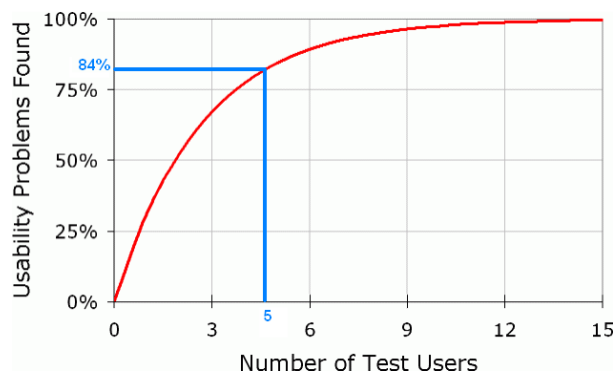


Figure 53 – Quantity of usability problems vs. number of test users (Nielsen, 2000)

Some people think that usability testing is very costly and complex and that user tests should be reserved for the rare web design project with a huge budget and a lavish time schedule. However, elaborate usability tests are a waste of resources. The best results come from testing no more than five users and running as many small tests as you can afford (Nielsen, 2000) (Figure 53).

In this context, we insisted in a set of tasks and tested frequently with four test users. These tasks lead us to the identification and further correction of many flaws, confirming that prototyping was the best model of development to use in our context.

7.4. Results

A deliverable is any tangible outcome that is produced by the project. Knowing that the project had many phases, and took more than a year, it is expectable to have different and several deliverables:

- *An application (CMDB) and its modules:*
 - CMDB engine;
 - CMDB front-office (known till here as “Web Viewer”);
 - Configuration management;
 - Release management (Typical Requests and Application Maintenance).

¹ Nielsen argues it is possible to have low costs using this technique (Nielsen, 2000)

- *Graphical data model*, in which entities are classified per architecture (business, service, application, technology, and information);
- *Documentation*:
 - Functional requirements specification;
 - Technical specification.
- *Case study*: current chapter.

7.5. Summary

The CMDB application was meant to:

- Gather all the relevant information related with IT management, in the scope of the enterprise IS department, thus describing the essential IT components and their relations;
- Provide the functionalities to manage that information: through the back-office or following specific determined processes;
- Be extendable and further adapted to the changes in the data model, with no need of further programming;
- Provide functions and services to attached modules and third party applications.

In order to support existing processes we had to:

- Analyze existing data models, documents, data tables, among others;
- Import existing IT data from several sources (namely sub-enterprises, its organization units, people, main business processes, applications and their modules, etc.);
- Duplicate existing application screens in our application (MS Access);
- Provide user centered documentation.

The development process was the prototyping, which in our context is the most adequate. The most important deliverables of this project were: an application (CMDB) with several modules, a graphical data model, a functional requirements specification, a technical specification and a case study analysis.

8. Evaluation

This chapter presents the proposal evaluation as well as the evaluation of two implemented prototypes: “Web Viewer” and “The cITy”. We consider the experience gathered from the state of the art study, and from the case study, thus looking at EAs and the proposal with “new eyes”.

8.1. Proposal

Considering the state of the EA art survey and the case study that were realized, we are now able to map the theory onto practice and evaluate the success of our proposal. Remembering that our proposal was:

Our proposal is an EA visualization system that uses visual metaphors to represent EA models. With the use of metaphors, the system benefits from the highly patterned and symbolic manner that the brain uses to think and learn. The visualization system has a back-office for the enterprise architect prepare viewpoints and views to stakeholders, and a dynamic front office that presents views according to the stakeholder.

We can argue that:

- Engineering is making things based on scientific principles – as opposed to the intuitive making that defines a craft. Information visualization is practiced as a craft today, based mostly on practical examples, but not on theoretical basics (Kosara, 2007). This proposal is one effort to make information visualization more “engineered”;
- This proposal is well supported by theory, especially in the use of metaphors and the adoption of the IEEE-1471-2000 standard, which is ideal for architecture visualization (IEEE-SA, 2000);
- It speaks the stakeholder language, given that he sees only the views prepared to him;
- It supports different mediums, given that one can copy the resulting view to MS PowerPoint, print it, etc.;
- It strictly follows the solid conceptual model defined, making easy to teach the system and its benefits to people, thus promoting EAs;
- The viewpoint library we proposed serves enterprise knowledge management, storing that information instead of closely depending of people to access it;
- Finally, besides the proposed metaphors (“person and friends” and “the city”), the proposal is open to other innovative and opportune visual metaphors.

8.2. “The cITy”

In what concerns the city metaphor, we think it encompasses a highly expandability and structured system as well it provides a comfortable environment where the user feels the will to explore and learn in a natural way. In short, it clearly helps people understanding of EA models. Finally, despite we used the city metaphor to represent the applications architecture; it could be adapted to other types of architecture. The important is to keep the consistency and follow the best visualization practices.

Unlike the “CMDB Web Viewer”, this prototype was not tested on an enterprise situation. However, its academic potential makes possible future work on him. “The cITy” ideas are well expressed and defined so a future implementation would not be a problem.

8.3. “CMDB Web Viewer”

One of the main deliverables of the case study was the CMDB. This CMDB works as a proof-of-concept with some qualities that we highlight bellow:

- *Portability*: since it was tested in different web browsers (mainly Internet Explorer and Firefox) with success;
- *Consistency*: concept names are transversally used in a consistent way in this thesis, but also in the prototype interface, project documentation, presentations, etc. (Ex.: CI, Meta-CI, Relation, Meta-Relation);
- *Reusability*: the API of the CMDB engine (Figure 41 and Appendix I) is highly reusable. The proof is that four modules (“Web Viewer”, “Navigator”, “change management”, and “release management”) were developed as its clients;
- *Package*: this is not a pure homemade solution: it is generic enough to be deployed on any enterprise without further programming;
- *Security*: depending on the authenticated user, it displays a different set of functions (“write permission”, “administration”, “metadata”, and “data” profiles).
- *Fault tolerance*: a good fault tolerance policy of error handling was implemented at three logical layers (Figure 47). That policy essentially handles database errors, user mistakes, wrong rules, etc.

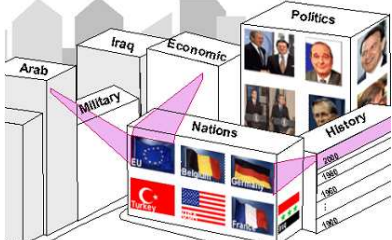
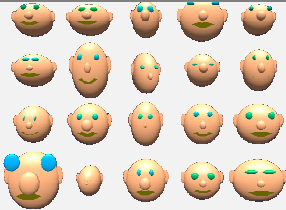
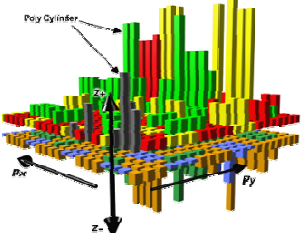

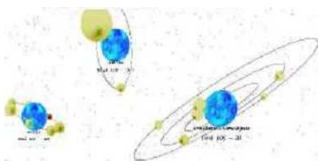
The most innovate and positive point of the CMDB solution is a flexible and dynamic data model. To adapt to it, the visualization system also had to be dynamic and integrated with the flexible subjacent data model; usually visualization systems are adapted to the dynamism of data, but not of the meta-data. This is one of the most important innovations of our system.

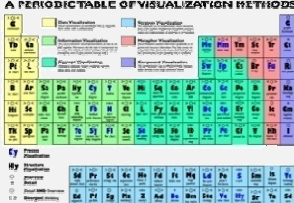
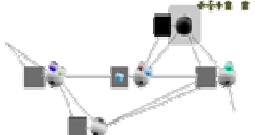
Our proposal uses a CMDB as its data bank. CMDB is the ITIL recommended data repository. We think this is the best way to store EA data, since it also serves multiple enterprise interests (e.g. several ITIL processes) and can be a highly integrated solution. In conclusion, our visualization system has much to win integrating with a CMDB database.

9. Related Work

Current chapter presents an overview list of the projects related to our proposal. Recall that our proposal is the utilization of visual metaphors in an EA visualization system oriented to viewpoints / views. In what concerns the utilization of visual metaphors, we summarized the related projects in the following table (the order is irrelevant):





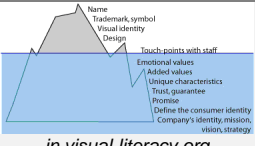
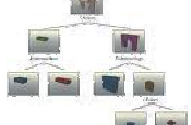
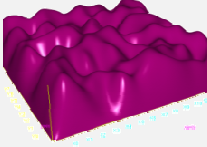

Table 16 – Related work

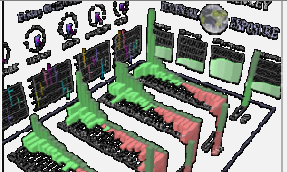

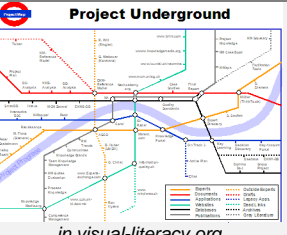
Project	Description
<p>«A City Metaphor to Support Navigation in Complex Information Spaces» [generic] Complex I.S. (like WWW) ↓ City</p>	<p>A major problem for users of modern information systems is the retrieval of new and previously viewed information from the system. Systems like the Word-Wide Web are heavily interlinked but do not communicate structure that helps users to navigate the information it contains. The use of appropriate navigation metaphors can help to make the structure of modern information systems easier to understand, and therefore, easier to use. The authors propose a conceptual user interface metaphor based on the structure of a city. Cities are very complex spatial environments and yet, people are used to navigating within cities. They know how to get information, how to reach particular destinations, and how to make use of the infrastructure (Dieberger, et al., 1998).</p>
<p>«A Cityscape Visualization of Video Perspectives» TV broadcasts and related multimedia documents ↓ Cityscape</p>	 <p>The authors wanted to organize the corpus or a subset along multiple dimensions, or perspectives, adding relevant background material, significantly expanding and accelerating the viewer's comprehension and integration of knowledge about broadcasts, etc. A perspective can provide factual background information, a history of an issue, the view of a biased source, a technical or medical perspective, etc. (Derthick, et al., 2003)</p>
<p>«Chernoff faces» Multivariate data ↓ Human faces</p>	 <p>Chernoff faces display multivariate data in the shape of a human face. The individual parts, such as eyes, ears, mouth, and nose represent values of the variables by their shape, size, placement, and orientation. The idea behind using faces is that humans easily recognize faces and notice small changes without difficulty. Chernoff faces handle each variable differently (Chernoff, 1973).</p>
<p>«3D Representations for Software visualization: sv3D» Software metrics (or other multidimensional data) ↓ 3D solids</p>	 <p>This system uses a 3D representation for visualizing large software systems. By utilizing a 3D representation, it can better represent higher dimensional data than previous 2D views. It uses containers, poly cylinders, height, depth, color and position to represent different dimensions (Marcus, et al., 2003).</p>
<p>«WebBook and WebFrager» WWW ↓ Workspace with books</p>	 <p>WWW has achieved global connectivity stimulating the transition of computers from knowledge processors to knowledge sources. However, the Web and its client software are seriously deficient for supporting users' interactive use of this information. This project consists of two related designs with which to evolve the Web and its clients. The first is the WebBook, a 3D interactive book of HTML pages. The second is the Web Frager, an application that embeds the WebBook and other objects in a hierarchical 3D workspace (Card, et al., 1996).</p>
<p>«A Solar System Metaphor for 3D Visualization of Object Oriented Software Metrics» Software metrics ↓ Solar system</p>	 <p>One way of potentially increasing empirical analysis activity on this realm is to contemplate visualization as a means to readily analyze either static or evolving code to perceive in real-time, suspected areas of risk within the codebase. This project represents a first attempt at 3D visualization of software metrics by using a familiar metaphor to present empirical concepts (Graham, et al.).</p>

<p>«Towards A Periodic Table of Visualization Methods for Management»</p> <p>Visualization methods ↓ Periodic table</p>		<p>This project defines compiles existing visualization methods in order to develop a systematic overview based on the logic, look, and use of the periodic table of elements (Lengler, et al., 2007) (Appendix F).</p>
<p>«3D Gadgets for Business Process Visualization»</p> <p>Business processes ↓ 3D solids</p>		<p>Business visualization is becoming increasingly important, since managers recognize the power of human visual intuition in information-rich decision tasks. The authors propose a 3D visualization system for business processes, through gadgets with behaviors, which include brushing, grouping, and (drill down) manipulation (Schönhage, et al., 2000).</p>

Many other projects are available but were not mentioned due to their similarities with the presented ones. Visual metaphors are used long ago in interface design and visualization. In visualization, certain types of visual metaphors have become popular:

Table 17 – Typical visual metaphors

Metaphor	Example	Possible uses
Temple		<p>Underpinning ideas Great to represent a set of supporting ideas of something in a PowerPoint slide. For example, liberty, equality, and fraternity are the pillars of the French motto.</p>
Pyramid		<p>Proportional relations Simple hierarchical relations</p>
Funnel		<p>Business processes inputs / outputs Sum of parts Filtering</p>
Pie		<p>Percentages How the individual parts makes the all.</p>
Iceberg	 <p><i>in visual-literacy.org</i></p>	<p>Simple ideas Used to contrast what “is seen” versus what is not. For example, in an enterprise, some organic units interact with the client, but others not.</p>
Tree		<p>Hierarchical or process Tree metaphor is useful when representing concepts organized in a recursive way (e.g. staff). Another common use is direct the use of the tree concepts: roots → origins or causes, trunk → transforming process, branches → results or consequences, leaves → detailed results.</p>
Surface (mountain)	 <p><i>in npaci.edu</i></p>	<p>Scientific visualization However, Tegarden suggests it could be used in business visualization helping managers to identify patterns (Tegarden, 1999).</p>
Map		<p>Geographic information representation Maps are a potential "natural" representation for entities that can be analyzed geographically. For example, retail sales per country or a disease spread.</p>

Room		<p>Business metrics visualization</p> <p>In this representation, information is assigned to various business graphics and is displayed on a wall of the room or on the floor of the room. It allows a great deal of information to be placed in a relatively small space.</p>
Dashboard (a car cockpit)	 <p><i>in bluespringsoftware.com</i></p>	<p>Business metrics visualization</p> <p>A dashboard (also known as cockpit or scorecard) is a <i>user interface</i> that organizes and presents information in a way that is easy to read. The name refers to the fact that it can sometimes look like the dashboard of a car.</p>
Metro map	 <p><i>in visual-literacy.org</i></p>	<p>Project visualization</p> <p>Using a common metro map representation, one can display project related information. Routes are connected activities and stations are milestones.</p> <p>Concept visualization</p> <p>General concept viewer: stations are concepts (people, websites, etc.) and routes are relations between concepts. Colors represent types of relations.</p>

Visualization through visual metaphors is always being reinvented through innovative usages, depending on the field and the designer imagination. For example, the Oscar metaphor (Figure 8), the heaven & hell metaphor, the camera metaphor (Figure 22), the bridge metaphor, the slide metaphor, etc. Other works on visual metaphors, besides not being too reusable, are great in their uses (Figure 54):

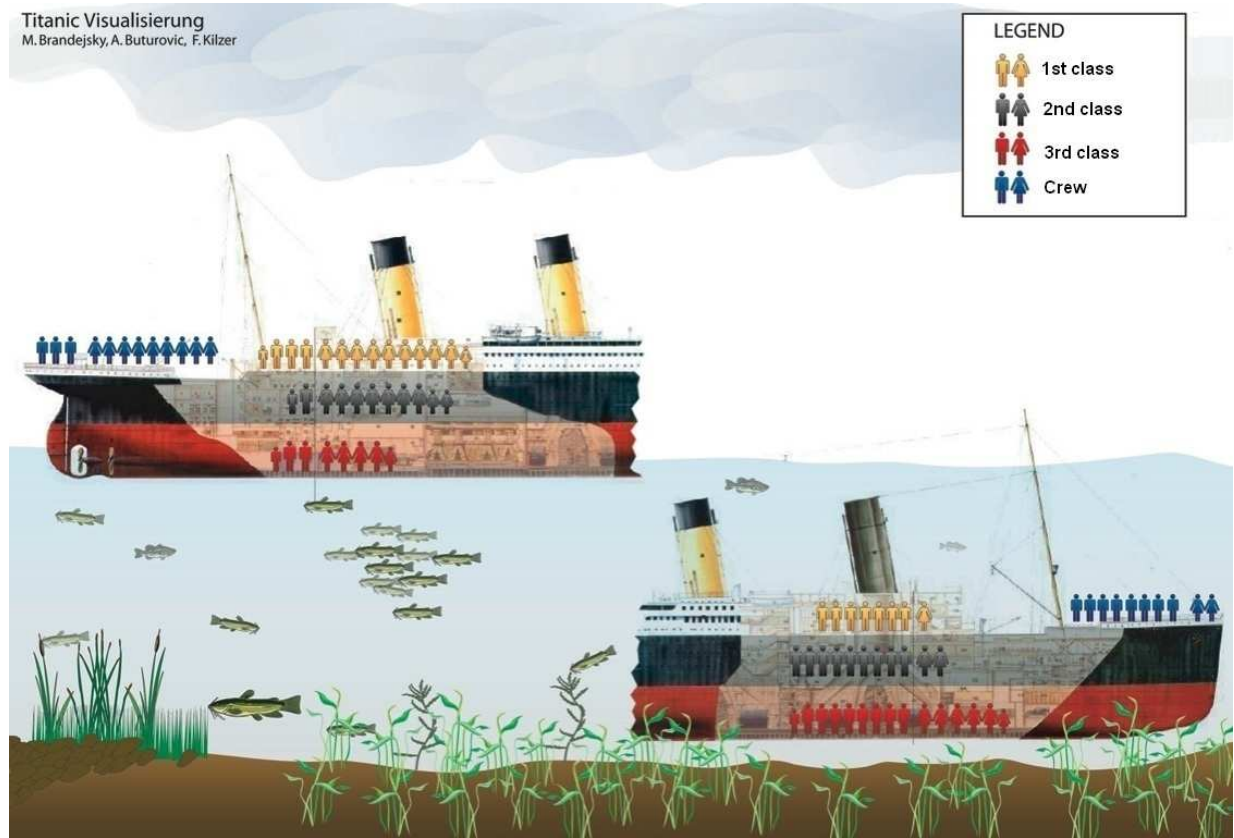


Figure 54 – Titanic disaster: deaths per class, sex, and age (Kosara, 2007)

10. Conclusion

An enterprise architect plays a central role in understanding stakeholder concerns, creating EA views from EA models, managing viewpoint library, etc. There are currently many frameworks and tools concerning this domain. However, EA delivery and visualization is not the central concern of most EA tools. In fact, even enterprises with an EA program, do not consider EA visualization as part of the process. One of the consequences is that nowadays, most people misunderstood EA modeling with EA visualization. This approach has many problems, but the most important one is that most part of the stakeholders does not feel comfortable looking at lots of boxes and arrows. Instead, they would prefer representations with more real objects and situations that speak their language and have the correct level of detail.

It is now clear and straightforward to think of *visualization* as a leading method for communicating within the organization: solving problems, supporting decision-making, justifying arguments, assimilating complex structures, among many other endings.

After a long study on *visualization* and EAs, we consider a fact that a proposal and a possible full implementation would enclose much more quality if accomplished by experts on those fields, due to the many precious ideas and experiences coming from distinct sources.

Many visualization systems are being developed in other fields due to the data complexity and need to understand it. In EA, we are in the same conditions, and therefore it is imperative to bet on those systems.

In what concerns the proposed city metaphor, we think it comprises a highly expandable and structured system, providing a comfortable environment where the user feels the will to explore and learn in a natural way. In short, it clearly helps people understanding of EA models. Finally, despite we used the city metaphor to represent the applications architecture; it could be adapted to other types of architecture. The important is to keep the consistency and follow the best visualization practices.

The presented system still needs a better formal language definition. Future work will better define the language between domain and visual artifacts. Additionally, the city metaphor could be better explored as well as other metaphors with many potential.

This work tries to deliver the best EA visualizations to people. In other words, it solves the “How?” question of EA visualization. However, we need to solve another important question: “Who?”: we need to define target stakeholders of visualizations as well as their concerns, through a formal algorithm to define viewpoints.

References

Ambler Scott W. Enterprise Unified Process (EUP) Home Page [Online] // EUP. - March 1, 2006. - August 4, 2007. - <http://www.enterpriseunifiedprocess.com/>. - Ambyssoft Inc..

Arbab Farhad [et al.] Visualisation of Enterprise Architectures [Report] / Centrum voor Wiskunde en Informatica (CWI) ; Telematica Instituut (TI) . - 2003. - ArchiMate/D3.4.1(Q4).

Averbukh V. L. Visualization Metaphors [Journal]. - New York, NY, USA : Plenum Press, September 2001. - 5 : Vol. 27. - pp. 227-237. - ISSN: 0361-7688.

Baida Ziv Architecture Visualization [Master Thesis in Computer Science]. - Amsterdam : Cap Gemini Ernst & Young, February 2002.

Bon Jan Van Foundations of IT Service Management: based on ITIL [Book]. - [s.l.] : Van Haren Publishing, 2005. - 2Rev Ed. - ISBN: 9077212582.

Bosma Hans [et al.] Requirements [Report] / Telematica Instituut. - Enschede, The Netherlands : [s.n.], 2002. - ArchiMate/D4.1.

Brockhoff Rainer User Interface Metaphors [Article]. - 2000.

Brooke J. SUS: a "quick and dirty" usability scale. - 1996.

Card Stuart K., Mackinlay Jock D. and Shneiderman Ben Readings in Information Visualization: Using Vision to Think [Book]. - San Francisco, CA : Morgan Kaufmann, 1999. - 1st Edition. - ISBN: 1558605339.

Card Stuart K., Robertson George G. and York William The WebBook and the Web Forager: an information workspace for the World-Wide Web [Conference] // Conference on Human Factors in Computing Systems. - Vancouver, Canada : ACM Press, 1996. - pp. 111-117. - ISBN:0-89791-777-4.

Chernoff Herman The Use of Faces to Represent Points in K-Dimensional Space Graphically [Journal] // Journal of the American Statistical Association. - 1973. - 342 : Vol. 68. - pp. 361-368.

Daniel Diann The Rising Importance of the Enterprise Architect [Online] // CIO. - CXO Media Inc., 2007. - April 20, 2007. - <http://www.cio.com/article/print/101401>.

Derthick Mark [et al.] A Cityscape Visualization of Video Perspectives [Conference] // Proceedings of the National Academy of Sciences Arthur M. Sackler Colloquium on Mapping Knowledge Domains. - Irvine, CA : [s.n.], 2003.

Dieberger Andreas and Frank Andrew U. A City Metaphor to Support Navigation in Complex Information Spaces [Journal] // Journal of Visual Languages and Computing. - 1998. - Vol. 9. - pp. 597-622. - 6.

Dürsteler Juan C. Visual Metaphors [Online] // Inf@Vis!. - InfoVis, June 17, 2002. - April 9, 2007. - <http://www.infovis.net/printMag.php?num=91&lang=2>.

FOLDDOC Three-tier [Online] // FOLDDOC - Free On-Line Dictionary of Computing. - Imperial College Department of Computing, May 13, 1998. - May 6, 2007. - <http://foldoc.org/?three-tier>.

Google SketchUp Home [Online] // Google SketchUp. - Google, 2007. - September 14, 2007. - sketchup.google.com.

Graham Hamish, Yang Hong Yul and Berrigan Rebecca A solar system metaphor for 3D visualisation of object oriented software metrics [Conference] // ACM International Conference Proceeding Series. - Darlinghurst, Australia : Australian Computer Society, Inc.. - Vol. 99. - pp. 53 - 59. - ISBN ~ ISSN: 1445-1336 , 1-920682-17-1.

Haan Johan den Visualizing Architecture [Online] // The Enterprise Architect. - September 5, 2007. - September 14, 2007. - http://www.theenterprisearchitect.eu/archive/2007/09/05/Visualizing_Architecture.

Horton William Illustrating Computer Documentation: The Art of Presenting Information Graphically on Paper and Online [Book]. - New York : Wiley, 1991. - 1st Edition : p. ISBN: 0471538450.

IEEE-SA IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [Journal]. - Piscataway, NJ, USA : IEEE Computer Society, September 2000.

iSixSigma Business Metric [Online] // iSixSigma. - iSixSigma, June 22, 2005. - July 18, 2007. - http://www.isixsigma.com/dictionary/Business_Metric-830.htm.

Jeffries Robin [et al.] User Interface Evaluation in the Real World: A Comparison of Four Techniques [Conference] // Proceedings of CHI'91 / ed. Laboratories Hewlett-Packard. - New Orleans : [s.n.], 1991.

Kirakowski Jurek Questionnaires in Usability Engineering: A List of Frequently Asked Questions [Online] / prod. HFRG. - Human Factors Research Group, June 2, 2000. - June 13, 2007. - <http://www.ucc.ie/hfrg/resources/qfaq1.html>.

Kleinmuntz Don N. and Schkade David A. Information Displays and Decision Processes [Journal] // Psychological Science. - [s.l.] : Blackwell Synergy, July 1993. - 4 : Vol. 4. - pp. 221–227.

Kosara Robert Caring about the Data [Online] // EagerEyes.org. - September 4, 2007. - September 18, 2007. - <http://eagereyes.org/blog/caring-about-the-data.html>.

Kosslyn Stephen Michael Elements of Graph Design [Book]. - January : W.H. Freeman & Company, 1994. - ISBN: 071672362X.

Kosslyn Stephen Michael Image and Mind [Book]. - Harvard : Harvard University Press, 1986. - Reprint edition. - ISBN: 0674443667.

Kruchten Philippe Architectural Blueprints The "4+1" View Model [Journal] // IEEE Software. - November 1995. - 6 : Vol. 12. - pp. 42-50.

Lankhorst Marc Enterprise Architecture at Work: Modelling, Communication, and Analysis [Book]. - [s.l.] : Springer, 1998. - 1st Edition. - ISBN: 3-540-24371-2.

Lengler Ralph and Eppler Martin J. Towards A Periodic Table of Visualization Methods for Management [Conference] // IASTED Proceedings of the Conference on Graphics and Visualization in Engineering (GVE 2007). - Clearwater, Florida, USA : ACTA Press, 2007.

Lewis James R. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use [Journal] // International Journal of Human-Computer

Interaction. - Mahwah, NJ, USA : Lawrence Erlbaum Associates, Inc., 1995. - 1 : Vol. 7. - pp. 57-78. - ISSN: 1044-7318.

Mackinlay Jock Automating the design of graphical presentations of relational information [Journal]. - New York, NY, USA : ACM Press, 1986. - 2 : Vol. 5. - ISSN:0730-0301.

Maletic Jonathan I., Marcus Andrian and Collard Michael L. A task oriented view of software visualization [Conference] // VISSOFT Proceedings of the 1st International Workshop on Visualizing Software for Understanding and Analysis. - Washington, DC, USA : IEEE Computer Society, 2002. - p. 32. - ISBN: 0-7695-1662-9.

Marcus Andrian, Feng Louis and Maletic Jonathan I. 3D Representations for Software Visualization [Conference] // 2003 ACM symposium on Software visualization. - San Diego, California : ACM Press, 2003. - pp. 27-ff. - ISBN:1-58113-642-0.

Miller George A. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information [Journal] // Psychological Review. - 1956. - Vol. 63. - pp. 81-97.

Nielsen Jacob Usability Engineering [Book]. - San Francisco : Morgan Kaufmann, 1994. - 1st Edition. - ISBN: 0125184069.

Nielsen Jakob Why You Only Need to Test With 5 Users [Online] // useit.com. - March 19, 2000. - May 29, 2007. - <http://www.useit.com/alertbox/20000319.html>.

Owen Scott G. Definitions and Rationale for Visualization [Online] // HyperVis. - SigGraph, February 11, 1999. - April 7, 2007. - <http://www.siggraph.org/education/materials/HyperVis/visgoals/visgoal2.htm>.

Perlman Gary Web-Based User Interface Evaluation with Questionnaires [Online] // acm.org. - ACM, January 2000. - June 12, 2007. - <http://www.acm.org/~perlman/question.html>.

Platt Michael Microsoft Architecture Overview. - [s.l.] : Microsoft Corporation, July 2002.

Price Blaine A., Baecker Ronald M. and Small Ian S. A Principled Taxonomy of Software Visualization [Journal] // Journal of Visual Languages and Computing. - New York, NY, USA : Academic Press, 1993. - 1 : Vol. 4. - pp. 211-266. - ISSN: 0-262-19395-7.

Pulkkinen Mirja Systemic Management of Architectural Decisions in Enterprise Architecture Planning. Four Dimensions and Three Abstraction Levels [Conference] // 39th Hawaii International Conference on System Sciences. - [s.l.] : System Sciences, 2006. - Vol. 8. - pp. 179a - 179a. - ISSN: 1530-1605.

Roman Gruia-Catalin and Cox Kenneth C. A Taxonomy of Program Visualization Systems [Journal] // IEEE Computer. - 1993. - 12 : Vol. 26. - pp. 11-24.

Schekkerman J. Enterprise Architecture Tool Selection Guide v. 4.0 [Report]. - [s.l.] : Institute for Enterprise Architecture Developments, 2007.

Schönhage Bastiaan, Ballegooij Alex van and Elliëns Anton 3D gadgets for business process visualization—a case study [Conference] // Virtual Reality Modeling Language Symposium. - Monterey, California, USA : ACM Press, 2000. - pp. 131-138. - ISBN: 1-58113-211-5.

Sherman Lee Enterprise Modeling Tools [Online] // Enterprise Architect. - Fawcette Technical Publications, 2004. - July 27, 2007. - <http://www.ftponline.com/ea/magazine/fall/columns/architectstoolbox/lsherman/>.

Shneiderman Ben Designing the user interface: strategies for effective human-computer interaction [Book]. - Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1986. - ISBN: 0-201-16505-8.

Shneiderman Ben The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations [Article] // IEEE Visual Languages. - Maryland, U.S.A : [s.n.], 1996.

Stasko John T. [et al.] Software Visualization: Programming as a Multimedia Experience [Book]. - [s.l.] : The MIT Press, 1998. - ISBN: 0-262-19395-7.

Steen M. W. A. [et al.] Supporting Viewpoint-Oriented Enterprise Architecture [Conference] // EDOC 2004. - Monterey, California : [s.n.], 2004. - pp. 201-211. - 9249_1099998796.

Tegarden David P. Business information visualization [Article] // Communications of the AIS. - Atlanta, GA, USA : Association for Information Systems, 1999. - 1es : Vol. 1. - 4.

The Open Group Developing Architecture Views [Online] // TOGAF 8.1.1 Online. - The Open Group, 2006. - April 13, 2007. - <http://www.opengroup.org/architecture/togaf8-doc/arch/chap31.html>.

Tory Melanie and Möller Torsten Rethinking Visualization: A High-Level Taxonomy [Article] // Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04). - Washington, DC, USA : InfoVis, 2004. - Vol. 00. - 10.1109/INFVIS.2004.59.

Tufte Edward R. Envisioning Information [Book]. - Cheshire, Conn : Graphics Press, 1990. - ISBN: 0961392118 .

Tufte Edward R. The Visual Display of Quantitative Information [Book]. - Cheshire, Conn : Graphics Press, 1983. - ISBN: 096139210X.

Vallecillo Antonio RM-ODP: The ISO Reference Model for Open Distributed Processing [Journal] // Software Engineering.. - March 2001. - pp. 69-99. - ISBN: 84-931933-2-1.

Vessey I. Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature [Journal] // Decision Sciences. - 1991. - 2 : Vol. 22. - pp. 219-241.

Zachman John A. A Framework for Information Systems Architecture [Journal] // IBM Systems Journal. - Riverton, NJ, USA : IBM Corp., 1987. - 3 : Vol. 26. - pp. 276 - 292. - ISSN: 0018-8670 .

Glossary

3-tier architecture – client-server architecture in which the *user interface*, functional process logic (business rules) and data storage/access are maintained as independent modules, most often on separate platforms.

Abstraction (, visual) – suppression of irrelevant detail during visual representations.

Analogy – generalization of *visual metaphor*; comparison between two different things, in order to highlight some form of similarity.

API – (Application Programming Interface) source code interface that a computer system or program library provides to support requests for services to be made of it by a computer program.

Applications architecture – (or systems architecture) contains the systems dimension, the information systems of the enterprise.

BPMN – (Business Process Modeling Notation) standardized graphical notation for drawing business processes in a workflow.

Business architecture – depicts the business dimension (business processes, service structures, organization of activities) .

Business metric – high-level existing management performance indicator that champions pay special importance (e.g. profitability percentage, customer satisfaction, inventory levels, time to market, yield etc.).

CI – (Configuration Item) IT component and the services this provides. CI's can include PC hardware, software, network components, servers, central processors, documentation, procedures, services, IT users, IT staff, business units, etc.

CMDB – (Configuration Management Database) logical database which keeps track of CI's, their versions, status, and the relationships between them.

Cognitive fit theory – problem-solving theory, which states that a solution to a problem is an outcome of the relationship between the *problem representation* and problem solving *tasks* (Vessey, 1991).

Communication – the process of exchanging information and ideas.

Computer graphics – (CG) field of visual computing, where one utilizes computers both to generate visual images synthetically and to integrate or alter visual and spatial information sampled from the real world.

Conceptual model – high-level description of the structure and functioning of a system, made by the designers with the objective of forming *mental models* on the system users.

Concern, stakeholder – key interests that are crucially important to the *stakeholders* in the system, and determine its acceptability.

Data – a representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or by automated means.

Data, abstract – data associated with not tangible things.

Data, concrete – data associated with tangible and/or real things.

Design pattern – in software engineering, general repeatable solution to a commonly occurring problem in software design.

Effectiveness, metaphor – metaphor metric that represents the success of the univocal mapping between data values and visual parameters.

Enterprise – organization created for business ventures which comprises a complex interplay of people, processes, and technologies in achieving business objectives.

Enterprise architect – responsible for EAs, who maps, defines, and standardizes technology, data, and business processes to make IT enables business strategy today and tomorrow.

Enterprise architecture – (EA) blueprint of the organization, which serves as a starting point for analysis, design, and decision-making.

Expressiveness, metaphor – capability of the metaphor to visually represent all the *information* we desire to visualize.

Framework, viewpoint – tool and standard for defining *viewpoints* which comprehends a formal a systematic way to analyze architectures.

Gestalt principles – laws that have guided the study of how people perceive visual components as organized patterns or wholes, instead of many different parts.

Heuristic, usability – general principle for user interface design.

Imagery – use of vivid language to create mental images of objects, actions, or ideas.

Information – result of processing, manipulating, and organizing data in a way that adds to the knowledge of the person receiving it.

Information architecture – captures the information dimension of EA; high level structures of business information and, at a more detailed level, the data architecture.

Information cost – the effort of finding and accessing *information*.

Information format – information structure.

Information graphic – (or infographics) visual representations of information, data or knowledge.

Information retrieval – process of recovering stored *information*.

Information visualization – the formation of mental visual images; the act or process of interpreting in visual terms or of putting into visual form.

Input channel, human – human sense used for information perceiving.

ITIL – (Information Technology Infrastructure Library) framework of best practice approaches intended to facilitate the delivery of high quality information technology (IT) services.

Landscape map – graphical resource that presents architectural elements in the form of an easy to understand 2D “map”.

Medium – where information is represented (slides, MS PowerPoint, sheets, etc.)

Mental model – mental representation that people use to understand a system. Its quality depends on the subjacent *conceptual model*.

Metaphor, visual – representation of a new system by means of visual attributes corresponding to a different system, familiar to the user, which behaves in a similar way.

Modeling – the act of formally representing a system or a sub-system.

Modeling language, graphical – using diagram techniques with named symbols that represent concepts and lines that connect the symbols and that represent relationships and various other graphical annotation to represent constraints.

MVC – (Model-View-Controller) software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modifications to the view component can be made with minimal impact to the data model component.

Navigation – process by which a user explores all the levels of interactivity, moving forward, backward, and through the content and interface screens,

Pattern, software – general repeatable solution to a commonly occurring problem in software design (e.g. proxy, client-server, OSI reference model, 3-tier ...).

Process, business – group of business activities undertaken by an organization in pursuit of a common goal. Typical business processes include receiving orders, marketing services, selling products, etc.

Prototype – a less formal experimental and experiential development process of a proposed application for the purpose of demonstrating some or all of its functional capabilities.

Process illustration – EA visualization technique used to represent *business processes* which abstract details regarding applications and technology involved and speaks stakeholders language.

Prototyping – the process of quickly putting together a working model (a prototype) in order to test various aspects of a design, illustrate ideas or features, and gather early user feedback.

Questionnaire – method for the elicitation, recording, and collecting of information.

Requirement – in engineering, a requirement is a singular documented need of what a particular product or service should be or do.

Stakeholder – an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.

Standard – a published specification that establishes a common language, and contains a technical specification or other precise criteria and is designed to be used consistently, as a rule, a guideline, or a definition.

Taxonomy, visualization – formal method to categorize and classify a visualization system. It may also introduce a formal language to ease communication, thus helping to solve related problems.

Technology architecture – covers the technologies and technological structures used to build the information and communication systems in the enterprise.

Translation rule – the definition of how to transform an artifact of a architecture model into a visual artifact.

UML – (Universal Modeling Language) industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

Usability – general term that encompasses everything having to do with "ease of use" (of an *interface* of a system)

Usability testing – process of measuring the ease which *users* can complete common *tasks* on an *interface* of a system

User – persons who makes use of an IT service, application, information system, etc.

User interface – (UI) the aggregate of means by which users interact with a particular machine, device, computer program, or other complex tool (the system).

User interface, graphical – (GUI) type of *user interface* that accepts input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor.

View – representation of a whole system from the perspective of a set of concerns, in terms meaningful to *stakeholders*.

Viewpoint – specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views.

Viewpoint framework – tool and standard for defining viewpoints. It comprehends a formal a systematic way to analyze architectures.

Visual imagery – the phenomenon of seeing in the absence of a visual stimulus.

Visual thinking (or picture thinking, or visual/spatial learning) – the common phenomenon of thinking through visual processing.

Visualization – systematic, rule-based, external, permanent, and graphic representation that depicts information in a way that is conducive to acquiring insights, developing an elaborate understanding, or communicating experiences.

Visualization, EA – the process that delivers understandable visualizations of EA models to stakeholders.

Appendixes

Appendix A – IEEE 1471-2000

We present below a summary of the IEEE 1471-2000 standard “Recommended Practice for Architectural Description of Software-Intensive Systems” (IEEE-SA, 2000):

Abstract: This recommended practice addresses the activities of the creation, analysis, and sustainment of architectures of software-intensive systems, and the recording of such architectures in terms of architectural descriptions. A conceptual framework for architectural description is established. The content of an architectural description is defined. Annexes provide the rationale for key concepts and terminology, the relationships to other standards, and examples of usage.

Keywords: architectural description, architecture, software-intensive system, stakeholder concerns, system stakeholder, view, viewpoint

[...]

3. Definitions

For the purposes of this standard, the following terms and definitions apply [...]

- 3.1 *acquirer*: An organization that procures a system, software product, or software service from a supplier. (The acquirer could be a buyer, customer, owner, user, or purchaser.)
- 3.2 *architect*: The person, team, or organization responsible for systems architecture.
- 3.3 *architecting*: The activities of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture.
- 3.4 *architectural description (AD)*: A collection of products to document an architecture.
- 3.5 *architecture*: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
- 3.6 *life cycle model*: A framework containing the processes, activities, and tasks involved in the development, operation, and maintenance of a software product, which spans the life of the system from the definition of its requirements to the termination of its use.
- 3.7 *system*: A collection of components organized to accomplish a specific function or set of functions.
- 3.8 *system stakeholder*: An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.
- 3.9 *view*: A representation of a whole system from the perspective of a related set of concerns.

[...]

4.4 Uses of architectural descriptions

Architectural descriptions are applicable to a variety of uses, by a variety of stakeholders, throughout the life cycle.

These uses include, but are not limited to the following:

- a) Analysis of alternative architectures
- b) Business planning for transition from a legacy architecture to a new architecture
- c) Communications among organizations involved in the development, production, fielding, operation, and maintenance of a system
- d) Communications between acquirers and developers as a part of contract negotiations
- e) Criteria for certifying conformance of implementations to the architecture
- f) Development and maintenance documentation, including material for reuse repositories and training materials
- g) Input to subsequent system design and development activities
- h) Input to system generation and analysis tools
- i) Operational and infrastructure support; configuration management and repair; redesign and maintenance of systems, subsystems, and components
- j) Planning and budget support
- k) Preparation of acquisition documents (e.g., requests for proposal and statements of work)
- l) Review, analysis, and evaluation of the system across the life cycle
- m) Specification for a group of systems sharing a common set of features, (e.g., product lines)

[...]

5.2 Identification of stakeholders and concerns

An AD shall identify the stakeholders considered by the architect in formulating the architectural concept for the system.

At a minimum, the stakeholders identified shall include the following:

- a) Users of the system
- b) Acquirers of the system
- c) Developers of the system
- d) Maintainers of the system

[...]

An AD shall identify the concerns considered by the architect in formulating the architectural concept for the system.

At a minimum, the concerns identified should include the following:

- The purpose or missions of the system
- The appropriateness of the system for use in fulfilling its missions
- The feasibility of constructing the system
- The risks of system development and operation to users, acquirers, and developers of the system
- Maintainability, deployability, and evolvability of the system

[...]

5.3 Selection of architectural viewpoints

An AD shall identify the viewpoints selected for use therein.

Each viewpoint shall be specified by

- a) A viewpoint name,
- b) The stakeholders to be addressed by the viewpoint,
- c) The concerns to be addressed by the viewpoint,
- d) The language, modeling techniques, or analytical methods to be used in constructing a view based upon the viewpoint,
- e) The source, for a library viewpoint (the source could include author, date, or reference to other documents, as determined by the using organization).

[...]

The following diagram relates the concepts of IEEE 1471-2000 standard:

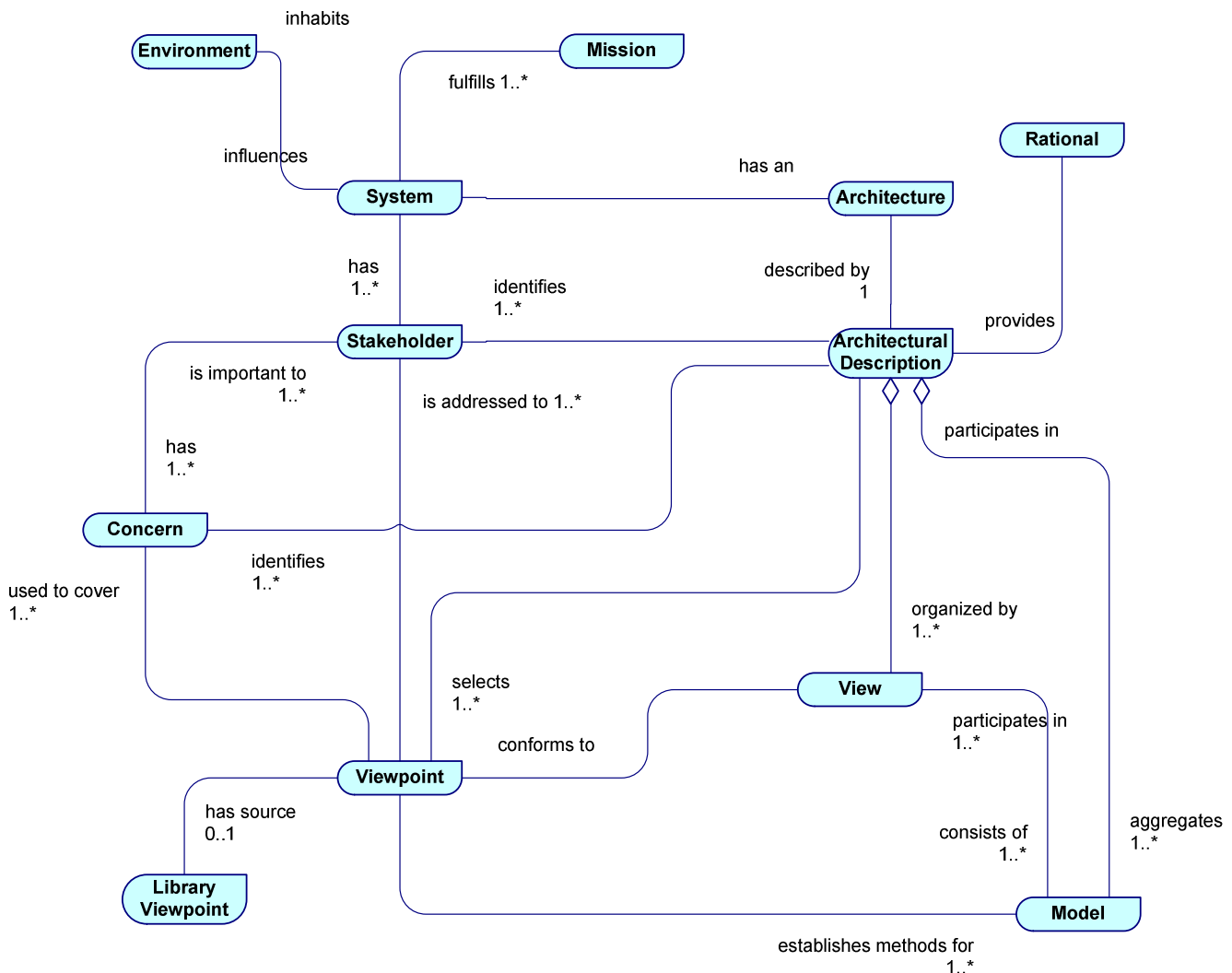
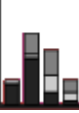





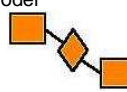
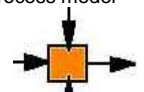
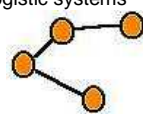
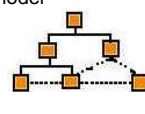
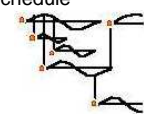
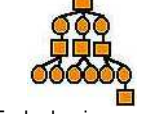
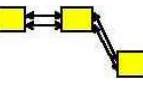
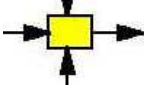
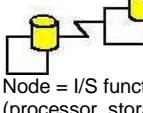
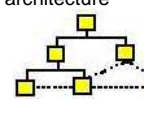
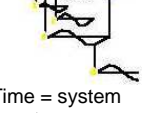
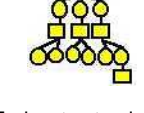
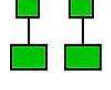
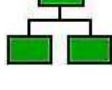
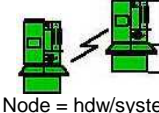
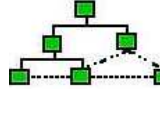
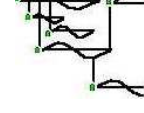
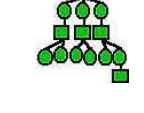








Figure 55 – Conceptual model of architectural description of IEEE 1471-2000 (IEEE-SA, 2000)

Appendix B– Zachman Framework

Table 18 – Zachman framework (Zachman, 1987)

	What Data	How Function	Where Network	Who People	When Time	Why Motivation
Scope {contextual} Planner	Things important to the business  Entity = class of business thing	Processes the business performs  Process = class of business process	Locations which the business operates  Node = major business locations	Organizations important to the business  People = major organizational unit	Events/cycles significant to the business  Time = major business event/cycle	Business goals/strategies  End/means = major business goal/strategy
Business model {conceptual} Owner	e.g.: semantic model  Entity = business entity, Relationship = business relationship	e.g.: Business process model  Process = business process, I/O = business resources	e.g.: Business logistic systems  Node = business location, Link = business linkage	e.g.: Workflow model  People = organization unit, Work = work product	e.g.: Master schedule  Time = Business event, Cycle = business cycle	e.g. Business plan  End = business objective, Means = Business strategy
System model {logical} Designer	e.g.: Logical data model  Entity = data entity, Relationship = data relationship	e.g.: Application architecture  Process = application function I/O = user views	e.g.: Distributed systems architecture  Node = I/S function (processor, storage, etc.), Link = line characteristics	e.g.: Human interface architecture  People = role, Work = deliverable	e.g.: Processing structure  Time = system event, Cycle = processing cycle	e.g.: Business rule model  End = structural assertion, Means = action
Technology model {logical } Builder	e.g.: Physical data layer  Entity = segment/table/etc., Relationship = Pointer/Key/etc.	e.g.: System design  Process = computer function, I/O = data elements/sets	e.g.: Technology architecture  Node = h/w/system software, Link = line specifications	e.g.: Presentation architecture  People = user, Work = screen formats	e.g.: Control structure  Time = execute, Cycle = component cycle	e.g.: Rule design  End = condition, Means = action
Detailed representation {out-of-context} Subcontractor	e.g.: Data definition  Entity = field, Relationship = address	e.g.: Program  Process = language statement, I/O = control block	e.g.: Network architecture  Node = address, Link = protocol	e.g. Security architecture  People = identity, Work = Job	e.g.: Timing Definition  Time = interrupt, Cycle = machine cycle	e.g.: Rule specification  End = sub-condition, Means = step
Functioning enterprise	e.g.: Data	e.g.: Function	e.g.: Network	e.g.: Organization	e.g.: Schedule	e.g.: Strategy

Appendix C – TOGAF Architecture Views

Table 19 – TOGAF views (The Open Group, 2006)

To address the concerns of the following stakeholders...

Users, Planners, Business Management	Database Designers and Administrators, System Engineers	System and Software Engineers	Acquirers, Operators, Administrators, Managers
---	--	--------------------------------------	---

... the following views may be developed:

Business Architecture Views	Data Architecture Views	Applications Architecture Views	Technology Architecture Views
Business function view	Data entity view	Software engineering view	Networked computing/hardware view
Business services view			
Business process view			
Business information view			
Business locations view			
Business logistics view	Data flow (organizational data use)	Applications interoperability view	Communications engineering view
People view (organization chart)			
Workflow view			
Usability view			
Business strategy and goals view			
Business objectives view	Logical data view	Software distribution view	Processing view
Business rules view			
Business events view			
Business performance view			
Systems engineering view (e)			
Enterprise security view			
Enterprise manageability view			
Enterprise quality of service view			
Enterprise mobility view			

Appendix D – Mapping between Zachman and TOGAF

Table 20 – Mapping between Zachman and TOGAF frameworks (The Open Group, 2006)

	<u>What</u> Data	<u>How</u> Function	<u>Where</u> Network	<u>Who</u> People	<u>When</u> Time	<u>Why</u> Motivation
Scope (contextual) Planner	Data entity view (class model) (b)	Business function view (a)	Business locations view (a)	People view (org. chart) (a)	Business events view (a)	Business strategy and goals view (a)
		Business services view (a)	Enterprise mobility view (i)		Enterprise quality of service view (h)	
Business model (conceptual) Owner	Data flow view (organization data use) (b)	Business services view (a)	Business logistics view (business-to- location mapping)	Workflow view (a)	Business performance view (master schedule) (a)	Business objectives view (SMART form business scenario) (a)
		Business process view (a)	Enterprise mobility view (i)		Enterprise quality of service view (h)	
System model (logical) Designer	System engineering view (e)	System engineering view (e)	System engineering view (e)	Usability view (a)	System engineering view (e)	Business rules view (a)
		System engineering view (c)	Standards view (d)		Processing view (d)	
	Logical data view (b)	Application-to- application communication view (c)		Enterprise mobility view (i)	Standards view (d)	Standards view (d)
		Standards view (d)	Standards view (d)			Enterprise quality of service view (h)
Technology model (logical) Builder	Physical data view [out of TOGAF scope]	Software distribution view (c)	Networked computing / hardware view (d) Communications engineering view (d)	Usability view (a)	Control structure [out of TOGAF scope]	Business logic (rules) design [out of TOGAF scope]
Detailed representation (out-of-context) Subcontractor	Data definitions [out of TOGAF scope]	Application program code [out of TOGAF scope]	[out of TOGAF scope]	[out of TOGAF scope]	Timing definitions [out of TOGAF scope]	Application program (rules specification) [out of TOGAF scope]
Functioning enterprise	Enterprise security view (f)	Enterprise security view (f)	Enterprise security view (f)	Enterprise security view (f)	Enterprise security view (f)	Enterprise security view (f)
	Enterprise mobility view (i)	Enterprise mobility view (i)	Enterprise mobility view (i)	Enterprise mobility view (i)	Enterprise mobility view (i)	Enterprise mobility view (i)
	Enterprise quality of service view (h)	Enterprise quality of service view (h)	Enterprise quality of service view (h)	Enterprise quality of service view (h)	Enterprise quality of service view (h)	Enterprise quality of service view (h)
	Enterprise manageability view (g)	Enterprise manageability view (g)	Enterprise manageability view (g)		Enterprise manageability view (g)	

Mapping with TOGAF architecture views (Appendix C):

- a) Business architecture views
- b) Data architecture views
- c) Applications architecture views
- d) Business architecture views
- e) Systems engineering views
- f) Enterprise security views
- g) Enterprise manageability views
- h) Enterprise quality of service views
- i) Enterprise mobility views

Appendix E – EA Tools Listing

Table 21 – EA tools list (Schekkerman, 2007)

Supplier	Tool	1	2	3	4	5	6	7	Supported frameworks
Aam tech	SAMU					■			[Not specified]
AcceptSoftware	Accept 360		■	■	■	■			[Not specified]
Adaptive	Adaptive EA Manager, IT Portfolio Manager, Metadata Manager, Project Portfolio Manager		■	■	■	■	■	■	Adaptive, Zachman, FEAF, ArchiMate, CWM
Agilense	EA Webmodeler	■		■	■	■	■		Agilense, Synthesis, Zachman, TOGAF, DoDAF, FEAF/TEAF, FE
Altova	Altova Enterprise Suite						■	■	UMODEL, XML Suite
Alfabet	Planning IT	■	■	■	■	■	■	■	SITM framework
ASG Software Solutions	ASG-Rochade	■				■	■	■	[Not specified]
BEA AquaLogic	Enterprise Repository						■	■	FEAF
BizzDesign	BiZZdesign Architect, BiZZdesigner, Risk Manager	■	■			■			ArchiMate
Casewise	Corporate Modeler Enterprise Edition	■	■	■	■	■	■		Casewise Framework, Zachman, FEAF, TEAF, eTOM, DoDAF, etc.
CACI International	SimProcess					■	■		Zachman
Enterprise Elements	ElementsRepository		■	■	■	■			CPIC / DoDAF
Forsight	Modelling & Validation Toolset						■	■	DoDAF
Framework Software, Inc	Structure					■			FEAF, DoDAF, Zachman
Future Tech Systems, Inc	ENVISION® VIP		■	■	■	■	■		[Not specified]
GoAgile	GoAgile MAP Product Suite					■			[Not specified]
IBM	IBM Rational Software Architect						■	■	[Not specified]
IDS Scheer	ARIS Process Platform	■	■	■	■	■	■		ARIS Framework, ArchiMate
TeleLogic (I-Logix)	Rhapsody						■	■	DoDAF
Intelligile Corporation	MAP Suite					■	■	■	Zachman, Togaf & 1, FEAF, DoDAF
Knotion Consulting	SYNAP-C SOLUTION					■	■	■	[Not specified]
LogicLibrary	Logidex						■	■	[Not specified]
Méga International	Méga (Process, Architect, Designer)	■	■	■	■	■	■	■	Zachman
NetViz	NetViz Suite						■	■	[Not specified]
Palisade	Risk & Decision Analysis	■	■						[Not specified]
Proforma	Provision Modeling Suite		■	■	■	■	■	■	Zachman, C4ISR
Select Business Solutions	Select Component Architect					■	■	■	Zachman
Simon Labs	SimonTool						■	■	Simon EA, Zachman
Sparx Systems	Enterprise Architect						■	■	[Not specified]
TeleLogic	System Architect Family	■	■	■	■	■	■	■	Zachman, TOGAF 8, DoDAF, MoDAF
Troux	Metis Product Family	■	■	■	■	■	■	■	Zachman, TOGAF, DoDAF, FEAF / TEAF, ArchiMate
US Government	FEAMS					■	■		FEAF, TEAF, C4ISR
Visible	Advantage						■	■	[Not specified]






¹ Governance, risk, compliancy
² Enterprise IT/portfolio management
³ Program management
⁴ Business/IT strategy
⁵ Enterprise architecture
⁶ Service oriented architecture
⁷ Software engineering

Appendix F – Periodic Table of Visualization Methods

<p>Data visualization Visual representations of quantities data in schematic form (either with or without axes)</p>		<p>Information visualization The use of interactive visual representations of data to amplify cognition: the data is transformed into an image; it is mapped to screen space. Can have interactivity.</p>		<p>Strategy visualization The systematic use of complementary visual representations in the analysis, development, formulation, communication, and implementation of strategies in organizations.</p>		<p>Metaphor visualization Visual metaphors position information graphically to organize / structure information and convey an insight about the represented information through the characteristics of the employed metaphor.</p>		<p>Compound visualization The complementary use of different graphic representation formats in one single schema or frame.</p>						<p>G graphic facilitation</p>			
C continuum																	
Tb table	Ca cartesian coordinates										Me meeting trace	Mm metro map	Tm temple	St story template	Tr tree	Ct cartoon	
Pi pie chart	L line chart										Co communication diagram	Fp flight plan	Cs concept sceleton	Br bridge	Fu funnel	Ri rich picture	
B bar chart	Ac area chart	R radar chart cobweb	Pa parallel coordinates	Hy hyperbolic tree	Cy cycle diagram	T timeline	Ve venn diagram	Mi mindmap	Sq square of oppositions	Cc concentric circles	Ar argument slide	Sw swim lane diagram	Gc gant chart	Pm perspectives diagram	D dilemma diagram	Pr parameter ruler	Kn knowledge map
Hi histogram	Sc scatterplot	Sa sankey diagram	In information lense	E entity relationship diagram	Pt petri net	Fl flow chart	Cl clustering	Lc layer chart	Py minto pyramid technique	Ce cause-effect chains	Tl toulmin map	Dt decision tree	Cp cpm critical path method	Cf concept fan	Co concept map	Ic iceberg	Lm learning map
Tk tukey box plot	Sp spectrogram	Da data map	Tp treemap	Cn cone tree	Sy system dyn./ simulation	Df data flow diagram	Se semantic network	So soft system modeling	Sn synergy map	Fo force field diagram	Ib ibis argumentation map	Pr process event chains	Pe pert chart	Ev evocative knowledge map	V Vee diagram	Hh heaven 'n' hell chart	I infomural

Cy Process visualization

Hy Structure visualization

-  Overview
-  Detail
-  Detail and overview
-  Divergent thinking
-  Convergent thinking

Su supply demand curve	Pe performance charting	St strategy map	Oc organisation chart	Ho house of quality	Fd feedback diagram	Ft failure tree	Mq magic quadrant	Ld life-cycle diagram	Po porter's five forces	S s-cycle	Sm stakeholder map	Is ishikawa diagram	Tc technology roadmap
Ed edgeworth box	Pf portfolio diagram	Sg strategic game board	Mz mintzberg's organigraph	Z zwicky's morphological box	Ad affinity diagram	De decision discovery diagram	Bm bcg matrix	Stc strategy canvas	Vc value chain	Hy hype-cycle	Sr stakeholder rating map	Ta taps	Sd spray diagram

Figure 56 – Periodic table of visualization methods (Lengler, et al., 2007)

(Full version available at http://www.visual-literacy.org/periodic_table/periodic_table.html)

version 1.5

© Ralph Lengler & Martin J. Eppler; www.visual-literacy.org

Appendix G – Nielsen Usability Heuristics

These are ten general principles for user interface design. These are called *heuristics* because they are more in the nature of rules of thumb than specific usability guidelines (Nielsen, 1994):

- H1. Visibility of system status:** the system should always keep users informed about what is going on, through appropriate feedback within reasonable time;
- H2. Match between system and the real world:** the system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order;
- H3. User control and freedom:** users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo;
- H4. Consistency and standards:** users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions;
- H5. Error prevention:** even better than good error messages is a careful design that prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action;
- H6. Recognition rather than recall:** minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate;
- H7. Flexibility and efficiency of use:** accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions;
- H8. Aesthetic and minimalist design:** dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility;
- H9. Help users recognize, diagnose, and recover from errors:** error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution;
- H10. Help and documentation:** even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Appendix H – Shneiderman Rules of Interface Design

Shneiderman proposed this collection of principles that are derived heuristically from experience and applicable in most interactive systems after being properly refined, extended, and interpreted (Shneiderman, 1986).

To improve the usability of an application it is important to have a well designed interface. Shneiderman's "Eight Golden Rules of Interface Design" are a guide to good interaction design:

- 1^o **Strive for consistency:** consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout;
- 2^o **Enable frequent users to use shortcuts:** as the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user;
- 3^o **Offer informative feedback:** for every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial;
- 4^o **Design dialog to yield closure:** sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions;
- 5^o **Offer simple error handling:** as much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error;
- 6^o **Permit easy reversal of actions:** this feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions;
- 7^o **Support internal locus of control:** experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders;
- 8^o **Reduce short-term memory load:** the limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.











Appendix I – CMDB API

Table 22 – CMDB engine API specification

Input ¹	Service	Output	Exceptions ²
→ CI identifier	1) Delete CI <i>Deletes a CI, its attributes and relations.</i>	[none]	CI not found
→ Meta-CI identifier	2) Delete Meta-CI <i>Deletes a Meta-CI, its Meta-Attributes, Meta-Relations, CI's and icon.</i>	[none]	Meta-CI not found
→ Meta-Relation identifier	3) Delete Meta-Relation <i>Deletes a Meta-Relation an its Relations.</i>	[none]	Meta-Relation not found
→ User identifier	4) Delete User <i>Deletes a User from the system</i>	[none]	User not found
→ Meta-CI identifier → Text to search	5) List CI's <i>Gets the list of CI's of a Meta-CI with "text" in its name.</i>	→ List CI's	[none]
→ Is in implementation? (flag) → Is qualifier? (flag)	6) List Meta-CIs <i>Gets the list of Meta-CI's. Respects the input flags.</i>	→ List of Meta-CI's	[none]
→ Meta-CI id. left → Meta-CI id. right → Meta-CI id. left or right → Only direct? (flag)	7) List Meta-Relations <i>Gets the list of Meta-Relations. If a Meta-CI identifier is provided only compliant Meta-Relations are returned..</i>	→ List of Meta-Relations	[none]
→ Meta-CI id. left → Meta-CI id. right → Meta-CI id. left or right → Meta-Relation id.	8) List Relations <i>Gets the list of Relations. Has the capacity to filter them according to the identifiers provided.</i>	→ List of Relations	[none]
→ CI identifier → Version identifier → Fast mode? (flag) → Only direct Relations? → Only inclusion Relations? (flag)	9) CI details <i>Reads the details of the requested CI.</i>	→ CI details (Attributes, Relations and Versions)	CI not found
→ Meta-CI identifier → Fast mode?	10) Meta-CI details <i>Reads the details of a the requested Meta-CI (identified through name or id.).</i>	→ Meta-CI Details (Meta-Attributes and Meta-Relations)	Meta-CI not found
→ Meta-Relation identifier	11) Meta-Relation details <i>Reads the details of a Meta-Relation.</i>	→ Meta-Relation (left, right, and qualifier Meta-CI's)	Meta-Relation not found
→ Relation identifier	12) Relation details <i>Reads the details of a Relation.</i>	→ Relation (left, right, and qualifier CI's)	Relation not found
→ CI identifier → Meta-Attribute id. → Version identifier	13) Get Attribute <i>Given a CI, gets a specified attribute</i>	→ Value of attribute	CI not found Attribute not found
→ CI identifier → Meta-Attribute id. → Version identifier → Value	14) Update Attribute <i>Given a CI, rewrites a specified Meta-Attribute with the new value.</i>	[none]	CI not found Attribute not found

¹ Mandatory parameters are in bold

² Does not include database or input errors exceptions

<ul style="list-style-type: none"> ➔ CI identifier ➔ Name ➔ Description ➔ Meta-CI type 	<p> 15) Create or update CI</p> <hr/> <p><i>If a CI identifier is specified, updates that CI, otherwise creates a new one.</i></p>	<ul style="list-style-type: none"> ➔ CI identifier (of created or updated CI) 	<ul style="list-style-type: none">  CI not found
<ul style="list-style-type: none"> ➔ Meta-CI identifier ➔ Name ➔ Description ➔ Data sources 	<p> 16) Create or update Meta-CI</p> <hr/> <p><i>If a Meta-CI identifier is specified, updates that Meta-CI, otherwise creates a new one.</i></p>	<ul style="list-style-type: none"> ➔ Meta-CI identifier (of created or updated CI) 	<ul style="list-style-type: none">  Meta-CI not found
<ul style="list-style-type: none"> ➔ Meta-Relation identifier ➔ Left Meta-CI id. ➔ Right Meta-CI id. ➔ Qualifier Meta-CI id. ➔ Description 	<p> 17) Create or update Meta-Relation</p> <hr/> <p><i>If a Meta-Relation identifier is specified, updates that Meta-Relation, otherwise creates a new one.</i></p>	<ul style="list-style-type: none"> ➔ Meta-Relation identifier (of created or updated Meta-Relation) 	<ul style="list-style-type: none">  Meta-Relation not found
<ul style="list-style-type: none"> ➔ Relation identifier ➔ Meta-Relation id. ➔ Left CI identifier ➔ Right CI identifier ➔ Qualifier CI ➔ Description 	<p> 18) Create or update Relation</p> <hr/> <p><i>If a Relation identifier is specified, updates that Relation, otherwise creates a new one.</i></p>	<ul style="list-style-type: none"> ➔ Relation identifier (of created or updated Relation) 	<ul style="list-style-type: none">  Meta-Relation not found  Relation not found  Unexpected CI's