Application of Object-Oriented Principles to the Methodology Area

Abstract. In this paper I discuss problems with software development methodologies and I introduce the Methodology Framework for IS/ICT MeFIS and its meta-version MetaMeFIS developed at the Department of Information Technology of the University of Economics in Prague. MeFIS encompasses a collection of methodology patterns along with principles and processes attached with defining a methodology for a concrete project. There are specialized methodology patterns for various types of solution, problem domains and project types. The idea of a methodology pattern is original and has arisen as a result of using object-oriented principles in the methodology area. Each methodology pattern has a defined description structure and is assigned into certain classification categories. Applying this classification we can build a methodology patterns hierarchy and express it as a class diagram in the Unified Modeling Language. This approach enables better methodology selection, its customization and the reuse of methodology elements across many projects.

Present State of Software Development and Importance of Using the Formal Methodology

First I'd like to discuss the role software is playing in society these days. There is a common admission that software is a key factor for the market competition and company success. Software helps to take strategic business decisions, to manage information and communication in all business departments and to support business. On the other hand today's economic climate forces companies to measure results according to revenue, costs and quality. Software vendors have been hard hit by the economic slump. One place to start meeting the challenges of revenue, cost and quality is to look for IT alignment with the business, that means companies have to focus on those projects with an immediate value to the business. IT projects under this pressure have to be made right the first time, on time, and match to customer requirements.

Other key factors influencing information systems and information and communication technologies (IS/ICT) are fast and permanent economic environment changes and changes in the area of IS/ICT alone which are even distinctive. Hardware, software and IS/ICT management processes are changing very fast. There are emerging new programming languages and integrated development environments, application frameworks, middleware technologies, new forms of IS/ICT operation (outsourcing, ASP) and new IS/ICT management attitudes (COBIT, ITIL). Workers, responsible for IS/ICT development and operation management, can use very sophisticated and powerful tools however they still face more complicated problems. Their bad or late solution may have a very distinctive impact on competitive company situation.

Many software development companies and their IT managers believe that using software development methodology can solve some of these problems. Methodologies impose a disciplined process upon software development with the aim of making this process more predictable and more efficient. They do this by developing a detailed process with a strong emphasis on planning inspired by other engineering disciplines (e.g civil and electrical engineering). Traditional or rigorous methodologies have been around for a long time, but they've not been noticeable for being terribly successful nor popular. The most frequent criticism of these methodologies is that they are bureaucratic. There's so much work required to follow the methodology that the whole pace of development slows down. As a reaction to these methodologies, a new group of methodologies have appeared in the last few years. These methodologies are known as agile methodologies. These new methodologies attempt a useful compromise between no process and too much process, instead providing just enough process to gain a reasonable payoff.

Many Methodologies - How to Categorized Them, Which One to Choose?

Methodologies have been around for a long time, but there is still a big part of all IS projects unsuccessful. According to the research of Standish Group [13] in 2000 only 28% of all application development projects satisfy success criteria, e.g. project finished in time, according to the budget and with all specified functions. That means 72% of projects completely fail or overran the budget, didn't fulfill the term or all contracted functions. There may be many reasons for this fact. Part of it is that software development in many companies is a chaotic activity, without any methodology. We can characterize this way of software development by the phrase "code and fix". Part of it is that rigorous methodologies don't suit for current projects where requirements aren't possible to define at the beginning or they are changing. Part of it is that we have a lot of difficulties with using of formal methodology. We can name some of them. Existing methodologies are neither adequately and uniformly described nor classified. Many methodologies are concentrated only on some phases of IS development life cycle (e.g. object oriented analysis and design) and don't deal with processes as requirements specification, installation, operation and maintenance etc. Second main fault, existing methodologies have, is that they don't focus on cross-project processes as information strategy definition, project portfolio management, enterprise architecture adoption etc. Many methodologies don't deal with new application domains as e-commerce, BI, CRM etc. and reflect only application development from scratch. The real situation is differrent, because many current IS projects deal with application integration or application enhancement. Most methodologies don't reflect the specific type of the project. There are neither criteria how to select an appropriate methodology for a concrete company and project conditions nor processes for methodology customization.

We can identify objective reasons, why multiple methodologies must exist. These reasons can be separated into three categories:

- technology

- people and companies
- projects

The first category emphasizes the fact that many methodologies have been developed for special technologies and development approaches (e.g. structured methodologies, object-oriented methodologies, Data warehouse building methodologies etc.) As we accept the idea that people represent the key element of the methodology, we come to a second category. Each development team need its own methodology. People vary in knowledge, experience, skills and other characteristics and similar differences are among teams. Corporate culture plays an important role in choosing or constructing the methodology. The third category is dealing with project differences. Different methodologies are needed depending on the project size (number of people being coordinated) the criticality of the systems being created, and the priorities of the project [5]. In my doctor thesis [4] I have made an analysis of available methodology information resources and I have proposed classification of methodologies consisting of six criteria explained in next paragraphs:

- Methodology focus
- Methodology scope
- Methodology weight
- Problem domain
- Type of solution
- Way of solution

Methodology Focus Criterion

Methodology focus criterion distinguishes between Enterprise methodologies that are focused on information system at the enterprise level and Project methodologies that are focused only on one project. Most methodologies belong to the project methodologies category but there are some focused on the enterprise level (Enterprise Unified Process-EUP, Model Driven Architecture - MDA, Capability Maturity Model - CMM).

Methodology Scope Criterion

This criterion is based on the methodology scope definition presented by A. Cockburn [5], but I have made a slight modification concerning the third factor. My conception of methodology scope is defined as a cross point of three factors:

- life cycle phases
- roles
- dimensions

Methodologies differs in the count of IS development life cycle phases. Many object-oriented methodologies focus only on object oriented analysis and design phases and don't deal with processes such as requirements specification, installation, operation and maintenance. Third factor (dimensions) is taken up from Multidimensional Management and Development of Information System (MMDIS), methodology developed at the Department of Information Technology of the University of

Economics, Prague early in the 1990's. This methodology emphasizes the need of dealing explicitly with various dimensions or views of software development and defines specific dimensions which are addressed with another weight in each life cycle phase:

- software dimension
- hardware dimension
- HCI dimension
- data dimension
- function/process dimension
- work dimension
- economic dimension
- legal dimension etc.

Methodology Weight Criterion

The term "methodology weight" is taken from A. Cockburn's papers [5], in which methodology weight is defined as a *methodology size* multiplied *methodology specific density*. We can diversify between "light methodologies" and "heavy methodologies". Methodology for a spaceship navigation system project is certainly different than a project for building a web presentation. In the first one there is much more formality caused by need for the highest quality and maintainability, which is needed for mission critical systems or life critical systems. Another reason for heavier methodologies is the project size measured as a team size. If we agree that project management and team communication build an essential part of methodology (see [6]), there must be a difference between a project made by 4 developers sitting in one room and a project for 150 developers all around the world.

Problem Domain Criterion

ERP (Enterprise Resource Planning) system development certainly differs from building datawarehouse or e–commerce application in processes, techniques, management etc. Some of methodology elements are shared but some are specific for particular domains. We can define problem domains that are in figure 1.

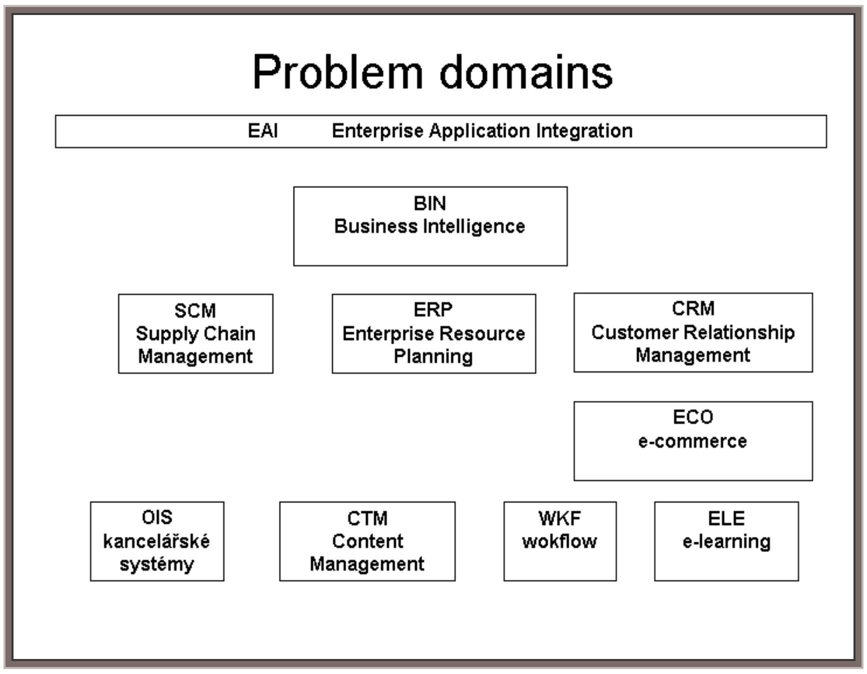

Fig. 1. Problem domains

Type of Solution Criterion

Type of solution criterion takes into account particularity of a new solution, upgrade, package implementation, and integration. Most methodologies address only new solutions in spite of the fact nowadays there are many integration projects that have to exploit existing applications.

Way of Solution Criterion

Criterion named "way of solution" distinguishes between in house solution and outsourcing which is nowadays also of growing importance.

Applying Object-Oriented Principles to the Methodology

In the previous paragraph I named some reasons for multiple methodologies existence. On the other hand I am convinced that there are some methodology elements we can share and reuse. In the realm of software development currently there is a big stress on object technology and object-oriented principles. Every protagonist of object-oriented approach hopes and discovers that object-oriented approach in software development can deliver many advantages e.g. more reuse,

higher development speed, better quality etc. Object-oriented analysis is more corresponings to the real world. Unified Modeling Language, standardized graphical modeling notation, has been developed and is widely used. I think we can use these advantages for improving software development processes and methodologies themselves. That means we can use object-oriented principles in a reflexive manner. To do so, we need to analyze and describe software development methodologies in a standardized form, classify them, record them and build the software development knowledge base. This way we can reuse either whole methodologies or part of them. In this sense we are encouraged to use object oriented approach in the methodology area. Let's have look at key object oriented principles - abstraction, encapsulation, reuse, inheritance, polymorphism, software patterns concept and its application in the methodology area.
Abstraction is a basic principle in software analysis and design, but it is very useful in the methodology field too. If we accomplish separating abstraction levels in methodology, we can gain more reusability. For example if we can build methodology independent of the development platform, we can reuse methodology elements across various CASE tools and IDEs. Encapsulation is the most important object-oriented principle, which mirrors also in component based development and service oriented development. It is useful for the methodology construction too and manifests in building of reusable methodology elements. Reuse is a buzzword these days, but it is difficult to achieve it. According to Ambler's paper [2] organizations that are successful at reuse they share several common philosophies. First, they recognize that you can reuse more than just code. Components, frameworks, documentation templates, software processes and development standards can all be reused.

Methodology Patterns Concept

The idea of using patterns in software development is of growing importance in the last decade. Pattern Based Development enables developing software faster, in higher quality and with increased level of reuse. There are many types of software patterns: architectural patterns, analysis patterns, design patterns and code templates. Patterns are useful to developers and architects because they:

- Document simple mechanisms that work.
- Provide a common vocabulary and taxonomy for developers and architects.
- Capture the knowledge of experienced developers and present it in the form of a patterns catalog.
- Enable solutions to be described concisely as combinations of patterns.
- Enable reuse of architecture, design and implementation decisions.

If patterns are useful in software development, why not use pattern concept for software development methodologies themselves? In the methodology area we also need a documentation tool, to build vocabulary and taxonomy, capture knowledge about "what works in certain context", reuse whole methodology or some of its elements. Idea of using pattern concept for methodology is original and is described in my doctor thesis [4]. Each methodology pattern matches to unified description structure and is assigned into classification categories. This way it defines context for its application and builds knowledge. Methodology patterns consist of three basic criteria - problem domain, type of solution, way of solution, and three main project criteria - system criticality, project size, and approach to solution. Applying this classification we can build a methodology patterns hierarchy. This hierarchy is modeled as a Class diagram in Unified Modeling Language (see figure 2).

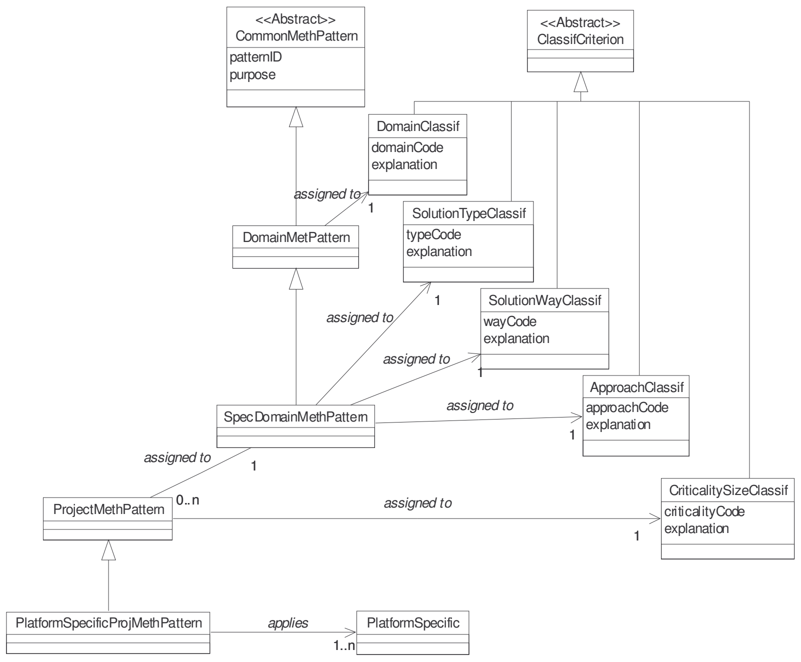

Fig. 2. Methodology patterns hierarchy

General Methodology Pattern is an abstract ancestor of all methodology patterns and defines common process structure (lifecycle phases and gates among these phases), general principles, processes and practices, which must be provided in each methodology pattern. Special aspects of a particular problem domain are solved within **Domain Methodology Patterns** and then specialized for type of solution and way of solution in **Specific Domain Methodology Patterns**. Particular problems of various project types according to defined project criteria are solved in **Project Methodology Patterns**. All these patterns are independent of the development platform. This way it is possible to reuse methodology elements across various CASE tools and IDEs. Platform Specific Project Methodology Pattern is built by means of mapping Project Methodology Pattern for concrete development platform.

MeFIS Methodology Framework for IS/ICT

Application of object oriented principles to the methodology area has resulted in defining the Methodology Framework for IS/ICT - MeFIS. The main issue for the methodology framework definition is the idea, that there can't be only one methodology, as I explained previously. The idea of defining the methodology framework isn't completely new. We can see this idea in OPEN process framework or in Cockburn's family of methodologies Crystal [5]. MeFIS results from these ideas but goes beyond in the scope. It defines specialized methodologies not only for various types of projects as to size and criticality but also for various problem domains, solution types (new IS development, upgrade IS, package implementation) and other criteria. These specialized methodologies are described as methodology patterns. Together with a collection of methodology patterns, MeFIS specifies also architecture, principles and processes for patterns customization.
The first order resource for MeFIS was Multidimensional Management and Development of Information System (MMDIS), methodology developed at the Department of Information Technology of the University of Economics, Prague early in the 1990's. Many principles of this methodology remain valid – especially the impact on the system integration and multidimensional approach, but it was necessary to revalue, replace and extend some principles - especially iterative and incremental development, bigger acceptation of object attitudes, concentrating on IS/ICT services, accenting new categories of applications like CRM, SCM and BI. That was the reason for creating new methodology framework. MeFIS results from other current methodological sources:

- Rigorous Methodologies - OPEN, CMM, RUP, etc.
- Agile Methodologies - XP, SCRUM, DSDM, FDD, ASD, etc.
- Model Driven Architecture
- Service Oriented Architecture and many others.

Meta-Methodology Framework MetaMeFIS

As we have seen MeFIS results from many sources and builds a collection of newly defined methodologies named as methodology patterns. Relation among these patterns and other methodology elements are expressed in the form of UML diagrams, which build a conceptual model of MeFIS.
As I mentioned before there are objectively many methodologies, some of them are very popular and are widely used. We, at our department, assume that together with building new methodology patterns it is very useful to analyze existing methodologies and create conceptual models of them. This collection of conceptual models for various existing IS/ICT development and operation methodologies we name MetaMeFIS. Describing methodologies in a unified and standardized form enables better understanding of methodology concepts and better choosing of the adequate methodology for each particular project. We want to develop an information portal for realization and maintenance of MetaMeFIS, which will serve as a communication and evolution platform. We expect participation of many subjects in the Czech Republic

and worldwide including universities, methodology suppliers , software development companies etc. We want to define processes for framework using, e.g. processes for methodology selection, customization for concrete company and concrete project, methodologies maintenance, building knowledge etc. We are prepared to define a UML profile for methodology .

Conclusions

This paper addresses problems with software development methodologies classification, selection and construction. It introduces an original concept of methodology patterns which transfers object oriented concepts (e.g. inheritance, encapsulation, reuse), software patterns concepts and abstraction levels separation into the methodology area. This way it brings new approach to IS development methodologies. At present Methodology Framework MeFIS defines architecture, conceptual model, methodology patterns classification criteria, meta-methodology principles and processes. Methodology pattern content is documented by defining methodology pattern for "New object oriented own common software development". There is much work before us. It is necessary to define content of various methodology patterns, generalize some their features and transfer them into a common ancestor of all methodology patterns, build an information portal for the Methodology Framework realization and maintenance, test this concept on pilot projects and modify conceptual model according to gained experience. As to MetaMeFIS we have started work on creating models of some existing methodologies.
I think it is worthwhile to take the object-oriented approach to software development processes with the aim of its improvement. When we teach object-oriented principles and UML, we have to use this means even in other areas and show its power in analyzing and describing matters.

References

1. Ambler, S.W.: Something's Gotta Give, Software Development, March 2003
2. Ambler, S.: Reuse for the Real world, Agile Modeling, April 2002, www.agilemodeling.com
3. Ambler, S.W.: Different Projects Require Different Strategies, 2002, www.agiledata.org
4. Buchalcevova, A.: Methodology Framework for IS/ICT, Ph.D. thesis, 2003
5. Cockburn, A.: Just-in-time methodology construction, Humans and Technology Technical Report TR 2002.01, 2000
6. Cockburn, A.: A Methodology Per Project, Humans and Technology Technical Report, TR 99.04,1999
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns
8. Highsmith, J.: Agile Software Development Ecosystems, Addison-Wesley, 2002, ISBN 0-201-76043-6

9. Summary of Results 2003 Worldwide IT Benchmark Report, 2002, www.metagroup.com
10. Enterprise Solution Patterns Using Microsoft .NET, www.microsoft.com
11. www.standishgroup.com
12. Vorisek, J., Strategic management of information systems and systems integration. Management Press, Prague 1997, ISBN 80-85943-40-9
13. www.standishgroup.com
14. Buchalcevova, A.: Methodology Patterns, concept, classification and processes for their application, CITSA 2004, Orlando, USA