Volume 5, Issue 9, September 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# An Integrated System for Collaborative Work in Software Development Education: Requirements and Basic Functionality

**Nadezhda Filipova***
Department of Informatics, University of Economics
Varna, Bulgaria

***Abstract— IS-SDE is an integrated system for collaborative work in software development education. It is a response to the innumerable problems and challenges of education, and the ever-growing needs of software companies. The research paper discusses the driving forces fueling the development of IS-SDE. The scope of the system is defined, and its key characteristics are described. The functional and nonfunctional requirements of IS-SDE are identified. Its major functions and modules are outlined. Some directions for the further improvement of the integrated system are also pointed.***

***Keywords— IS-SDE, software engineering, CALM, ALM, collaborative software development, development tools***

## I. INTRODUCTION

Software development process nowadays can be extremely complex, and it has changed significantly during the 21st century. The requirements towards software systems are also exposed to rapid changes. The systems undergo many modifications, and it is necessary to maintain various system versions. Qualitative changes occur within software development methodologies and methods. Very often, different development methodologies, tools, and technologies have to be integrated for the purposes of building a new system. The pace of innovation in software development today is faster than ever: new software products and development environments appear, new versions of theirs come into being, development methodologies and standards are updated. Hence, to preserve and increase their competitiveness, software companies must quickly adapt to **constant changes**.

There is no doubt, teamwork is the backbone of modern software development. There are large, geographically distributed and multi-national, teams. This is a serious challenge in software development management: besides planning the process, it is necessary to support effective communication among team members, to organize sharing of various work products (such as specifications, models, and code), to distribute tasks, and to track their fulfillment status. Some authors affirm that globally distributed software development is an issue of increasing significance nowadays [1].

Being in a constantly changing business environment, software companies must adopt new development approaches, methodologies, technologies, environments and tools. As a result, a new class of integrated development environments has appeared in the recent years that aim at collaborative work and application life cycle management. These tools are designated as CALM or C/ALM (Collaborative Application Lifecycle Management). CALM is a strategic necessity in today's business-driven world. These tools ought to be deployed in software companies, on the one hand, and be adapted for the purposes of software development education in universities, on the other. So, universities must move towards training the students in **collaborative** (distributed) **software development**.

In response to contemporary challenges, an integrated system for collaborative work in software development education (IS-SDE) has been developed. It is grounded on a platform for collaborative development, and supports various approaches and methodologies for developing different kinds of software applications. Thus, the aim of this research paper is to define the scope of IS-SDE, and to systematize the requirements towards the system and its basic functionalities.

## II. DRIVING FORCES FOR THE DEVELOPMENT OF IS-SDE

The development of the integrated system for collaborative work in software development education has been fueled by two major driving forces (fig. 1):

1) The problems within software development education in universities, and
2) The evolution of development environments.

**The problems within software development education** concern both teachers and students. These problems stem from the dynamics in the field of IT, and in particular from the changes in software development. The dominant practice is to use many non-integrated software products in the different courses offered. Usually, the access to servers is only within the university campuses, and this hinders students' self-preparation.

It should also be noted that installing, configuring, and integrating modern development tools is a task too difficult: the requirements towards the IT infrastructure are higher and higher; more and more hardware and software resources are needed – e.g., servers and server operating systems, database management systems, web servers, application servers,

 

program frameworks, etc. On the one side, it means huge demands on universities' computers, which increase year after year. High rates of renewal are necessary, so that universities are a step ahead of software companies. But very often this is just impossible.

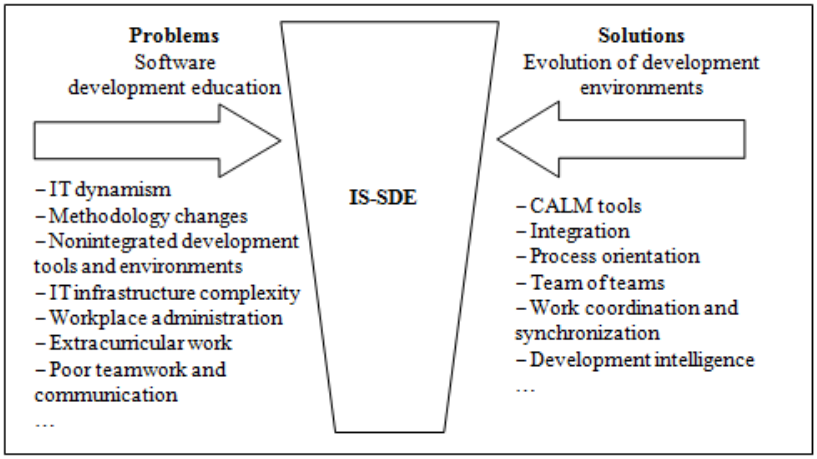


Fig. 1. Driving forces for the development of IS-SDE

On the other side, the experience in software development training shows that students face many difficulties when configuring a working environment on their own machines for the purposes of their extracurricular work. Usually they use the development tools in autonomous mode; often their computers are too weak to meet the requirements of the modern tools, and that hampers their adequate usage. It should be pointed, that similar problems occur in configuring and administering students' workplaces for software development education in universities by the IT staff.

It is necessary to work in teams in the professional development of software – this is a key requirement. According to the author's observations and surveys among students, principally they use some communication tools (such as Skype, Facebook, etc.), as well as some tool for remote access and computer management (e.g. TeamViewer) for the purposes of their teamwork. But this is not enough. The embracement of special tools for centralized storage of data and work products in software development appears to be indispensable – a trend which is already underway in software companies.

**The emergence of CALM tools** is the second prerequisite for the development of the integrated system for collaborative work in software development education – IS-SDE. In brief, CALM tools support the organization of software developers in teams, and the management of their collaborative work [2]. They enable the overall integration of application life cycle activities – from requirements definition to deployment, and maintenance. CALM tools facilitate the tracking of project progress and risk, and provide many reports. Among their features are [3]:

- Project management;
- Version control;
- Work item tracking;
- Test scenario management;
- System builds maintenance, etc.

CALM tools are characterized by a high degree of *integration*: they cover the overall life cycle of software systems, and support the work of various roles in the development process – system analysts and architects, application developers, testers and QA specialists, project managers, and so on. Various development tools (from different providers), used during the stages of the life cycle, can be integrated on the basis of CALM. There are conditions created for parallel work on several projects, and for organizing "team of teams".

CALM tools can be determined as *process-oriented*. They support various development methodologies. Moreover, methodologies can be adapted for special purposes, and used together (which is the dominant practice in software companies). If necessary, users can define a development process of their own. CALM provide a repository, and a centralized access control to the work products.

CALM environments bear the responsibility for coordination and synchronization of the participants in the development process [4]. Real-time communication tools are provided, and online discussions are supported. Event notifications are sent – e.g. change of a work product or deadline for task submission. CALM provides rich information for the work products, and manages the access to them.

Development intelligence is a key advantage of CALM [5]. Its essence is in applying business intelligence techniques, ingrained in DBMS, to software systems development. Using innovative analysis techniques, development intelligence traces the status of the projects and produces trend assessments. Many reports, graphs, and charts visualize the progress of the project: they are derived directly from team activities in a real-time mode. Development intelligence grows over time, and penetrates team's culture. It helps project managers in steering the teamwork, managing risks, and assessing projects [6].

Some examples of CALM and ALM products are: Microsoft Visual Studio ALM, IBM Jazz, Oracle Team Productivity Center, CollabNet TeamForge, HP ALM, Serena Software, etc.

 

Evidently, moving towards modern tools such as CALM is indispensable. On the one hand, they should be studied as tools for development, because they are already adopted and applied in software companies. On the other hand, they are a prerequisite for solving some problems in software development education pointed above.

It is essential to be noted that the introduction of CALM environments in education process involves a set of difficulties. These are complex system, and the cycle of their learning, adoption and deployment might be too long. Their requirements towards the IT infrastructure are high. Besides, they have to be adapted for the specific purposes and requirements of the education in software development.

Therefore, in order to apply CALM tools in software development education, and solve the problems of the process organization, it is necessary to build and configure an own integrated system for collaborative work in software development education. Combining various development tools and environments through CALM, this system should support the collaboration of students and teachers altogether. It should be compatible with the existing IT infrastructure in the university. The system should include appropriate tools for supporting, configuring, adapting, and managing all of its own elements – such as network, storage, virtualization, operating systems, execution environments, middleware, data, and applications.

## III. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS TO IS-SDE

The software system requirements describe what the system should do, but now how. They represent the services provided by the system, as well as some constraints on its work. There are numerous classifications of system requirements, but in this research paper their categorization into functional and non-functional is accepted [7].

The functional system requirements are statements about the services of the system. They define the reaction of the system to user inputs, or its behavior under certain circumstances. Sometimes, system requirements state what the system should not do. On the basis of the functional requirements the architecture of the system is defined.

In order to fulfill their objectives, the functional requirements must be comprehensive, and compatible (i.e. uncontradictory). Defining comprehensive and compatible system requirements might be extremely difficult (if not impossible) when building large and complex systems. One of the main reasons is that system users themselves often declare inconsistent and conflicting requirements, or they just don't know what they want. Sometimes system architects may also make mistakes in requirements definition, regardless of the techniques and tools they use, and their expertise.

The non-functional requirements represent some constraints on system operations and functions. Usually they are not concerned with a specific service of the system, and refer to the overall system architecture. It is needed to note that quite often some non-functional requirements (e.g. performance, response time, availability, and reliability) may have a higher priority than the functional requirements, as they affect all the users. Failing to meet some non-functional requirements can make the system unavailable. Sometimes the non-functional requirements may represent characteristics of the system development process: e.g. deadlines, budget, and resources needed. Others arise from the quality standards, security policies, or laws.

Usually the specific functional requirements are implemented by separate system components. But it might be more difficult to relate non-functional requirements to system components: their support might be dispersed through the system.

While defining system requirements, it is necessary to take into consideration that this is a complex nonlinear process. The multiple dependencies among the system requirements are the main reason for this situation – e.g. a non-functional requirement can be in relation or in conflict to other non-functional or functional requirements. The requirements can be prioritized: some of them could be implemented at further stages of system improvement. While developing the software system, its requirements can be refined and decomposed, and new requirements may be derived. Moreover, sometimes it is difficult to determine if a certain requirement is functional or non-functional.

Having in mind all that has been said above about the system requirements, let's continue with IS-SDE.

On the one hand, IS-SDE targets solving some problems in software development education (stated in §II); on the other hand, the system should admit rapid adoption and deployment of new development technologies, in accordance with the achievements of software engineering theory and best practices. Students must be able to use and practise modern development tools.

Hence, IS-SDE has to integrate a set of miscellaneous development tools (local and remote), supporting the overall life cycle of software systems. Simultaneously, the integrated system should provide the conditions necessary for the collaborative work of students and teachers in developing examples, homeworks, and course projects – both within and outside the university. It is necessary to provide opportunities for flexible enlargement of the set of development tools (which change at a rapid pace). Students should be granted a permanent access to the development tools and the software systems that they work on, regardless of their geographic location, and so they will be able to manage their time in a convenient way and practise themselves in real conditions. Students must work on joint projects and share work products. IS-SDE ought to support the work of many students' teams which are changed each term: their establishment and management is a true challenge.

The requirements to IS-SDE are identified, taking into consideration the research of software development methodologies and development environments, as well as the peculiarities of software development education, and the experience in this field.

The functional and non-functional requirements to IS-SDE, that have been determined initially, are summarized in Table 1 and Table 2, respectively.

 

TABLE 1 MAIN FUNCTIONAL REQUIREMENTS TO IS-SDE

| N | Functional requirement description |
|---|---|
| F1. | Providing development tools for the individual and collaborative work of students on their assignments that can be used inside and outside of the university. |
| F2. | Maintaining the core processes of software development in some courses (such as "Development Environments", "Business Modeling", "Programming and Data Structures", etc.) in the university. |
| F3. | Integrating development tools (local and remote), supporting the overall life cycle, and creating development infrastructure. |
| F4. | Faster and easier adoption of new tools and permanent version renewal. |
| F5. | Flexible distribution of resources needed for the students – access to software products and systems, disk storage, databases, program code, models, etc. |
| F6. | Creating and supporting infrastructure for application hosting and execution, and for storing their data. |
| F7. | Flexible assignment of responsibilities for access and permissions management among all the users (incl students). |
| F8. | Providing web and graphic tools for access to the system repository. |
| F9. | Providing a variety of communication tools – e.g. instant messenger, video, voice, email, event notifications, etc. |
| F10. | Support of a centralized repository for work products of software systems which being developed, and repository management tools. |
| F11. | Providing personal local workplaces for students and teachers. |
| F12. | A permanent and controlled access to system resources during the study period (if necessary, the access is scheduled). |
| F13. | Possibilities for teachers self-preparation and practice for laboratorials (individual or teamwork); maintenance of presentation materials. |
| F14. | Access virtualization and integration of resources from different networks. |

TABLE 2 MAIN NON-FUNCTIONAL REQUIREMENTS TO IS-SDE

| N | Non-function requirement description |
|---|---|
| N1. | Applying available technologies, development environments and tool in building IS-SDE, as well as in the education process. |
| N2. | Minimal software installation, and a strong focus on minimizing hardware requirements – disk storage, RAM, CPU, etc. |
| N3. | Reducing software costs. |
| N4. | Reducing hardware costs. |
| N5. | Rapid creation and development of the system through using prebuild solutions, software packages, services, OS services and roles, etc. |
| N6. | Creating a scalable system. |
| N7. | Overall virtualization of the system. |

As it has already been pointed, in some cases it is under discussion if a certain requirement should be classified as functional or non-functional. Anyhow, grouping requirements into functional and non-functional facilitates their identification; and this is also important for building the system architecture.

By analyzing the group of the functional requirements to IS-SDE, they are divided into three subgroups, as follows:

1) Requirements concerning the overall infrastructure for software development: NN F2, F3, F4, F6.
2) Requirements concerning the system management: NN F5, F7, F12, F13, F14.
3) Requirements concerning the collaborative work: NN F1, F8, F9, F10.

## IV. BASIC FUNCTIONALITIES AND MODULES OF IS-SDE

IS-SDE requirements, summarized above, outline some of its key characteristics, and to be more precise:

- IS-SDE is based on prebuild solutions which are adapted, configured, and integrated to meet the system requirements;
- It is placed on a CALM tool;
- The system supports the processes of software development education which are based on software development methodologies;
- A set of widely used development tools and environments are integrated in IS-SDE;
- The system has a client-server architecture, the clients and the server are virtualized, a VPN is configured;
- Web tools are used to manage the system resources, the centralized repository, and the users.

IS-SDE is process-oriented: on the one side, the system has a process-oriented organization, and, on the other side, it supports process-oriented software development through the usage of conventional and agile development methodologies. According to the process approach, software development in IS-SDE consists of three types of processes (fig. 2):

 

1) Core or primary processes – they cover the software systems development, and hence they can be characterized as production processes.
2) Management processes – they are directed towards organizing the collaborative work in the system, e.g. establishing teams, managing teams' permissions, structuring the repository, and managing the work products saved there.
3) Support processes – they are concerned with system administration and configuration, installing and integrating new development tools, environments and tools.

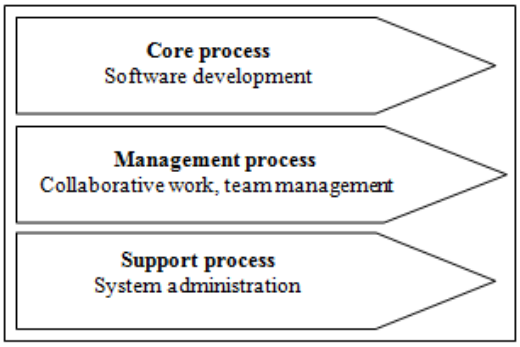


Fig. 2. Major processes in IS-SDE

An important requirement to IS-SDE is to integrate development environments and tools, and to cover the variations of the software development process in education. More especially, IS-SDE supports the following variations of the core process:

1) Team development of databases.
2) Team development of desktop applications and databases (this process includes the team development of databases).
3) Team development of web applications and databases (this process includes the first and the second one).
4) Team development of business models through UML CASE tools.

In these processes, IS-SDE provides the users with a set of tools necessary for the implementation of their individual responsibilities and for their collaborative work. These features of the system are accomplished by the **"Tools for software development in education"** module.

The organization of the "Tools for software development in education" module makes it easy to adapt IS-SDE, and support other variations of the core development process – e.g. further the system may be enlarged to support cloud applications development, and web services development.

The major roles in IS-SDE include students, teachers, and administrators. To use the system, users must register. The **"User management"** module manages user registration. The system supports online registration, and thus the teachers' burden significantly lessens. Students' accounts are managed through web tools by teachers.

IS-SDE users may play different roles depending on the development process in which they participate: e.g. a student performs the role of a Student in the registration module, he/she is a standard user in the operating system, and a contributor in CALM. Apart from these roles, the teacher may be a DBMS administrator, a project administrator in the CALM tool, or a site administrator in the web server. He/she also distributes resources in the system for collaborative work, using various tools some of which are web-based. One of the most important responsibilities of teachers and administrators is team creation and team permissions management.

The IS-SDE functions for DBMS management, teams' collaborative work organization, resources distribution, and web infrastructure management are scattered in specific modules for each role participating in the development process. In particular, these are the **"Software development management by students"** and **"Software development management by teachers"** modules.

The IS-SDE support processes are a responsibility of system administrators. The administrator can perform many roles such as DBMS administrator, CALM administrator, OS administrator, etc. The system administration is performed from the server part. Web-based tools placed on the system server are used for the purpose. The administrator maintains the VPN, the user accounts, the DNS server, etc. These functions are in the **"Administration"** module.

The "Administration" and "User management" modules are in the basis of the support process. The "Software development management by students" and "Software development management by teachers" modules sustain the management process, and the "Tools for software development in education" module is concerned with the core development process.

## V. CONCLUSIONS

IS-SDE provides infrastructure for the overall development of software systems in a number of university courses. It supports conventional and agile development methodologies. Students and teachers can play a variety of roles in the development process – e.g. business analysts, software architects, programmers, and project managers. The system equips the individual self-preparation and the collaborative work of both teachers and students, and it is accessible from any place. Within the university, the access to IS-SDE is from the computer laboratories, or from the students' devices

 

connected in wireless networks. There is VPN providing access to the system outside of the university. IS-SDE includes a number of graphic and web tools for software development management during the education process. Teachers, students, and administrators can use them to structure the centralized repository of IS-SDE, and manage the permisiions for the workproducts in the repository. They can organize teams and subteams, and grant access to specialized web portals for collaborative work, web servers, and FTP servers. The users register online to use the system, and their access to its resources is precisely managed. A hosting environment for the software applications being developed is also supported.

IS-SDE has been used in a number of courses in the area of programming and business modeling. It is essential to note that the system can be enlarged and adapted to support new variations of the core development process, and new development tools and environements can be integrated in it. Thus, the scope of the system would broaden to cover new courses.

The perspective is to integrate IS-SDE with an e-learning system, as well as with other university systems. The hardware infrastructure could be improved to raise system's relilability, and some administration tasks may be automated. It is much possible to use IS-SDE for other pusposes beyond software development – e.g mutual work on projects in various areas.

**REFERENCES**

[1] H.-Christian Estler, M. Nordio, C. Furia, and B. Meyer (2014), "Awareness and Merge Conflicts in Distributed Software Development". [Online]. Available: http://se.ethz.ch/~meyer/publications/empirical/awareness_icgse14.pdf

[2] M. Göthe, et al. *Collaborative Application Lifecycle Management with IBM Rational Products*. International Business Machines Corporation, 2008. [Online]. Available: http://www.redbooks.ibm.com/redbooks/pdfs/sg247622.pdf

[3] Dennis Minium (2006), "Team Foundation Server Fundamentals: A Look at the Capabilities and Architecture". [Online]. Available: https://msdn.microsoft.com/en-us/library/ms364062(v=vs.80).aspx

[4] Yvette Francino, "Enterprise software development collaboration with CALM and CDEs". [Online]. Available: http://searchsoftwarequality.techtarget.com/tip/Enterprise-software-development-collaboration-with-CALM-and-CDEs

[5] Carolyn Pampino (2011), "Five Imperatives for Application Lifecycle Management". [Online]. Available: https://jazz.net/library/article/637/#devintelligence

[6] Monica Luke (2011), "Improve predictability with Development Intelligence". [Online]. Available: https://jazz.net/blog/index.php/2011/10/28/improve-predictability-with-development-intelligence/

[7] Ian Sommerville, Software Engineering, 9th Ed. Addison-Wesley, 2011.

 