Consistent Enterprise Software System Architecture for the CIO - A Utility-Cost Based Approach -

Mathias Ekstedt, Pontus Johnson, Åsa Lindström, Magnus Gammelgård, Erik Johansson, Leonel Plazaola, Enrique Silva, Joakim Lilieskold
Dept. of Industrial Information and Control Systems
KTH, Royal Institute of Technology
SE - 100 44 Stockholm, SWEDEN
{mathiase, pontusj, asa, magnus, erikj, leonelp, enriques, joakiml}@ics.kth.se

Abstract

Previously, business operations of most large companies were supported by a number of isolated software systems performing diverse specific tasks, from real-time process control to administrative functions. In order to better achieve business goals, these systems have in recent years been extended, and more importantly, integrated into a company-wide system in its own right, the enterprise software system. Due to its history, this system is composed of a considerable number of heterogeneous and poorly understood components interacting by means of equally diverse and confusing connectors. To enable informed decision-making, the Chief Information Officer (CIO), responsible for the overall evolution of the company's enterprise software system, requires management tools.

This paper proposes Enterprise Software System Architecture (ESSA) as a foundation for an approach for managing the company's software system portfolio. In order to manage the overwhelming information amounts associated with the enterprise software system, this approach is based on two concepts. Firstly, the approach explicitly relates the utility of knowledge to the cost of its acquisition. The utility of knowledge is derived from the increased value of better-informed decision-making. The cost of knowledge acquisition is primarily related to the resources spent on information searching. Secondly, the approach focuses on ensuring the consistency of the architectural model.

1. Introduction

Today, most of the larger companies within various industrial domains such as power, telecommunications, pulp and paper, finance, etc., possess truly complex enterprise software systems; in large organizations several hundred software systems may be employed. The size of each single system may vary extensively from company-wide enterprise resource planning systems to smaller custom-made niche products. Furthermore, the interconnections among the systems are numerous and heterogeneous [1][2][3]. An organizational role called the Chief Information Officer (CIO) has been proposed for the management of the enterprise software system [4][5][6][7][8]. This organizational role typically consists of a small group or an individual close to the senior management of the company [4][8][9][10]. Given the large quantity of information available about the enterprise software system, the CIO faces a significant problem. An approach that has been proposed to overcome these overwhelming amounts of information is Enterprise Software System Architecture (ESSA). This architecture is devised to reflect the technical aspects of the systems as well as the business context that they are intended to support.

1.1 Related works

Several frameworks have been proposed, that address the ESSA, such as TOGAF [11][12], C4ISR [11][13] and FEA [14]. Also, the CIO typically employs a number of different models for enterprise software system management, gathered from different disciplines such as software engineering, information systems, systems engineering, and business analysis [15][16][17][18][19][20][21]. Although many of these frameworks and models are highly useful, this paper addresses two pertinent remaining problems. Firstly, these frameworks and models are oftentimes fairly ambitious in their approach. The resources needed for searching and gathering the information required for using them are quite extensive [11]. This means that the cost for employing the models and tools can become very high, which in turn easily leads to an unplanned and incomplete model of the enterprise software system situation. Secondly, ESSA models are typically managed in terms of different viewpoints[1]. However, in order to depict the whole ESSA model, several viewpoints have to be employed. Unfortunately, many of these do not have a defined relation to each other, leading to potentially

[1] Viewpoints is the definition of the language for describing views[22].

inconsistent analyses. For instance, there may be inconsistencies between a viewpoint of how some software systems support the business processes and some other viewpoints of interactions between software systems.

This paper proposes a utility-cost based approach, founded on a consistent ESSA meta-model[2], for answering the questions of the CIO. The utility-cost based approach is addressing the first of the two problems discussed above. The focus on consistency is aiming at tackling the second problem. The proposed approach is based on relating the ESSA meta-model with the questions and answers of the CIO. In this sense, the ESSA model can be used as basis for providing answers to questions filtered by a utility-cost analysis.

1.2 Outline

The main proposal, the utility-cost based approach, founded on a consistent ESSA meta-model, for answering the questions of the CIO, is elaborated in Section 2. Due to the limited textual space, several issues that would benefit from deeper considerations can only be superficially considered. Sections 3 and 4 focus on two of the most central parts of the approach. Section 3 outlines concerns, or questions, relevant to the CIO, and Section 4 presents a first draft of the ESSA meta-model. Finally, the paper is concluded in section 5.

2. An Approach for ESSA Management

CIOs usually have a model of their enterprise software system. However, this model is often fraught with problems. Typically, it consists of a large number of seemingly unrelated pieces of information in the form of natural language documents, structured databases, UML diagrams, source code, etc. It is however, as proposed in this paper, more beneficial to employ one consistent model for the most pertinent information.

The proposed ESSA model is a high-level abstraction of the enterprise software system and its context. The ESSA meta-model defines the language in which the ESSA model is expressed (cf. Figure 1). For instance, the meta-model may prescribe that entities of the model be *software components*, *business processes*, *data*, etc. The meta-model further defines relations between the entities in the model; a software system for example may *store* certain data.

This paper argues that one of the main purposes of the model is to answer the questions that the CIO considers relevant. Depending on the questions asked, different parts within the model will be addressed. For instance, certain questions concerning the modifiability of an enterprise software system might depend on the coupling between software systems. The parts of the model that are addressed in this case may thus be the *software components* and their *connectors*.

It is important that the answers to the questions provide utility for the CIO. However, these answers do not come for free. Every answer is associated with a certain cost, foremost in terms of the effort to search for the needed information and introduce it into the ESSA model. This paper argues that it is of significant benefit to explicitly consider the utility of an answer in relation to its cost. To put it differently, some questions are worth answering, while others are not.

The following four subsections elaborate on 1) the utility of answers, 2) the relation between the questions, answers and the ESSA meta-model, 3) the cost of information, and 4) utility-cost analyses of answers. In the fifth subsection, a method for ESSA management is proposed. The sixth subsection briefly discusses the use of viewpoints.

2.1 Utility of Answers

Before any model can be produced, the purpose of developing it must be identified. Since not all stakeholders have the same tasks and assignments, they are interested in different information. The designer of an application program is typically interested in matters such as functional division and data interchange, not data types and programming languages, which instead is the interest of the programmer. Consequently, the CIO must decide on what questions are useful in the context of the enterprise software system. Furthermore, the utility of the answers to these questions must be estimated. This utility comes in the form of better-informed decisions. The outcomes of these decisions may be valued, e.g. in terms of money, which in turn provides a base for assessing the value of the answers.

However, estimating the utility of the specific concerns of all CIOs cannot be done once and for all. Some general methodological support may be provided, but the main assessment must be performed on an enterprise-specific basis.

2.2 Relations Between Questions, Answers and the Meta-Model

Providing an explicit description of how answers can be derived from the meta-model is vital. These relations can be expressed as analysis procedures linked to the meta-model; i.e., given a certain model (an instance of the meta-model), what answers can be elicited by applying the analysis procedures? The enterprise software system discipline is full of such rules and heuristics. These analysis procedures do not have to be at all complicated or convoluted, they can very well be trivial, claiming for instance that two software systems will be hard to integrate if they do not share at least one common

[2] The ESSA meta-model is the definition of the language for describing the ESSA model.

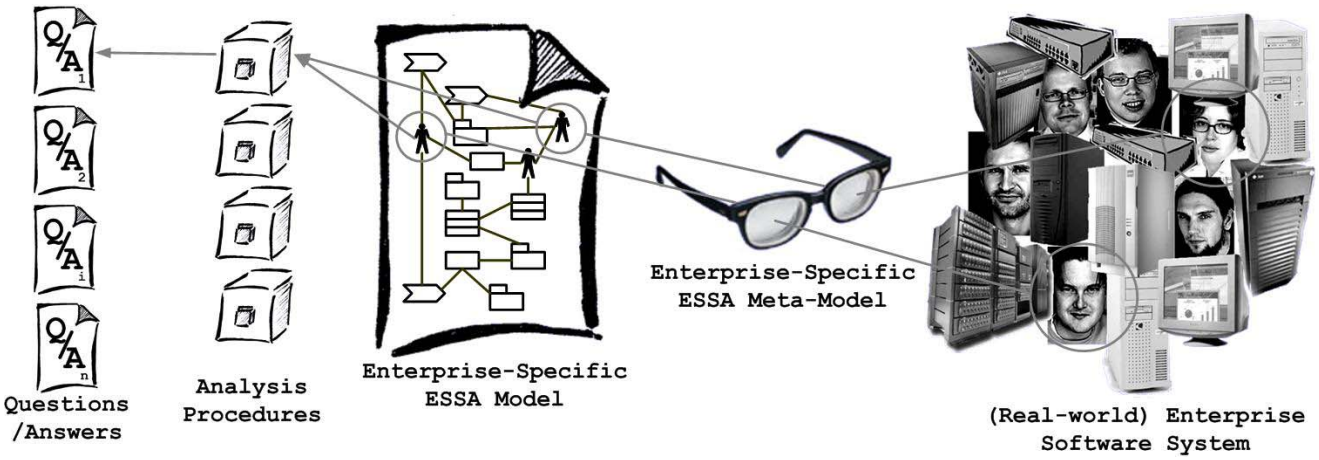

Figure 1. The (real-world) enterprise software system of a specific company is modeled using the language defined by the enterprise-specific ESSA meta-model. The meta-model may thus be viewed as glasses that project certain aspects of the real world into the resulting ESSA model. This enterprise-specific ESSA model is analyzed by different analysis procedures in order to answer the different questions of the CIO.

protocol. In general, analysis procedures will be indicative rather than conclusive on the (high) level of abstraction that the CIO is limited to; e.g., a simple protocol name matching analysis is an indicator for integrability, not a conclusive measure. The ESSA-specific analysis procedures in turn influence the design of the ESSA meta-model. It is important to remember that the meta-model is developed for the purpose of answering some specific questions, and since the analysis procedures indicate what architectural information is required to answer these questions, the procedures influence the design of the ESSA meta-model. However, in practice these two steps are not performed serially; rather, the meta-model and the analysis procedures are elicited in several iterations.

2.3 Cost of Information

There is a cost associated with capturing the necessary information for the ESSA model in order to gain utility from the answers to the CIO's questions. In order to determine whether it is rational to answer specific questions this cost need to be estimated. The cost is mainly related to the resources spent on searching for and acquiring the required information. It is often the case that relevant documentation or knowledgeable personnel are difficult to find. Imagine, for instance, information that has to be reverse engineered from source code; this information is very expensive.

Thus, acquisition of information must be traded off against the utility gained. In order to do this, the search cost must be estimated in advance. This aspect is oftentimes underestimated in many related architectural initiatives in industry, and likewise in literature. There are several ways in which this search cost may be estimated. For instance, a statistical approach, assessing the average search cost of different architectural entities, is feasible. Another option may be the use of a dedicated *architectural information view*, considering the distribution of architectural information across the enterprise [24]. These methods will not be elaborated on in this paper.

2.4 Utility-Cost Analyses of Answers

An important activity is the utility-cost trade-off, relating the utility of answers to their costs. This activity may be employed to select the most relevant set of questions for the CIO. Briefly, every question is related to an answer with an expected utility and cost. The interesting questions are, of course, those associated with a higher utility than cost. However, since the cost of an answer is related to the architectural information required to perform the related analysis procedure, it is quite possible that answering one question will lower the cost of some other answer, since this second answer may be dependent on the same architectural information. In other words, information once collected may be reused. Therefore, costs of answers should not be assessed in isolation and the objective of the utility-cost analysis should be to find the set of questions that are associated with the highest joint net utility[3].

2.5 A Method for ESSA Management

Seen as a process for obtaining architecturally significant information by creating an ESSA model, not all of the above described activities must be done specifically for every company. Rather, there are several generic activities that could be packaged as tools for the CIO.

[3] Net utility being defined as utility minus cost.

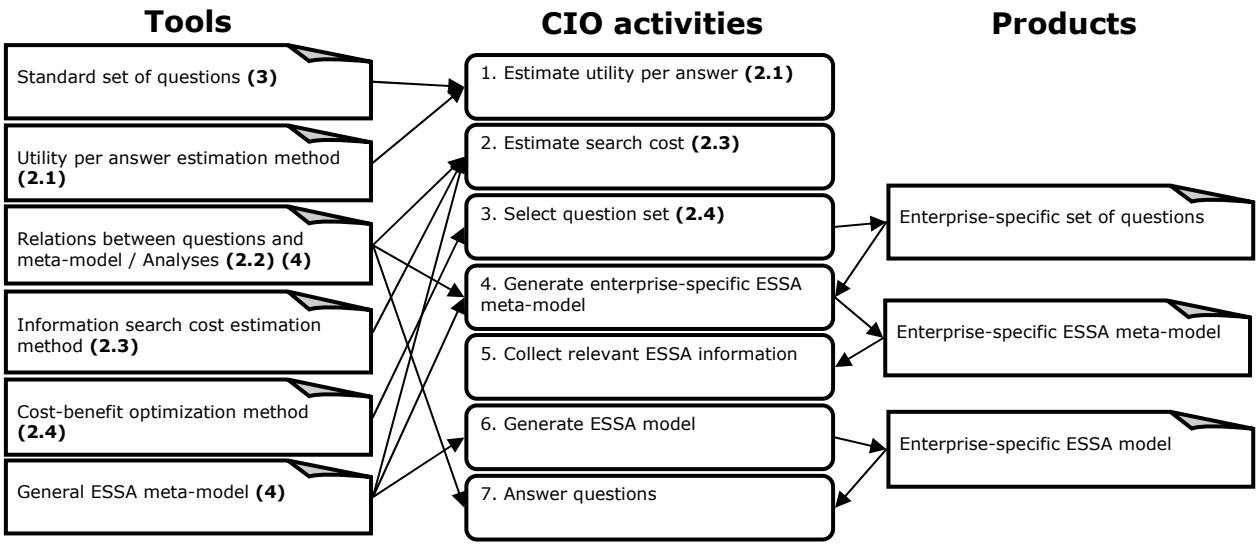

Figure 2. ESSA method divided into general and specific work tasks. The tool list, as well as the first three CIO activities, contains references to the sections in the paper treating aspects of each tool and activity.

In Figure 2, the above discussion is summarized as a method for the CIO. The figure presents the tools, the CIO activities required for every specific ESSA initiative, and the products generated by the CIO activities. The tool list, as well as the first three CIO activities, contain references to those sections and sub-sections in this paper that treat the concepts embodied in each tool. The purpose of the fourth CIO activity is to extract an enterprise-specific ESSA meta-model from the general one, based on the enterprise-specific set of questions generated in the third CIO activity. This is something e.g. TOGAF [11][12] does not address. It rather implicitly states that all information should be gathered. The fifth, sixth and seventh CIO activities are standard architectural activities, collecting information, documenting it in the form of an enterprise-specific ESSA model, and employing the model for answering the questions.

2.6 Viewpoints

Even if the final meta-model contains all the information for answering the questions of the CIO, it is inconvenient to present this information all at once. The employment of viewpoints is a standard strategy for abstraction, used within the discipline of software architecture, information systems, and business modeling [12][13][23][18][21]. Viewpoints are adopted for abstracting concerns expected to be relevant for the CIO in some specific situation. For instance, at a given moment, the main concerns of the CIO may include business processes and software systems, but not interconnections between the systems. These interconnections might, however, very well be relevant at some other point in time. The amount of information that should be included in a view is mainly limited by the complexity as perceived by the observer [25].

A viewpoint can thus answer a limited set of related questions. This means that only a certain part of the ESSA meta-model is depicted at a time. All the relations to the other entities of the model are still there, but they are not depicted in the specific viewpoint. Of course, if different analyses are made based on different views of the ESSA model, these analyses should not diverge. Meta-model consistency, discussed previously, is a prerequisite for sound analysis.

2.7 Summary

In this section, an approach for ESSA management for the CIO has been presented. The primary message of the section is the explicit consideration of the utility and cost of answers to the questions of the CIO. Furthermore, the importance of meta-model consistency has been emphasized. The approach presented in this section will serve as a guideline and the driving force for further development of the discussed ESSA management concerns.

The remainder of the paper will be spent on outlining and describing the scope for two of the general tools, namely, a standard set of questions, or concerns, relevant for the CIO, as well as the ESSA meta-model. Furthermore, the relations between the questions and the meta-model will be highlighted. However, the model and questions presented here will be subject to change, since the development of the ESSA model and its support tools is a considerable task, an iterative process is adopted where new findings and revisions will be published as they become available.

3. Concerns of the CIO

The general concerns of the CIO are of great variety [4][7][8][10][11][26][27]. However, this paper is restricted to those concerns that are related to concrete ESSA issues, consequently excluding duties such as communicating with other functions within the company e.g. CEO and CFO, and detailed management of individual enterprise software system projects. To clarify the relation to the previous section, concerns may be viewed as sets, or groupings, of questions. As previously mentioned, the ESSA model constitutes a base for answering these questions. The main intention of this section is not completeness, but instead to indicate the variety and the scope of the questions within each set. The concerns considered below are *alignment*, *maintenance*, *quality*, *knowledge* and *cost*.

3.1 Alignment

Alignment between the business processes and the software systems of the enterprise is a topic of outmost importance to the CIO [7][8][10][11][26]. The alignment issue deals with how software systems, i.e., the components of the enterprise software system, can efficiently be exploited by the organization and how the two cooperate in a coordinated and harmonious manner [7][28]. Consequently, in order to achieve alignment, the CIO needs to maintain a correct description of what software systems are used and needed by what business processes, and the resources that are consumed and produced within the two. With such knowledge, the CIO can react on the needs of different parts of the organization.

3.2 Maintenance

The CIO has the overall responsibility for ensuring that the systems are functioning; therefore maintenance is a concern for the CIO [4][7][11]. System maintenance as a concern for the CIO is mainly related to keeping track of systems that need to be maintained and with what level of ambition, including lifetime assessments. For example, the CIO needs to know what support is available and if any outsourcing contracts are connected with the different software systems [5][29]. The procurement, development, and integration of new systems as well as the replacement and disposal of old systems can be seen as a part of the maintenance process for the whole enterprise wide software system [7]. In addition, the software systems must be upgraded according to new requirements, and old bugs must be fixed [7]. Furthermore, the CIO is responsible for making the prioritizations among the maintenance needs.

3.3 Quality

Determining quality attributes for different parts of the enterprise software system, such as reliability, modifiability, security, performance, etc, is a means for the CIO to assess the value and appropriateness of the systems [16][30][31][32][33][34][35][36]. Ensuring a certain level of quality of the ESSA is, of course, a basic concern [7][11]. Historically, performance has been a driving quality when developing software systems. As hardware has grown more powerful, and software larger, other attributes such as modifiability and security, have gained in importance. However, all different qualities cannot be achieved at the same time, which leads to an inevitable quality trade-off for the ESSA [33]. Quite common is the balance between performance and modifiability, where for instance, low-level, direct-access integration typically renders good performance and poor modifiability. Estimating and assessing quality attributes is a standard software system architectural activity.

3.4 Knowledge

Another area of interest for the CIO is the total amount of information and knowledge about ESSA-related issues kept within the company [4][27][37][38]. This could be knowledge kept by people as well as information stored in documents and databases. Key persons and central documentation hold information and knowledge about the enterprise software system that the company cannot afford to loose [27][38]. Consequently, the CIO is typically interested in at least a rough picture of how much and what kind of knowledge and information is available. Moreover, it is important to know where this knowledge and information is located within the organization and how it could be retrieved. Furthermore, the CIO cannot trust all information concerning the enterprise software system. Information stored within documents, as well as held among people, is frequently obsolete or otherwise incorrect [24]. As this is influencing the CIO's understanding of the ESSA, information credibility is of significant importance.

3.5 Cost

The main reason for aligning the business processes and software systems, as described previously, is to improve the organizational efficiency, which in turn means increased profit or reduced costs [27]. It is evident that economic aspects are most important in all the above described CIO concerns [7][11]. But, of course, the systems are not only beneficial to the economy of the enterprise. Maintaining, as well as achieving qualities of the enterprise software system is expensive, both in terms of resources and money. Thus, in principle, this parameter could have been distributed all over the other concerns. However, since this aspect is so important, many CIO's manage it as a separate concern [10][11].

4. An Enterprise Software System Architecture Meta-Model

Given the concerns of the CIO outlined in the previous section, this section will elaborate on how a suitable ESSA meta-model can be constructed. Containing a vocabulary of entities and relationships, the meta-model is a language in which the ESSA model can be expressed. Entities and relationships may be specified on different levels of abstraction. For instance, we will subsequently propose *data* as an entity of the ESSA meta-model. But in order to describe the kind of data that is used in an architecture, a data taxonomy[4] is required. In most companies such a taxonomy would contain concepts such as *customer*, *price*, *contract*, etc. However, these data entities could be further decomposed into fairly deep hierarchies.

Turning to the specific context of the enterprise software system, there are a number of fundamental aspects that need to be modeled, such as software systems, business processes, and data within the company. These aspects, presented within the proposed ESSA meta-model as entities, are discussed in the first subsections below. The entities possess certain relationships to each other, which are considered in the penultimate subsection. In the last subsection some viewpoints are exemplified.

4.1 Entities

The entities that have been chosen for this ESSA meta-model are software components, software connectors, users, data, functions, business processes, organizational units, and information carriers. The authors' intention is not to be innovative or controversial in the choice of the entities and the description of them. Below, the entities are first described on a high abstraction level. Then the relations between the entities and the concerns of the CIO are considered.

4.2 Software components

Software components are probably the most central entities in the ESSA meta-model. These entities are what we naturally think of as the actual software and its execution, i.e. software components that execute functions and handle data. Traditional software architecture literature [15][16][39][40][41] speaks of software components as one of two (along with software connectors) fundamental entities that build up the software architecture.

Software components address the CIO's major concerns described earlier in Section 3. For example, this entity is needed in order to support the modeling of system ownership and the relationships between systems and business processes in the context of business alignment [7][11]. The software components entities also need to be modeled when the CIO focuses on the maintenance [11] or any of the quality issues of the ESSA.

4.3 Software connectors

Software connectors are together with software components the major entities in traditional software system architecture and can be seen as links between components [15][16][39][40][41][42]. In this sense, software connectors are abstractions of communication mechanisms. These communication mechanisms may be simple, as protocols or communication standards, or complex ones, incorporating the workings of operating systems or middleware.

Software connectors are related to several of the CIO's concerns [38]. When a new software component is deployed, the CIO needs to know how this software component communicates with other software components. The software connectors are also of interest when considering how the failure of some specific software component affects other parts of the enterprise software system. Furthermore, the software connectors are of interest when analyzing performance issues. This entity is also of interest in the context of modifiability analyses as well as for other quality attributes [7].

4.4 Users

The users in the enterprise software system, as the entity name implies, refer to the people interacting with the software components. Commonly, humans are regarded as the context to software components and connectors [15][16][17]. However, just as software, humans may execute and transfer functions and data between entities. For example, people perform trading services or read recommended production data from one software component and write it in another software component, or tell someone else to do it. These 'manually' performed operations need to be depicted in the model, in order to give the CIO a comprehensive understanding of the enterprise software system.

Human interaction is of importance for several analyses, such as the usage and knowledge of software components [7][11], the overall business performance and efficiency, and what business processes they take part in, which is of value when considering business alignment [21].

4.5 Data

Keeping an up-to-date and consistent picture of the enterprise's total amount of data is a resource-demanding activity for many companies, since the amount of data usually is quite significant. Data is typically stored within

[4] In this paper, *taxonomy* is used to represent a hierarchical classification system, relating entities to their sub-entities.

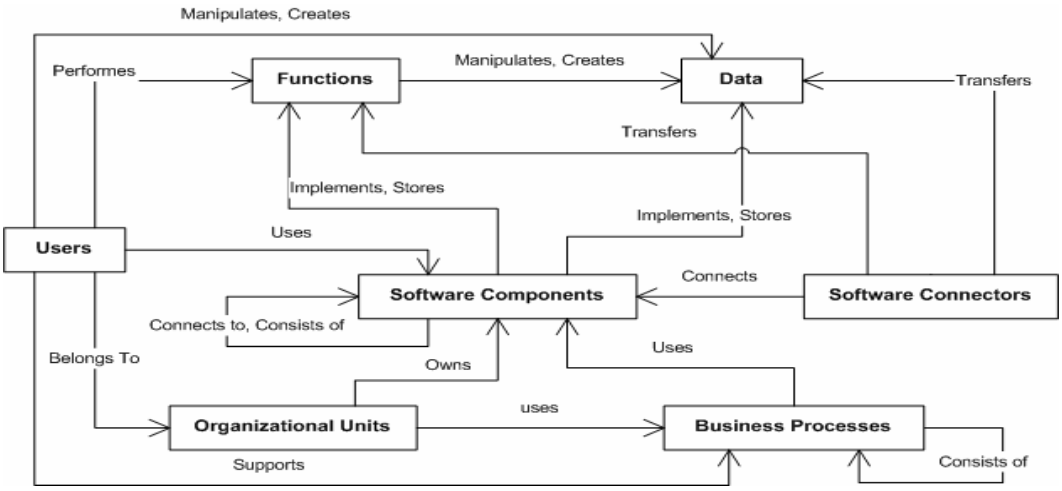

Figure 3. A basic ESSA meta-model. (Information carriers are not included in the figure since they would only make the figure more complex and their relationship to the other entities are simple).

databases and might, in the enterprise context, for instance represent customers, accounts, agreements, and production results [7]. Other aspects of data that may be modeled include how the data is structured and manipulated, relationships among data, how different data are associated and influencing each other, etc. [11][43][44]. The same data can be used in different business processes and software systems, which therefore become relevant to model.

The data entity is needed in the context of business alignment, for instance when considering what data a certain business process needs and where that data is located [7]. From a business alignment perspective, it is also of interest to consider how data relates to each other and to software components. This is also of interest to the quality concern, for example for identifying redundant data. Furthermore, the CIO needs to model the reliability of the data, i.e. the probability that the functions get hold of a particular data. There are of course other questions concerned with quality attributes where the data entity is of interest.

4.6 Functions

Typically there is a substantial amount of functions in an enterprise software system. In general, functions operate on data according to business rules [43][44]. Examples of functions include high-level concepts, such as enterprise resource planning, as well as lower-level tasks, such as adding a customer record to a database. In analogy to data, functions may be classified by taxonomies. Furthermore, they may be associated with attributes, such as performance or reliability properties.

The relation of the CIO's concerns to functions is similar to the that of data. Besides issues related to business alignment, quality in the terms of functional redundancy are of interest [7][37][43]. Furthermore, the CIO might be interested in the performance of certain functions and the security, for instance in terms of preventing unauthorized access to certain functions [5][44].

4.7 Business processes

A business process is a goal and production-oriented activity of the enterprise that takes a certain input and creates output adhering to the process goal, e.g. customer attention, production of goods, or business support [21][27][28][45]. Business processes are in turn broken down into a number of steps or activities that are performed according to a set of rules [21].

Traditionally, business process modeling has preceded the software system development, but recently the separation between the two domains has become more blurred. Today, for instance, software applications often shape the business processes rather than vice versa [27][28]. Consequently, managing the relationship between the two domains is becoming increasingly important. The business process entity is an obvious part in the analysis of the business alignment concerns, but also from a quality perspective.

4.8 Organizational units

In organizations, people are, with few exceptions, divided into different, typically hierarchically ordered, organizational units, such as divisions, departments, sections, etc [46]. These organizational units are related to various ESSA entities [7][21][46]. For example, they might be owners or users of software components, perform business processes, and, of course, users belong to them. The organizational units are therefore an important aspect for the CIO to consider [11].

For the business alignment concern the question of what organizational units are using a software component is interesting [11][46]. Within the system maintenance concern the CIO might want to know what organizational

unit is responsible for maintaining which software component in order to make sure that no unauthorized modifications are made, to keep track of maintenance contractors etc. [7][11]. In connection to the competence of personnel the CIO might want to know to what organizational unit users and information carriers belong to and what organizational units have information about some parts of the enterprise software systems [27].

4.9 Information carriers

Apart from modeling the technical and business-oriented aspects of the enterprise, the CIO is also interested in knowledge of the enterprise software system that is to be found in documents and people [11][27]. The information carrier is modeled as anything within the company that has information relevant to for the ESSA. Typically, there are plenty of documents within the company describing the above presented entities. One might suspect to find descriptions of software system functionality, business processes, data taxonomies, and so on. Furthermore, all this information is also residing within the heads of the personnel. This entity as a part of the ESSA meta-model is further elaborated in [24].

Clearly, this entity relates to the knowledge concern of the CIO. However, keeping track and managing the company's collected information resources is also of vital importance when deciding on maintenance initiatives [11][27][44]. Modelling information carriers can, for instance, support the CIO with identifying information inconsistencies, inefficient information flows, and incomplete architectural awareness, as well as identifying central persons, functions and documents.

4.10 Relationships between entities

A sound ESSA meta-model requires consistency among the entities; well-defined relationships are important for this objective. The entities and their primary relationships to each other are presented in Figure 3. An example of the relationships between entities illustrated in the meta-model is as follows: The users belong to one or many organizational units and they are allowed to manipulate and create data and can perform functions. As seen in Figure 3, information carriers are not included in the model. The reason for this is to keep the picture of the meta-model simple since information carriers have relationships to all of the other entities. The information carriers contain information about the other entities except for organizational units, which the carriers belong to.

4.11 Viewpoints

One single document presenting a complete ESSA model would surely be unmanageably complex. Therefore, viewpoints are used (cf. 2.6). Literature gives many examples of standard viewpoints [11][12][15][16][17][18][21][23]. Two examples of traditional software architecture viewpoints are the "conceptual architecture view" [15] and the "component-and-connector viewtype" [37], which can be used for answering questions concerned with communication between and integration of software components within the enterprise software system, as well as some questions about functionality distribution. For organizational aspects, the "business process view" [21] can be employed. This viewpoint answers questions that have to do with how business processes are organized and how they communicate, as well as how software components and organizational units relate to the processes.

These two viewpoints can be implemented in the presented ESSA meta-model. So could also several other possible viewpoints, such as the architectural information view focusing on the information carriers [24], or for instance an ownership view, highlighting organizational units' relationships to software systems. By focusing on some of the entities of the model, while excluding others, is it possible to create a large number of views of the ESSA model. Thus, as argued previously in this paper, views are rather a matter of convenience than a means for defining an architectural model.

5. Example

In order to clarify how the suggested approach could serve the CIO with answers to important questions, a very brief and somewhat trivial example is described below. The example constitutes a small part of a case study conducted at a large North European energy company [24].

Suppose the CIO has estimated that there is good utility in the concern of architectural knowledge within the company (as suggested in chapter 3), then there are several aspects in which it becomes relevant to gain insight. One such issue is persistency of architecturally relevant information scattered throughout the enterprise. This aspect in turn could be indicated or estimated by answering concrete questions such as: "is a specific piece of information documented or not," and "is the information stored redundantly?" In order to acquire answers to these questions an enterprise-specific ESSA model including information carrier meta-model entities is needed. Further, suppose that the "specific piece of information" happens to be how functionality and data is distributed over software systems, then also these three entities must be included in the ESSA model. So, a model depicting information carriers (in terms of humans and codified repositories such as documents or databases) and a relationship indicating whether they possess data and functionality information of a certain system can be used for answering these questions. If information is documented or not will be revealed by the trivial analysis

procedure connecting the information to what kind of carrier it is stored in; human or codified (thus assuming that codified information is persistent while human knowledge is not). Analogously, information redundancy is simply answered by counting how many carriers that store the same information. (Just for clarification, it is worth noting that the granularity of the information may vary with needs of the CIO: the model might only depict that certain information carriers stores *data* or *functionality*, or more ambitiously, the model might contain information about which functionality and data that is stored.)

Incidentally, a slightly more elaborated model than the one presented in this example may be used as a support for estimating the search cost in obtaining the actual content of the architecturally significant information.

6. Conclusions and Further Works

This paper has proposed an approach for Enterprise Software System Architecture (ESSA) management, primarily intended for the CIO. Firstly, this approach argues for the importance of a utility-cost-based reasoning with respect to architectural knowledge. The utility of knowledge is derived from the increased value of better-informed decision-making. The cost of knowledge acquisition is primarily related to the resources spent on searching for the information required by the ESSA model. In this respect, the paper addresses the problems related to the management of the sizeable amounts of architectural information distributed across the enterprise. Secondly, as opposed to defining architecture by their views, the approach places the meta-model in the center of the architectural effort, thus ensuring the consistency of the architectural model.

Future publications will delve into the details of the proposed approach by elaborating on the individual tools and methods outlined herein.

7. References

[1] Andersson J., *Enterprise Information Systems Management – An Engineering Perspective Focusing on the Aspects of Time and Modifiability*, Ph.D. Thesis, Royal Institute of Technology (KTH), 2002.

[2] Haglind M., *Information Systems Planning in the Deregulated Electric Power Industry – Addressing Small and Medium-sized Enterprises*, Ph.D. Thesis, Royal Institute of Technology (KTH), 2002

[3] Johnson P., *Enterprise Software System Integration – An Architectural Perspective*, Ph.D. Thesis, Royal Institute of Technology (KTH), 2002

[4] Brown C., The Successful CIO: Integrating organizational and Individual Perspectives, *In the proceedings of SIGCPR '93*, 1993.

[5] Wilson D., Information System Security in an Airport Environment, *In the proceedings of the 36th Annual International Carnaham Conference on Security Technology*, 2000.

[6] Office of Roger Baker, *Guidance to Operating Units on CIO Roles and Responsibilities*, Department of Commerce, www.osec.doc.gov/cio/guideCIO.html 4/25/2003, 2003.

[7] Boar B., *The Art of Strategic Planning for Information Technology*, John Wiley & Sons Inc., 1993.

[8] Gottschalk P., Taylor N., Strategic Management of IS/IT Functions: The Role of the CIO, *In the proceedings of the 33rd Hawaii International Conference on system Sciences*, 2000.

[9] Enns H., Huff S., CIO Influence Behaviors: Antecedents, Consequences, and Moderators, *In the proceedings of SIGCPR '99*, 1999.

[10] CIO Role Survey, *CIO Insight Magazine April 2002*, (350 senior American executives) 2002.

[11] Perks C., Beveridge T., *Guide to Enterprise IT Architecture*, Springer Verlag, 2003.

[12] The Open Group, *The Open Group Architectural Framework Version 8*, The Open Group, 2002.

[13] Department of Defense C4ISR Architectures Working Group, *C4ISR Architecture Framework Version 2.0*, Department of Defense, 1997.

[14] Federal Enterprise Architecture Program Management Office, *Federal Enterprise Architecture,* www.feapmo.gov/fea.asp 9/29/2003, 2003.

[15] Hoffmeister et al. *Applied Software Architecture*, Addison-Wesley, 2000.

[16] Bass, L. et al, *Software Architecture in Practice*, Addison-Wesley, 1998.

[17] Garland J., Anthony R., Large-Scale Software Architecture – A Practical Guide Using UML, John Wiley & Sons ltd, 2003.

[18] Zachman, J., A Framework for Information Systems Architecture, *IBM Systems Journal,* Vol. 26, No 3, 1987.

[19] Martin J. N., *Systems Engineering Guidebook – A Process for Developing Systems and Products*, CRC Press, 1997.

[20] Stevens R., *Systems Engineering – Cooping with Complexity, Prentice Hall*, 1998.

[21] Eriksson H-E., Penker M., *Modeling business with UML*, OMG Press, 2000.

[22] IEEE Std 1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 2000.

[23] Kruschten, P., The 4 + 1 View Model of Architecture, *IEEE Software*, 1995.

[24] Ekstedt M. et al., The Architectural Information View – A New Perspective for Enterprise Software System Management, (To be published).

[25] Ekstedt M. et. al., Making project complexity understandable – the elegance of notations, *In Proceedings of IAMOT-03*, 2003.

[26] Gottschalk P., Strategic Management of IS/IT Functions: The Role of the CIO in Norwegian Organisations, *International Journal of Information Management*, 1999.

[27] Ward J., Griffiths P., *The Strategic Planning for Information Systems*, John Wiley & Sons Inc.,1996.

[28] Nilsson A. G. et al., *Perspectives on Business Modeling*, Springer Verlag, 1999.

[29] Session R., *Software Fortresses*, Addison Wesley, 2003.

[30] Barbacci, M., M. Klein, T. Longstaff, C. Weinstock, *Quality Attributes*, Technical Report CMU/SEI-95-TR-021,1995.

[31] Kazman, R., et al., SAAM: A method for analyzing properties of software architectures, *Proceedings of the 16th International Conference on Software Engineering*, 1994.
[32] Kazman R., *Toward Deriving Software Architectures from Quality Attributes*, Technical Report CMU/SEI-94-TR-10, 1994.
[33] Kazman, R. et al., The Architecture Tradeoff Analysis Method, *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems*, pp. 68 –78,1998.
[34] Lassing et al, Towards a Broader View on Software Architecture Analysis of Flexibility, *Proceedings of the Sixth Asia Pacific Conference on Software Engineering*, 1999.
[35] Magee, J., et al., Analyzing the Behaviour of Distributed Software Architectures: a Case Study, *Proceedings of the Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems* ,1997.
[36] Moriconi, M., et al., Secure Software Architectures, *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, 1997.
[37] Clements P. et al., *Documenting Software Architectures*, Addison Wesley, 2003.
[38] McLeod R. et al., 1995, The Difficulty in Solving Strategic Problems: The Experience of Three CIOs*, Business Horizon* January-February, 1995
[39] Allen, R., *A Formal Approach to Software Architecture* (Ph.D. Thesis), Carnegie Mellon University, 1997.
[40] Shaw, M., Larger Scale Systems Require Higher-Level Abstractions, *Proceedings of the Fifth International Workshop on Software Specification and Design*, pp. 143-146, 1989.
[41] Shaw, M., Garlan, D., *Software Architectures – Perspectives on an Emerging Discipline*, Prentice Hall, 1996.
[42] Mehta N. R. et al., Towards a Taxonomy of Software Connectors**,** *Proceedings of the International Conference on Software Engineering*, 2000.
[43] Linthicum D. S., *Enterprise Application Integration*, Addison-Wesley, 2000.
[44] Ruh W. A., *Enterprise Application Integration,* John Wiley & Sons Inc., 2001.
[45] Scholz-Reiter B. et al., *Process Modeling*, Springer Verlag, 1999.
[46] Morabito J. et. al., *Organization Modeling, Innovative Architectures for the 21st century*, Prentice Hall, 1999.