Improving Productivity in the Development of Software-based Systems

Frances Paulisch

CT SE SWI
Siemens AG
Otto-Hahn-Ring 6
80200 München
frances.paulisch@siemens.com

Abstract: This paper describes various techniques for improving productivity in the development of software-based systems. These build on having a strong foundation regarding people and processes. On this solid foundation, it is important to do requirements engineering well so that you develop the right functionality, elicit also the non-functional requirements, and have testable requirements. Structure the system to avoid unnecessary complexity and to take advantage of reuse. During the development use an iterative and lean approach to ensure that the product is realized as efficiently as possible. Finally, work at a higher level of abstraction through model-driven approaches and take advantage of automation provide additional improvements to productivity.

1 Overview

This paper describes various techniques for improving productivity in the development of software-based systems. These build on having a strong foundation regarding people and processes. On this solid foundation, it is important to do requirements engineering well so that you develop the right functionality, elicit also the non-functional requirements, and have testable requirements. Structure the system to avoid unnecessary complexity and to take advantage of reuse. During the development use an iterative and lean approach to ensure that the product is realized as efficiently as possible. Finally, work at a higher level of abstraction through model-driven approaches and take advantage of automation provide additional improvements to productivity.
2 Useful Techniques for Improving Productivity

First of all, it is important to have a solid foundation to build on, in particular in regards to people and processes. The “people factor” is very important as it is the most variable and has a strong influence. You need to have experienced and skilled people on the project and also the ability of the team to work well together is very important. Don’t underestimate this aspect. Having an open and trustful communication and ensuring that all have the same understanding of the status of the project throughout the lifecycle is very important especially when dealing with uncertainties in a project. Furthermore, having appropriate and well-defined processes in place helps reduce risks and helps make the various responsibilities clear which is important since so much time and effort can be wasted through poor coordination.

Secondly, have a good understanding of the system you will build. Not only the requirements in terms of functionality, but also the non-functional requirements of the system such as performance, reliability, dependability, scalability, etc. Aim to have testable requirements, also for such non-functional requirements. Be aware of which features are most important in terms of customer value, which have the most influence on the architecture of the system, and which ones are likely to change. Use that information to define the architecture of the system. Making the right choices early and ordering the realization appropriately can help to avoid significant rework during the project. Also aim to structure the system so that complexity and dependencies between components are reduced. If one is able to reuse components or use a product-line engineering approach, this is another very effective way to reduce effort and also often increase the quality of a system.

During the development of the system, an iterative approach is often useful to provide feedback and motivation to the whole product development team. An iterative approach helps alleviate some misunderstandings across phases e.g. between product management and the realization roles. If a problem arises, the sooner one knows about it, the better. This often goes hand-in-hand with test-driven development and continuous integration. Use the techniques of “lean development” to analyze the value stream, identify sources of “waste” and consider, taking all aspects into account, how to adjust appropriately to achieve the best overall benefit. In other words, as is in the slogan of lean development “think big, act small, fail fast, learn rapidly”.

There has been much progress made in the past ten to twenty years in the area of modern software engineering for example in the areas of model-driven software engineering and domain-specific languages to enable the development teams to work at a higher level of abstraction. Take advantage of the latest proven techniques and tools to improve the productivity of the development of software-based systems.