INFORMATICA, 2009, Vol. 20, No. 3, 343–368

Requirements Elicitation in the Context of Enterprise Engineering: A Vision Driven Approach

Albertas ČAPLINSKAS
Institute of Mathematics and Informatics
Akademijos 4, LT-08663 Vilnius, Lithuania
e-mail: alcapl@ktl.mii.lt

Received: October 2008; accepted: March 2009

Abstract. In the context of enterprise engineering, strategic planning, information systems engineering, and software engineering activities should be tightly integrated. Traditional, interview-based requirements gathering and elicitation techniques are suited for this aim not enough well and often lead to the violation of the strategic alignment. The vision-driven requirements engineering has been proposed to solve this problem. The paper contributes to the further development of vision-driven requirements engineering techniques. It proposes a methodical framework that defines a complete scheme to organize different level requirements and allows to flowdown requirements from business to software level preserving their business-orientation.

Keywords: enterprise engineering, methodical frameworks, requirements engineering, strategic alignment vision driven approach.

1. Introduction

The development of an enterprise system is a kind of strategic innovation. Usually, it initiates the business reengineering process which intends to introduce a new, more progressive business model and in order to solve problems, which faces current business; to explore new opportunities created by modern ICT; and to provide new values for the customers. Consequently, the development of an enterprise system should be started with strategic analysis because the strategic alignment of the enterprise system to the business goals and needs cannot be achieved without linking the system requirements with the business vision, goals, objectives and strategies. It is the main point of the vision-driven enterprise engineering. Every member of the project team, from the top managers to the programmers and testers, should be able to articulate the reasons why the project has been undertaken, and which business strategic objectives it should meet. In other words, if the team aims to develop the system truly supporting the business, it must understand what the real business needs are or, quoting Paul A. Strassmann,

> "*Before one tries to prescribe solutions to problems, one must necessarily understand and interpret the problems correctly*" (Strassmann, 1998).

The traditional requirements gathering techniques suppose that all real business needs can be discovered investigating the customers, end users and other stakeholders. However, the information gathered from the stakeholders usually is not enough trustworthy

and only partly reflects real business goals and needs. It happens for many reasons. The top management has no time for long interviews and discussions and, mainly, is not familiar enough with the details of modern ICT. It cannot envisage what new business opportunities can be created by an advanced enterprise system today. The middle-level managers and line-of-business employees are occupied with the hot operational problems, which mostly are only indirectly related with the strategic goals of the enterprise. Consequently, this personal expects first of all that the to-be enterprise system will help to solve these hot problems. As a result, the gathered requirements to a large extent are subjective. The real operational needs are tangled with the desires and wishes of the investigated persons. Shortly, such requirements usually lead to the violation of the strategic alignment.

A number of architectural and methodical frameworks aiming to lower the risk to violate strategic alignment have been proposed. The most popular ones are the Zachman framework (Zachman, 1987) and the TOGAF (The Open Group, 2002). Architectural frameworks are static and, mainly, purposed to classify artefacts. Methodical frameworks are dynamical ones. They define the order in which artefacts must be developed and provide guidance how it should be done. Encompassing all phases of enterprise engineering, they do not go into details of any particular phase or activity, including the requirements engineering. To provide guidance in which order the requirements should be elicited and analyzed, a requirement-oriented framework is necessary in addition. This paper discusses such a framework that defines a complete scheme to organize different level requirements of a to-be enterprise system and allows to flowdown these requirements from the business level to the software level preserving the business-orientation. However, it does not pretend to propose neither a new requirements engineer process nor a new requirements engineering methodology. It does not consider requirements traceability, process scalability and many other issues that should be considered describing a process or a methodology. The proposed framework is supposed to be used together with the most of current requirements engineering processes, methodologies and enterprise architectures. It is oriented to the middle scale enterprise systems because to define in advance all requirements for large enterprise systems is impossible.

The remaining of the paper is organized as follows. Section 2 discusses the Zachman framework. Section 3 describes the vision driven approach to the requirements engineering. Section 4 surveys the related works. Section 5 highlights relations between the systems engineering and the philosophy beyond the proposed methodical framework. Section 6 describes this framework in details. Section 7 concludes the paper.

2. The Zachman's Framework

The methodical framework proposed in this paper is organized in a similar way as the Zachman's framework (Zachman, 1982, 1987; Sowa and Zachman, 1992), although the recent is an architectural one. In this original version, the Zachman's framework supposes that it is possible to manage the complexity of an enterprise system using multiperspective approach or, more exactly, explicitly looking at every important issue from

every important perspective – planner's, owner's, designer's, builder's, integrator's and user's. The views describe the system by collection of artefacts (models) at the contextual, conceptual, logical, physical, integrated and operational levels. The Zachman's framework also supposes that all important issues (focus areas) for each perspective can be described answering six questions: what? (data), how? (functions), where? (network), who? (people), when? (time), and why? (motive). So, the rows representing perspectives and the columns representing important issues form matrix with the dimension 6×6. The semantic of the Zachman's framework is defined by seven rules (Sowa and Zachman, 1992; Inmon *et al.*, 1997):

- *Perspective uniqueness* – Each row represents a distinct and unique perspective.
- *Dimension simplicity* – Each column has a simple basic model describing an aspect of the enterprise architecture. Models are interdependent and interact continuously (a change in one column usually affects one or more other columns).
- *Dimension uniqueness* – The basic model of each column is unique. It means that each artefact of an enterprise system can be unambiguously classified.
- *Cell uniqueness* – Each cell is unique.
- *Dimension importance* – Columns have no order.
- *Dimension necessity* – All six dimensions are needed for the complete description each of the perspectives.
- *Logic recursiveness* – The framework may be used to describe different variants of the enterprise system (e.g., as-is and to-be) and each variant can and may be described at various levels of detail/granularity (i.e., the cells can and may be described at various levels of detail/granularity).

To sum up, I conclude that the Zachman's framework is organized using combination of abstraction, decomposition, concern separation and unification principles. Zachman defines the 6 abstraction levels or, in his terminology, perspectives. The decision to define 6 perspectives is well-motivated but not principal one. The number of perspectives is context-dependent. Developing other frameworks one may define as many perspectives as it is necessary.

Zachman decomposes each perspective into six focus areas named "what", "how", "where", "who", "when", and "why". I call it the H3W decomposition. The H3W decomposition is exhaustive and ensures the complete separation of concerns. So, at least in the context of enterprise engineering, the number six is principal one. Developing other frameworks one always must use the H3W decomposition to decompose perspectives regardless how many perspectives are defined.

3. Vision-Driven Requirements Engineering

The proposed framework follows the philosophy of the vision-driven approach. This approach is not a new one. Already James Martin in his classical work on enterprise engineering emphasized the leading role of vision for the success of enterprise engineering projects (Martin, 1995). S.M. Grotevant put this in the following way:

"*Information Technology provides the infrastructure and tools, which fundamentally change organizations, but management provides the strategic business vision that transforms technology into competitive advantage*" (Grotevant, 1998).

This approach is widely accepted by the requirements engineering community (De la Vara and Díaz, 2007; Dumas *et al.*, 2005; Lamsweerde, 2001) as well as by many important players in the field of systems engineering, including the MITRE Coorporation (McLean, 2006). In the context of the requirement engineering, the vision-driven approach means that most of the system requirements can be and should be derived directly from the results of the strategic planning, first of all from the business vision. It is a natural point of view because the enterprise systems are purposed to support the strategic goals of the enterprise. Although the stakeholders play a very important role in the requirements discovery and elicitation, nevertheless, they are only supplementary source of information serving to complete and to refine requirements derived from the results of the strategic planning. In line with this approach, the strategic analysis and the strategic planning should be modified in such a way that they can be considered also as inherent parts of the enterprise engineering process. More exactly, it is assumed that the enterprise architecture planning must be included into strategic planning because it is "*the process of defining architectures for the use of information in support of the business and the plan for implementing those architectures*" (Spewak and Hill, 1995).

In line with the vision-driven approach, the aim of the strategic analysis is to discover the threats and problems hindering to achieve strategic as well as tactical goals of the enterprise and, as a consequence, to implement successfully its mission. The unused business opportunities must be analysed and described, too. It is also highly recommended to prepare the forecast of challenges, which with high probability will be faced by the enterprise in near future. The final aim of the strategic planning is to develop the detailed vision statement which should be formulated on the basis of results of the strategic analysis and describe how, using new opportunities, to improve the business that at least the most serious problems and threats would be eliminated and that the enterprise will become able to cope with the forecasted challenges. The vision statement is considered as a part of the to-be enterprise system requirements, more exactly, as the highest-level requirements of this system defining its purpose as well as reasons why it should be developed. The vision statement is formulated in such way that it will serve as a basis to define the scope and the depth of the project. To be realistic and implementable, the vision should be constrained by a number of financial, political, legal and ethical constraints. Financial constraints define the reasonable amount of investments for the development of the to-be system. Political constraints define what business policies, first of all security policies, must be preserved in the to-be system. Legal and ethical constrains define what legal and ethical norms cannot be violated by the to-be system. So, the term "vision statement" is used here in slightly different meaning as in the business where it usually means "a short and inspiring statement of what the organisation intends to become and to achieve in the future, often stated in competitive terms". Here this term addresses a precise description of the desirable future state of the enterprise, which first of all performs a directional function. It expresses the goals, aspirations and intentions of the enterprise and must be

approved by the top management and may be by the other important stakeholders. Nevertheless, it should be developed by the professionals, mainly, by the business analysts, consultants and architects involving also the information systems analysts and even the application systems analysts. The details of the vision development process are beyond the scope of this paper and will not be discussed here.

4. Related Works

In the context of vision-driven enterprise engineering, some attempts already have been done to use the Zachman's architectural framework (Zachman, 1987; Sowa and Zachman, 1992) as a methodical one. D. Hay proposed (Hay, 2002) how to use this framework to translate the business owner's view into the architect's view. He emphasizes the importance of strategic planning and supposes that it should be considered as the first phase in developing any business-oriented software (Hay, 2002). Hay describes the tasks of strategic planning as follows:

> "*Lay out the vision, mission, priorities, and constraints of the enterprise. From this, define a set of projects, carefully setting the boundaries among them so as to make the whole coherent. These boundaries then define the scope of each project. This phase is carried out from the perspective of the planner's view (Row One) of the Architecture Framework*" (Hay, 2002).

The most important shortcoming of the proposed approach is that Hay, similarly as the Oracle methodology (OC, 2000), views the whole enterprise engineering process from the point of view of database system. He supposes that at each abstraction level the data requirements first of all should be specified and the data model should be developed. As a result, the motivation serves not as the starting point from which other requirements must be derived, but only for the explanation of reasons of already defined requirements. In addition, Hay is not consequent in dealing with the information processing requirements (i.e., IS-level requirements). On the one hand, he states that the task of requirements analysis phase is:

> "*The detailed examination of a particular area of the business. In that area, what are the fundamental, underlying structures, what are the information-processing gaps, and what kinds of information technology might address these? What data are required, when, and where, for each function to be performed? What roles perform each function, and why? What constraints are in effect?"* (Hay, 2002).

On the other hand, he interprets the Zachman's "information-system designer's view" as "a model of fundamental concepts" which defines the business in more rigorous terms. In fact, he tangles IS requirements partly with business level requirements, partly with software level requirements and supposes that the flowdown of requirements can be performed directly from the business level to the application system level. Such approach infringes the concern separation principle and increases the risk to violate the strategic alignment.

K. Wiegers proposed (Wiegers, 2003) another way how to translate the business software requirements into the business ones. Wiegers does not refer to any enterprise architecture. He also does not pretend to propose "an elaborate methodology that purports to solve all of your requirements problems" (Wiegers, 2003). Nevertheless, the scheme used to organize different levels requirements is closely related to the Zachman's framework because this scheme groups requirements into layers using criteria of views or perspectives. Wiegers defines three perspectives: business requirements, user requirements, and software requirements. Information processing requirements do not belong to any of these groups and are considered as 'system requirements', which are treated as external information source to be used defining software level requirements. It seems, that Wiegers follows here the IEEE Std 1223-1998 (IEEE, 1998) approach. His book is a valuable source for any requirements engineer, but it does not describe in details how one can perform correctly the requirements flowdown from the business to the software level. He mentions both business and software visions, but explains how to explore further the software vision only. How business vision and other deliverables of the strategic planning should be used, is not explained. There is also some obscurity in how the software requirements can be derived from the user requirements.

Highly related to the vision-driven enterprise engineering approach is also the famous Circular No. A-130 (OMB, 1996). The document requires that developing an enterprise[1] system it is necessary:

- to identify the work performed to support enterprise's mission, vision and performance goals;
- to analyze the information utilized by the enterprise in its business processes, identifying the information used and the movement of the information;
- to identify, to define, and to organise the activities, that capture, manipulate, and manage the business information to support business processes, and to describe the logical dependencies and relationships among business activities;
- to define the data and to describe the relationships among data elements used in the enterprise's information systems, and to identify how data is created, maintained, accessed, and used;
- to describe and to identify the functional characteristics, capabilities, and interconnections of the hardware, software, and telecommunications.

It is easy to see that, even if it is not stated explicitly, this document treats the IS requirements as an autonomous group of requirements which is clearly separated from the business level as well as from the application level requirements. It requires aligning the IS requirements with the business processes that support the enterprise's mission and goals. However, the document does define neither any framework nor process nor methodology how to organise, elicit, derive and flow down the requirements.

The leader in the practical application of vision-driven approach in enterprise engineering is the USA Department of Defence (DoD), which already in 1996 developed the first version of its C4ISR Architecture Framework (DoD, 1997). In 2003 this framework was supplanted by the first version of the new framework DODAF (now already

[1]The document uses the term "agency". The term "enterprise" is more general one.

the version 1.5 (DoD, 2007) is available). In the DoD approach the primary driver is the business mission. The DODAF provides four kinds of views or perspectives for the enterprise architecture: all view (AV), operational architecture (OA), systems architecture (SA), and technical architecture (TA). The AV describes the entire architecture and defines the scope and context of the architecture. The OA describes the operational context, the SA describes the system capabilities, and the TA describes the arrangement, interaction, and interdependence of the system parts. The Enterprise Architecture planning process is based on the Business Systems Planning (BSP) approach (Zachman, 1982) and takes a data-centric approach for the architecture planning. A similar approach has been advocated by Spewak and Hill (1995). Goikoetxea (2007) follows this approach, too. He argues that an architectural framework should provide five views or perspectives: Business Process Architectural View, Business Systems Architectural View, Data Architectural View, Applications Architectural View, and Technologies Architectural View. Like many other researchers, Goikoetxea advocates data-centric approach. He supposes that the Vision and Strategy document must be translated into a set of business processes describing the day-to-day business of the enterprise and each business process must be decomposed into a number of business activities. After this, the existing set of logical business systems must be modified and extended into new set of logical enterprise business systems. Goikoetxea uses the term "business system" rather to address the functional areas of a business such as human resources or finances. The new set of business systems must be represented hierarchically as a tree of business systems. Interfaces for each business system must be defined in terms of the services provided by this system. Further the business hierarchy must be broken into groups having business affinity for each other and each such group must be considered as a separate project.

To sum up, a number of approaches how to translate the business level requirements into the software level requirements has been proposed but up to time any exhaustive and detailed methodical framework still is not developed for this aim.

5. The Systems Engineering View on an Enterprise System

Enterprise engineering deals with three different kinds of systems – business, information, and application systems. However, there is no generally accepted agreement on what these terms mean. Different authors define quite differently what a business system, an information system, or even an application system is. The vision-driven requirements engineering requires to conceptualise all layers of an enterprise system in a unified manner and, consequently, to define business, information, and application systems using the same ontology and unified terminology. An attempt to do this has been made in Caplinskas *et al*. (2002a, 2002b, 2003) using the system engineering paradigm. The system engineering is a holistic, integrative discipline, wherein the contributions of many engineering disciplines are evaluated and balanced, one against another, to produce a coherent whole that is not dominated by the perspective of a single engineering branch (Griffin, 2007). Even if today many of main players in the field, including the International

Counsel on Systems Engineering (INCOSE), consider the system engineering primarily as an interdisciplinary field of engineering, even as a separate engineering discipline, it is possible to consider it also as a generic discipline, which focuses on the ideas, concepts, principles, methodologies, methods, techniques and practices that, after specialization, adaptation and enrichment, can be applied for engineering almost of any kind of systems. A similar point of view among others advocates also the generic design science (Warfield, 1994). For sure, not all engineering methods, techniques and approaches, especially ones related to the implementation of system components, can be generalized and formulated in terms of abstract system. Besides, there are still many unsolved issues related to the engineering of self-organising emerging systems, first of all, social and socio-technical ones. Because the business systems and the IS are socio-technical systems, any enterprise engineering effort will be, probably, fallen if it does not take into consideration also the human components of the system. In recent time, this problem has been intensively discussed and some solutions have been proposed. For example, Haskins (2005, 2007, 2008) suggests that using patterns and pattern languages it is possible to extend the language of systems engineering and to include both social and technological contexts. Another solution is to consider the enterprise system as a composition of only "mechanistic" elements, including elements needed to support the social reengineering issues, supposing that the decisions how to redesign social components have already be done and are reflected in the reengineering strategy. Nevertheless that the problem is still far to be solved in full extent, the systems engineering methodology can and should be applied for the engineering of the enterprise systems. This point of view is supported by many authors.

An abstract artificial system can be defined from outside as well as from inside perspectives. From the inside perspective it is usually defined as a whole, formed out of sets of elements and relationships between them that distinguish itself by emerged properties of connectedness and functionality. From the outside perspective it is defined as a black box, that is, as an entity characterized only by its external interface behaviour. It means that the system is defined in terms of inputs, outputs and a mathematical relation between them. Some authors, for example, Myers and Kaposi (2004), see a system rather as a model of a real-world entity or, in terms of Myers and Kaposi, as a representation of a referent, but not as a real-world entity itself.

> "*It is not true that anything and everything is a system, but it is true that anything and everything can be modelled as a system*" (Myers and Kaposi, 2004).

In line with this approach, the outside perspective is described by a *black box* representation of system and the inside perspective is described by its *structural* representation. Taking a snapshot of the referent at the time instant, "we obtain its representation as a *product"*, and, modelling its operation over a period of time, "we shall have its representation as a *process"* (Myers and Kaposi, 2004).

The idea that any system under development should be represented firstly as a black box and that this representation should be step by step transformed later into structural representation is widely accepted in many engineering fields, including software engineering and enterprise engineering. In SE this idea was proposed by H.D. Mills (Linger *et al.*, 1979) and elaborated later in the Cleanroom methodology (Prowell *et al.*, 1999)

where the externally visible behaviour of a black box is specified in the terms of a total mathematical function that maps every possible sequence of input stimuli to the appropriate response. Frappier and St-Denis (1998) extended this definition in that they use a relation instead of a function. This model allows representing the behaviour of state-aware systems without defining internal states. The systems with the states observable from the outside perspective can be described using pre- and post-conditions. Using invariants, and pre- and post-conditions it is possible to describe some non-functional characteristics of system, too. However, not all non-functional characteristics can be specified using this approach because some characteristics cannot be related to any specific function or relation and are generated by a whole system. Linear and non-linear, deterministic and non-deterministic systems can be represented as functional style black boxes. However, it is possible to represent the systems using also black box models of other styles: object-oriented, service-oriented, task-oriented, etc. For example, in the object-oriented paradigm an object can be considered as a black box, which receives and sends messages. According to the contract principle proposed by Meyer (1988), the specification of an object-oriented system can be treated as a black box representation of this system. A more elaborated representation of an object-oriented black box system is a use case model (Jacobson *et al.*, 1992). A use case is a pattern of behaviours the system exhibits or, in other words, it specifies a set of transactions initiated by an external actor. It means that it describes how the system responds to the external stimuli. A stimulus is an event that initiates a goal-oriented interaction between an external actor and the system. This interaction produces a result that is observable from the outside perspective. An actor is someone or something external to the system and interacting with it. A use case description may include the pre- and post-conditions, invariants, goal description, description of abnormal situation and other related information. Thus, "*a use cases capture who (actor) does what (interaction) with the system, for what purpose (goal)*" (Malan and Bredemeyer, 1999). A use case model describes all the actors of the system and all use cases initiated by them. It specifies all the ways of using the system or, in terms of the use case approach, describes the system as having a set of responsibilities. Apart of the use case model, this approach provides also other black box views of the system (Gomaa and Olimpiew, 2005): use case interaction model, use case collaboration model, use case context model and use case state model. A use case interaction model describes all system usage scenarios. The system is treated as a black box object, actors as objects, and stimuli and responds as the messages. This model explicitly describes the sequences of inputs and outputs of the system. A use case collaboration model represents the functionality of the system by roles that each actor plays in collaborations with it. This model is used for black box representation of an enterprise system in the SEAM methodology (Wegmann *et al*., 2005). A use case context model represents all external classes that interact with the black box system that is represented as an aggregate class (Gomaa and Olimpiew, 2005). External classes represent users, external devices and external systems. The difference between actors and external classes is that actors are intended to be proactive, providing inputs to the system. An actor can interact with the system using several external classes. For example, using an ATM to withdraw the cash from the machine, the actor (customer)

uses several external hardware devices (card reader, keyboard, display, cash dispenser, printer), which are represented as external classes (Gomaa and Olimpiew, 2005). Thus, comparing with the use case model, the use case context model provides some additional information. A use case state model describes the interactions between the actor and the black box system by protocol state machines. This model describes the state-depend systems more effective as the use case model because it explicitly describes the inputs from each actor and from the actor and state dependent responses.

The main conclusion of this discussion is that from the outside perspective it is reasonable to consider an abstract system as a constrained black box of certain style that aims at a set of goals and objectives. Goals specify the intended use of the system and the output to be provided by it. Objectives are defined measurable levels against which non-functional characteristics defining the quality of the system can be measured[2]. The constraints specify external restrictions imposed on the behaviour of the system. The style specifies the way in which the functionality of the system can be accessed.

In the enterprise engineering context, black boxes usually are purposed to process the events. The term *event* is used here in a very broad sense. It encompasses commands, situation and everything that directly or indirectly triggers the system. Such black boxes can be specified using the H3W decomposition (Sowa and Zachman, 1992).

Let me go back now to the definition of an abstract system from the inside perspective. Usually, it is defined as a whole, formed out of sets of elements and relationships between them that distinguish itself by emerged properties of connectedness and functionality. Although this definition is accepted by most researchers working in the fields of systems and enterprise engineering, it is mainly ignored when defining business, information and application system. I advocate that these systems should be defined by the concretisation of the definition of an abstract system and that in the context of enterprise engineering it can be done in the below described way.

An enterprise business system is a whole formed out of sets of business actors, resources, and interrelated business processes (possibly, including some production processes) that implements business policies and business rules accepted by the enterprise. Any business system is purposed to achieve certain, usually long term mission-related business goals delivering particular domain-specific products or/and services. A business actor is an entity capable of exerting behaviour. Each actor in the system is responsible for performing some set of business and/or production operations and is implemented using some combination of peopleware, hardware and software, including information and application systems. A business system domain (or simply business domain) is a sphere of business activity related to the mission of enterprise. Sometimes such domains are called mission domains and the term business domain encompasses not only mission but also the so-called resource domains (Wells, 2006). Typically, an enterprise has several business domains and, consequently, several business systems. For example, a university may have separate business systems to deliver the education, research and community services. Each business system produces, process, uses or in other way manipulates some

[2]This definition has been suggested by Ferrstl and Sinz (2006).

business objects. The term *business object* is used here as a generic one and refers to a product, service, resource or business event. Mostly, it is interchangeable with the term *accounting object.* Each business system provides a number of interfaces. The parts of business system are related to each other by architecture of this system (static relationship) and by system's workflow (dynamic relationship). All the enterprise business systems form together a collaboration in which each business system plays some prescribed role required to fulfil the mission of the enterprise.

An enterprise information system (IS) is a whole formed out of organisational memory and sets of information processing actors (IPA), information flows, and interrelated information processing processes implemented in accordance with the enterprise information processing policies and standards. It includes also various system integration and management mechanisms including organizational ones. An enterprise information system is purposed to deliver information, computing and communication services required by internal business actors and, possibly, by some external actors (customers, clients, governmental bodies, etc.). An IPA is an entity capable to deliver some set of required services and is implemented using some combination of peopleware, hardware and software, including application systems. Shortly, an information system is system that provides the information required to perform everyday business activities and supports the business decisions (Davis and Olson, 1985). Typically, it consists of a number of relatively autonomous subsystems, a part from which support so-called functional areas (finance, human resources, materiel resources, procurements, management, etc.) and other part provide specific services required to support particular enterprise business systems. All subsystems of an enterprise IS are integrated and form together a collaboration in which each subsystem plays some prescribed role required to support the enterprise business as a whole. An enterprise IS provides a number of interfaces used to access the services delivered by it. It produces, process, uses or in other way manipulates some *information objects,* a part from which models accounting objects and other part are information entities that exist only at the IS level. The parts of an IS are related to each other by architecture of this system (static relationship) and by system's workflow (dynamic relationship).

As a part of enterprise system, an application system can be defined as a whole formed out of sets of hardware agents, protocols, data stores, knowledge bases, and interrelated software application programs. An application system is purposed to support a particular information processing function and to deliver the services required by the IPA related to this function and, possibly, by the other application systems. Some application systems may fully implement the appropriate IPA or, in other words, act as IPA and deliver services directly to internal or/and external business agents. Such application systems are called *software agents.* Hardware agents (computers, networks, etc.) execute software applications programs. Typically, an application system consists of a number of subsystems (components) providing specific services required to support a particular information processing function. An application system has a number of interfaces used to access the services provided by it. It produces, process, uses or in other way manipulates some *digital objects* a part from which models information objects and other part are software objects

used for internal purposes. The parts of an application system are related to each other by software architecture of this system (static relationship) and by system's control flows (dynamic relationships). All application systems form together a collaboration in which each application system plays some prescribed role required to support the enterprise IS as a whole.

6. The Proposed Framework

The proposed methodical framework (Fig. 1) is intended to be used for requirements elicitation, specification, analysis and evaluation. This paper discusses the structure of this framework and the order in which requirements must be elicited. The description of the concrete requirements derivation and flowdown techniques is omitted due to the restriction of the size of the paper.

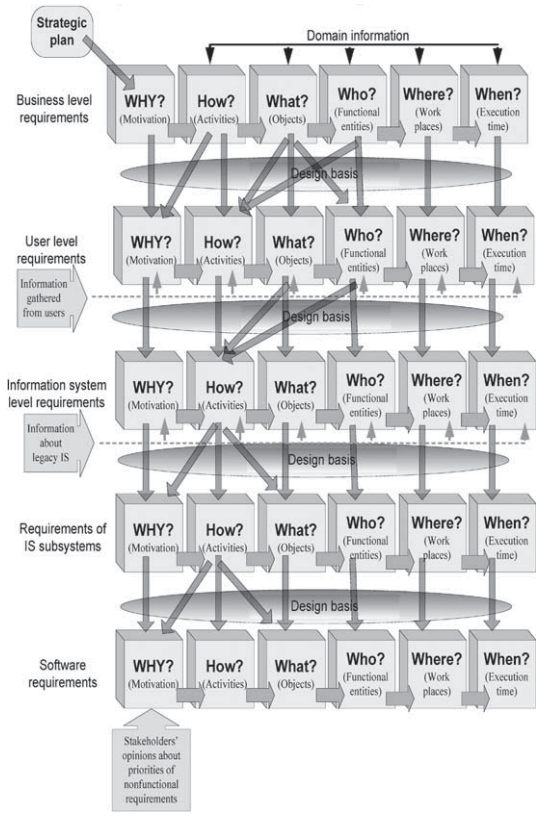

Fig. 1. The proposed methodical framework for requirements elicitation, analysis, specification and validation.

6.1. *Perspectives*

The framework (Fig. 1) provides five perspectives describing the business level requirements (the view of business analyst), the user level requirements (the view of stakeholders), the IS requirements (the view of IS analyst), the requirements of IS subsystems (the view of IS engineer), and the software requirements (the view of software analyst). This list provides the levels of requirements specified normally for any enterprise system under development. To be complete, it should additionally include the requirements of software components (the view of software architect), the implementation requirements (the view of software engineer), the process requirements (the view of process engineer), and the testing requirements (the view of tester). However, these perspectives are omitted here because they are well-researched ones and described in many textbooks, for example, in Bray (2002). The first five perspectives differ from corresponding ones provided by the Zachman's framework because they are designed for different purposes. The requirements of a to-be enterprise system should be elicited, specified, analyzed and evaluated for each of the proposed perspectives.

6.2. *Focus Areas*

The proposed framework (Fig. 1) is built using the H3W decomposition. The focus areas are defined as following:

- "Why": Motivation (vision of the system as it seen from the corresponding perspective).
- "How": Service requirements (what services are required to support the vision; what accuracy, reliability, and safety constraints shall meet these services?); architectural requirements (what architecture should be chosen to produce required services?).
- "What": Objects requirements (what kind of objects shall process the system in order to deliver required service; how these objects should be protected in the system?).
- "Who": Accessibility requirements (who will use the system and in which way? how many and what kind of interfaces should provide the system? what level of usability should be ensured for this aim?).
- "Where": Workplaces requirements (what workplaces are required for each "*who*"? how these workplaces should be equipped?).
- "When": Efficiency requirements (what deliver time is acceptable for each of services provided by the system?).

The development of architectural requirements is a design activity. However, it is considered here as a part of the proposed methodical framework because the functional and architectural requirements are tightly related and should be developed in parallel. Besides, it is impossible, for example, to produce application systems requirements without decomposing the IS into subsystems and deciding what application systems are required to support these subsystems.

6.3. *Rules*

The semantic of the proposed framework (Fig. 1) is described by the following rules:

- *Perspective uniqueness* – Each row represents a distinct, unique perspective. Each perspective describes the requirements of to-be enterprise system from the point of view of some group of stakeholders. The requirements should be produced in the order provided by the ordering of rows. The perspectives are interdependent, most of the lower level requirements follow from the higher level ones; all higher level requirements must flowdowned to the lower level; the requirements coming from other sources (additional requirements) are constrained by the higher level requirements.
- *The dominant role of business vision* – Top level requirements are derived from the constrained business vision.
- *Design basis* – The flowdown of the requirements from the higher level to the lower one should be done taking into account the design basis document that lists the features that may be potentially provided by the lower level requirements (Fig. 1). This document helps to avoid old-fashioned solutions that may be suggested by the stakeholders, which often are unfamiliar with the possibilities of the advanced ICT technologies.
- *Alignment of perspectives* – Each perspective except higher level one is constrained by higher level perspective. On the other hand, each perspective is augmented using additional (external) sources of information (Fig. 1). Additional requirements elicited from the external sources augment the derived requirements but cannot contradict higher level requirements.
- *Dimension ordering* – Columns are ordered according to their dependency in the following way: “why”, “how”, “what”, “who”, “where”, and “when”.
- *Dimension simplicity* – Each column contains a group of requirements which describes an aspect of enterprise system. The later column may depend only on previous columns. A change in one column usually affects one or more other columns.
- *Dimension uniqueness* – The group of requirements described in each column is unique. It means that each requirement of enterprise system can be unambiguously classified.
- *Cell uniqueness* – Each cell is unique.
- *Dimension necessity* – All six dimensions are needed for the complete description of each perspective.
- *Logic recursiveness* – The framework may be used to describe requirements at various levels of granularity.
- *Procedural independency* – The framework establishes the classification of enterprise system requirements, defines the dependencies between the groups of requirements and the order in which different groups should be elicited, analyzed, and specified. However it does not define any particular methods or techniques that should be used for this aim. It does not also require some particular methods or techniques be used for the requirements flowdown or derivation. It is independent from any particular requirements engineering process, methods or techniques.

6.4. *The content of the Cells*

The content of the cells of the proposed framework has been discussed shortly in Caplinskas and Paskeviciute (2009). Here it is discussed in more details (Table 1).

6.4.1. *Motivation (why-requirements)*

Vision driven approach requires that during the strategic planning the vision statement would be developed and that the financial, policy, legal, and ethical constraints for this statement would be defined. The vision statement states what business threats and problems shall eliminate the to-be business system and what new business opportunities it shall provide. The business level why-requirements must be derived from this statement. They consist of a prioritised business goal tree and the financial, political, legal and ethical constraints that constraint the implementation of the business goals provided by this tree.

At the user level, the why-requirements are formulated in terms of business processes that shall support the to-be enterprise system. They define the list of business processes and, for each process, its priority, inputs, outputs, events that initiate this process, and its implementation constraints (Blick, 2000; Rukanova *et al.*, 2006).

The list of the business processes is derived from the business goal tree. The inputs, the outputs and the events of a business process are defined refining the business goal that is implemented by this process. To derive the implementation constraints of the business processes, the implementation constraints of the business goal tree must be allocated to the business processes and the flowdown of the allocated constraints must be performed. In other words, the decision should be made what financial constraints must be established for implementation of each process and how this process should be constrained in other ways. The priorities are assigned to the business processes taking into account the priorities of corresponding business goals and hearing the voice of the stakeholders. The priorities define the order in which the processes shall be implemented and should be used later to take decisions about the versioning of the to-be system.

At the IS level, the why-requirements include the to-be IS vision statement, the feature tree, its implementation constraints and the priorities of the features. The vision statement is derived from the higher level why-requirements and expressed in terms of the user-oriented IS services supporting the required business processes. It highlights the differences between the current IS and the to-be IS. The product vision statement template (Moore, 1999) may be used to formulate the vision. The vision is refined by an IS feature tree (van den Broek *et al*., 2008; Roubtsova and Roubtsov, 2006). The design basis supports the development and refinement of vision statement suggesting what services may provide and what features can implement an advanced IS today. The financial, policy, ethical and legal constraints of feature tree are derived from the constraints of business processes. The priorities of the features are derived from the priorities of business processes. If the project provides an incremental development, the feature tree should be decomposed into separate increments.

At the IS subsystems level, the why-requirements are formulated for each subsystem separately. The list of subsystems is defined by the IS level how-requirements. To derive the why-requirements, the IS features and implementation constraints are allocated to the

Table 1
The content of the cells

Why	How	What	Who	Where	When
		Business level requirements			
Business vision, goal tree and its implementation constraints	Requirements of business services	Types of business objects with which shall manipulate the business system	Stakeholders, access rights and privileges of stakeholders	The list of system access points related to the stakeholders and to the business services	Time constraints for service deliver
		User level requirements			
Business processes, their priorities and implementation constraints	Operational needs (business use cases), business policies and rules	What information is required to produce required service and how it shall be protected (conceptual requirements)	To whom shall be delivered IS services, access rights, privileges and competence of services receivers	Where shall be delivered the services	Time constraints for business use cases
		IS requirements			
IS vision, feature tree and its implementation constraints	IS use cases, information processing standards, policies and rules. Trustworthiness of IS services. Architectural requirements	What information shall process IS and how this information shall be protected (detailed requirements)	Access, interface and usability requirements	How should be equipped workplaces	Time constraints for IS use cases
		Requirements of IS subsystems (for each subsystem)			
Subsystem vision, feature tree and its implementation constraints	Subsystem level use cases, trustworthiness of results, architectural requirements	What information shall process each of the subsystems and how this information shall be protected	Access, interface and usability requirements	Detailed workplaces requirements	Time constraints for subsystem use cases
		Software requirements (for each application system)			
Product vision, implementation constraints	Application system use cases, trustworthiness of results	Data requirements, security requirements	Access, interface, usability and ergonomic requirements	Run-time environment, resource behaviour requirements	Time constraints for application system use cases

IS subsystems and their flowdown is performed. So, the why-requirements of a subsystem consist of its vision statement, its feature tree and its implementation constraints.

At the software level, the why-requirements are formulated for each application system supporting the to-be IS. The list of application systems is defined by the subsystem level how-requirements. The requirements of an application system consist of the vision of this system and its implementation constraints. The vision statement is statement is produced refining the feature tree of the IS subsystem, which shall be supported by this application system, and is expressed in terms of application system features. The product vision statement template (Moore, 1999) may be used to formulate the vision. The design basis suggests what kind of features can be implemented by an advanced application system today. To derive the implementation constraints, the constraints of each to-be IS subsystem must be allocated to the application systems supporting this subsystem and the flowdown of the allocated constraints must be performed. An application system may support several IS subsystems and inherit their features and constrains. Nevertheless, except the case when the business goal tree includes contradictory goals, any contradictions ought not to arise.

6.4.2. *Functional and architectural requirements, trustworthiness of delivered results (how-requirements)*

At the business level, the how-requirements define the list of business services, which shall provide the to-be system in order to implement the business vision, the required functional and extra-functional properties of the services, and their architecture. A business service is a unit of business system capability. It produces results that are delivered to a customer and it is implemented by a combination of business transactions. Thus, its architecture is defined by the set of tightly coupled or dependant on similar business events business processes. The architectural requirements of a business service define how it is decomposed into business processes and how these processes are related each to other. The list of all business processes defines the scope of the project (Wiegers, 2003) because it describes what business processes shall be supported by the to-be enterprise system. The extra-functional properties of a business service define the required quality of service (QoS) and the required level of its trustworthiness. The business level how requirements are derived from the why-requirements taking into account the additional domain information collected from the stakeholders.

At the user level, the how-requirements define the business use cases, and related business policies and business rules. That is, they describe how shall be implemented the business transactions supported by the to-be business system. The list of primary and secondary agents participating in the use cases is derived from the business-level who-requirements. The list of business events that initiate the use cases and the list of business objects processed or/and used by these use cases are derived from the business-level what-requirements. The business polices and business rules determine how the to-be business system shall conduct with regard to each business use case. They may be specific to a use case or applied across the entire business system. The how-requirements should define when and where business policies and rules are applicable. Each business use case

must provide references to the specific business rules that are active during this use case. However, policies and rules should not be embedded in the use cases.

At the IS level, the how-requirements define the IS level use cases, the list of indispensable information processing standards, information processing policies and rules, and the list of the to-be IS subsystems. So, they define what services shall provide the to-be IS in order to fulfil the operational needs of the stakeholders. The IS level use cases are derived from the business use cases eliminating their non-relevant parts and refining the relevant parts. The user-level who- and what-requirements should be taken into account refining the business use cases. The information processing standards, and the information processing polices and rules determine how the IS shall conduct with regard to each IS use case. Policies and rules may be specific to a use case or applied across the entire IS. The how-requirements should define when and where the information processing policies and rules are applicable. Each IS level use case must provide references to the specific information processing rules that are active during this use case. However, policies and rules should not be embedded in any use case. The QoS and the trustworthiness level of the to-be IS services including accuracy, reliability, and safety of delivered results are derived from the IS implementation constraints. The IS level how-requirements include also the architectural requirements defining how the to-be IS shall be decomposed into subsystems of different kinds including transaction processing, decision support, information flow management, workflow management, content management, document management, data processing, group work support and resource management ones. The design basis suggests from what kinds of subsystems can be composed an advanced IS today. The IS may provide also expert systems, search engines and other sophisticated software components. However, mainly, they are regarded as the internals of the appropriate IS subsystems. On the other hand, the global data stores usually are implemented as IS subsystems.

At the IS subsystems level, the how-requirements for each subsystem define its use cases, the required QoS and the required level of trustworthiness of delivered services, and the list of application systems and other software components necessary to implement the required use cases. To produce the requirements, the to-be IS use cases, the trustworthiness requirements and the QoS requirements must be allocated to the IS subsystems and the flowdown of allocated requirements must be performed. The how-requirements of a subsystem cannot violate implementation constraints of this subsystem defined by its why-requirements.

At the software level, the how-requirements of any application system define the use cases of this system, and the required QoS and the required level of trustworthiness of services delivered by it. The application system use cases are derived from the use cases of the corresponding subsystem of the to-be IS taking into account the who- and what-requirements of this application system.

6.4.3. *Object requirements (what-requirements)*

At the business-level, the what-requirements define the kinds of business objects necessary to implement the business services defined by the how-requirements. The requirements are produced analyzing these services.

At the user level, the what-requirements define what kinds of business objects and in which way shall model and protect the to-be IS. The requirements are produced confronting the business level what-requirements with the business use cases. The design basis suggests what kinds of information objects can process information systems today and, consequently, in which ways the required business objects can be modelled. The information objects, which model the accounting objects of the business system, normally, are represented as records. Other business objects are modelled as pictures, photos, maps, audiovisual objects or in some other way. Usually, the stakeholders expect that the to-be IS will manipulate also with some auxiliary information objects modelling any business object. The decision on what kinds of auxiliary information objects are required is done analyzing the business use cases and the additional information gathered from the stakeholders. The information necessary to formulate security requirements should be collected also from the stakeholders. So, the user level objects requirements define the kinds of the information objects that shall be stored in so-called organization memory and the required protection level for each kind of these. The term "*organization memory"* addresses here the total body of enterprise's data stores including computer data bases, data warehouses, intranet pages, paper documents and other stores. The objects requirements must describe the structure, identification, representation accuracy and integrity constraints of the information objects stored in this memory. They must describe also who and where shall create each kind of information objects, how these objects shall get into enterprise memory, how they shall be protected and how they shall be presented to the users. The information protection requirements must define the required confidentiality class for each kind of information objects and the required protection level for each confidence class.

At the IS level, the what-requirements define the structure and other properties of the to-be organizational memory. They should define which kinds of data stores (computer DB, repository of the paper documents, repository of the Web documents, etc.) shall provide this memory, how shall be organized (centralized, distributed, etc.) each data store, and what replication requirements it shall meet. The requirements are derived from the user level ones. First of all, the requirements of information objects must be allocated to the data stores and the list of required stores must be produced. Then the allocated requirements, including the security requirements and the information presentation requirements, are refined taking into account the specific properties of the data store, where the information objects shall be stored, and corresponding IS use cases. Refining the information presentation requirements, for each information object the decision about media, used to present this object for the users, must be done.

At the IS subsystems level, the what-requirements define with which kinds of information objects shall manipulate each subsystem. To produce the requirements, the higher-level what-requirements must be allocated to the to-be IS subsystems and the flowdown of allocated requirements must be performed. The list of the to-be IS subsystems already is defined by the IS level how-requirements. The subsystems' use cases define to which subsystem the requirements of which information objects must be allocated. A subsystem may access internal as well as external data stores. If subsystem accesses some external

data store, the requirements may define a special view of information objects stored in this store.

At the software level, the what-requirements define how the information objects shall be digitalised and how the digital objects shall be protected from accidental loss, corruption and/or deliberate unauthorized attempts to access or to alter them. To produce the requirements, the what-requirements of each to-be IS subsystem are allocated to the application systems defined by how-requirements of this subsystem. To which application system the requirements must be allocated, define the use cases defined by the software level how-requirements. The required accuracy of the digitalisation is defined refining the allocated requirements. The exact form of the digital representation is the subject of the future design decisions. The data security requirements are derived from the higher level security requirements.

6.4.4. *Access, interface and usability requirements (who-requirements)*
At the business level, the who-requirements define the list of stakeholders of the to-be enterprise system, their access rights and privileges. The stakeholders are defined in terms of roles (i.e., as the departments, positions, customers, clients, suppliers, regulators, etc.). External systems and processes, including software, hardware, devices and other "things" that shall interact with the to-be enterprise system are regarded also as the stakeholders. The list of the stakeholders is derived from the business goal tree considering which roles contribute to the achieving the goals provided by this tree. The bottom-level goals are allocated to the roles directly contributing to this process, the higher level goals to the managers of different levels, including the top management of the enterprise. How and with which business objects shall be allowed to operate a stakeholder, follows from the role, which this stakeholder plays in the goal achievement process, and from the requirements of the objects produced or used by this process.

At the user level, the who-requirements define the list of the consumers of the services delivered by the to-be IS, their access rights, privileges, preferences and competences. The list is produced confronting the list of the stakeholders with the business use cases. The business use cases define also how and with which information objects shall operate each consumer. Together with the business-level requirements, they serve also as a basis to define the consumer's access rights and privileges. For each consumer, the requirements describe also his competences, including computer literacy, and preferences (easy-to-use, performance, etc.). These requirements must be taken into account defining later the interface and usability requirements. A special category of consumers is the external systems, processes and devices. The access rights, privileges and, possibly, competence requirements should be defined also for this category of consumers. For any category of the services consumers the access rights and privileges are defined taking into account the objects requirements.

At the IS level, the who-requirements define the list of the to-be IS interfaces, the usability requirements, the services delivered through each interface, and the access rights and privileges of the service consumers. They are derived from the higher-level who-requirements. First of all, the list of the IS interfaces is produced. Then the IS services

are allocated to these interfaces and the flowdown of the allocated requirements is performed defining for each interface which IS services and how can be accessed through this interface. An IS interface is a protocol which defines how a services consumer shall interact with the IS in order to receive required services. Because the IS may deliver some information, computing or even communication services without using any software system (i.e., manually), an interface may provide also such forms of interaction as the oral or written interaction. The access rights and privileges, competences, and preferences of a service consumer are derived from business-level requirements taking into account the IS-level use cases and the requirements of the information objects. Analyzing consumers' competences and preferences the usability requirements are produced for each IS interface.

At the IS subsystems level, the who-requirements define the access, interface and usability requirements of each to-be IS subsystem. To produce the requirements, the IS level who-requirements must be allocated to the to-be IS subsystems and the flowdown of the allocated requirements must be performed. The subsystem-level use cases are used to decide which requirements to what subsystem must be allocated. They are also used to decide which service consumers shall use each interface of each subsystem, which information objects they shall access and, consequently, which access rights and privileges must be allocated to this subsystem.

At the software level, the who-requirements for each interface of each application system define the kind of the task description language, the users' access rights and privileges, the users' authorization requirements, the usability requirements and the ergonomic requirements. The requirements are produced allocating the higher level requirements and performing their flowdown. The flowdown of the requirements is performed taking into account the how- and the what-requirements of the corresponding application system.

6.4.5. *Workplaces requirements ("where")*

At the business level, the where-requirements define the list of the points from which the system shall be accessible (i.e., system access points, SAP), and who (in terms of roles) and for which purposes shall access the system through these SAPs. All services delivered by the to-be IS can be divided into three categories: the services that shall be delivered securely to the internal users (i.e., to the enterprise departments, to its employees, etc.); the services that shall be delivered securely to the special categories of the external users (e.g., the bank services delivered to the depositors); and the services that shall be delivered to the general public. By analogy with intranet, extranet and internet, it is reasonable to speak about the internal services network (ISN), the external services network (ESN), and the public services network (PSN). Normally, the intranet is a part of the ISN, and the extranet is a part of the ESN. However the ISN and the ESN are broader concepts as the intranet and the extranet. The requirements for the SAPs provided in the ISN, the ESN and the PSN are different. At the business level, the ISN requirements define where (terrains, buildings, rooms, etc.) shall be deployed the workplaces of the employees of the enterprises and for which purposes they shall be used performing the business transactions. They define also how workplaces shall be related to each other in order to

form the ISN. The ESN and the PSN requirements define where inside of the enterprise shall be deployed client service terminals (CST) (e.g., the client service counters, the automated tell machines, or computers through which clients may access some services) and how many and what kinds of SAP shall be provided by the system to deliver the services for the remote users of ESN and PSN, and to interact with the external systems, processes and devices. The business lever where-requirements also define how the SAPs of the ESN and PSN shall be connected with the ISN.

At the user-level, the where-requirements define the intended usage of each SAP when performing different kinds of the business transactions provided by the to-be enterprise system. To produce these requirements, the business use cases are allocated to the SAPs and the business level where-requirements are refined performing the flowdown of the allocated use cases.

At the IS level, the user level where-requirements must be completed and refined. First of all, it is necessary to decide where the to-be IS data stores shall be deployed and to add the "workplaces" of the data stories to the list of the SAPs. Similarly as the "normal" workplaces, the data stores and the services to save and to retrieve the information objects shall be provided for them. It is a reason to speak about the "workplaces" for the data stores. Next, the IS level use cases must be allocated to the SAPs and the decision must be done which services provided by the to-be IS shall be delivered to each SAP. Finally, analyzing how the IS level use cases are related each to other, must be defined what communication services shall provide the to-be IS to ensure required communication between different workplaces inside the ISN, between the ISN and the ESN and between the ISN and the PSN.

At the IS subsystems level, the where-requirements define with what kind of hardware and software shall be equipped each to-be workplace and each to-be SAP. They also define which parts of ISN, ESN and PSN shall be implemented by the intranet, the extranet and the internet technologies. To produce the requirements, the higher-level where-requirements must be allocated to the to-be IS subsystems and the allocated requirements must be refined taking into account the subsystem-level use cases.

At the software level the where-requirements must define the required run-time environment (system software, hardware, network environment, etc.) and the required resource behavior of each application system. To produce the requirements, the higher-level level where-requirements must be allocated to the application systems and the flowdown of the allocated requirements must be performed.

6.4.6. *Performance requirements ("when")*

At the business level, the when-requirements must define the required performance of the to-be business system. The performance of a business system can be defined in many different ways. In the context of enterprise engineering, the performance of a to-be business system usually is defined in the terms of business services. For each business service, the average or the maximal acceptable deliver time, including the time consumed in the added value and logistic chains must be defined. It is supposed that the business-level when-requirements for each business event, business situation, initiative and/or directive must define maximum or average its processing time.

At the user level the when-requirements must define the time constraints for each business use case. Similarly, at the IS level they must define the time constraints for each IS use case, at the IS subsystems level – for each subsystem-level use case, and at the software level – for each application system level use case. In all mentioned cases, the time constraints must be stronger as the appropriate higher level constraints and must be derived from theses.

7. Conclusions

The development of an enterprise system should be started with the strategic analysis because the strategic alignment of the enterprise system to the business goals and needs cannot be achieved in any another way. The vision-driven requirements engineering has been proposed to support the derivation of software requirements from the results of strategic planning, first of all from the business vision. A number of architectural and methodical frameworks aiming to lower the risk to violate strategic alignment have been proposed. However, they encompass all phases of enterprise engineering, do not go in very details of requirement engineering and do not support directly the vision driven requirements engineering techniques. A special phase-oriented methodical framework is needed for this aim. It should be designed in such a way that can be used together with the general purpose architectural and methodical frameworks. In addition, it should be based on the systems engineering paradigm and define the business, information, and application systems as the special kinds of an abstract system because the vision-driven approach requires to conceptualise all layers of an enterprise system in a unified manner. The paper proposes and investigates such methodical framework purposed for the elicitation of requirements of a to-be enterprise system. Together with the philosophy described above the Zachman's (1987) idea to decompose the system into a number of perspectives and focus areas has served as a theoretical basis for this framework. The framework defines a complete scheme to organize different level requirements, supports the flowdown of requirements from the business to the software level preserving requirements' business-orientation and contributes to the further development of the vision-driven requirements engineering techniques.

References

Blick, G. (2000). Defining business process requirements for large-scale public sector ERP implementations: A case study. In: *Proc. of the 8th Eur. Conf. on Information Systems* (*ECIS 2000*), Vienna, Austria, July 3–5. Accessible at: `http://is2.lse.ac.uk/asp/aspecis/20000156.pdf`.

Bray, I.K. (2002). *An Introduction on Requirements Engineering*. Addison-Wesley. An imprint of Pearson Education.

Caplinskas, A., Paskeviciute, L. (2009). A methodological framework for enterprise information system requirements derivation. In: Papadopoulos, G.A., Wojtkowski, W., Wojtkowski, W.G., Wrycza, S., Zupancic, J. (Eds), *Information Systems Development*: *Towards a Service Provision Society*, Springer-Verlag (in press).

Caplinskas, A., Lupeikiene, A., Vasilecas, O. (2002a). A framework to analyse and evaluate information systems specification languages. In: Manolopoulos, Y., Navrat, P. (Eds.), *Advances in Databases and Informa-*

tion Systems. *6th East European Conf.*, *ADBIS 2002*, Bratislava, Slovakia, September 2002. *Proc. LNCS*, Vol. 2435, Springer, pp. 248–262.

Caplinskas, A., Lupeikiene, A., Vasilecas, O. (2002b). Shared conceptualisation of business systems, information systems and supporting software. In: Haav, H.-M., Kalja, A. (Eds.), *Databases and Information Systems II: Selected Papers from the Fifth International Baltic Conference, BalticDB&IS'2002*. Kluwer Academic, pp. 109–320.

Caplinskas, A., Lupeikiene, A., Vasilecas, O. (2003). The role of ontologies in reusing domain and enterprise engineering assets. *Informatica*, 14(4), 455–470.

Davis, G.B., Olson, M.H. (1985). *Management Information Systems*: *Conceptual Foundations, Structure and Development*, 2nd ed. McGraw-Hill.

De la Vara, J.L., Díaz, J.S. (2007). Business process-driven requirements engineering: A goal-based approach. In: *Proc. of the 8th Workshop on Business Process Modeling, Development, and Support* (*BPMDS'07*), 11–15 June 2007, Trondheim, Norway. Accessible at:
`http:// lamswww.epfl.ch/conference/bpmds07/program/Gonzalez_23.pdf`.

Department of Defense (DoD) (1997). *Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance* (*C4ISR*) *Architecture Framework*. Version 2.0, 18 Dec. 1997.

Department of Defense (DoD) (2007). *DoD Architecture Framework*. Version 1.5, 23 April 2007:
Vol. I. *Definitions and Guidelines*. Accessible at:
`http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf`.
Vol. II. *Product Descriptions*. Accessible at:
`http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_II.pdf`.
Vol. III. *Architecture Data Description*. Accessible at:
`http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_III.pdf`.

Dumas, M., Aalst, W. van der, Hofstede, A. (2005). *Process-Aware Information Systems*: *Bridging People and Software Through Process Technology*. Wiley.

Ferrstl, O.K, Sinz, E.J. (2006). Modeling of business systems using SOM. In: Bernus, P., Mertins, K., Scmidt, G. (Eds.), *Handbook on Architectures of Information Systems*, 2nd ed. Springer-Verlag, pp. 347–368.

Frappier, M., St-Denis, R. (1998). A specication method for cleanroom's black box description. In: *Proc. of the Thirty-First Hawaii Intern. Conf. on System Sciences*, Kohala Coast, Hawaii, USA, January 6–9. Electronic ed., Vol. 6: *Organizational Systems and Technology*. IEEE Computer Society, pp. 112–121. Accessible at:
`http://computer.org/proceedings/hicss/8248/82480112abs.htm`.

Goikoetxea, A. (2007). *Enterprise Architectures and Digital Administration*: *Planning, Design and Assessment*. World Scientific.

Gomaa, H., Olimpiew, E.M. (2005). The role of use cases in requirements and analysis modeling. In: *Proc. of the 2nd Intern. Workshop on Use Case Modeling* (*WUsCaM-05*): *Use Cases in Model-Driven Software Engineering*, Montego Bay, Jamaica, October 2–7. Electronic ed. Accessible at:
`http://www.ie.inf.uc3m.es/wuscam-05/`.

Griffin, M. (2007). *System Engineering and the Two Cultures of Engineering*. Boeing Lecture at the Purdue University, 28 March 2007. Accessible at:
`http://www.spaceref.com/news/viewsr.html?pid=23775`.

Grotevant, S.M. (1998). Business engineering and process redesign in higher education: Art or science? In: *Online Proc. of EDUCAUSE Conf. on Information Technology in Higher Education* (*CAUSE 98*), December 8–11. Washington State Convention & Trade Center, Seattle, Washington. Accessible at:
`http://net.educause.edu/ir/library/html/cnc9857/cnc9857.html`.

Hay, D.C. (2002). *Requirements Analysis*: *From Business Views to Architecture*. Prentice-Hall.

Haskins, C. (2005). Application of patterns and pattern languages to systems engineering. In: *Proc. of the 15th Annual Intern. Symposium of the INCOSE on Syst. Eng.*: *Bridging Industry, Government,& Academia* (*INCOSE 2005*), Rochester, July 10–15.

Haskins, C. (2007). Using patterns to transition systems engineering from a technological to social context. *Syst. Eng.*, 11(2), 147–155.

Haskins, C. (2008). Using systems engineering to address socio-technical global challenges. In: *Proc. of the 6th Annual Conf. on Systems Engineering Research* (*CSER 2008*), Crowne Plaza Redondo Beach & Marina Hotel, Redondo Beach, April 4–5. Accessible at:
`http://www.gilb.com/tiki-download_file.php?fileId=235`.

IEEE (1998). Std 1233-1998: *IEEE Guide for Developing System Requirements Specifications*. Software Engineering Standards Committee of the IEEE Computer Society, USA.

Inmon, W.H, Zachman, J.A., Geiger, J.G. (1997). *Data Stores, Data Warehousing, and the Zachman Framework*: *Managing Enterprise Knowledge*. McGraw-Hill.

Jacobson, I., Christerson, M., Jonson, P., Overgaard, G. (1992). *Object-Oriented Software Engineering*: *A Use Case Driven Approach.* Addison-Wesley.

Lamsweerde, A. van (2001). Goal-oriented requirements engineering: A guided tour. In: *Proc. of the 5th IEEE Intern. Symposium on Requirements Engineering*, Toronto, Canada. Accessible at: `http://www.info.ucl.ac.be/Research/Publication/2001/RE01.pdf`.

Linger, R.C., Mills, H.D., Witt, B.I. (1979). *Structured Programming*: *Theory and Practice*. Addison-Wesley.

Malan, R., Bredemeyer, D. (1999). *Functional Requirements and Use Cases*. Bredemeyer Consulting. Accessible at: `http://www.bredemeyer.com/pdf_files/functreq.pdf` (last modified: July 25, 2006).

Martin, J. (1995). *The Great Transition, Using the Seven Disciplines of Enterprise Engineering to Align People, Technology and Strategy*. American Management Association.

Mayer, B. (1988). *Object-Oriented Software Construction*. Prentice-Hall.

Mayers, M., Kaposi, A. (2004). *A First Systems Book*: *Technology and Management*. Imperial College Press.

McLean, V.A. (2006). *The MITRE Corporation, NET Recruiter*. Accessible at: `http://www.netrecruiter.net/currentreqs.html`.

Moore, G.A. (1999). *Crossing the Chasm*: *Marketing and Selling High-Tech Products to Mainstream Customers*. Harper Business.

Office of Management and Budget (OMB) (1996). *Circular No. A-130. Revised. Transmittal Memorandum No. 4*. Accessible at: `http://www.whitehouse.gov/omb/circulars/a130/a130trans4.html`.

Oracle Corporation (OC) (2000). *CDM Quick Tour*. Release 2.0.0.

Prowell, S.J., Trammell, C.J., Linger, R.C., Poore, J.H. (1999). *Cleanroom Software Engineering – Technology and Process*. Addison-Wesley.

Roubtsova, E.E., Roubtsov, S.A. (2006). A feature computation tree model to specify requirements and reuse. In: *Proc. of the 8th Intern. Conf. on Enterprise Information Systems* (*ICEIS-2006*), Vol. 3, *Information Systems Analysis and Specification*, pp. 118–125.

Rukanova, B., van Slooten, K., Stegwee, R. (2006). Business process requirements, modeling technique, and standard: How to identify interoperability gaps on a process level. In: Konstantas, D., Bourrières, J.-P., Léonard, M., Boudjlida, N. (Eds.), *Interoperability of Enterprise Software and Applications*. Springer, pp. 13–24.

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

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

Strassmann, P.A. (1998). What is alignment? *Cutter IT Journal*, August 1998. Accessible at: `http://www.cutter.com/itjournal.html`.

The Open Ggroup (OG) (2002). *The Open Group Architecture Framework TOGAF*. Version 8.1.1, Enterprise Edition. Accessible at: `http://www.opengroup.org/architecture/togaf8-doc/arch/`.

Van den Broek, P., Galvão, I., Noppen, J. (2008). Elimination of constraints from feature trees. In: *Proc. of the 12th Intern. Software Product Line Conf., 12 Sept.*, Limerick, Ireland. Lero International Science Centre, University of Limerick, Ireland, pp. 227–232.

Warfield, J.N. (1994). *A Science of Generic Design*: *Managing Complexity Through Systems Design*. 2nd ed. Iowa State University Press.

Wegmann, A. (2004). On the systemic enterprise architecture methodology (SEAM). In: *Proc. of the Int. Conf. on Enterprise Information Systems* (*ICEIS 2003*), Angers, France.

Wegmann, A., Balabko, P., Le, L-S., Regev, G, Rychkova, I. (2005). A method and tool for business-IT alignment in enterprise architecture. In: Belo, O., Eder, J., Cunha J.F., Pastor, O. (Eds.), *Proc. of the 17th Conf. on Advanced Information Syst. Engineering* (*CAiSE '05*), Porto, Portugal, 13–17 June. *CAiSE Forum, Short Paper Proc.*, electronic ed. Accessible at: `http://www.informatik.uni-trier.de/~ley/db/conf/caise/caisefo2005.html#WegmannBLRR05`.

Wells, D. (2006). Modeling business metrics. Part 1: TDWI flash point. *Electronic Journal*, March 9. The Data Warehousing Institute. Accessible at:
http://www.tdwi.org/Publications/display.aspx?id=7884&t=y.
Wiegers, K.E. (2003). *Software Requirements*. 2nd ed., Microsoft Press.
Zachman, J.A. (1982). Business systems planning and business information control study: A comparison. *IBM Syst. J.*, 21(1), 31–53.
Zachman, J.A. (1987). A framework for information systems architecture. *IBM Syst. J.*, 26(3), 276–292.

A. Čaplinskas is a professor, principal researcher and the head of the Software Engineering Department at the Institute of Informatics and Mathematics, Vilnius, Lithuania. His main research interests include software engineering, information system engineering, legislative engineering, and knowledge-based systems.

Organizacijų integruotų informacinių sistemų reikalavimų formulavimas vadovaujantis tų organizacijų vizija

Albertas ČAPLINSKAS

Kuriant organizacijų integruotas informacines sistemas, programų sistemų inžinerijos procesai turi būti betarpiškai susieti su organizacijos strateginio planavimo ir informacinių sistemų inžinerijos procesais. Tradiciniai reikalavimų rinkimo būdai, grindžiami vartotojų bei užsakovų apklausos arba panašiomis technikomis, tam nėra pakankamai gerai pritaikyti, dėl ko dažnai pažeidžiama suformuluotų reikalavimų atitiktis realiems organizacijos poreikiams. To galima išvengti, jei reikalavimai yra išvedami iš organizacijos vizijos. Straipsnyje pasiūlytas metodinis karkasas, naudojant kurį galima vienareikšmiai klasifikuoti visų lygmenų reikalavimus ir per kelis tarpinius lygmenis organizacijos poreikius transformuoti į programinės įrangos reikalavimus užtikrinant pastarųjų atitiktį tiems poreikiams.