

# From Software Engineering to Enterprise Engineering – Introduction to a Language-critical Approach

Erich Ortner  
Technische Universität Darmstadt  
Development of Application Systems  
64289 Darmstadt, Germany  
Ortner@winf.tu-darmstadt.de

**Abstract**– With service-oriented architectures and the organization-centric development of application systems, we are experiencing the beginning of a reorientation in how information technology is used in the global economy as well as in our private lives. This trend has already impacted on the “System and Application Development” branch of study at colleges and universities. Language-critical Organization Theory has thus become a discipline in research and teaching. In the years to come, this shift of emphasis will broaden present endeavors and originate new interdisciplinary ones in research and development. Because of the global nature of these endeavors, it is important to look for a common basis for possible cooperation partners in this field. The necessary efforts are enormous and can only be managed successfully by working together. Looking at how the philosophy of science has developed throughout the twentieth century seems to imply that a) the primarily American analytic philosophy of science paired with b) the methodical constructivism of predominantly German origin (Erlangen and Constance) could form this common basis. In this paper, both philosophies will form the foundation substantiating practical development in the field of service-oriented architecture. The common foundation will also disclose what people must be able to do or, respectively, what they must know (understand) if they want to work successfully in the IT-industry in the future.

## I. INTRODUCTION

According to Aristotle (384-322 BC), *Architectonics* refers to the art and science of building, while the term *architecture* denotes the structure. In civil engineering, the proportion of “art” within the structuring activity is usually drawn upon to distinguish an architect (emphasis on art) from an engineer (emphasis on methodology).

For vendors and users of information technology, the term *service-oriented architecture* (SOA) has been an issue for about four years. Organization-centric Requirements Engineering (on a constructivistic basis) with reference to Applied Computer Science was proposed for the first time in 1980 [1]. The following observations are important to this concept today:

- *Organizational processes*, which, to some extent, offer a free choice (e.g. specific knowledge gained from

experience) to the acting persons in particular steps of an occurrence are principally to be distinguished from the inherently different algorithmic computer processes (software and hardware). Therefore, organizational processes ought to be specified further by *language-critical organization theory* [2].

- Ubiquitous (= found everywhere) computer technology leads to the fact that presently almost any object (thing or occurrence) can be a medium in this technology.
- *Common languages*, (e.g. “ontologies”, conceptual schema, ortho-languages) whether spoken by people or used in technology, always serve as a means for integration. This includes, to some extent, the integration of heterogeneous elements in a system or architecture.
- Today, the *meta-language* in application systems – the showpiece of system informatics – is firmly established within the overall architecture by means of repositories. For example, repositories are used to enable and facilitate the management of component-based solutions.
- In the field of constructive languages and consequently in research methods of various application fields, there has been considerable progress in the past few years ascribed to the *Unified Modeling Language (UML)*, whose development is still ongoing.
- There are predominantly three factors, which have instigated the necessity to start *organizing work globally*. These are: The fact that Enterprise Organization Theory is a part of Applied Computer Science, the comprehensive concept of application systems as *service-oriented architectures*, and the development of new technologies on the internet (Web 2.0), for example interactive applications [3]. It is essential to approach this organizational task from the position of dynamically organized enterprise networks that interact globally.

In the following, we will describe how the ProCEM® method (Process-Centric Enterprise Modeling & Management) meets the above-mentioned requirements, taking into

consideration the human needs. The basis for our description is the experience gained from accompanying the project “Best Process Architecture”, a contribution to the BITKOM college competition 2007, the seminar “SOA in Accountancy” held in the summer term of the same year, and the lecture organization-centric “Development of Application Systems”, which is an ongoing lecture at TU Darmstadt as of 1996.

Since the problems to be solved in the upcoming years are of an interdisciplinary nature and will have a strong and global impact on the working world, the last section of this paper will include an appeal for global cooperation.

## II. ORGANIZATION-CENTRIC DEVELOPMENT OF APPLICATION SYSTEMS

The iterative orchestration of application systems (see fig. 1) is one particular feature of service-oriented architectures in the following aspects:

- The optimized work processes,
- the ideal assignment of employees, and
- the dynamic use of information technology using services.

Services, or more precisely service schemas, are application software that implements work procedures. They are developed on the basis of components and specified as algorithms.

Rather than having oboes, violins or triangles at ones disposal, the orchestration of application systems (see fig. 1) makes use of human beings, organizational structures and technology. Whereby anyone who specifies e.g. organizational processes in the same way as computer processes [4] and does not distinguish between human-related symbol processing and computer-based symbol processing [5], is not suitable for the development of organization-centric application systems. Such a person lacks the interdisciplinary knowledge taught by some of the forward-looking chairs of Applied Computing and Application Systems at universities worldwide.

## III. INTERDISCIPLINARY LANGUAGE-CRITICAL SPECIFICATION OF IT-USE

Which skills and what kind of knowledge do developers, i.e. “business architects,” “application developers,” “solution architects,” and so on, need for organization-centric application development in order to successfully participate in projects of this kind or even execute such a project on their own? In his book “Der Flug der Eule” (The flight of the owl, [6]), Jürgen Mittelstraß gives us an answer that is as clearly defined as it is simple:

“Anyone [...] who has not studied interdisciplinary cannot perform interdisciplinary research”.

The acquisition of interdisciplinary schemas and the understanding of them is a prerequisite for interdisciplinarity. Anyone who has only studied how to apply something will not be able to develop organization-centric application systems. The following is a simple example that illustrates the profound

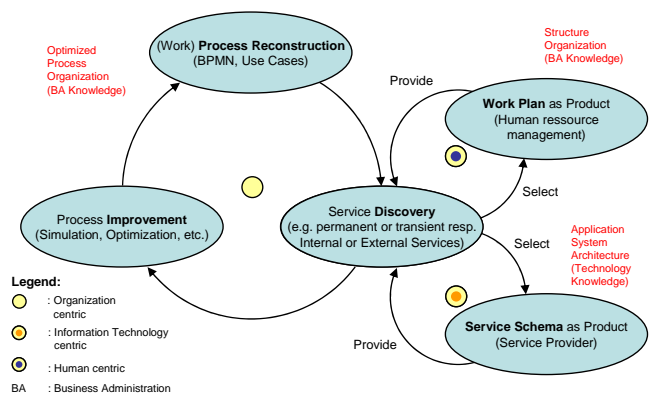


Fig. 1. Iteration paths of orchestration: man - organization – technology

understanding of the development process. The example reconstructs a data schema in Applied Computer Science.

Schematize the following sentence in object-language,

a) “Smith is a customer who is willing to pay.”

by means of computer sciences, specifically the meta-language “relational model”:

b) “Relation Name (key attribute(s); non-key attributes)”

in an interdisciplinary way, i.e. by different disciplines simultaneously. In order to accomplish this, a developer must not solely understand the sentence in object-language a) with respect to its business-driven generalization (schematization, norm), but additionally, the user must have agreed on the norm derived from it. In our example, this would mean that a society (language community) tolerates the following object-language norm:

a’) “If we identify a person as a customer, we are allowed to characterize the customer more closely by the attribute ‘payment behavior’.”

Furthermore, it is necessary to realize or understand that the “relational model” is merely a different grammar (meta-schema) for representing the standardized (schematized) object-language “content”. It is our goal to maintain customer data efficiently on the computer. Through modeling, we achieve the logical result of our interdisciplinary schematization:

b’) “Customer (name; payment behavior)”.

Interdisciplinary schematization (modeling) is one of the core tasks in Applied Computer Science such as Business Informatics. For the acquisition of interdisciplinary knowledge, e.g. in university courses of study, there is even a so-called “methodical order.” We can formulate it as follows, whereby the figures in brackets indicate the “sequence”, i.e., the methodical order.

Computer Science: form (4) follows function (3)

Business Informatics: applications (3) follow

processes (2)

Business and Social Sciences: means (2) follow ends (1)

Theoretically, the methodical order, or course can be avoided. However, in practice, it is recommended to adhere to it. It is most advisable to “put on the socks before putting on the shoes”, although, at least in theory, it may be possible to consider the reversed order. The problem in some of the programs of study in Computing Sciences is that interdisciplinary knowledge is not taught – even at the recently appointed German superior universities, an apparent lack in IT-architects has lead to the fact that students in bachelor programs of study merely concern themselves with pure computer sciences, i.e. (3) and (4). For those students who have not entered a practical profession by then, the Masters program of study will “ensure that they are acquainted with matters of Applied Computer Science” [7]. Well, it is conceivable that disciplines become extinct!

Modeling is language-based. This is true on the part of object-languages, users, and ends, as well as on the part of meta-languages and the means of computer sciences. The linguistic fundamentals of computer science for modeling have recently been published in [8]. Wittgenstein ([9], § 132) describes the language-critical, practical aspect very well in the following statement:

“The confusions which occupy us arise when language is like an engine idling, not when it is doing work.”

Language works when we are talking with users in the relevant application fields about the development-relevant topics while they are working. The famous linguist Karl Bühler (1879-1963) called this “empractical”. Due to their gift for languages, female developers will typically have a natural advantage over their male colleagues in this context. Clearly, interdisciplinarity is much more of a decisive factor for successful projects than mere speaking will be. The author’s practical experience leads him to three principles of language-critical development of application systems [10] that are emphasized in the following:

1. Ordinary language is alright. (Ludwig Wittgenstein)
2. Languages have already made a great many, sometimes too many, distinctions. (Paul Lorenzen)
3. We will be held responsible for what we are saying. (Peter Janich)

The first principle shows why development work is practically relevant, while the second emphasizes “engineering” and the foundation of the results with a rational language. The third principle clarifies that the organization-centric development of application systems is interdisciplinary and together with the ends, that is with the ethics as a “critical examination of practical reason” (Kant, 1724-1804), they form the beginning. If we want to evaluate this work, e.g. for later quality management, we must do this one level higher, on a meta-meta-language level. Therefore, we can speak of

transdisciplinary, ethic-political education (meta-knowledge) of managers. Both, universities and the industry, or society respectively, are open to a new field of vocational training. However, this will only become possible if researchers and teachers are proficient in their respective fields in an interdisciplinary as well as a transdisciplinary way.

### A. Organization Modeling

For successful organization modeling – especially with respect to optimization – differentiation is vitally important. Figure 2 illustrates the possibilities regarding work processes and structures.

The capability to differentiate clearly is critical to the ability to optimize. This is important for the object-language level, the application field, as well as for the meta-language level, the diagram languages or, another grammar. On both levels, the point is the reconstruction of connector words (e.g. to do) and topic words (e.g. to work). On the meta-language level, the developer gets to know the modeling method in greater detail. On the object-language level, the grammar of the modeling language plays a vital role. In the latter case, the organizational expert knowledge of the relevant application domain must be represented in a structured way.

Modeling (topic words of the application field) and structuring (connector words of the modeling language) are different but they supplement each other as complementary parts. We do not approve of the so-called formalistic (only syntax) approaches, as they do not occur in practical use. Even if only one modeling method or one diagram language (e.g. UML) is considered, we are not formalistic but meta-linguistic and use general schematic letters, so-called syntactical variables, for our modeling field (the “content”).

In the meta-proposition

“With one proper name  $n$  and a single-figure predicator  $P$ , the elementary proposition  $n \varepsilon P$  is built.“

the letters “ $n$ ” and “ $P$ ” are syntactical variables. Hereby, “ $n \varepsilon P$ ” is to be understood in such a way that the copula “ $\varepsilon$ ” (read: is a) is a sign of the object language, that forms a link between a particular proper name (e.g. Peter) and a particular predicator (e.g. customer). The copula links object-language “contents” but does not appear between syntactical variables on the meta-

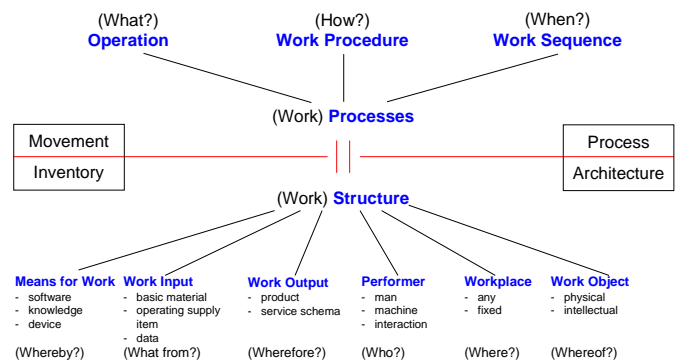


Fig. 2. Differentiated work organization from the position of process and structure

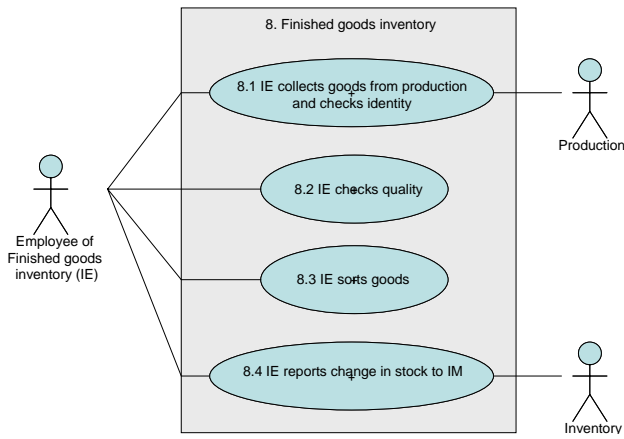


Fig. 3. Use case diagram: Finished goods inventory

language-level.

Even before the object-oriented system design, diagram languages have proved to be suitable modeling languages for the organization of a company's structures and processes. Use cases for example are especially useful for the structural aspect (see fig. 3), while the Business Process Modeling Notation (BPMN) is suited ideally for the procedural aspect (see fig. 4).

Detailed descriptions of modeling languages can be found in various case collections [11] or OMG manuals. However, anyone who later, in the system design, intends to specify the flow of work processes in greater detail is well advised to distinguish the aspects like "operations", "work procedures" and "sequence of work" or "workflows" orthogonally. The same applies to structural organization and aspects such as

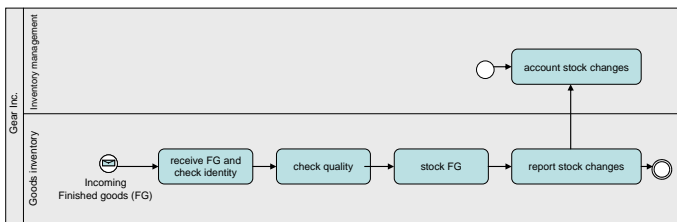


Fig. 4. BPMN diagram: Stock finished goods

"workplace", "vacancy", "employee" or "work material" (see fig. 2). The optimization can now be considered sensibly and from different angles (aspects).

### B. Method-Neutral Knowledge Reconstruction

The method-neutral knowledge reconstruction [12] is primarily communicative and hardly any diagram representations are used.

In order to get a first picture of the important tasks, which are performed in close cooperation with the users, we classify them roughly in the following three parts:

- Assessment of propositions that are relevant for development by talking to the users.

| Object         | STRUCTURE      |                     | PROCESS |
|----------------|----------------|---------------------|---------|
|                | Thing oriented | Occurrence oriented |         |
| INTERNAL       | a)             | b)                  | c)      |
| EXTERNAL       | e)             | f)                  | g)      |
| Constraints d) |                |                     |         |

Fig. 5. Classification schema for development-relevant propositions

- Clarification and reconstruction of the expert terminology that has been used.
- Establishment of a common enterprise expert language.

The assessment of propositions relevant to the development can be done by using a model, as shown in figure 5.

The model is intended to aid the assessment and to ensure that all potential types of results for the system design have been scrutinized in consideration of their underlying expert knowledge. The following list contains several propositions that can be assigned to the above fields (see fig. 5):

- An account has an account number.
- An account is opened.
- Opening an account results in an opening balance.
- The total of all debit line items must be the same as the total of all credit line items in double entry accounting.
- A (personal) account is assigned to a business partner.
- Shipment of goods is related to posting business transactions.
- At the end of an accounting period, all of the accounts are closed, their values are entered in a profit and loss account and ultimately gathered in the balance.

In the clarification and reconstruction of the identified expert terminology the following "defects" are discussed and examined thoroughly with the future users or the company's experts.

- Checking synonyms**  
Check for words with the same meaning (extension and intension) that can be interchanged.  
e.g.: MEMBER and ASSOCIATE have the same meaning for DATEV<sup>1</sup>.
- Eliminating homonyms**  
Check for words that are written or pronounced in the same way but have a different meaning.

<sup>1</sup> DATEV is a computer center and software house for the German-speaking tax profession where the author worked as executive manager in software development for seven years.

e.g.: STALK, which can mean either part of a plant or to follow someone around

3. Identifying equipollences

Different names are used for the same objects (extension) from different perspectives (intension).

e.g.: Goods or merchandise of a company is referred to as STOCK from a quantitative perspective and INVENTORY ACCOUNT from a value perspective.

4. Clarifying vagueness

As there is no clear delimitation (definition) of the terms in regard to their content (intension), it may not be clear which objects belong to each term (scope, extension)

e.g.: Does RESIDENCE, the place where a CONSULTANT works, belong to the term CHAMBERS for DATEV or not?

5. Replacing wrong designators

Discrepancies between the actual meaning of a word and the meaning assumed at first (intension and extension)

e.g.: For DATEV, the CONSULTANT NUMBER does not define the function of a tax CONSULTANT, but it defines the USER RIGHTS a tax CONSULTANT has within DATEV.

This clarification results in further propositions relevant for development. Their relevance for the result types (system design) can be examined with the help of a classification schema (see fig. 5). Work on building a common expert language for a company, which is aimed at integrating all of a company's knowledge resources, can be organized in different ways.

1. With the help of a repository, a kind of glossary will be created and administered. This glossary will contain all the terms that are important for an organization (language community), and should be designed for internal and external use.
2. A much more complex way, in comparison to (1.), of representing a company's knowledge is with an encyclopedia. The encyclopedia amounts to a conceptual schema for data but will go substantially further in respect to terminological coherences. This approach will distinguish inward and outward knowledge, which will be administered in a repository as an enterprise knowledge base.
3. The enterprise expert language is a rational interim language that is implemented on a meta-meta language level in the repository [13]. It is used for integrating and translating other languages used in the utilized in a company. For users, it is not necessary to know the expert language itself.

Currently, the three variants discussed above can be found in industry worldwide. Vendor-independent research is done in the field of SOA under the catchword *Enterprise Application Integration* (EAI). Furthermore, companies like Oracle look into *Application Integration Architecture* (AIA) and offer products such as Fusion. Other vendors offer products like WebSphere (IBM) or NetWeaver (SAP).

C. Entirely Object-oriented System Design

After the development-relevant knowledge has been reconstructed neutral to specific methods and technology (e.g. according to [12]), and integrated into the overall knowledge base of an enterprise using common language, then, in the system design, this knowledge is transformed into the result types of an object-oriented solution to the task. Figure 6 shows an object-oriented system design according to Schienmann [14] that has been extended for the design of service-oriented architectures.

When we speak of entirely object-oriented development of application systems, the enlightening step is the introduction of objects from computing sciences as grammatical objects. Grammatical objects are target points of language actions (e.g. writing, speaking, thinking), whereby they can also be replaced by pronouns in the sentence (e.g. one, he, him, this one). At school we have learned to speak of direct and indirect objects, genitive objects and various prepositional objects.

In contradiction to what many computer scientists still believe, when modeling and programming in computer science we do not concern ourselves with concrete or "ontological objects" such as this chair, that apple or my laptop. When speaking of objects "informatically" (i.e. modeling and programming in a meta-language), it is of particular importance that abstract types such as "class" in a repository or the term "invoice" in an application, are to be thought of as target points. The concrete, "ontological objects" are usually found in the application fields.

Computer science is the science where students learn how to talk constructively about language (grammatical) objects, or more precisely, about abstract objects. Needless to say, we can still start from the concrete objects of the application fields in "Requirements Engineering" and when introducing the implemented solution, we can refer back to the users' concrete (ontological) objects.

The object-oriented approach in the development of application systems goes back to Platon. Platon classifies

| Result Type | Inventory                       | Procedure                            | Process                          |
|-------------|---------------------------------|--------------------------------------|----------------------------------|
| Internal    | Conceptual Schema               | Service Application (Procedure Part) | Organization of Work Occurrences |
| External    | Service Application (Data Part) | Participation                        |                                  |

Restrictions

Fig. 6. Extended object oriented system design acc. to [14]

objects from the perspective of human beings and their languages into things (nouns, proper names) and actions, which can also be considered occurrences (verbs). If we transfer this classification to operating with data on a computer, the object-orientation (resp. its object) will be classified into the fields of data orientation (things) and procedure orientation (occurrences). This classification shows why object-orientation is universal. It encompasses data orientation (data classes) as well as procedure orientation (procedural classes).

Based on the results of organization modeling and system design, the results from figure 6 are modeled in the following order:

1. Process modeling:
  - BPMN diagrams (from organization modeling)
  - State machine diagrams
  - Activity diagrams
  - ...
  - Constraints (e.g. in Object Constraint Language (OCL))
2. Participation modeling:
  - Use cases (from organization modeling)
  - Sequence diagrams
  - Job descriptions (in the sense of structural organization)
  - ...
  - Constraints (e.g. organizational standards such as signature regulations)
3. Procedure modeling:
  - Class diagrams (data classes and procedural classes)
  - State machine diagrams
  - Activity diagrams
  - ...
  - Constraints (e.g. plausibility checks at data entry in service-oriented applications)
4. Inventory modeling:
  - Object type diagrams (for the conceptual schema)
  - Dataflow diagrams (for specification of data that are exchanged)
  - External schemas (extended as data classes)
  - ...
  - Constraints (e.g. semantic integrity rules for DBMS-enforced integrity)

Organization modeling and the reconstruction of development-relevant expert knowledge from the application fields are highly communicative processes. Here, users and developers communicate very “intensively” (in great detail and clearly) with each other. Diagram languages play a minor role in this context. In contrast, the (entirely) object-oriented design of a SOA with diagram languages must be performed in an already highly "designative" way. This means that the terms of

the object language and the meta-language (e.g. error report service as an object-language terminus and procedural class as a meta-language terminus) should be displayed as independent as possible from their use in the judgment-context. The focus is on disclosing the types (software) that shall be implemented later. Diagrams are ideally suited for this purpose.

The diagram languages for procedure modeling are of course very similar to the diagram languages for process modeling. Procedure modeling comprises of the process parts (algorithms), which run as service-oriented applications while a process is being executed, as well as of those process parts, which can be specified in less detail since they involve human work (e.g. following work plans).

The order (1.-4.) chosen here serves merely as a recommendation. The modeling process is an iterative process, as every well-educated developer will know from practical projects (see fig.1).

#### IV. CONCRETION IN THE LARGE

Organization-centric development of application systems has been derived from data-centric [1] development. The development paradigm “applications follow processes”, which is valid in today’s service-oriented architectures, complements, but does not replace the data-centric approach. Therefore, SOA stands for a new paradigm, not a shift in paradigm. The data-centric approach remains as important as ever, but due to the triumphant progress of object-orientation and component-based development, it is integrated in the overall architecture and work processes in a more “intelligent” way (Platon was right!). In addition to data processing, work organization (industrial engineering) has become a subject in applied computer science.

Figure 7 illustrates an enterprise that is organized as an application system in a process-centric way, considering the expert field (domain) and the logical position (architecture).

Figure 8 shows an enterprise represented in an entirely object-oriented way. On the right, implementation aspects can be found; the right side lists the specific details of the information technology available for implementation today. We are therefore only talking about a “concretion in the large”. “IT and solution architects”, “integration developers” and

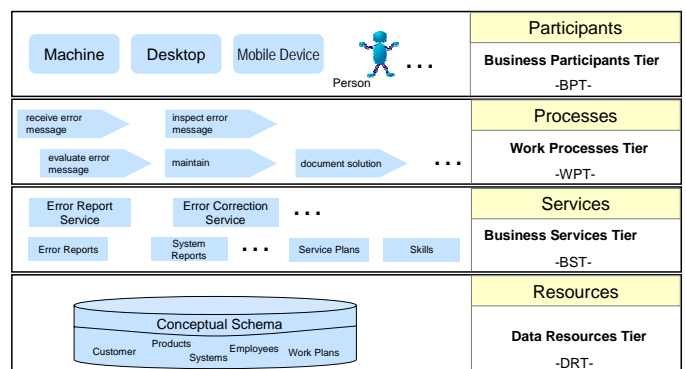


Fig. 7. Tier architecture model of an enterprise as application system

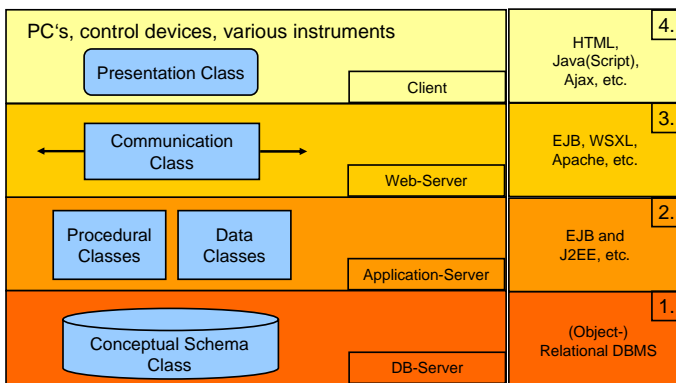


Fig. 8. Entirely object oriented concretion of SOA

“deployment managers” must be able to deliver this concretion for the enterprises (domains) that use information technology.

“Code development”, which is of course also important, in particular from the point of view of the “service and solution testers” on site, is currently done in so-called “low-wage countries” by well-trained people (near and offshoring). In complementation to a “concretion in the large”, we are now talking about a “concretion in the small.”

This sort of *software industrialization* [15] will continue to develop successfully in information technology if enterprises have at their disposal highly qualified „IT-service managers“, „IT-operators“, „business analysts“ and „project managers“. It will be shown in the following section, that aside from the software industrialization we will soon encounter a knowledge *industrialization*, which is based, for example, on work plans administered on the net, accumulating knowledge of work execution.

It is advisable to pay attention to admonishers like Mittelstraß when he writes [16]:

“Knowledge which is only viewed as a good that must be purchased, taught, managed or used, will lose its inherent character, namely to be an expression of the epistemic nature of mankind.”

In this context, every country on earth can calculate where its investments would be able to compete, in order to ensure employment and reasonable wealth for its citizens in the future. Given the immense differences in culture, shortages and aggressions in the world, a peaceful competition will be difficult to realize. We would like to refer to the excellent article “Euroskepsis, Markteuropa oder Europa der (Welt-) Bürger”(Euro-skepticism, market Europe or (World) Citizens’ Europe) by Jürgen Habermas [17] and in particular to [18] for methodology.

#### V. DYNAMIC SUPPORT AND OPTIMIZATION OF WORK PROCESSES

For the dynamic management of application systems (see fig. 7), it is necessary to create and use a meta-information system whose most important part is the repository system [13] as for example described by [19]. In accordance to the much-

noted work “The Quest for Resilience” by Hamel and Välikangas [20], future enterprise networks will be implemented as elastic ecosystems [21] built from components of different categories. These systems must be assembled in the best possible way, thereby facilitating the systems to respond, possibly even self-actingly, to changing situations.

The ever changing job design (e.g. due to product changes) and work organization is crucial to this approach. This is done considering

- the aspects: optimized processes, best possible employee assignment and dynamic IT-support (e.g. IT-services), as well as
- the fact that some of the jobs that are part of these processes, are performed by employees who come from everywhere, or respectively, the jobs are done where personnel is available at low cost.

In this regard, organizing potential assignments for employees in work plans and establishing a global work base (see fig. 9) is exceedingly relevant. Such a database allows neutral (assignment-free) storage and maintenance. It could contain the IT-services (data and program schemas) that are used anywhere in the world as program-technological means (to work plans) for work in those processes. This way, an enterprise’s IT-department organizes and controls the company’s work processes worldwide in division of labor and dynamically using the Internet.

The possibility of a global *informatization of our work* as well as an *informatization* of machines arises due to the fact that software (for machines) as well as knowledge (for people) can be managed globally as resources.

Platon (427-347 BC) was also right concerning the core of his ideology. In its modern version based on Frege/Lorenz’s abstraction theory [22] the basic verdict reads as follows:

Abstracta as knowledge or software are assets. This is analytically true.

Software and knowledge represent marketable resources that can be made available and put to use anywhere via the internet

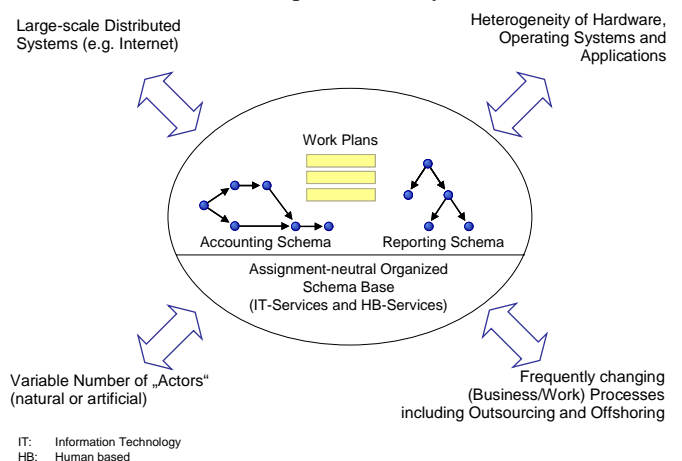


Fig. 9. Worldwide work place management out of the internet

(see fig. 9). Due to their vital role for the continuously growing human population, they should be treated like assets for which there are various balancing and depreciation rules. While work plans accumulate knowledge, program and data schemas represent software that is used for controlling machines. Knowledge represented in work plans can be considered the resource that offers orientation for human work. It can be traded globally and independently from the executor.

The protruding innovation of SOA is the extension of the concept of application systems by work and organizational processes of enterprises. This makes organization theory as it is found in Business and Social Sciences, the Engineering disciplines or in Industrial Engineering an “integral”, that is an interdisciplinary part, of Applied Computer Science. And this in a way that has not been seen in previous years. Concepts and institutions like the German REFA-Association for Work Design or Methods-Time-Measurement (MTM) founded in 1924 (These are systems for time allotment that have been used in Sweden as of 1950, in Switzerland since 1957 and in Germany since 1960) suddenly constitute a field of activity and provide IT-businesses and enterprises worldwide with the knowledge that information and computing scientists possess. Due to Ubiquitous Computing, however, this also affects the courses of study of Industrial Engineering, Business and Social Sciences, Mechanical Engineering, Electrical Engineering or Civil Engineering, as all of them are concerned with work science and process organization.

The following is a list of typical issues in optimization as they were recently elaborated by a student team of the TU Darmstadt at the Bitkom competition “Best Process Architecture” [23].

- *Parallelization*: Operations that are independent of each other must run in parallel and thereby shorten the overall process duration.
- *Optimization of single processes*: In addition, we must analyze each process individually to make improvements.
- *Integration*: Existing systems are integrated seamlessly in the new architecture so that the available resources can be used efficiently and effectively.
- *Elimination of information deficits*: The interfaces between operations are analyzed thoroughly so that the expected input or output will be found at the right time in the right place.
- *Reorganization or sequence optimization*: Process analysis takes into account an increase in efficiency due to reorganization of the order of single operations.
- *Outsourcing*: It is considered an alternative to outsource single processes, especially maintenance activities at the customer’s site, to external service providers.
- *Integration*: It is necessary to systematically analyze the overall process for operations that belong together and can therefore be considered a unit.
- *Elimination*: During process analysis those operations must be eliminated that are useless or do not contribute beneficially to the process result.

- *Acceleration*: Specific measures for shortening the overall process duration are vital, but not at the expense of quality and cost.
- *Introduction of additional test steps*: To ensure higher quality, it is useful to integrate additional steps for checking the process.

From the perspective of an employee, there are three possibilities to be considered when setting out to optimize work processes using IT:

1. to reduce people’s workload through *automation* (resource: “software”)
2. to *support* human work as for example using interactive applications (resources: “software” and “knowledge”), or
3. to improve people’s work *qualifications* (resource: “knowledge”)

Industrialization and automation were so successful in the previous decades that it is very advisable to revert our efforts with respect to the listing above. With globalization in mind as well as taking into consideration our worldwide division of labor, we should “invest much more in education and as little as possible in further automation efforts.” Technological progress cannot be stopped, but a world which is becoming increasingly compact, can only cope with progress, if it is flanked by human education.

## VI. OUTLOOK

In Applied Computer Science, from a global standpoint, service-oriented architectures constitute a new paradigm, but do not result in a paradigm shift. Managing data and managing processes are complementary and lead to entirely new job descriptions. In the expert languages of globally interacting IT-enterprises these new professions are called:

- IT-Architect
- Business Analyst
- Application Developer
- Service and Solution Tester
- Software Developer
- Deployment Manager
- Integration Developer
- Solution Architect
- Code Developer
- etc.

Nevertheless, people, who perform these jobs throughout the world, have the least say in *who performs which kind of work when and where*.

Today we are facing a great lack in researchers and teachers at colleges and universities who are able to support young people in acquiring interdisciplinary knowledge. This may be due to the fact that a *systematic uneducation* has apparently made its entry into numerous universities. Liessmann ([24], p. 72) comments this as follows:

“Therefore, uneducation today is not an intellectual deficit, not a lack in knowledge, no defect in cognitive competence – although all this will also continue to occur – but it is *not wanting* to understand. Whenever we are talking of knowledge nowadays, it is not about understanding. [...] It is either about the development of technologies that enable us to dominate nature or people or, it is about producing management ratios that have less and less to do with the subject matter it is allegedly about.”

Our universities have changed – in Europe for example in the course of the Bologna process – from “Universities of Understood Sciences” to “Universities of Applied Sciences”.

What should we do if Liessmann’s description of the situation was correct - and practical experience seems to confirm his assessment? This contribution has tried to sketch an answer to this question. There is nothing more important for our survival than that the humanities take up the challenge to newly enter in a process of enlightenment. *Logic, Mathematics, Linguistics, Computer Science and Philosophical Anthropology*, for example, are studies of the humanities. “Normative Logic and Ethics” [25] as well as their advancement to an “Encyclopedia Philosophy and Philosophy of Science” [26] provide us with the necessary *fundamental education and terminology*, in the sense of a *Universal Literacy*, to fulfill this task.

Therefore, we appeal for constructive computer sciences to become basic education for all citizens. As a matter of course, this basic education should be graded and differentiated into interdisciplinary (rather universities) and infradisciplinary (rather schools) knowledge. The root of the matter is teaching a disciplined use of language. Anyone who is a democrat and who is interested in participating in remodeling our pluralistic democracies into republics with a “plurality-tolerating form of life” [18] is well-advised to try this in a language-critical way. This form of life is characterized by the fact that it teaches people how they can think correctly instead of teaching them what they should think. – Parlemus!

## VII. REFERENCES

- [1] H. Wedekind and E. Ortner: *Systematisches Konstruieren von Datenbankanwendungen – Zur Methodologie der Angewandten Informatik*. Munich/Vienna: Carl Hanser Verlag, 1980.
- [2] F.R. Lehmann: *Fachlicher Entwurf von Workflow-Management-Anwendungen*. Stuttgart/Leipzig: B.G. Teubner Verlagsgesellschaft, 1999.
- [3] D. Nussbaum, E. Ortner, S. Scheele and J. Sternhuber: “Discussion of the Interaction Concept focusing on Application Systems”, IEEE International Conference on Web Intelligence, in press.
- [4] A. Oberweis: *Modellierung und Ausführung von Workflows mit Petri-Netzen*. Stuttgart/Leipzig: B.G. Teubner Verlagsgesellschaft, 1996.
- [5] M. Broy et al.: “Ein Requirements-Engineering-Referenzmodell”, *Informatik-Spektrum*, 30 (2007) 3, pp. 127-141.
- [6] J. Mittelstraß: *Der Flug der Eule – Von der Vernunft der Wissenschaft und der Aufgabe der Philosophie*. Frankfurt: Suhrkamp Verlag, 1989.
- [7] A. Oberweis and M. Broy: “Informatiker disputieren über Anwendungsnähe der Disziplinen“, *Computer Zeitung*, No. 29, Monday, 16.07.2007.
- [8] H. Wedekind, E. Ortner and R. Inhetveen: “Informatik als Grundbildung“, 6 articles in *Informatik-Spektrum*, 2/2004-1/2005.
- [9] L. Wittgenstein: *Philosophische Untersuchungen*. Frankfurt: Suhrkamp Taschenbuch Verlag, 1977.
- [10] E. Ortner: *Software-Engineering als Sprachkritik – Die sprachkritische Methode des fachlichen Software-Entwurfs*. Constance: Universitätsverlag Konstanz, 1993.
- [11] A. Cockburn: *Writing Effective Use Cases*. Boston, MA: Addison-Wesley, 2001.
- [12] E. Ortner: *Methodenneutraler Fachentwurf – Zu den Grundlagen einer anwendungsorientierten Informatik*. Stuttgart/Leipzig: B.G. Teubner Verlagsgesellschaft, 1997.
- [13] E. Ortner: “Repository Systeme, Teil 1: Mehrstufigkeit und Entwicklungsumgebung“, “Repository Systeme, Teil 2: Aufbau und Betrieb eines Entwicklungsrepositoriums“, *Informatik-Spektrum*, 22 (1999) 4, pp. 235-251 and 22 (1999) 9, pp. 351-363.
- [14] B. Schienmann: *Objektorientierter Fachentwurf – Ein terminologiebasierter Ansatz für die Konstruktion von Anwendungssystemen*. Stuttgart/Leipzig: B.G. Teubner Verlagsgesellschaft, 1997.
- [15] T. Grollius, J. Lonthoff and E. Ortner: “Softwareindustrialisierung durch Komponentenorientierung und Arbeitsteilung“, *HMD-Praxis der Wirtschaftsinformatik*, vol. 256, August 2007, pp. 37-45.
- [16] J. Mittelstraß: *Wissen und Grenzen. Philosophische Studien*. Baden-Baden: Suhrkamp Verlag, 2001.
- [17] J. Habermas: *Zeit der Übergänge*. Kleine Politische Schriften IX. Frankfurt: Suhrkamp Verlag, 2001, pp. 85-103.
- [18] P. Lorenzen: “Konstruktivismus“, *Journal for General Philosophy of Science*, 25 (1994) 1, pp. 125-133.
- [19] R. Berbner et al.: “Management of Service-oriented Architecture (SoA)-based Application Systems“, *Enterprise Modelling and Information Systems Architectures*, vol. 2, No. 1, May 2007, pp. 14-25.
- [20] G. Hamel and L. Välikangas: “The Quest for Resilience“, *Harvard Business Review*, Sept. 2003.
- [21] A. Corallo, G. Passiante and A. Prencipe: *Digital Business Ecosystems*. Cheltenham: Edward Elgar Publishing, 2007.
- [22] P. Lorenzen: “Gleichheit und Abstraktion“, *Konstruktive Wissenschaftstheorie*. Frankfurt: Suhrkamp Taschenbuch Verlag, 1974, pp. 190-198.
- [23] H. Ghani et al: *Konzept zum Hochschulwettbewerb “Beste Prozessarchitektur“* for the BITKOM University Challenge 2007, <http://www.bitkom.org>.
- [24] K.P. Liessmann: *Theorie der Unbildung – Die Irrtümer der Wissensgesellschaft*. Vienna: Zsolnay-Verlag, 2006.
- [25] P. Lorenzen: *Normative Logic and Ethics, 2nd annotated edition*. Zurich: B.I.-Wissenschaftsverlag, 1984.
- [26] J. Mittelstraß (ed.): *Enzyklopädie Philosophie und Wissenschaftstheorie*. Stuttgart/Weimar: J.B. Metzler Verlag, vol. 1(1980), vol. 2 (1984), vol. 3 (1995), vol. 4 (1996).