IT Trends - Software Development

1 Introduction

Main purpose of this paper is to show a summary of trends in the field of software development. It concentrates on trends visible in Software Engineering (the process oriented more theoretical part in software development) and also in Programming (the more practical oriented part of software development). It will also show some major side effects which will influence software development in future. Finally it shows core challenges in the field of software development.

2 Trends in Software Engineering

Software Engineering (SE) is a systematic approach to the analysis, design, implementation and maintenance of software. One core component of SE is the software development process (SDP). Beside the more common and proven software development processes like “RUP” (Rational Unified Process) and the “V-Modell” several more uncommon processes will be in place. One of these approaches is called “eXtreme Programming” (XP). The classic processes focus on an iterative and incremental approach and are separated in phases with well defined deliverables. The process itself is heavy weighted and is more management oriented. XP on the other hand is a more light weighted software development approach and focuses more on the developers’ point of view. The search for the best process approach will continue and also unusual approaches will evolve.

Regardless of the “Fight for the best process” the **software development process itself plays an increasing role in software development in the future**. RUP from Rational Software is a candidate process for standardisation within the OMG (Object Management Group).

Other approaches focus more on improving aspects of processes. Analysis and design for example are key phases in every process. “Agile Modelling” (AM) is such an extension to software development processes. AM is a collection of values, principles and practices for modelling software. Goal of this approach is not to have a perfect design model but a good one. Not only the process itself will be candidate to improve but also **aspects of the software development process will become more efficient and effective using new approaches**.

Over all there will be an **increasing role of patterns not only in software design – also in software architectures and project management**. While software patterns are well documented since 1995 in the book “Design Patterns” (E. Gamma, R. Helm, R. Johnson, J. Vlissides) patterns in software architectures and also in project management are just evolving. Patterns are a easy way to preserve knowledge in special fields of interest. It is assumable that patterns will play a really increasing role in software development and also in management tasks in general.

3 Trends in Programming

Standardisation is one of the major trends in software development. Standardisation is a big issue when talking about software architectures. The Model Driven Architecture (MDA) from the Object Management Group (OMG) is one example. Main goal is to provide an open, vendor-neutral approach to the challenge of interoperability. Also in the space of WebServices standardisation in the sense of interoperability is one of the major efforts. Industry leading companies as Microsoft, IBM, BEA, Oracle, Qwest and others have recovered the need for vendor neutral standardisation to extract the real strength of the WebService initiative and founded the WS-I (WebServices

Interoperability Organisation). All these standardisation efforts makes live easier for software developers. They profit from standardised interfaces and reliable software developer kits. Standardisation reduces the effort to become common with major technologies.

Standardisation is also in the area of Integrated Development Environments (IDE) a visible trend. The IDEs will be more like generic horizontal platforms wherein the developer plugs the tools in he/she needs. IDEs like “Eclipse” or “NetBeans” are freely available and support exactly this model. Tools like compiler, debugger, UML design tools and many others can be combined to create the best fitting IDE for the developers need.

Integrated Development Environments will evolve to simple to use “drag & drop” toolboxes. Major idea behind these development environment is to hide the complexity of existing and future technologies. Using a components based visual “drag & drop” based programming paradigm hides the complexity and opens these technologies to a wider developer community. This approach enables also non-professional programmers to participate with the technology progress. Using tools like “BEA Workshop” or “Microsofts Visual Basic” show this trend towards easy to use software generators. These toolboxes are based on framework technologies and fastens development. Major disadvantage using these tools is the effect of canalising creativity. The tool restricts the results in advance because of its own limitations.

The simple to use IDE hides the evolving complexity of future technologies. This evolution of complexity promotes also an **evolution of two different skilled developers: expert developers and amateur developers**. The expert developer needs to have a deep understanding of basis technologies and the relationships between themselves to build new software architectures and technologies. He/she has a common understanding of software development processes. The amateur developer has an overall understanding of technologies and relies on “drag & drop” programming techniques. He/she uses results of expert developers and uses a semi professional software developer environment. A business oriented person could be an amateur programmer as well. The group of amateur developer is much more bigger than the group of expert developers.

Beside standardisation and the IDE evolution there is also an **evolution in Open Source usage**. Open Source projects as Linux, JORAM, … or Open Source like projects like Apache, JBoss, … will get more and more influence in the markets in the future. The strength of the Open Source community lies in fast implementation and non-business but more technical oriented approaches. More reliable and stable software will be produced by the Open Source community in the future. More and more software development projects will be based on Open Source software projects.

Looking more inside the programming topic there is almost a **stabilisation in the field of programming models**. While there was a shift from structural programming (SP) towards object oriented programming (OOP) in the last twenty years industry doesn’t introduce revolutionary new programming models beside OOP. Some fine adaptation on top of OOP were made (aspect oriented programming, component oriented programming, …) but at the end they are all based on top of OOP. It seems like OOP will remain the programming model for the next years.

The major platform opponents “J2EE” (Java 2 Enterprise Edition, Sun) and “.NET” (DotNet, Microsoft) are two examples of component oriented programming (COP). J2EE, .Net and COP are examples for the **increasing importance and usage of reusable generic software components and software component frameworks**. COP could be seen as an implementation of Generic Programming (GP) and Domain Engineering (DE). Main goal in GP is to program with concepts. Concepts in this context are defined as a family of abstractions that are all related by a common set of requirements. These concepts are tried to be mapped to generic software components. DEs' major goal is to identify and build reusable software components in special domains.

Software components and frameworks could be seen as macro units in software development. Inside this components when it comes to pure software development there is a trend towards the **increasing usage of software patterns in implementation**. Beside the 23 patterns identified by Erich Gamma and his team there are several new patterns in new areas discovered. New technologies enable new patterns. These patterns preserve knowledge in these new areas of technology. Patterns increase the development process and help developers to avoid common pitfalls.

4 Side effects affecting Software Development

The coming future technology will get more and more complex. The software architectures for coming technologies will be based on rather complex concepts and basis technologies. The resulting technology will be much more complex and difficult to understand since it needs a good and deep understanding of the basis technologies. In the future an **expert developer have to have a very good and deep technical knowledge base** to understand all the relations between different technologies.

The forthcoming technologies follow a trend of abstraction of underlying hardware, software and concepts beginning in the past. The development of abstraction starts with the assembler language and structural programming on a very concrete level. Next step of abstraction was the evolution of 3rd generation languages as C, C++, Java, … coming along with object oriented programming. **Technologies in the future will result in very abstract layers** supported by software generators, very powerful programming languages and software frameworks. By abstracting from underlying technologies and concepts it will be possible for the developer to concentrate more on solving the real problem.

The industry identifies the following core technology for the future:

- Java Technology
- Mobile Internet Technology
- .Net
- OpenSource in general

5 Core Challenges

Major challenges for the future will be the **new complex technologies**. People have to be trained to cope with these technologies. Technologies have to be known good to estimate their business relevance for the industry. Beside the pure technological view there are also challenges in the field of management. **Coping with project management problems and finding the appropriate software development approach** are major management tasks to make software development effective, efficient and trustable. The **developer as a service provider have to have a better understanding of the specific problem domain** wherein he/she is implementing a solution.

6 Summary of trends

6.1 Software Engineering

- software development process itself plays an increasing role in software development in the future
- aspects of the software development process will become more efficient and effective using new approaches
- increasing role of patterns not only in software design – also in software architectures and project management

6.2 Programming

- Standardisation is one of the major trends in software development.

Summary *of trends*

- Standardisation is also in the area of Integrated Development Environments (IDE) a visible trend
- Integrated Development Environments will evolve to simple to use “drag & drop” toolboxes
- evolution of two different skilled developers: expert developers and amateur developers
- evolution in Open Source usage
- stabilisation in the field of programming models
- importance and usage of reusable generic software components and software component frameworks
- increasing usage of software patterns in implementation