A Service Based Development Environment on Web 2.0 Platforms

Xabier Larrucea , Rafael Fernandez , Javier Soriano , Andrés Leonardo Martínez , and Jesus M. Gonzalez-Barahona

European Software Institute, Parque Tecnologico de Zamudio 204,
48170 Zamudio, Spain
Xabier.Larrucea@esi.es
Computer Networks & Web Technologies Lab., School of Computing,
Universidad Politecnica de Madrid, 28660, Boadilla del Monte, Madrid, Spain
{rfernandez,jsoriano}@fi.upm.es
Telefonica Research & Development, 28043 - Emilio Vargas 6, Madrid, Spain
almo@tid.es
GSyC/LibreSoft, Universidad Rey Juan Carlos
jgb@gsyc.escet.urjc.es

Abstract. Governments are investing on the IT adoption and promoting the so-called e-economies as a way to improve competitive advantages. One of the main government's actions is to provide internet access to the most part of the population, people and organisations. Internet provides the required support for connecting organizations, people and geographically distributed developments teams. Software developments are tightly related to the availability of tools and platforms needed for products developments. Internet is becoming the most widely used platform. Software forges such as SourceForge provide an integrated tools environment gathering a set of tools that are suited for each development with a low cost. In this paper we propose an innovating approach based on Web2.0, services and a method engineering approach for software developments. This approach represents one of the possible usages of the internet of the future.

Keywords: Service, Web2.0, Method engineering.

1 Enterprise 2.0 Technologies and Quality Assurance

Nowadays organisations are worried about mainly two main issues: collaboration and quality assurances. As global market opportunities and competition increase, collaboration is becoming more and more essential for improving productivity and accelerating innovation at the personal, team, group, enterprise and business coalition levels. Many enterprise collaboration platforms have already been developed and successfully deployed in both large, and small- and medium-sized enterprises (SMEs). Enterprise collaboration has recently come to benefit from the emergence of an enterprise-oriented specialization of the Web 2.0 vision, commonly referred to as

Enterprise 2.0 providing new models and tools for emergent collaboration and co-creation. Enterprise collaboration is thus being enhanced by virtual communities that leverage social linking and tagging tools (like tools for social networking, social bookmarking and social search), user-contributed content management platforms (like enterprise Wikis, blogs and forums), tools that leverage user opinions (like tools supporting comments and voting), subscription-based information distribution tools (like Enterprise RSS feeds), etc. Used in the context of a carefully engineered collaboration strategy, these technologies provide a wealth of collaborative services for software developers

On the other side quality is still representing a nightmare for too many organizations. In fact quality cost is one of the most important considerations in software production Quality assurance practices and software products quality represent in most of cases the forgotten requirement and it is becoming a hard task to select an appropriate infrastructure in order to fulfil at the same time customers' requirements and some level of quality assurance. The resulting solutions are usually defined in terms of what functionalities are exposed and the question about "what is the quality required?" and "How do we achieve this quality?" are effaced from stakeholders' memory.

This happening generalises a broad range of situations such as consultancy, in-house and outsourcing developments. The evolution of Internet technologies such as Web2.0 and mash up platforms are supporting collaboration mechanisms but at the same time they need to fulfil quality models requirements (e.g., Capability Maturity Model Integrated-CMMI) In order to facilitate the fulfilment of these two challenges the proposed architecture combines the evolution of these new Enterprise 2.0 technologies and quality assurance facilities. In fact the resulting collaborative environment is enriched with the savoir-faire (knowledge management) in software production environments.

2 Ezforge: New Generation of Networked Forges Supporting Collaborative Software Development

Organizations tend to behave like dynamically reconfigurable networked structures that carry out their tasks by means of collaboration and teamwork. Effective teamwork is an essential part of any non-trivial engineering process, and collaborative capabilities are an essential support for these teams. Software development is no exception; it is in itself a collaborative team effort, which has its own peculiarities. Both in the context of open source software development projects and in organizations that develop corporate products, more and more developers need to communicate and liaise with colleagues in geographically distant areas about the software product that they are conceiving, designing, building, testing, debugging, deploying and maintaining. In their work, these development teams face significant collaborative challenges motivated by barriers erected by geographic distances, time factors, number of participants, business units or differences in organizational hierarchy or culture that inhibit and constrain the natural flow of communication and collaboration. To successfully overcome these barriers, these teams need tools by means of which to communicate with each other and coordinate their work. These

tools should also take into account the functional, organizational, temporal and spatial characteristics of this collaboration. Software product users are now becoming increasingly involved in this process, for which reason they should also be considered. In response to this necessity, forges are gaining importance both in the open source context and the corporate environment.

Following the ideas a forge can be described as a kind of collaborative development environment (CDE) that provides a virtual space wherein all the stakeholders of a software development project –even if distributed by time or distance - may negotiate, brainstorm, discuss, share knowledge, and labor together to carry out a software product and its supporting artifacts. It integrates multiple collaborative tools and resources, thanks to which it offers a set of services to aid all the stakeholders in the software development area, including managers, developers, users, commercial software manufacturers and software product support enterprises, to communicate, cooperate and liaise. Forges consider software development's social nature and assure that the people who design, produce, maintain, commercialize and use software are aware of and communicate about the activities of the others simply, efficiently and effectively, also encouraging creativity and driving innovation. In doing so, forges provides with a safe a centralized solution conceived to optimize collaborative and distributed software development generally based on Internet Standards. This solution serves a number of essential purposes, including:

- A holistic integration of disparate collaborative processes and tools through a collaborative environment,
- an expansion of visibility and change control,
- a centralization and administration of resources, and
- a reinforcement of collaboration, creativity and innovation.

3 EzForge

The appearance of Enterprise 2.0-based forges, such as EzForge enable software development teams to find, customize, combine, catalogue, share and finally use tools that exactly meet their individual demands. Supported by the EzForge platform, they can select and combine development tools hosted by third parties rather than buying a pre-determined, inflexible and potentially heavyweight software development environment.

EzForge as the main part of the proposed architecture is based on the idea of considering forges not as single sites providing a monolithic set of services to host projects as isolated silos of knowledge, but as a collection of distributed components providing services among which knowledge is shared. Each project decides on its own customized set of services, and users can configure their own mashup-based user interface. Fig. 1 depicts the 3-tier EzForge architecture.

Back-end tier is where integrated and legacy systems reside. It is important to take into account that the EzForge architecture imposes no limitations to where the different components may be hosted. Those systems have their own set of basic forge services, such as source code management, wiki, and issue/bug tracking services, and they are integrated into the forge following a Web 2.0 approach consisting of

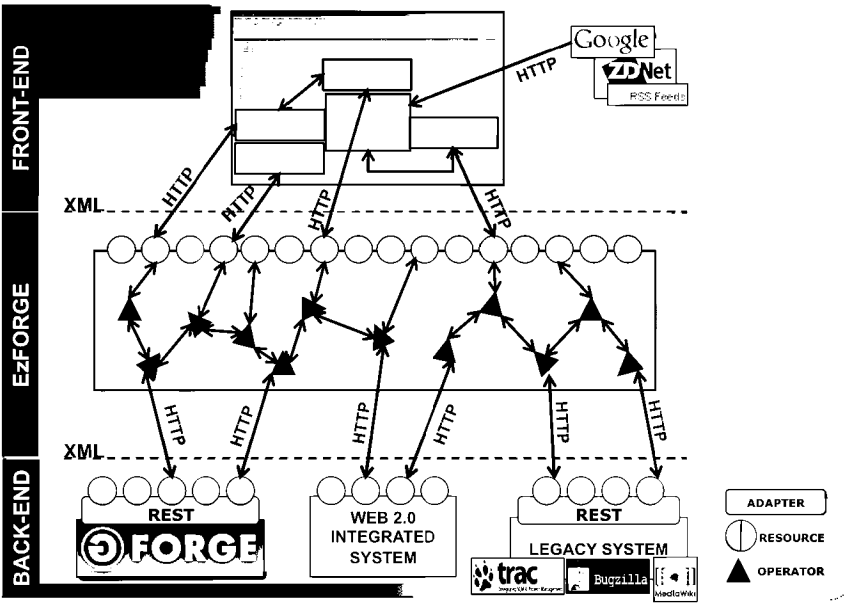

Fig. 1. EzForge architecture

transforming their legacy services into a uniform layer of resources. These resources are components designed following the REST architectural style that can be accessed through an URI via HTTP. Integrated systems already follow this approach, while legacy systems needs adapters to perform this task. Thanks to the aforementioned layer of resources, the EzForge tier can access them to gather and process their data by means of special resources called operators, elements designed to get data from resources and use it to produce new data that can be processed by other resources, enabling their remix and composition. Doing so, EzForge creates the set of resources the forge will deliver to its final users.

Once the EzForge tier has its forge resources set, final users are empowered by allowing them to design their own front-end layer (or forge user interface) by means of composing user interface building blocks called gadgets, which are endowing with the forge resources. Following this approach, users can mix and compose forge resources on their own, allowing them to choose the best resources to meet their needs. User can even include external resources, such as Google Maps or RSS feeds, into their UI, using all of them as a whole. They will use whichever resources they like to create ad hoc instant forge UI, encouraging resources mashup, and following the DIY ("do it yourself") philosophy.

4 Savoir-Faire in Software Production Environments

More often than we can imagine, developers are plunged in the ocean of tools and procedures required in their daily work. Until now we have defined a forge (EzForge)

as a development platform but project responsibilities are delegated to developers and the management of developers' know-how is not taken into account. How do we materialize the know-how of your developments? How can we assure that our software products are developed as defined by the organization? These questions represent some of the factors that guide organizations to consider the materialization of their know-how and of their internal procedures in somehow for helping organisations to avoid or overcome barriers and hurdles raised during their work. For example some of these elements are the integration of new developers within development teams and quality assurance with respect to the requirements of quality models such as CMMI®.

This is a cornerstone in our software developments and it is part of the knowledge management (KM) broached by our architecture. One of the competitive advantages for organizations is their know-how, their human capital. Therefore we need to make explicit tacit knowledge in order to share information and to promote the savoir-faire within the organizations. In this sense in the area of KM Peter M. Senge [9] defines the learning organizations and he states five interrelated disciplines for the creation of smart and competitive organizations. In our approach we have used method engineering approach as the way to make explicit tacit software production processes and methods in order to spread knowledge within the organization.

Method engineering approach is used for several software developments and approaches and we have applied this approach in our context. In fact we have adopted Software Process Engineering Metamodel (SPEM) 2.0 as a language for the definition of software development processes and by the Eclipse Process Framework (EPF) (www.eclipse.org/epf),as a tool support for defining processes and methods in a Eclipse-based environment. The main idea is to define a methodology and relates method elements EzForge resources required for the software development. The huge number of resource-oriented services that are envisioned to be available in an Internet-scale networked forge will become unmanageable and thus useless for its users. Even if a repository service is provided, it will eventually become difficult for software development stakeholders to find out which resources (i.e. tool services) are appropriate for their development process.

This is the reason why we have created dedicated catalogues. In fact they provides navigation services for software development stakeholders and help them to find out which resources (i.e. tool services) they need to create the mash-ups they want. EzForge provides a user-contributed, "living" catalogue of resources founded on the Web 2.0 vision for user co-production and harnessing of collective intelligence (see Fig. 2.). This would provide all stakeholders with a collaborative semantic Wiki, and tagging and searching-by-recommendation capabilities for editing, remixing and locating resources of their interest.

The catalogue sets out the knowledge available within a certain community for composing resources (e.g. a method from its fragments) in a graphical and intuitive fashion and for sharing them in a world-wide marketplace of forge services.

The catalogue allows users to create complex mash-up solutions by just looking for (or being recommended) "pre-cooked" or off-the-shelf resources and customizing these resources to suit their personal needs and/or the project requirements, interconnecting resources, and integrating the outcome in their development workspace. These decisions are defined during the development process definition.

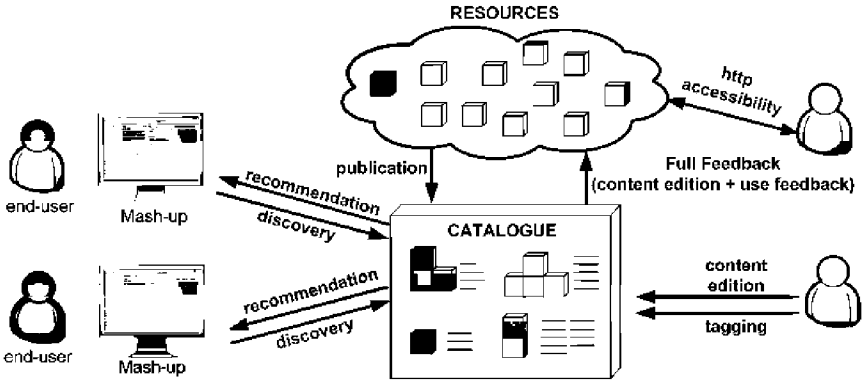

Fig. 2. Cataloguing Resources

"Folksonomies" of user-created tags will emerge, grow as users add information over time and act as important facilitators of a useful marketplace of resources for the networked forge.

Earlier approaches to service discovery and description like UDDI are not adequate to support human beings in easy resource retrieval and evaluation. By contrast, the exploitation of collective intelligence and user-driven resource categorization is beneficial for users.

A straightforward application of our savoir-faire approach using the catalogue is split in four steps:

- Evaluation of new developments: taking into account previous experiences, method engineers evaluate a new software development. In this phase, Knowledge Management plays a relevant role identifying software development phases, tasks and problems that are resident in developers' minds. Method engineers should evaluate previous experiences and clearly specify what objectives of this new development are.
- Selection of method fragments based on previous experiences: method engineers select the appropriate set of method fragments fitting software requirements. In fact in this context each method fragment is related to a set of Web2.0 resources. A basic catalogue contains the relationships between software processes and Web2.0 resources. Each task is related to workproducts representing a resource and therefore method engineers could specify the appropriate tools support at each software development stage.
- Composition of method fragments in order to produce a software development process used in the organization. In this step the selected method fragments are composed defining a flow that it is guided by the methodology. This composition states which are the selected resources at each stage of the development process. This approach is similar to Business Process Execution Language (BPEL) where Web Services are called following a specific order and sequence.

- Deployment within the organization. The resulting software development process is represented as a model and it is used by our forge. At this step and following CMMI® terminology, the result represents a defined and managed Standard Software Process (SSP) for an organization. This is a requirement for organizations aiming to achieve compliancy with CMMI® level 3.

This novel approach uses a method engineering approach in order to make explicit the savoir-faire in software developments within an organization (see Fig. 3.).A catalogue contains relationships between method fragments represented by the methodology using SPEM2.0, and resources that are represented within the forge as aggregators or connectors. Method engineers select the required method fragments needed for their software developments. In this context they select indirectly a set of aggregators and/or connectors that are related to specific resources. These resources are the basic tools within the development environment. Therefore we are reducing the gap between methodologies and software development tools support. This process allows the customisation of the resources and therefore the user's interfaces.

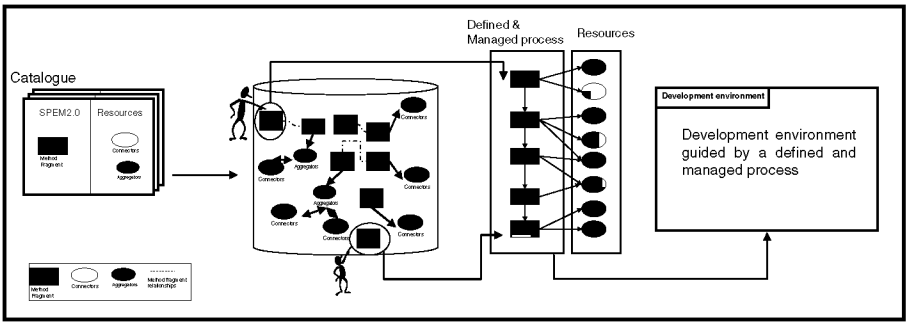

Fig. 3. A developer's environment customisation

5 Method Engineering and EzForge Architecture: A Holistic View

EzForge is a highly configurable and extensible user-centric collaborative software development tool that follows a novel mashup-based lightweight approach, given by the EzWeb core technology. Its user interface is defined by the user himself, who is able to make it up by assembling a set of small web applications called gadgets, which are the face of the services being offered by the forge. Up to now, there have been several attempts to bring mashup-based tools to the organizations with satisfactory results.

But with regard to software development, open source development tools don't take into account a key point in the software development within organizations: quality.

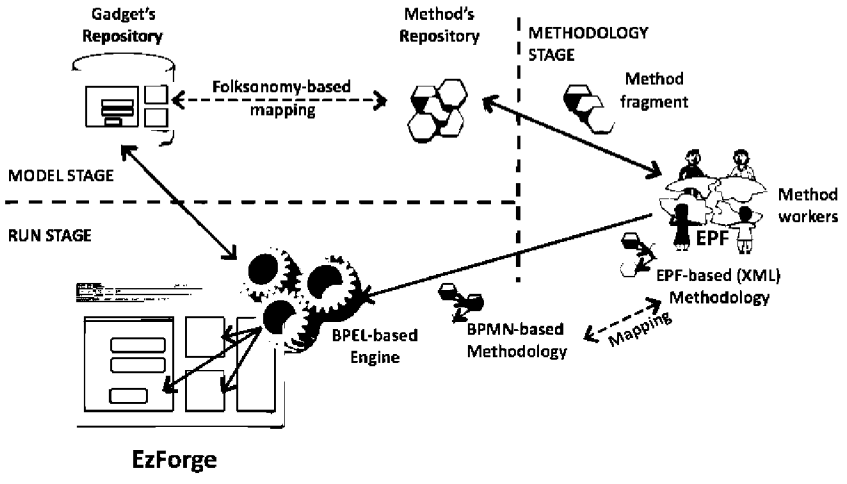

Fig. 4. A holistic view of Method Engineering and EzForge

Method Engineering and EzForge architecture (Fig. 4.) is compatible with the savoir-faire process defined previously and technically it is defined into three levels:

Model stage. The goal of this stage is to link the available gadgets from the EzForge catalogue with the method fragments that exist in the method repository and that will conform lately the used methodology. This catalogue provides access in an automated way to EzForge catalogue in the execution level. For this purpose, we have developed a folksonomy-based mapping, which allows us to create that link by using social tagging techniques. By using these tags we will be able to choose the gadget or gadget group labelled with the method identifier in methodology run time in an easy way. Besides, it gives us a way to incorporate the organization internal knowledge about how things work better, as it is their own developers who carry out this tagging process.

Methodology stage. It is in between model and run stages, and as we said before, it is where method workers select the method fragments that will make up the organization's methodology. To do so, method workers use the Eclipse Process Framework, which helps us to get, among other things, an XML representation of the methodology.

Run stage. Once we've got the methodology, the next step is to put it in execution by means of a workflow engine. Thanks to this, EzForge can choose the appropriate and required development tools depending on the ongoing development phase.

Thus, our proposed infrastructure takes the advantages of method engineering and brings them all to EzForge, allowing companies and organizations to have a user-centric collaborative development tool which can guide its users through the development process.

At this level the resulting and running application, Fig. 5. is shown using a web browser. Gadgets presented in this interface are those that have been defined by the method engineer when he was defining the organization's standard software process.

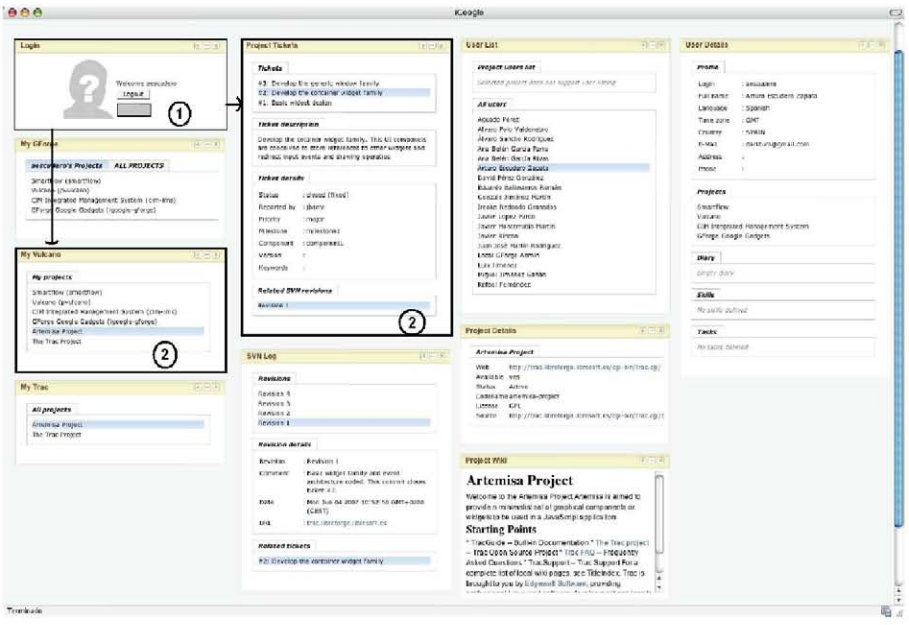

Fig. 5. Runtime execution overview

Obviously there are some permanent gadgets in this interface but most of the gadgets are configured during the model and methodology levels. Once we start/continue a software development, these gadgets are modified accordingly to a software development phase. In Fig. 5. the main gadget marked as "1" acts upon the existence of gadgets marked as "2". Moreover there are other relationships amongst gadgets as they are highlighted in this figure. When we select an element in one gadget automatically there is a selection of the related elements in other gadgets. These relationships are not specified by the defined and managed methodology, they are implemented by the forge.

Why this approach covers quality practices?

Managers are not worried about the selected architecture but they are more focused on costs and quality requirements used for the developments carried out in their organizations. Method engineering & EzForge architecture combines quality practices and a development infrastructure based on Web2.0 assuring quality aspects, with low cost, open to new OS tools, it represents an integrated tool and it overcomes new developer's barriers.

But why this approach covers quality practices?

CMMI® is one of most used quality reference model and it comprises two representations: staged and continuous. Whatever CMMI® representation stakeholders select for their adoption, there is a common problem: a separation between process areas due to a scarce tool support from a holistic perspective. Nowadays engineering practices

and process/project practices are separated and one of the main tasks for adopters is to assure that all process areas are coherent and consistent among them. Method engineering & EzForge architecture assures quality practices because the process defined and managed is used accordingly to its specification, and the development forge is guided by the methodology designed. In addition it also covers engineering and support process area because it provides an integrated development environment gathering requirements and configuration managements.

6 Conclusions

The presented approach combines method engineering and Web2.0 technologies in order to create a new generation of software developments tools and methods. Our approach starts from an explicit definition of the main tasks that a developer should carry out and its development environment is modified with respect to the organisation's development process. This novel approach combines an extendable development environment based on Web2.0 technologies with quality practices and tools. As results, we reduce the gap between development tools with low cost; we implement an extendable environment where we can select the appropriate tools support, and quality practices and tools are assured by this service based architecture. Web2.0 technologies relevance is becoming during these last years a new wave in several environments and method engineering approach provides architecture to make explicit the knowledge managed by organizations. Our experience combining both approaches places us on the road for a new tool generation of software developments which benefits are:

- Facilitate collaboration in heterogeneous contexts: Web2.0 technologies facilitate software developments on the Web.
- User interface configuration: another advantage that comes directly from using method engineering and EzForge altogether is the possibility of generating the user interface based either on the methodology used. Thus, when creating a new project on the forge, it will be able to help the user to choose the tools to be used
- Help developers to add new tools within the development process.
- Compliancy at CMMI® levels 2 and 3 through the definition of defined and managed standard software processes. The use of method engineering opens the door to the definition and management of the development processes. That is why EzForge will give support, for example, to the use of CMMI®. This will make it possible to ensure that carried out developments will place the organization in a certain maturity level, allowing an improvement of the methodology used
- Establish a relationship between process a project management tools (EPF) and engineering tools (forge).
- Provide a new tool in the knowledge management area through the use of method engineering

Currently we are applying this approach in several test cases and projects. A step forward in this development is to provide facilities to the forge in order to collect all kind of metrics. This characteristic will provide a better understanding of our current developments.

Acknowledgements

This work was funded by the Vulcano project (Proyecto Singular Ministerio Industria, Turismo y Comercio- FIT-350503-2007-7). We would like to thank Vulcano partners for their great feedback.

References

McAfee, A.: Enterprise 2.0: The Dawn of Emergent Collaboration. MIT Sloan Management Review 47(3), 21–28 (Spring, 2006)

Soriano, J., Lizcano, D., Cañas, M.A., Reyes, M., Hierro, J.J.: Fostering Innovation in a Mashup-oriented Enterprise 2.0 Collaboration Environment. System and Information Sciences Notes 1(1) (July 2007); ISSN 1753-2310

Huang, L., Boehm, B.: How much Software Quality Investment is Enough: a value-based approach. IEEE Software 23(5), 88–95 (2006); Digital Object Identifier 10.1109/MS.2006.127

Campanella, J.: Principles of Quality Costs. American Society for Quality Press; ISBN: 0-87389-443

Crisis, M.B., Konrad, M., Shrum, S.: CMMI® Second Edition. Guidelines for Process Integration and Product Improvement. Addison-Wesley, Reading; ISBN 0321279670

Booch, G., Brown, A.W.: Collaborative Development Environments. Advances in Computers 59 (2003)

EzForge project website, `http://ezforge.morfeo-project.org/lng/en`

Fielding, R.T.: Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine (2000)

Senge, P.M.: The fifth discipline. Doubleday (1990); ISBN 0-385-26095-4

Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Inf. Software Technol. 38(4), 275–280 (1996)

Henderson-Sellers, B., France, R., Georg, G., Reddy, R.: A method engineering approach to developing aspect-oriented modelling processes based on the OPEN process framework. Information and Software Technology, doi:10.1016/j.infsof.2006.08.003

Larrucea, X.: Method Engineering Approach for Interoperable Systems Development. Journal Software Process: Improvement and Practice (2008), doi:10.1002/spip.371

Software Process Engineering Metamodel (SPEM) 2.0. Object Management Group, `http://www.omg.org/docs/ptc/07-11-01.pdf`

Driver, E., et al.: Road Map to an Enterprise Collaboration Strategy. Forrester Research, August 2 (2004)