FLORIDA INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE

TECHNICAL REPORT

2000-06

INFORMATION ENTERPRISE ARCHITECTURES: PROBLEMS AND PERSPECTIVES

Konstantin Beznosov
beznosov@cs.fiu.edu
http://cs.fiu.edu/~beznosov

June 14, 2000

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.

Keywords

Information enterprise architecture, software architecture, software engineering.

Information Enterprise Architectures: Problems and Perspectives

Konstantin Beznosov
beznosov@cs.fiu.edu

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 in most cases a lack of them, have become more and more topical. Periodic studies by Brancheau et al [1, 2] showed that during the period 1990-1994 the problem of building "responsive" information infrastructures moved to the top of US chief information officers' (CIO) list of issues. According to Zachman [3], an 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 and 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 CIO's job at this level is one of the most highly 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 IEA "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 the principles and patterns of software systems architecture and has been collecting positive knowledge about building such systems [6-9]. As for the architecture of information enterprises, there is much less public experience and systematic knowledge in this area. This report reviews the area of IEA, identifies the problems, and proposes ways for achieving progress. The problem of IEA will serve as a large context of the work on engineering application-level access control in enterprise distributed applications. It shows why any distributed application functionality should always be considered from the perspective of the ever-changing enterprise environment.

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 adapted it according to their experience and available knowledge. Meanwhile, the research community took a systematic approach in the field of software system architecture, and many of the their results can be extrapolated into IEAs. However, there are differences between architectures at the software system level and the enterprise level. 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 report is organized as follows. In the next section we define what an information enterprise is. The definition of the IEA concept follows in Section 3. Then, in Section 4, we survey the issues in IEAs identified by various sources and systemize them to separate the essential problems from the marginal ones. Afterwards, we analyze existing approaches in designing IEAs by practitioners (Section 5) and researchers (Section 6). Finally, we propose directions where research can go to solve the main problems in Section 7. In this thesis, we will always mean "information enterprise" when we use the word "enterprise" unless otherwise stated.

2. INFORMATION ENTERPRISE

We must clarify the notion of an enterprise before discussing the problems it presents. An intuitive perception of an IE tells us that it is a system of information systems. Such a description, 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 can be seen, they define enterprise level as the largest scale within an organization according to their scope hierarchy: object, collection of objects, frameworks, application, system, enterprise, global scope. Enterprise level software is comprised of multiple systems, where several applications constitute each system. Zachman [5] places enterprise in the following scope hierarchy: program, system, application, department, enterprise, conglomerate enterprise, industry enterprises. 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. Although these hierarchies do not match exactly, they are relatively close to each other.

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

After defining an IE and its scope in the hierarchy of software constructs, we identify enterprise goals and constraints. The former will allow 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 for 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 [10] that the rate of change in business enterprises has grown from a full cycle period of approximately 7 years in the 1970s and 1980s to 12-18 months in the 1990s. He suggests the main goal for IEs is not only to align the enterprise with the business work-flow but also "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 the evolution of systems constituting the enterprise. We define a *well constructed* IE 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* has to be determined.

2.2. Constraints

Architectural choices in constructing an IE are limited by the imposed constraints. We found that the constraints 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 to be 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 with which an IE can accommodate those changes. The demand in enterprise capacity is a dynamic factor due to frequent work-flow changes. Different business work-flow processes have different change rates. For example, manufacturing, financial and human resource work-flows change at a comparatively slow rate. Meanwhile, customer support, supply chain, and decision support change more quickly.

- **Architecture development** [10, 11]: 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 resources 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 organizational forms introduce additional constraints on how an enterprise is built.

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

3. ENTERPRISE ARCHITECTURE

After defining the concept of an IE and discussing its goals and constraints, we proceed to define the notion of enterprise architecture. First, it is necessary to agree on exactly what architecture is. According to the dictionary [13], *architecture* is "the art or practice of designing and building structures."[1] The next step is to define the meaning of architecture in the context of a software system in order to see how it is different from IEA.

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 something that "defines that system in terms of computational components and interactions among those components" [8]. Software Engineering Institute even has a WWW page[2] with a discussion and various definitions of the term "software architecture."

The most precise and useful definition of software system architecture, which will be used for this thesis, 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."

The meaning of IEA also varies. 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 architect's main 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 a 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, system operators, and technical specialists. Zachman [5] views an IEA 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 IEA as a set of principles which guide design, selection, construction, implementation, deployment, support, and management of information infrastructure.

Darnton and Giacoletto [17] define an IEA 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 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

1. The word was found originally in the English of 16th century.
2. http://www.sei.cmu.edu/~architecture/definitions.html

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 applying principles and techniques of developing software system architecture directly to enterprise architectures, 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. There are several main differences system and enterprise architecture:

- The requirements for a system and for an enterprise are different as discussed in Section 2.1. We believe this 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. In contrast, an enterprise architecture has its requirements in the form of company work-flow, and an architect has to produce specifications for each of the enterprise systems according to the IEA.
- The focus of concern in IEA is moved from single system structure, its performance, and decomposition, to the structure of the whole enterprise, change management, and its decomposition into services and systems that support the work-flow.
- Another difference is the level of abstraction. For an IEA to have reasonable complexity in order to avoid delivering an architecture which is already out of date due to rapid work-flow changes and the lengthy design time, the IEA languages have to be of a higher abstract level than system architecture languages.

As discussed later in this report, we recommend extrapolating some approaches to software system architecture and applying them to IEA due to the similarities between them. However, different techniques are needed in order to succeed in developing IEAs.

4. Problems with Enterprises and Their Architectures

Architecture is responsible for the final success or failure of an IE. The main problems enterprises face today are due to the fact that most of them are built in ad hoc fashion. As von Halle observes, quick and dirty information systems projects are the rule rather than the exception for enterprises [18]. The academic community can help to address this problem by, for example, 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

There are four major problems with IE reported in the literature:

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

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

3. The increase of costs and time for maintenance is approximately exponential to the increase in the number of applications deployed at the 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, 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 construction.

4.2. Causes

Our research suggests that the following are the main causes of the problems by IE owners:

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. The 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 IEAs that are made in ad hoc fashion, are informal, un-analyzable, un-maintainable and not reusable.

3. Architectural mismatch is described by Garlan et al [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 the software engineering field was first identified for information systems, clearly architectural mismatch prevents the deployment of “well constructed” enterprises.

4. We believe some problems with today’s practice of documenting software system architectures identified by Shaw and Garlan [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 the development of IEAs and analyze them from the perspective of these main causes.

5. 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 International Standards Organization (ISO) and International Telecommunications Unit (ITU).[1]

5.1. Zachman's Framework

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

Description

The framework is essentially based on the premise derived from observations of some manufacturing industries (particularly airplane business). 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 that capture an enterprise architecture from different perspectives. This should be done in such a way that it would be sufficient for every main "player" to work on their own level of abstractions and concepts. Such "players" and the corresponding representations are:

1. *Planner* -- scope/objectives
2. *Owner* -- model of business
3. *Designer* -- model of the information system
4. *Builder* -- technology model
5. *Sub-contractor* -- detailed representations

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

1. ITU used to have another name -- CCITT.

The proposed framework is a place-holder for a set of views allocated in the two-dimensional matrix of representations (scope, business model, etc.) and "aspects" (what, how, etc.) as depicted in Table 1. Each

Presentation/Aspect / Scope/objective	Data (what)	Function (how)	Network (where)	People (who)	Time (when)	Motivations (why)
Scope/Objective Model (planner)						
Business Model (owner)	data entity-relationship logical model					
Model of Information System (designer)						
Technology Model (builder)						
Detailed Representations (sub-contractor)						

Table 1: Zachman's Information Systems Architecture Framework

cell is filled out with a particular view according to the meaning of the corresponding row and column. 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.

The use *conceptual graphs*[1] for describing data and control flow aspects is the main contribution of the extensions to the framework by Sowa and Zachman [20]. 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 has showed that the graphs have been used to represent models for the framework cells.

Sowa and Zachman state [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 promises to be helpful for setting up a common ground for IEA view so that all 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."

Analysis

The main contribution of the framework is the explicit decomposition of an enterprise architecture into distinctly defined views. For an average[2] IT professional involved in the design, 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 IEA. It also allows the accumulation and communication of pos-

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

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

itive knowledge about building enterprises to others. However, since the approach does not state how various views are specified, it is not clear if such knowledge will be of high quality and usefulness.

Another significant merit of the framework is explicit recognition of business workflow. We consider the 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, the main requirement for any IE is the support for business work-flow. Including work-flow into IEA can help architects 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 [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 disadvantages are a lack of a means to derive a view in one cell from a view in another cell and methods that would ensure consistency and absence of conflicts among different views.

Other important steps needed in order to make the framework more useful are techniques for specifying each of those 30 views precisely. A potential direction for developing the framework in this respect is to define exact techniques for specifying each view. For instance, instead of re-inventing work-flow descriptions for the last three columns, the framework author might take advantage of already accumulated[1] knowledge and experience with work-flow and process management.

In summary, Zachman's framework proposes to decompose enterprise architecture into various views. It addresses the issue of an enterprise structure description as well as of the communication and accumulation of positive knowledge about building enterprises. The approach lacks a scientific foundation. The framework does not address most of the problems for IEAs, particularly architectural mismatch, change management, and various problems pointed by Shaw and Garlan and discussed in Section 4.2.

5.2. RM-ODP

Another approach (more popular in Europe and Australia than in the US) used by practitioners is the reference model of open distributed processing (RM-ODP) recommendations [22-25]. Most of their parts were completed in the mid-1990's by ISO jointly with ITU. Revisions and additions to the RM-ODP were made at the end of 1998 [26].

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 IE.

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 the general concepts that it offers as specification tools; and the principles for assessing conformance for ODP systems.

The separation of concerns is established by the 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 via interfaces,

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

4. engineering -- mechanisms supporting system distribution (i.e. networked computing infrastructure that supports the system),

5. technology -- the details 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 design layers. Thus it is wrong to assume that enterprise viewpoint is layered on top of information one, which is layered on top of computational viewpoint and so on. Furthermore, the viewpoints are not independent. The same entities can be represented in more than one viewpoint. For example, a system function of providing access to the information is presented at the following viewpoints: enterprise -- by defining its users; information -- by specifying data it accepts and returns; computational -- by defining its syntax and semantics as well as dependencies on other functions and its pre/post-conditions.

Object modeling was chosen as a technique for defining RM-ODP viewpoints 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*. Anything of interest that can happen is an *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, properties of systems, objects, policies, naming, etc.

A set of concepts provides a language for writing specifications of systems from that viewpoint. 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. Architecturally, however, the ODP is neutral, requiring only that an appropriate specification is generated. The enterprise language introduces basic concepts necessary to represent an ODP system in the context of the enterprise in which it operates. The system is represented by one or more objects and their roles.

The RM-ODP recommendations define the responsibilities of languages for all five views for specifying an ODP system. Here are descriptions of three view languages:

1. **Information language** contains concepts enabling an architect to specify the meaning of information manipulated by and stored within an ODP system. The information is represented either by atomic or composite information objects and possible relationships over a set of these 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 specification model notations (for example Unified Modeling Language (UML) [27, 28]) can be used as long as they represent concepts required by the recommendations.

2. **Computational language** enables the expression of constraints on the application distribution: an object model that defines an object interface, the way that interface can be bound, and the possible forms of interaction. Computational interfaces are characterized by a signature, behavior, and an environmental contract -- a contract between an interface and its environment.

3. **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 [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 type system as well as an algorithm for type checking.

In the section on architectural semantics [25], the recommendation discusses how the formal languages LOTOS [29] and/or Z [30] may be used to formalize the concepts and rules of the computational viewpoint language. It also discusses how the specification and description language (SDL) [31] may be used to formalize the concepts and rules of the information viewpoint language. The intent to describe the usage of formal languages for specifying other viewpoints is stated.

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

The recommendations include a basic framework for conformance. They also define the form the conformance statements should take in each viewpoint. Standard ODP-compliant specifications are obligated to contain a conformance statement that must document which reference points, according to [32], should be used during the conformance testing.

Analysis

The RM-ODP was apparently influenced by Zachman's framework, object orientation, and formal methods. The main contribution of the ODP reference model to the work on IEA and enterprise application development is a provision of standard definitions of most concepts and terminology. Another significant merit is the definition of the concepts required for specifying distributed information systems and interfaces between them and their environment -- IE. These contributions allow architects to use common notations and languages for describing and communicating IEA. When formal languages for describing all five views 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 viewpoint languages are standardized, IE architects will be able to communicate enterprise architecture and specifications for its components to system vendors and other sub-contractors involved in enterprise development and maintenance. The RM-ODP is a step towards standardized specification languages, terminology, viewpoints, and concepts.

The reference model addresses to some degree the following issues from the list of IEA 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 define the systems in such a manner that they are considered as black boxes with defined properties and interfaces.
  - Assumptions about the global architectural structure: the structure is supposed to be explicitly described for each viewpoint in the corresponding 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: properties of specified systems are abstracted via formal languages used in viewpoint descriptions. 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 the 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, as well as the support for components with incompatible packaging, multi-paradigm systems, and

legacy systems. There is no algorithm for translation from one viewpoint language into another. Neither are there any algorithms for transforming a specification written in one language into into one in another language.

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 enterprise language that can be used to describe enterprise requirements -- company work-flow. Abstractions suitable for the enterprise level have to be defined in the model. Issues not addressed by the reference model should be worked out in order to make it suitable for describing IEAs.

6. THE STATE OF RESEARCH

Most of the research in software architecture is concentrated on systems, and not on enterprises. So far, only practitioners have made noticeable efforts to understand how the architecture of information enterprises should be developed. In this section, we will discuss those achievements in software architecture that have applications in IEA.

Architecture description languages (ADL): ADLs are developed to provide expressive notations for representing architectural designs and styles [33]. When enough knowledge and experience about enterprises are accumulated, enterprise-oriented ADLs will help to express and communicate enterprise architectures, and to avoid implicit assumptions about information systems 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 the resolution of IEA issues by developing formal foundations for approaches similar to Zachman's framework and the ODP reference model.

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

Architecture recovery and re-engineering: These processes are needed for extracting the 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 engineering field. Some work on IEA codification and guidance has already been done by the software patterns community [4]. More involvement of researchers from knowledge engineering will help in effectively extracting and recording the experience.

Tools and environments for architectural design: After languages and notations are developed, the practicing community will need environments for enterprise architects to work. The environments should have tools for architecture analysis, recovery and re-engineering, codification, and guidance, as well as for IEA description and documentation.

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 case studies have been reported in the system architecture field [34, 35]. Equivalent work is needed for IEA.

7. PROPOSED DIRECTIONS

We have discussed the state of the practice in developing IEAs and the directions of software architecture research applicable to the IEA problem domain. Now, we propose particular ways to make a positive impact in this area.

Taking into account the current issues with IEAs, 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 and nonfunctional properties from different perspectives,
- accumulate, codify and communicate common knowledge and experience about designing enterprises,
- directly derive specifications for the enterprise systems,
- provide a common ground for formal analysis of IEAs,
- allow for codifying solutions to architectural mismatches and other common IEA problems.

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 for practicing IT professionals to understand; allow the expression of multiple views of an enterprise and the coding of multiple aspects including the main requirement -- work-flow support; assure the consistency of representation; avoid 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 the ODP reference model, enterprise architects will have a powerful methodology for tackling IEAs. But just a methodology is not sufficient for the success. Practicing architects need enabling technology for efficient work. They need architecture development environments and tools.

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. Some efforts are already emerging among researchers [36, 37]. Once such languages are developed and become mature, the next step is to provide techniques to ensure that a particular enterprise supports a given work-flow. That is, there should be ways of checking if an IEA supports its requirements.

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

8. SUMMARY AND CONCLUSIONS

Information enterprise architecture is similar to software system architecture although there are differences in the form of requirements, the focus of concerns, and the level of abstraction. The major problems encountered in the IEA construction are low semantic compatibility of resulted systems, high re-alignment and maintenance cost, and its exponential increase to the increase in the number of deployed applications, enterprise modeling takes too long and becomes outdated too soon. The main causes of the problems 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 have made more progress in the IEA area than researchers. However, working in the similar field of software system architecture, the research community took a more systematic and better founded approach. Most of the directions in this field can be applied to the problem domain of IEA.

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 a precisely and completely described work-flow process can be used as the main requirement for an enterprise architecture,
- development of enterprise prototyping and modeling techniques.

In this report we defined the notion of information enterprise architecture, described the general issues of IEAs and reviewed existing solutions, as well as discussed prospects of addressing the issues.

9. REFERENCES

[1] F. Neiderman, J. Brancheau, and J. Wetherbe, "Information Systems Management Issues for the 1990s," *Management of Information Systems Quarterly*, vol. 15(December), pp. 475-502, 1991.

[2] J. C. Brancheau, B. Janz, and J. Wetherbe, "Key Issues in Information Systems Management: A Shift Toward Technology Infrastructure," *Management of Information Systems Quarterly*, 1995.

[3] B. Greene, D. McDavid, and J. Zachman, "Back to the Issue of the Century," *Database Programming and Design*(June), pp. 8-9, 1997.

[4] T. J. Mowbray and R. C. Malveau, *CORBA Design Patterns*. New York: Wiley Computer Publishing, 1997.

[5] J. A. Zachman, "Enterprise Architecture: The Issue of the Century," *Database Programming and Design*), pp. 44-53, 1997.

[6] D. Garlan and D. Perry, "Introduction to the Special Issue on Software Architecture," *IEEE Transactions on Software Engineering*, vol. 21(4), pp. 269-274, 1995.

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

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

[10] L. R. DeBoever, "Concept of "Highly Adaptive" Enterprise Architecture," In Proceedings of Enterprise Architecture Conference, 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*, 1994, pp. 78 - 86.

[13] Merriam-Webster, "Merriam-Webster's Collegiate Dictionary," , F. C. Mish, Ed.: Merriam-Webster, 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 Proceedings of the Seventeenth International Conference on Software Engineering, 1995.

[15] T. J. Mowbray, “What is Architecture?,” *Object Magazine*), pp. 20-22, 1997.

[16] T. J. Mowbray, “Architecture in the Large,” *Object Magazine*), pp. 24-26, 1997.

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

[18] B. v. Halle, “Architecting in a Virtual World,” *Database Programming and Design*(November), pp. 13-18, 1996.

[19] J. A. Zachman, “A Framework for Information Systems Architecture,” *IBM Systems Journal*, vol. 26(3), pp. 276-292, 1987.

[20] J. F. Sowa and J. A. Zachman, “Extending and Formalizing the Framework for Information Systems Architecture,” *IBM Systems Journal*, vol. 31(3), pp. 590-616, 1992.

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

[22] I. T. U. ITO, “Recommendation X.901: Information technology -- Open Distributed Processing -- Reference model: Overview and Guide to Use,” : International Telecommunication Union, 1995.

[23] I. T. U. ITO, “Recommendation X.902: Information technology -- Open Distributed Processing -- Reference model: Foundations,” : International Telecommunication Union, 1995.

[24] I. T. U. ITO, “Recommendation X.903: Information technology -- Open Distributed Processing -- Reference model: Architecture,” : International Telecommunication Union, 1995.

[25] I. T. U. ITO, “Recommendation X.904: Information technology -- Open Distributed Processing -- Reference model: Architectural Semantics,” : International Telecommunication Union, 1995.

[26] I. T. U. ITO, “Status of X-series Recommendations,” : International Telecommunication Union, 1998.

[27] X. Blanc, M.-P. Gervais, and R. Le-Delliou, “Using the UML Language to Express the ODP Enterprise Concepts,” In Proceedings of The 3rd International Conference on Enterprise Distributed Object Computing, 1999.

[28] P. F. Linington, “Options for Expressing ODP Enterprise Communities and Their Policies by Using UML,” In Proceedings of The 3rd International Conference on Enterprise Distributed Object Computing, 1999.

[29] I. S. O. ISO, “Information processing systems -- Open Systems Interconnection -- LOTOS -- A formal description technique based on the temporal ordering of observational behaviour,” : International Standards Organization, 1989.

[30] S. Valentine, “The Programming Language Z,” *Information and Software Technology*, vol. 37(5-6), pp. 293-302, 1995.

[31] I. T. U. ITU, “CCITT Specification and description language (SDL),” : International Telecommunication Union, 1993.

[32] I. S. O. ISO, “Information technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 4: Test Realization,” : International Standards Organization, 1991.

[33] M. Shaw and D. Garlan, “Characteristics of Higher-level Languages for Software Architecture,” CMU Software Engineering Institute CMU-CS-94-210, December 1994.

[34] 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 International Symposium on Software Engineering of Computer Based Systems, 1995.

[35] 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, 1995.

[36] P. Makey, “Workflow: Integrating the Enterprise,” Butler Group June 1996.

[37] W. Du, S. Peterson, and M.-C. Shan, “Enterprise Workflow Architecture,” In Proceedings of The 11th International Conference on Data Engineering, Los Alamitos, CA, USA, 1995, pp. 63-64.