Architecture of Information Enterprises: Problems and Perspectives[*]

Konstantin Beznosov

April 20, 1998

Abstract

Current problems, constrains, goals, and approaches in developing architecture of information enterprises are reviewed. Research directions for solving the main problems of information enterprise architecture field are proposed.

1 Introduction

The problems caused by inadequate enterprise architectures, or even a lack of them most of the time, have become more and more topical. Periodic studies by Brancheau et al [1], [2] showed that during the period 1990-1994 the issue of building "responsive" information infrastructures moved to the top of the list of issues considered important by the chief information officers in US companies. According to Zachman [3], a recent IBM internal report on interviews with CEOs and CIOs of 108 of IBM's largest customers registered nearly universal frustration with the state of information availability for management purposes, as well as extreme frustration with the inability of the current information systems to respond to business enterprise changes.

In the field of information technologies (IT), the enterprise level represents supreme challenges for information departments and IT professionals whose responsibilities span enterprise scopes. The Chief Information Officer's job at this level is one of the most highly

[*]This paper was written for the "Advanced Topics in Software Engineering" seminar given by Dr. Michael Evangelist during spring of 1998 at School of Computer Science, Florida International University.

compensated, but it also experiences very high turnover. CIO positions typically turn over every two years, often due to burn-out and project failures [4].

Zachman [5] even names the problem of building good information enterprise architectures "the issue of the century," due to its complexity, growing importance, and the amount of resources that will be allocated to address it.

The software engineering community has been studying principles and patterns of software systems architecture and has been collecting positive knowledge about building such systems [6], [7], [8], [9]. As for architecture of information enterprises, there are much less public experience and systematic knowledge in this area. The main purpose of this paper is to review the area of information enterprise architecture, identify the problems, and propose ways for achieving progress.

We found that practitioners made more progress in developing enterprise architectures than researchers. They borrowed the practice of constructing complex systems from other manufacturing and construction industries, and they adopted it according to experience and available knowledge. Meanwhile, the research community took a systematic approach in the field of software system architecture, and many results of their work can be extrapolated into enterprise architectures. However, there are differences between a software system level and enterprise level architectures. Development of enterprise architecture definition languages, bridging the languages with work-flow description languages, and development of enterprise prototyping and modeling techniques can impact the progress in this field the most.

The paper is organized as follows. First we define what an information enterprise is. Then we survey issues in information enterprises identified by various sources and systemize them to separate the essential problems from the marginal ones. After we understand the field's problems, we define the concept of information enterprise architecture and analyze existing approaches in designing enterprise architectures by practitioners and researchers. Finally, we propose directions where research can go to solve the main problems.

In this paper, we will always mean "information enterprise" when we use the word "enterprise" unless otherwise stated.

2 Information Enterprise

We clarify the notion of an enterprise before discussing the problems it faces. An intuitive notion of an information enterprise tells us that it is a system of information systems. Such a notion, although correct, is not complete. If we have two computers connected through a network that run several different software programs, they comprise a “system of systems,” although it is not an enterprise yet. It is important to identify scope and scale when we refer to enterprises. Mowbray and Malveau [4] define an enterprise more precisely – “an organizational scope upon which a common set of information technology policies can be imposed.”

As it can be seen, they define enterprise level as the largest scale within an organization for the following hierarchy defined also by them: object, collection of objects, frameworks, application, system, enterprise, global scope. Enterprise level software comprises multiple systems, where each system comprises several applications. Zachman [5] places enterprise in the following scope hierarchy: program, system, application, department, *enterprise*, conglomerate enterprise, industry enterprises. Although these hierarchies do not match exactly, they are comparatively close to each other. Zachman’s hierarchy takes into account traditional division of a company into departments and the fact that federations of different companies might constitute a conglomerate. Mowbray and Malveau have finer than just *program* granularity. However, they concentrate on object oriented approach.

We combine advantages of both views, while slightly modifying them, and provide the following hierarchy: module, collection of modules, framework, program, application, system, department, enterprise, conglomerate enterprise, industry enterprises, and global infrastructure.

After defining what we mean by information enterprise and what its scope is in the hierarchy of software constructions, we identify enterprise goals and constraints. The former will allows us to understand liveliness properties of an enterprise – what we want eventually to happen. The latter will serve as a necessary limit on the freedom of proposed solutions.

2.1 Goals

Clearly, the main goal of an enterprise, as any other informational construction, is to satisfy its functional and non-functional requirements. For a software system, requirements are its input and output information, and the system's functional and non-functional properties. For an enterprise, the requirement is the business work-flow it is to support. Today, business work-flow changes more and more rapidly. DeBoever observes in [10] that the rate of change in business enterprises has grown from a full cycle period of approximately 7 years in 1970s and 1980s to 12-18 months in 1990s. He suggests the main goal for information enterprises is not trying to align the enterprise with the business work-flow, but "to have such an enterprise that will allow quick re-aligning when the business work-flow changes."

Another important goal for an enterprise is to allow the gradual migration towards new technologies with the retirement of old ones as well as evolution of systems constituting the enterprise.

We define a *well constructed* information enterprise as one that fully supports business work-flow and allows sufficiently quick re-aligning, according to the work-flow changes, while requiring only a reasonable amount of resources to maintain and manage the enterprise. In each case the notion of *quick* and *reasonable* have to be determined.

2.2 Constraints

We found that constrains on enterprise level fall into the following groups:

Amount and nature of change [4], [10] At the enterprise level, change is aggregated (from many system-specific changes). At this level, change appears frequent and continuous. The nature of change itself is different. The rate of change in business processes, especially in administrative work-flow, is often faster than the rate an information enterprise can accommodate those changes. Due to frequent work-flow changes, demand in enterprise capacity is a dynamic factor.

Different business work-flow processes have different change rates. For example, *manufacturing*, *financial* and *human resource* work-flows change comparatively slow. Meanwhile, *customer support*, *supply chain*, and *decision support* change fast.

Architecture development [11], [10] There is an upper limit to the size of a chunk of enterprise modeling that can be tackled at one time. Due to the significant amount of money invested in any enterprise, the existing information infrastructure has to be reused – no “cold turkey” solutions.

Organizational restructuring [12] Such broad organizational change initiatives as process re-engineering, quality management, down-sizing, employee improvement, and transition to flexible and adaptable forms of organizations introduce additional constraints on how an enterprise is built.

The main constraints are the amount and the nature of change on the enterprise level, and the necessity to reuse existing information infrastructure.

3 Enterprise Architecture

In this section, we discuss the notion of enterprise architecture and how it is different from an information system architecture. According to the dictionary definition [13], *architecture* is “the art or practice of designing and building structures.” The word was found originally in the English of 16th century. First, we proceed with defining the meaning of architecture in the software system context.

There is no consensus on the definition of software system architecture in the literature. Here are only some samples: a set of design artifacts, or descriptive representations, that are relevant for describing a system such that it can be produced to requirements as well as maintained over the period of its useful life [5]; ”a description of the subsystems and components of a software system and the relationships between them” [9]; abstract view of a system as a configuration of components and connectors [14]; and “architecture of a software system defines that system in terms of computational components and interactions among those components” [8]. Software Engineering Institute even has a WWW page[1] with a discussion and various definitions of “software architecture.”

[1]http://www.sei.cmu.edu/˜architecture/definitions.html

The most precise and useful definition of software system architecture, which will be used for this paper, was given by Garlan and Perry [6]:

> "The structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time."

Now, we discuss the notion of an enterprise architecture.

Mowbray and Malveau [4] define enterprise architecture as "a set of rules, guidelines, interfaces, and conventions used to define how applications communicate and interoperate with one another." They add that the main architect's concern is defining a robust set of abstractions that manage complexity, change, and other forces. In other sources [15], [16], Mowbray describes an enterprise architecture as "a set of diagrams, text, and tables that define a system or family of systems from various stake-holder viewpoints." He suggests that each view of the architecture addresses the issues posed by principal stake-holders, such as end-users, developers, systems operators, and technical specialists.

Zachman [5] views an enterprise architecture as a "set of descriptive representations (i.e. "models") that are relevant for describing an enterprise such that it can be produced to management's requirements (quality) and maintained over the period of its useful life (change)."

DeBoever[10] defines enterprise architecture as a set of principles which guide design, selection, construction, implementation, deployment, support, and management of information infrastructure.

Darnton and Giacoletto [17] define an information enterprise architecture as "a framework for agreement that provides the art and science of identifying, planning, and implementing integrated information systems and their related infrastructure, and that provides activities of an organization or enterprise having certain desirable properties."

As it can be seen from the diverse definitions above, enterprise architecture is defined these days more broadly and less precisely than a software system architecture (we could even say "very vaguely"). Such looseness and diversity is mostly due to the facts that enterprise architecture is a field in its infancy and that no scientific research methodology has been applied to this area.

3.1 Differences Between System and Enterprise Architecture

Someone might suggest to apply principles and techniques of developing software system architecture directly to produce an enterprise architecture, since we can sometimes consider an enterprise simply as a system of information systems.

However, information enterprise architecture has its own specifics and differences from system architecture. The main differences are:

- The difference in requirements for a system and for an enterprise discussed in section 2.1. We believe it is the key to the difference between a system architecture and an enterprise architecture. A requirement for software system architecture can define precisely its behavior, input/output, and its properties. Whereas an enterprise architecture has its requirements in the form of company work-flow, and an architect has to produce requirement specifications for each of the enterprise systems.

- The focus of concern in enterprise architecture is moved from single system structure, its performance, and decomposition, to the structure of the whole enterprise, change management, and decomposition of it into services and resources that support the work-flow.

- Another difference is the level of abstraction. For an enterprise architecture to have reasonable complexity in order to avoid problems of delivering an architecture, which is already out of date due to rapid work-flow changes and the lengthly design time, languages of expressing the architecture have to be of a higher abstract level than for system architecture.

As it is discussed later in this paper, we recommend to extrapolate some approaches to software system architecture and apply them to enterprise architecture due to similarities between system and enterprise architectures. However, we believe different techniques are needed in order to succeed in developing enterprise architectures.

4 Problems with Enterprises and their Architectures

Architecture is responsible for the final success or failure of an information enterprise.

The main problem today's enterprises face is the fact that most of them are built in ad hoc fashion. As von Halle observes [18], quick and dirty information systems projects are the rule rather than the exception for enterprises. Academia can help to address this problem only by educating and popularizing ideas of developing enterprises with a well defined process. In the rest of this section, we will concentrate mainly on technical problems of constructing well built enterprises.

We will describe two groups of problems. The first group is consequences of actual causes, which is the second group.

4.1 Consequences

1. When an enterprise is built out of systems, piece by piece and product by product, it does not work as it is expected because the built systems do not fit together well [5], [11].

2. Maintenance of systems' relevance to a company work-flow in an information enterprise is time-consuming and expensive [5].

3. The costs and time of maintenance approximate exponential increases with an increase in the number of systems deployed at an enterprise [5].

4. Enterprise-wide modeling, as opposed to single system modeling, takes too long. By the time it is completed, the result is usually discredited and out of date [11].

Problem 2 is featured in software systems as well. On the enterprise level, maintenance difficulties are aggregated from all systems constituting an enterprise, and they become one of the major obstacles for IT professionals as Zachman [5] points out. The other three problems are not experienced in software system development.

4.2 Causes

Our research suggests that the following are the actual causes of enterprise problems.

1. Resources and changes are accumulated from each system and become orders of magnitude more than for any single system, which leads to unmanageable enterprises [4].

2. Lack of ability to consistently and meaningfully describe, analyze, change, and evaluate an enterprise structure as well as to communicate and accumulate positive knowledge about building well constructed enterprises [3]. This leads to enterprise architectures that are made in ad hoc hand-crafted fashion, are informal, unanalyzable, unmaintainable and not reusable.

3. *Architectural mismatch* is described by Garlan et al in [14]. If we project their mismatch taxonomy onto the enterprise level, we have the following main reasons for architectural mismatch:

   (a) assumptions about the nature of the systems constituting an enterprise;

   (b) assumptions about the nature of the connectors between systems;

   (c) assumptions about the global architectural structure – about topology of the enterprise communications and about the presence or absence of particular components and connectors;

   (d) assumptions about the construction process.

   Even though the notion of architectural mismatch in software engineering field was first identified for information systems, not for enterprises, clearly architectural mismatch prevents deploying "well constructed" enterprises.

4. We believe some problems with today's practice of documenting software systems architectures identified by Shaw and Garlan in [8] are also projected on enterprise architectures:

   (a) inability to localize information about interactions,

   (b) poor abstraction,

   (c) lack of structure on interface definitions,

(d) poor support for components with incompatible packaging,

(e) poor support for multi-paradigm systems,

(f) poor support for legacy systems.

In the next two sections, we describe current approaches in practice and research of software engineering to developing enterprise architectures and analyze them from the perspective of these main causes.

5 The State of the Practice

We found two major approaches in developing enterprise architectures. They are Zachman's framework and the reference model for open distributed computing recommended by ISO and ITU.

5.1 Zachman's Framework

The approach most popular among IT practitioners is the information systems architecture (ISA) framework introduced by John Zachman in 1987 in [19] and enhanced five years later together with Sowa in [20].

5.1.1 Description

The essence of the framework is based on the premise derived from observations of some manufacturing industries (particularly airplane business) that there are three "fundamental" architectural representations in manufacturing and construction works, one for each ”player in the game": the owner, the designer, and the builder. Those representations correspond to the few main views in the framework that are supposed to capture an enterprise architecture from different perspectives in such a way that it would be sufficient for every main "player" involved in the enterprise construction to work on their own level of abstractions and concepts. Such representations for an enterprise and the corresponding "players" are:

1. Scope/objectives – planner,

Presentation \ Aspect	DATA *what*	FUNCTION *how*	NETWORK *where*	PEOPLE *who*	TIME *when*	MOTIVATIONS *why*
Scope/objective						
Business model						
Model of the information system						
Technology model						
Detailed representations						

Table 1: Zachman's Information Systems Architecture Framework

2. Model of business – owner,
3. Model of the information system – designer,
4. Technology model – builder,
5. Detailed representations – sub-contractor.

The framework considers six aspects that include information processing and work-flow:

1. *What* – data organization,
2. *How* – control flow,
3. *Where* – systems location,
4. *Who* – people and departments involved,
5. *When* – duration of the business processes,
6. *Why* – motivation of the enterprise via objectives and strategies.

The proposed framework is a place-holder for a set of representations/views allocated in the two-dimensional matrix of representations (scope, business model, etc.) and "aspects" (what, how, etc.) as depicted in table 1. Each cell is supposed to be filled out with a particular view according to the meaning of the corresponding row and column it is in. For example, the cell corresponding to the *business model* row and *data* column is supposed to be filled with the data entity-relationship logical model.

Different techniques and different graphic representations are appropriate for different cells. The composite or integration of all cell models in one row constitutes a complete model of the enterprise from the perspective of that row.

Extensions to the framework by Sowa and Zachman described in [20], besides adding the last three columns, propose to use *conceptual graphs*[2] for describing data and control flow aspects. They claim that conceptual graphs are general enough to represent any cell in the framework and the relationships among cells. No other literature about the framework and its usage showed that conceptual graphs have been used in practice for representing models for the framework cells.

Sowa and Zachman state in [20] that the framework is useful for segmenting the descriptions of the enterprise: for separating independent variables into understandable, designable components; for developing appropriate design formalisms; and for establishing an enterprise infrastructure in which change can be assimilated in a manageable fashion. The proposed framework is promised to be helpful for setting up a common ground for enterprise architecture view so that all those who are involved in the enterprise construction can communicate with each other. Sowa and Zachman even claim that "the framework serves as a 'periodic table' for information entities."

5.1.2 Analysis

The main contribution of the framework is explicit decomposition of an enterprise architecture into distinct defined views. For an average[3] IT professional, who has to be involved in the development of enterprise architecture and later in construction and maintenance of the enterprise, such a matrix of views with simple labels (e.g. "business model of data" or "technology model of network") is much better than no "recipes" at all. The framework gives provisions to describe and analyze (to some degree) an enterprise architecture, as well as to accumulate and communicate positive knowledge about building enterprises to others. However, since the approach does not state how various views are specified, it is not

[2]Conceptual graphs [21] are a graphic notation for typed first-order logic, which has applied domains including natural language processing, information systems modelling, program specification, information retrieval, machine learning and case based reasoning.

[3]We regard those IT professionals who do not have a formal computer science background but have several years experience of work in IT industry as average.

clear if the knowledge accumulated and communicated to others will be of high quality and usefulness.

Another significant merit of the framework is explicit recognition of business work-flow. We consider last three columns (people, time, motivations) as a way of naturally including work-flow description into the enterprise architecture. As we discussed in section 2.1, an enterprise requirement is support of business work-flow. Including work-flow into enterprise architecture can help architects to follow work-flow requirements.

The main disadvantage of Zachman's approach is the lack of a scientific foundation on top of which the framework is constructed. Sowa and Zachman admit in [20] that the framework "is not so much an invention as it is an observation – an observation of some natural rules for segmenting an enterprise into understandable parts without losing the definition of its total integration." For this approach to be developed further, we need to understand the laws and principles that govern those "natural rules" in order not only to observe them but also to discover new rules and to be in a position to explain them.

Other important steps needed in order to make the framework more useful are techniques for specifying each of those 30 views precisely. Potential directions for developing the framework in this respect are defining exact techniques for specifying each view. For instance, instead of re-inventing work-flow descriptions for the last three columns, the author should take advantage of already accumulated[4] knowledge and experience with work-flow and process management in information systems in research and practicing community.

Neither does it provide a means to derive a view in one cell from a view in another cell. The framework approach lacks methods that would ensure consistency and absence of conflicts among different views.

In summary, Zachman's framework proposes to decompose enterprise architecture into various views, and it addresses the issue of an enterprise structure description as well as of communication and accumulation of positive knowledge about building enterprises. The approach lacks scientific foundation. The framework does not address most of the problems for enterprise architectures, particularly architectural mismatch, change management, and

[4]See, for example, WORP – a repository of resources for researchers and practioners in work-flow and process management in information systems at http://holbrooke.cs.uga.edu/worp/.

various problems pointed by Shaw and Garlan and discussed in section 4.2.

5.2 RM-ODP

Most parts of the Reference Model of Open Distributed Processing (RM-ODP) recommendations [22, 23, 24, 25] were completed in the mid-1990's by International Standards Organization (ISO) jointly with International Telecommunications Unit (ITU) (formerly CCITT). New revisions and additions to the RM-ODP are work in progress, and they are supposed to be completed by September, 1998 [26].

5.2.1 Description

The goal of the reference model is to provide a common, well defined language of terminology and notations for specifying functional and non-functional properties of a distributed system and its environment, that is information enterprise.

The RM-ODP defines the following [22]:

- a division of an ODP system specification into viewpoints, in order to simplify the description of complex systems;
- a set of general concepts for the expression of those viewpoint specifications;
- a model for an infrastructure supporting, through the provision of distribution transparencies, the general concepts that it offers as specification tools;
- principles for assessing conformance for ODP systems.

The separation of concerns is established by identification of five viewpoints:

1. enterprise – the role of the system in the business,
2. information – use and interpretation of information,
3. computational – description of the system as a set of objects that interact at interfaces,
4. engineering – mechanisms supporting system distribution (networked computing infrastructure that supports the system),

5. technology – the detail of the components (hardware, software) from which the distributed system is constructed.

The viewpoints should be considered as projections onto certain sets of concerns rather than layers of design methods. The viewpoints are not independent. The same entities can be represented in more than one viewpoint.

Object modeling was chosen because it seemed most suitable for ODP. Objects and actions are the most basic modeling concepts in the reference model. All things of interest are *objects*[5]. Anything of interest that can happen is *action*. Other concepts are *roles*, *contracts*, *contexts*, *liaisons*, *templates*, object *instantiation* and object *introduction*. The reference model defines a set of modeling concepts:

- Basic concepts – object-based model.
- Specification concepts which allow their user to describe and reason about ODP system specifications. They include notions of type and class to reason about properties of objects.
- Concepts of organization, of the properties of systems and objects, of policy, of naming, etc.

A set of concepts provides a language for writing specifications of systems from that viewpoint, and such a specification constitutes a model of a system in terms of the concepts. The reference model provides a means to define and specify different types of transparencies: access, failure, location, migration, relocation, replication, persistence, and transaction.

The recommendations introduce the notion of enterprise language, which is a contract, linking the performers of various roles and expressing their mutual obligations. Different notations for enterprise specification can be expected to support specific organizational structures and business practices, but architecturally, the ODP is neutral, requiring only that an appropriate specification be generated. The enterprise language introduces basic concepts necessary to represent an ODP system in the context of the enterprise in which it

[5]Object is a model of any concrete or abstract thing of interest, according to the RM-ODP terminology.

operates. The system is represented by one or more objects within a community of enterprise objects that are present at the enterprise and by the roles in which these objects are involved.

The RM-ODP recommendations define responsibilities of languages for all five views an ODP system is supposed to be specified. Here are descriptions of some view languages:

Information language contains concepts to enable the specification of the meaning of information manipulated by and stored within an ODP system. Information held and manipulated by an ODP system is represented either by atomic or composite information objects and possible relationships over a set of information objects. The information specification consists of a set of related schemata. A distinction is made between invariant (always true relationships), static (assertions that must be true at a single point in time) and dynamic (how information can evolve, and deletion/addition of information objects) schemata. Different information specification model notations can be used as long as they represent concepts required by the recommendations.

Computational language enables a specifier to express constraints on the distribution of an application: an object model that defines the form of interface an object can have, the way that interface can be bound, and the forms of interaction which can happen. Computational interfaces are characterized by a signature, a behavior, and an environmental contract – a contract between an interface and its environment.

Engineering language supports the notion of clusters of objects, capsules (conventional notion of processes) and nodes, as well as stubs, binders and protocol objects.

The recommendations define in [24] a first order type system that consists of a simple type language together with type equality rules and signature sub-typing rules. Further, they define signature interface types using the typing system as well as an algorithm for type checking.

In the part on architectural semantics [25], the recommendation discusses how the formal description language LOTOS [27] and/or formal language Z may be used to formalize the concepts and rules of the computational viewpoint language. It also discusses how the specification and description language (SDL), version 92 [28], may be used to formalize the concepts and rules of the information viewpoint language. The intent to describe usage of formal languages for specifying other viewpoints is stated.

For every terming used in the RM-ODP recommendations, there is either a definition in natural, and sometimes, formal languages, or a reference to another recommendation where it is defined. This ensures use of predefined terminology and concepts.

The recommendations include a basic framework for conformance to RM-ODP, and define the form conformance statements should take in each viewpoint. Standard ODP-compliant specifications are obliged to contain a conformance statement that must state which reference points, according to [29], should be used during the conformance testing.

The reference model is known and used much better in Europe and Australia than in North America.

5.2.2 Analysis

The RM-ODP was apparently influenced by Zachman's framework, object orientation, and formal methods.

The main contribution of the ODP reference model in the work of enterprise architects and system developers is a provision of standard definitions on most of the concepts and terminology. Another significant merit is a definition of most concepts required for specifying distributed information systems and interfaces between them and their environment – information enterprise. This addresses the issue of common notations and languages for describing and communicating enterprise architecture. When formal languages for describing all five views of the RM-ODP model are identified, practicing architects will be able to specify precisely systems constituting an enterprise and their properties. Such languages are already available from the formal methods community. We hope that, when various viewpoints and particular languages used for those viewpoints are standardized, practicing architects will be able to communicate enterprise architecture and specifications for its components to system vendors and other sub-contractors involved in the enterprise development and maintenance. The RM-ODP is one such a step towards standardized specification languages, terminology, viewpoints, and concepts.

The reference model addresses to some or other degree the following issues from the list of enterprise architecture problems discussed in section 4.2:

- Architectural mismatch:
  - Assumptions about the nature of the systems constituting an enterprise – a specification language for each viewpoint is required to specify constituting systems in such a manner that the systems are considered as black boxes with explicitly listed interfaces and their syntactic and semantical definitions, and systems' properties.
  - assumptions about the global architectural structure – the global architectural structure is supposed to be explicitly described for each viewpoint in the corresponding specification language.
- Other architectural problems:
  - Inability to localize information about interactions – all interactions between enterprise systems are explicit and defined for each interaction instance.
  - Poor abstraction – formal languages used in viewpoint descriptions to abstract properties of specified systems. However, we believe this level of abstraction is not sufficient for specifying properties of an enterprise as a whole.
  - Lack of structure on interface definitions – the reference model requires a definition for every interface exposed to enterprise environment by the means of computational language.

The RM-ODP recommendations do not address other problems of enterprise architecture, particularly: management of change, specifications of the connectors between systems, definition of the construction process, support for components with incompatible packaging, support for multi-paradigm systems, and support for legacy systems.

There is no automatic routine or formal algorithm for translation from one viewpoint language into another one. Neither are there any transformation algorithms for transforming one specification written in one language into another specification written in another language.

In summary, the ODP reference model is a good starting point towards standardized notations and languages for describing enterprises and constituting systems. It needs to be augmented with a complete set of formal languages for describing each of the viewpoints. Especially important is a language for enterprise viewpoint that can be used to describe enterprise requirements – company work-flow. Abstractions suitable for the enterprise level have to be defined in the model. Pointed out issues not addressed by the reference model should be worked out in order to make it suitable for describing architecture of information enterprises and their systems.

6 The State of Research

Most of the research in software architecture is concentrated on systems, and not on enterprises. So far, only practitioners made noticeable efforts in understanding how the architecture of information enterprises should be developed. In this section, we will discuss those achievements in software architecture listed in [6] that, we believe, have applications in enterprise architecture.

Architecture description languages: ADLs are developed to provide expressive notations for representing architectural designs and styles [30]. When enough knowledge and experience about enterprises and ADLs are accumulated, enterprise-oriented ADLs will help to express and communicate enterprise architectures, and to address the issue of implicit assumptions about information systems on the enterprise and their environment.

Formal underpinnings of software architecture: Researchers in this area are working on formal models of architectures and mathematical foundations for architectural styles. This direction can potentially contribute the most to enterprise architecture issues by helping to develop formal foundations of approaches similar to Zachman's framework and ODP reference model.

Architecture analysis techniques: These techniques are aimed to determine and predict properties of software system architectures. Application of such techniques in enterprise architectures will help to avoid expensive mistakes and to analyze enterprise models.

Architecture recovery and re-engineering: It is needed for extracting architecture of existing enterprises in order to reconstruct them. This direction is important for understanding what enterprise architectures are successful and for learning from already built enterprises.

Architectural codification and guidance: The work on extracting positive experience and codifying expertise in enterprise architectures is a must as in any other software engineering field. Some work has already been done by the software patterns community [4]. More involvement of researchers from knowledge engineering will help in effectively extracting and managing the experience.

Tools and environments for architectural design: After languages and notations are developed, the practicing community will need a working environment for enterprise architects.

Case studies: One of the efficient ways to popularize ideas of enterprise architecture and to educate IT professionals, as well as to share experiences, is to publish case studies of enterprise architectural design. More and more precedents of case studies occur in the system architecture field [31], [32]. Equivalent work is needed for enterprise architectures.

7 Proposed Directions

In the previous two sections, we have discussed the state of the practice in developing enterprise architectures and the directions of software architecture research that have applications in the architecture of information enterprises. Now, we propose particular ways to achieve impact in this area.

Taking into account the current problems an enterprise architecture has, and the state of the practice and research, it is imperative to develop a set of enterprise architecture definition languages (EADL). Such languages will allow practitioners and researchers to do the following:

- precisely describe architecture of an enterprise capturing functional/nonfunctional properties from different perspectives,
- accumulate, codify and communicate common knowledge and experience about designing enterprises,
- directly derive specifications for the systems to be deployed on the enterprise,
- provide a common ground for formal analysis of enterprise architectures,
- allow for codifying solutions for the architectural mismatches and other common problems featured in enterprise architectures.

A set of EADLs might include several languages. Each of them will be good for a particular enterprise view or aspect. The EADLs from such a set should: be precise and expressive, have notation easy to understand by practicing IT professionals, allow the expression of multiple views of an enterprise, allow to code multiple aspects including the main requirement for any enterprise – work-flow, allow to insure consistency and absence of conflicts between different views and aspects, and have an appropriate level of abstraction.

Having such a set of EADLs and using either Zachman's framework or ODP reference model (provided that problems pointed out with these approaches are resolved), enterprise architects will have a powerful methodology for tackling enterprise architecture. But just a methodology is not sufficient for success. Practicing architects need enabling technology for efficient work. They need tools and architecture development systems.

Another direction towards solving enterprise problems is to learn how to derive information enterprise requirements (even better, architecture) directly from a company's business model represented by its work-flow. For this, we want to have languages to describe business work-flow. Once such languages exist, the next step is to provide techniques to insure that a particular enterprise described in the set of views using previously described EADLs supports a given work-flow, which constitutes the enterprise requirements.

Yet another important direction is to develop a means of prototyping and modeling an enterprise before implementing it to avoid costly mistakes in requirements of analysis and design. Again, once a set of EADLs exist, such modeling will be feasible.

8 Conclusions

Information enterprise architecture is similar to software system architecture although there are differences in the form of requirements, the focus of concern, and the level of abstraction. The main causes of the problems with enterprise architectures are the lack of efficient solutions to manage changes accumulated across an enterprise, the lack of an efficient and precise way to describe, analyze, and communicate the architecture, architectural mismatch, poor abstraction, and poor support for legacy, component-based and multi-paradigm systems.

So far, practitioners made more progress in developing the area of enterprise architecture than researchers. However, working in the similar area – software system architecture, the research community took a more systematic and better founded approach. Most of the directions in software system architecture can be extrapolated into the enterprise architecture field.

The R&D directions that can highly impact the enterprise architecture progress are as follows:

- development of enterprise architecture definition languages,
- bridging work-flow languages and EADLs so that precisely and completely described work-flow can be used as requirements for an enterprise architecture,
- development of enterprise prototyping and modeling techniques.

References

[1] F. Neiderman, J. Brancheau, and J. Wetherbe, “Information systems management issues for the 1990s,” *MIS Quarterly*, vol. 15, pp. 475–502, December 1991.

[2] J. C. Brancheau, B. Janz, and J. Wetherbe, “Key issues in information systems management: A shift toward technology infrastructure,” *MIS Quarterly*, 1995.

[3] B. Greene, D. McDavid, and J. Zachman, “Back to the issue of the century,” *Database Programming and Design*, pp. 8–9, June 1997.

[4] T. Mowbray and R. Malveau, *CORBA Design Patterns*. John Wiley & Sons, 1997. ISBN 0-471-15882-8.

[5] J. A. Zachman, “Enterprise architecture: The issue of the century,” *Database Programming and Design*, pp. 44–53, March 1997. also at http://www.zifa.com/zifajz01.htm.

[6] D. Garlan and D. Perry, “Introduction to the special issue on software architecture,” *IEEE Transactions on Software Engineering*, April 1995.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[8] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, May 1996.

[9] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, 1996. ISBN 0 471 95869 7.

[10] L. R. DeBoever, “Concept of ”highly adaptive” enterprise architecture.” Enterprise Architecture Conference keynote address, December 1997.

[11] M. Fowler, *Analysis patterns: reusable object models*. Addison Wesley Longman, 1997.

[12] W. D. Nance, “Growing pains in information systems: transforming the IS organization for client/server development,” in *Reinventing IS managing information technology in changing organizations*, pp. 78 – 86, 1994.

[13] F. C. Mish, ed., *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster, ten ed., 1994.

[14] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch or why it's hard to build systems out of existing parts," in *Proceedings of the Seventeenth International Conference on Software Engineering*, April 1995.

[15] T. J. Mowbray, "What is architecture?," *Object Magazine*, pp. 20–22, July 1997.

[16] T. J. Mowbray, "Architecture in the large," *Object Magazine*, pp. 24–26, December 1997.

[17] G. Darnton and S. Giacoletto, *Information in the Enterprise: It's More than Technology*. Bedford, MA: Digital Equipment Corporation, 1992.

[18] B. von Halle, "Architecting in a virtual world," *Database Programming and Design*, pp. 13–18, November 1996.

[19] J. A. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 26, no. 3, pp. 276–292, 1987. IBM Publication G321-5298. 914-945-3836 or 914-945-2018 fax.

[20] J. Sowa and J. A. Zachman, "Extending and formalizing the framework for information systems architecture," *IBM Systems Journal*, vol. 31, no. 3, pp. 590–616, 1992. IBM Publication G321-5488. 914-945-3836.

[21] J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984.

[22] International Telecommunication Union, Telecommunication Standardization Sector, *Recommendation X.901: Information technology – Open Distributed Processing – Reference model: Overview and guide to use*, May 1995.

[23] International Telecommunication Union, Telecommunication Standardization Sector, *Recommendation X.902: Information technology – Open Distributed Processing – Reference model: Foundations*, May 1995.

[24] International Telecommunication Union, Telecommunication Standardization Sector, *Recommendation X.903: Information technology – Open Distributed Processing – Reference model: Architecture*, May 1995.

[25] International Telecommunication Union, Telecommunication Standardization Sector, *Recommendation X.904: Information technology - Open Distributed Processing – Reference model: Architectural Semantics*, May 1995.

[26] International Telecommunication Union, Telecommunication Standardization Sector, *Status of X-series Recommendations*, Januar 1998. http://www.itu.int/itudoc/itu-t/rec/x/xseries_25636.html.

[27] International Standards Organization, *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*, 1989. ISO 8807:1989.

[28] International Telecommunication Union, Telecommunication Standardization Sector, *CCITT Specification and description language (SDL)*, March 1993.

[29] International Standards Organization, *Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 4: Test realization*, 1991. ISO/IEC 9646-4.

[30] M. Shaw and D. Garlan, "Characteristics of higher-level languages for software architecture," tech. rep., CMU Software Engineering Institute, December 1994. CMU-CS-94-210.

[31] A. W. Brown, D. J. Carney, and P. C. Clements, "A case study in assessing the maintainability of a large, software-intensive system," in *Proceedings of the International Symposium on Software Engineering of Computer Based Systems*, IEEE Computer Society, March 1995.

[32] A. Brown, D. Carney, P. Clements, C. Meyers, D. Smith, N. Weiderman, and B. Wood, "Assessing the quality of large, software intensive systems: A case study," in *Proceedings of European Conference on Software Engineering*, September 1995.