Delft University of Technology
Software Engineering Research Group
Technical Report Series

Tool Support for Distributed Software Engineering

Hans Spanjers, Maarten ter Huurne, Dan Bendas, Bas Graaf, Marco Lormans, Rini van Solingen

Report TUD-SERG-2006-008

TUD-SERG-2006-008

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
`http://www.se.ewi.tudelft.nl/techreports/`

For more information about the Software Engineering Research Group:
`http://www.se.ewi.tudelft.nl/`

Tool Support for Distributed Software Engineering

Hans Spanjers and Maarten ter Huurne
Philips Applied Technologies
The Netherlands
{hans.spanjers, maarten.ter.huurne}@philips.com

Bas Graaf and Marco Lormans
Delft University of Technology
The Netherlands
{b.s.graaf, m.lormans}@tudelft.nl

Dan Bendas
University of Oulu
Finland
dan.bendas@oulu.fi

Rini van Solingen
LogicaCMG and Drenthe University
The Netherlands
rini.van.solingen@logicacmg.com
solingen.r.van@hsdrenthe.nl

Abstract

Developing a software system in collaboration with other partners, and on different geographical locations is a big challenge for organizations. In this article we first discuss a system that automates build and test processes: SoftFab. This system has been successfully applied in practice in the context of multi-site projects. Then, we discuss a case where it was applied to a more challenging type of collaboration: a multi-partner development environment. Furthermore, we investigate the underlying concepts of SoftFab and use them to define a list of features for systems that support distributed software engineering.

1. Introduction

Many forces make software development more and more an activity that is distributed over multiple geographical locations. Examples of such forces are acquisitions, outsourcing, mergers, time-to-market (round-the-clock development), and the (un)availability of a trained workforce [1][2]. Additionally, software is more and more developed in collaboration with partners located at different geographical locations. For example, within Philips an internal prediction was made that within the next five years, more than 90% of its software development is done in some form of collaboration. This does not mean that all software development is outsourced or done by suppliers, but that less than 10% of Philips' software will be completely developed internally.

Especially for software the trend towards engineering on different sites and with different partners is of interest, because software, compared to hardware, has negligible reproduction and transportation cost. Copying software code, reusing it, and sending it around the globe can be done in a split second and free of charge in a multi-site and multi-partner development environment. But as simple as it sounds, so difficult it is to apply this idea in a world where cultural and time differences, intellectual property interests, complex development environments, confidentiality issues, and so on, make collaboration difficult, leading to decreased development performance [3].

A number of problems involved in multi-site development have been described by Grinter et al. [4]. It appears that software engineering largely builds upon informal communication. This informal communication is essential for creating understanding among developers of what is going on in their software development processes, also referred to as *awareness* [5]. For multi-site development, a lack of such awareness leads to unexpected results from other sites, resulting in, for example, misalignment and rework [4]. Another problem for multi-site projects is finding the right experts when they are needed. Such communication problems not only exist for (remote) multi-site development, they already exist when developers are apart as little as 30 meters [6].

Beside communication, also technical issues play a role. The use of different tools and data formats, for instance, makes it difficult to easily exchange information and development artifacts [7].

Different types of solutions exist for specific distributed software engineering (DSE) problems. Typically improvements can be realized in the processes, technologies or organization of software engineering [8]. Some of the difficulties related to DSE can be addressed by the use of technical infrastructures that explicitly support DSE. Such DSE support systems should provide a means to connect the software development environments of different development organizations in a way that is both *acceptable* and *convenient* for the collaborating partners. This not only involves access to the created software development products, but also access to technical software development resources, such as tools and test equipment. At the same time the different development organizations should be able to stay in control of the work products and resources located at their site.

At Philips a system that was originally developed to automate build and test processes, called SoftFab, is now applied to support multi-site development as well. Its web interface makes it particularly suited for such projects. Already 40 projects at Philips have used SoftFab in a multi-site setup and the results are promising. Measurements show, for instance, that projects using SoftFab can reduce the budget required for testing by 30-35%. Therefore, we decided to take a closer look at this DSE support system. We asked ourselves the question: "If projects are so enthusiastic about this DSE tooling, what are the underlying concepts that make it successful?" We investigate the applicability of SoftFab to other types of collaboration that involve multiple partners, and take a closer look at it to find out the reasons for its success, with the intention to formulate them as features a DSE system should provide.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. Then, in Section 3 we provide an overview of SoftFab, discussing the main parts of the SoftFab infrastructure and its features that improve distributed software development. In Section 4 we present a case study describing experiences with SoftFab and show the benefits and shortcomings of SoftFab in a multi-partner environment. An extention to SoftFab, called SkyFab, is introduced in Section 5 and in Section 6 we discuss the features a multi-partner DSE support system, such as SkyFab, should have. We end this paper with a discussion in Section 7 and some concluding remarks in Section 8.

2. Related Work

Research in the area of global and distributed software development mainly addresses the lack of informal communication in such settings. Proposed solutions basically follow two strategies: 1) reduce the need for informal communication, or 2) ease, stimulate, and support informal communication, often by the use of Internet technologies.

Grinter et al. [4] follow the first strategy by proposing an organizational solution. They define several coordination models for dividing the work across the different sites. These models use different dimensions along which to divide the work. The idea is to co-locate work by the dimensions for which coordination is most difficult, thus facilitating informal communication for the coordination along that dimension. Other types of coordination mechanisms are then required to deal with coordination along the other dimensions. Such mechanisms can be either technical or procedural (processes). Interface definitions are an example of such a mechanism. The dimensions they propose are: functional area of expertise (co-locate experts), product structure (organization follows software architecture, c.f. Conway's Law [9]), and process steps (every site is responsible for a (series) of development activities).

In practice typically multiple dimensions are important leading to hybrid models. In the customization model, for example, one site develops the core product and other sites add features specific for a certain customer base. This model divides work along the process steps as well as the product structure dimensions.

The distribution of work across different sites inevitably introduces the need to transfer work products from one site to the other. Hand-off points define how and when sites perform these transfers and also specify the requirements for the involved work products [4]. Many technical solutions can be found to support the handling of these hand-off points.

Typically industrial companies develop their own supporting infrastructure for DSE. Fujitsu, for example has developed such a system [7][10]. This solution uses internet technology to allow for remote access to software development products and resources, within their company. Their work is focused on the technical challenges and does not specifically address multi-partner DSE.

Other research work on tool support for distributed software engineering often uses standard internet technologies or groupware technologies, such as peer-to-peer [11]. Lanubile et al. [12] present a web-based support system for distributed software inspection that supports both synchronous and asynchronous communication between the inspectors.

The tool we present not only allows communication by sharing of information, but in a sense also alleviates the need for informal communication by improving awareness [5] in an alternative way.

Next to an organizational and technological perspective, DSE problems can also be addressed from a process-based perspective, i.e. by deploying software engineering processes that explicitly support DSE. The CMMI, for instance, addresses some DSE problems in the process areas: Supplier Agreement Management and Integrated Supplier Management [13].

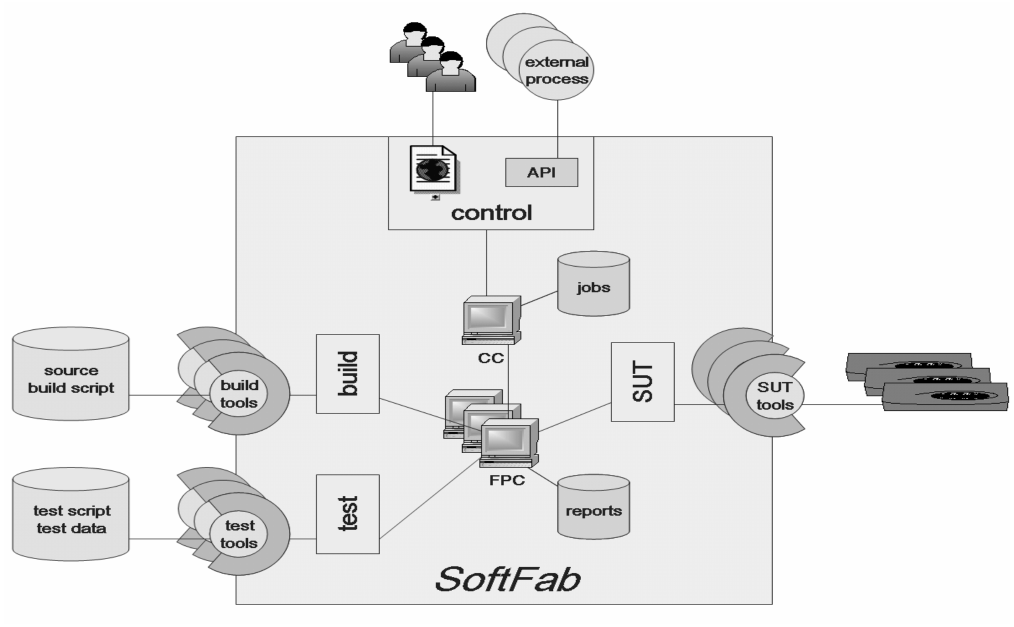


Figure 1. SoftFab architecture

3. SoftFab: A DSE Infrastructure for Automated Building and Testing

At Philips a software infrastructure is used to automate testing and building. This infrastructure, which is called SoftFab, enables projects to automate the build and test process, and control them remotely. SoftFab has been applied already in more than 40 projects. Figure 1 shows the SoftFab architecture. We call a SoftFab installation a "factory"; each factory consists of the following major parts:

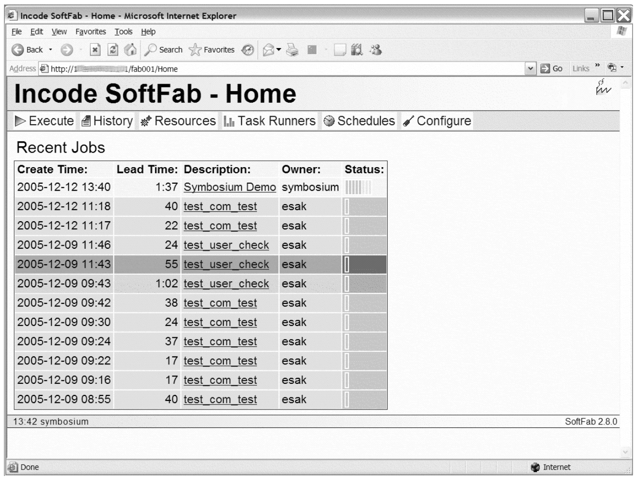
1. A Control Center (CC) for managing and controlling the factory
2. One or more Factory PC's (FPC's) capable of performing one or more tasks
3. A network for connecting the Control Center and all the Factory PC's

The Control Center is a central server that manages all the tasks involved in the test and build process. A task can be anything as long as it is finite and produces a result that can be interpreted by the Control Center. Tasks typically compile source code, execute tests, or transfer files. All defined tasks are stored in a database. SoftFab users can interact with SoftFab via a web interface. This interface allows users to define the properties of tasks (e.g., location of input files), cluster tasks that are often used together and schedule jobs for execution. Jobs execute a single task or a group of tasks. The web interface makes it possible to control these processes remotely in multi-site environments. The results of build and test tasks are also available via the web interface, giving all sites access to reports and log files. Role-based access rights ensure that selected users can manage or operate the SoftFab, while others are only allowed to track the status of jobs.

Figure 2 shows the execution queue in the main view of the SoftFab web interface. It contains a list of recent jobs that are either waiting, currently in execution, or finalized. Jobs are composed from individual tasks, which are simple, atomic activities, performed on Factory PC's. The progress of a job can be tracked by its tasks in the "status" column of the execution queue, where individual tasks are represented as vertical bars. A coloring scheme is used to indicate the status of a job. As tasks are completed, their color changes from white (waiting), via blue (executing), to green (complete success), orange (success with warnings) or red (failure). Failure of a task can prevent other tasks from executing, for example if a build fails then there is nothing to test.

Each executed job is stored together with all its configuration parameters, task results and reports, which can be inspected later. Figure 3 shows this job view for a specific job. It shows involved tasks, inputs, and links to the generated reports. This view represents a single line (job) from the job list in Figure 2 and can be opened by simply clicking that line.

A Factory PC is capable of performing one or more tasks. As such, it can be seen as a software development resource offering a specific set of capabilities. The Factory PC's in a factory are



Incode SoftFab - Home

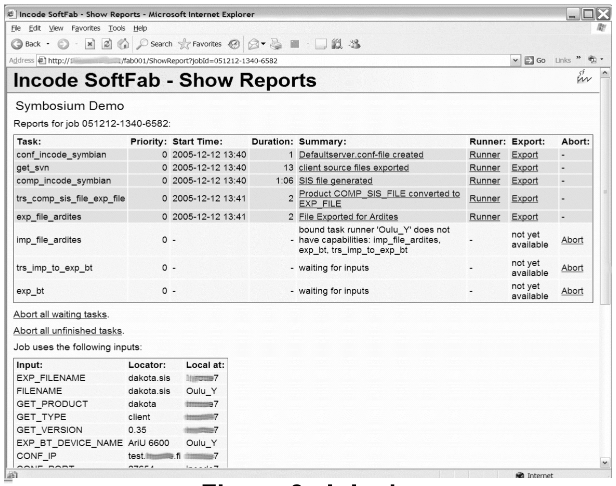
Recent Jobs

Create Time:	Lead Time:	Description:	Owner:	Status:
2005-12-12 13:40	1:37	Symbosium Demo	symbosium	
2005-12-12 11:18	40	test_com_test	esak	
2005-12-12 11:17	22	test_com_test	esak	
2005-12-09 11:46	24	test_user_check	esak	
2005-12-09 11:43	55	test_user_check	esak	
2005-12-09 09:43	1:02	test_user_check	esak	
2005-12-09 09:42	38	test_com_test	esak	
2005-12-09 09:30	24	test_com_test	esak	
2005-12-09 09:24	37	test_com_test	esak	
2005-12-09 09:22	17	test_com_test	esak	
2005-12-09 09:16	17	test_com_test	esak	
2005-12-09 08:55	40	test_com_test	esak	

13:42 symbosium SoftFab 2.8.0

Incode SoftFab - Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://1[redacted]1/fab001/Home
Go Links
Execute
History
Resources
Task Runners
Schedules
Configure
Done
Internet

Figure 2. Main view



Incode SoftFab - Show Reports

Symbosium Demo

Reports for job 051212-1340-6582:

Task:	Priority:	Start Time:	Duration:	Summary:	Runner:	Export:	Abort:
conf_incode_symbian	0	2005-12-12 13:40	1	Defaultserver.conf-file created	Runner	Export	-
get_svn	0	2005-12-12 13:40	13	client source files exported	Runner	Export	-
comp_incode_symbian	0	2005-12-12 13:40	1:06	SIS file generated	Runner	Export	-
trs_comp_sis_file_exp_file	0	2005-12-12 13:41	2	Product COMP_SIS_FILE converted to EXP_FILE	Runner	Export	-
exp_file_ardites	0	2005-12-12 13:41	2	File Exported for Ardites	Runner	Export	-
imp_file_ardites	0	-	-	bound task runner 'Oulu_Y' does not have capabilities: imp_file_ardites, exp_bt, trs_imp_to_exp_bt	-	not yet available	Abort
trs_imp_to_exp_bt	0	-	-	waiting for inputs	-	not yet available	Abort
exp_bt	0	-	-	waiting for inputs	-	not yet available	Abort

Abort all waiting tasks.

Abort all unfinished tasks.

Job uses the following inputs:

Input:	Locator:	Local at:
EXP_FILENAME	dakota.sis	[redacted]7
FILENAME	dakota.sis	Oulu_Y
GET_PRODUCT	dakota	[redacted]7
GET_TYPE	client	[redacted]7
GET_VERSION	0.35	[redacted]7
EXP_BT_DEVICE_NAME	AriU 6600	Oulu_Y
CONF_IP	test.[redacted].fi	[redacted]7

Incode SoftFab - Show Reports - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://1[redacted]1/fab001/ShowReport?jobId=051212-1340-6582
Go Links
Internet

Figure 3. Job view

connected via a network to the Control Center for exchanging information. Factory PC's are connected via standard network protocols, making it irrelevant where they are located and straightforward to set up a distributed factory.

Via Factory PC's, the SoftFab infrastructure interacts with the actual test and build tools. These tools are used as plug-ins in the SoftFab architecture. For the integration of specific tools and test equipment (SUT) it is necessary to develop glue-ware (wrappers) to allow SoftFab to interact with them. Besides tools, also test equipment can be made available via Factory PC's. As such, the capabilities a Factory PC offers are determined by what software is installed on it and what hardware is connected to it. On the other hand, a task defined on the Control Center requires a certain set of capabilities. The Control Center uses these capabilities to assign tasks to appropriate Factory PC's.

Besides the web interface, a Control Center also offers a programmable interface (API) that makes it possible to automate the control of a SoftFab. This makes it possible to integrate SoftFab in existing automated processes.

The Control Center is implemented in Python and the software that runs on the Factory PC is implemented in Java. It can be deployed on various operating systems, even multiple operating systems within a single factory. Furthermore, it is based on an open architecture: wrappers allow any tool that has a programmable interface (command line, COM, SOAP, etc.) to be integrated in SoftFab, enabling usage of both off-the-shelf development tools and in-house developed tools. Many wrappers are already available for mainstream development tools including Make, Doxygen, and JUnit. New ones can be developed by users, making SoftFab an effective backbone for tool interoperability.

Features of SoftFab that improve distributed software engineering include:

- The execution view displays the current activities at other locations, informing about the status of work and thereby increasing awareness.
- Engineers at different sites have access to the same reports and log files through their web browsers, facilitating communication.
- Work products and resources can be accessed, regardless of location, local time, language or availability of human resources.
- The Control Center coordinates tasks involving distributed resources, such as tasks involving building *and* testing executed on different computers, possibly on different sites.
- An overview of build and test procedures (tasks) is available on the Control Center. All details of the procedures are contained in the wrappers. This makes implicit knowledge explicit, facilitating new employees' training and allowing procedures to be easily transferred to other project teams.
- In-depth knowledge of a task is only required for initial implementation and for maintenance of associated wrappers, not for their execution. Therefore, tasks can be run by any user, without depending on the availability of an expert.

4. SoftFab Experiences

SoftFab's possibilities for DSE support can best be illustrated by a real-life example. In Finland we conducted a case study to investigate the applicability of SoftFab in collaborations that involve multiple partners. Due to, for example, intellectual property interests and confidentiality issues such collaborations pose extra requirements to DSE support systems compared to single-company, multi-site projects. We use this case study to explain the problems that are encountered in such a collaboration, and how SoftFab is setup and used. The case study involved three partners, each on a different development site:

- **System integrator.** Develops environment aware applications for smartphones.
- **COTS supplier.** Sells a database management system for mobile and embedded applications.
- **Testing subcontractor.** A research group at a university, which is specialized in software testing.

In our case study the system integrator replaces a certain layer of one of its mobile products with the data management solution developed by the COTS supplier. The testing subcontractor provides services related to the validation of the integrated product by executing tests and measuring performance for different software configurations. This migration project uses SoftFab as infrastructure.

One SoftFab factory is distributed over the COTS supplier and system integrator to share the releases of the COTS components. Another SoftFab factory is distributed over the integrator and the testing subcontractor. The first collaboration focuses primarily on the sharing of work products. The latter also allows the integrator to use the resources of the subcontractor. In the remainder we will focus on the latter as this application is more complex.

For deploying the SoftFab infrastructure a couple of activities need to be done at every site:

- Training of employees and analyzing the existing building and testing procedures.
- Installation and configuration of the SoftFab software on Control Center and Factory PC's.
- Development of scripts to automate several tasks previously preformed manually (e.g., retrieving configurations, or deploying installation packages to target devices).
- Development of wrappers linking existing automated tasks to SoftFab.

For the collaboration between the testing subcontractor and the integrator a testing facility is build at the subcontractor's site, which is connected to the integrator's site using SoftFab. The SoftFab Control Center is situated at the integrator's site, in a demilitarized zone (DMZ) that is accessible from outside of the company's intranet, and also acts as a bridge for transferring files over the firewall. One Factory PC is available at the same location, having access to company specific assets and tools. A computer at the subcontractor is connected as a second Factory PC. Its capabilities are limited to importing files from other locations and deploying and executing applications on target devices.

Via SoftFab the testing subcontractor can use Factory PC's at the integrator's site to build components from arbitrary versions of the source code, which can be used to build and test any version of the system. At the other site, the integrator can remotely execute tests at the subcontractor's site, using different versions of target hardware. Without SoftFab this would have required the integrator to request a test from the subcontractor. Then, the subcontractor has to wait for the right version of the application to be send, after which it can be tested and the test results returned. This illustrates that SoftFab can speed up the integration process significantly.

The mobile application is built at the integrator site from source code stored in their configuration management system, using their tools, licenses and scripts. The parameters for this task, such as product versions, library versions, and hard-coded constant values can be specified. After the application is built and packaged as a binary installation file, it is encrypted and exported to the subcontractor's Factory PC. In the final phase the install package is deployed on a target device via a Bluetooth link, after which the application can be executed for testing. This complete scenario can be controlled from a single (remote) location using SoftFab's web interface.

The testing subcontractor and the integrator used different types of tools that were easily plugged into the local SoftFab infrastructure by writing wrappers using their command line interface.

The benefits of this setup were especially visible during one specific experience at the subcontractor during a complete build-deploy-run cycle. This operation failed at the point where the binary should have been transferred outside from the integrator's intranet to the test equipment at the subcontractor's site. SoftFab's job view (see Figure 3) revealed that one of the engineers at the integrator's site was already re-executing some of the tasks involved in this job. So, somebody already noticed the problem and was busy fixing it. No telephone or email was required to understand the situation, to see latest progress, or to know that a solution was on the way.

This example shows that SoftFab increases developers' awareness, without the need for informal communication. Partners are able to understand the

Table 1. Collaboration scope

Scope	*Terminology*	*Sharing and collaboration*
multi-site	SoftFab	• 'standard' SoftFab benefits • test equipment, tools, test cases, test data
multi-project	inter factory resource sharing	• software licenses • equipment
multi-partner	SkyFab	• software development processes and procedures, • test equipment, tools, test cases, test data • protection of IP • respecting firewalls

situation based on the information shown by SoftFab and take action if needed.

We have used SoftFab in this case study in a multi-partner collaboration. This is a different type of collaboration than for which SoftFab was developed until now. Obviously, the partners involved also identified some shortcomings:

- All partners have the same level of access to all resources. Some partners wanted to have more control over their own Factory PC's. This is especially important when a project has to deliver to multiple customers.
- All partners have access to all work products. In general, however, partners require more control on sharing of work products, for instance, the ability to allow sharing between Factory PC's within one partner, but to limit sharing between *different* partners.
- Implementation details of processes of one partner are visible for all partners. It might take several tasks to produce a particular work product. This is not relevant for other partners; only the final product is. The processes of each partner need to be encapsulated, hiding its inner workings. Each partner should be able to control which of its tasks and work products are visible from the outside for reasons of sensitivity (IP) and clarity (abstracting from implementation details).

5. SkyFab: A Support System for Multi-Partner DSE

It turned out that the application of SoftFab in a collaborative environment with multiple companies was more difficult than collaboration within a single company. This suggests that there are different types of collaboration possible that might require specific support from a DSE infrastructure. Table 1 presents three collaboration levels and summarizes what is shared at each level. The levels are ordered according to their scope of sharing: the larger the sharing scope, the more difficult the collaboration.

A standard SoftFab setup, as discussed earlier, easily shares several resources, data, and procedures (e.g., via test scripts). Typically, every project implements its own factory. At Philips, SoftFab has also been used to share resources *between* projects. In such a case, each project owns a factory and the Factory PC controlling the resource is listening to the Control Centers of both projects, sharing the unique resource transparently for the end-user. The most challenging level of sharing, the collaboration between multiple partners (companies, universities, and so on), requires some additional features. A Multi-Partner DSE (MP-DSE) support system providing those features is currently being implemented as an extended version of SoftFab, and is called SkyFab.

The involvement of multiple partners makes software development more complex, for instance, when partners have their own policies on security and protection of intellectual property. To address this SkyFab will allow each partner to implement its own local factory behind a corporate firewall. A globally shared SkyFab Control Center is placed in a DMZ and is connected to the Control Centers of the local factories, thus forming a hierarchy of SoftFabs. Now one can run a job of which the tasks are executed in different partners' factories using the local available resources without breaking the partner's security rules. Figure 4 shows the architecture of a SkyFab factory. The setup of our case study can be seen as an intermediate step towards this architecture.

6. Features of a MP-DSE Support System

Below we list a set of desirable features for MP-DSE support systems. These features not only follow from the case study discussed above, but also from the experience of using SoftFab in about 40 different projects at Philips. Some of these features are supported by SoftFab; others are not supported by SoftFab, but appeared to be desirable in practice. These will be implemented in SkyFab.

6.1. Work product sharing

When software is developed collaboratively, work products (designs, documents, test results, code, executables, etc.) need to be shared among the different development sites. In our case study, for instance, the testing subcontractor can only test the application when the executable binaries, compiled at the integrator's site, are provided. This does not mean, that everything needs to be shared, it means that

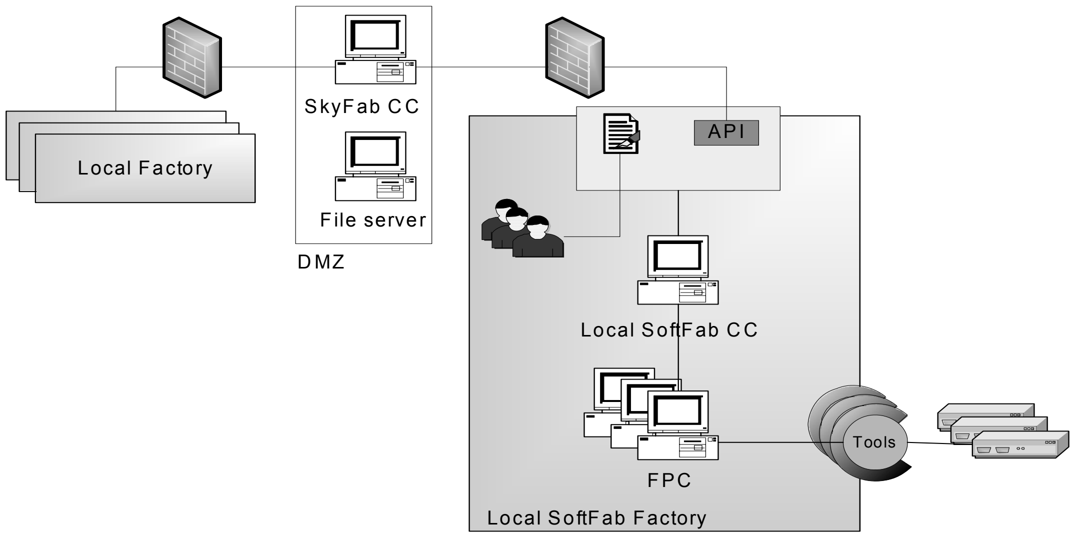


Figure 4. SkyFab architecture

sharing of work products needs to be decided upon and organized, i.e. handoff points [4] need to be defined. Sharing can be downloading or uploading of work products.

In SoftFab sharing of work products is already arranged both explicitly and transparently (and as extension to SoftFab, SkyFab has this ability as well). On the one hand, it is possible to explicitly define tasks that have the purpose of retrieving information from, e.g., a configuration management system and deliver the result via the web interface. On the other hand, tasks can be configured to require input from the site of a partner. During task execution this input is then retrieved transparently for the user.

6.2. Development resource sharing

Each partner typically has its own development tools and equipment, which are often even different between sites of one partner. Especially for embedded software development it can be difficult and expensive to replicate test environments on different sites, which, in turn, makes it difficult to reproduce test results on the different locations. In our case study, for instance, a testing facility was only available at the testing subcontractor's site. DSE support systems must be able to deal with these different technical environments without requiring technical expansions on other sites. In a collaborative environment, especially in the case of a multi-partner collaboration, one cannot expect other partners to rigorously change or expand their technical development environment.

Therefore, in some cases, collaborating partners should be able to start and control tasks such as compiling, building, testing, code generation, static analysis, etc. at other sites. This does not mean that every task must be fully remotely accessible or controllable, but it means that a dedicated selection of these tasks should be externally executable. As such, the testing subcontractor was able to compile a specific version of the application to test using the integrator's build environment remotely. The configuration of the tasks and their required tooling is the responsibility of the site where a task runs. Executing a task typically produces a work product, which again can be shared.

In a multi-site SoftFab setup, this is supported via the access to and control over the Factory PC's in other sites. Each Factory PC declares its capabilities explicitly: it declares which tasks it is able to perform and which output it is able to produce based on which inputs. In SkyFab a hierarchy of SoftFab factories will be introduced, enabling one partner to execute tasks in the local factory of another partner. This allows a partner to share its development resources, without releasing all control over them (see also Sec. 6.3).

6.3. Product and resource access control

Although it is necessary to share work products and resources between partners, not everything needs to be shared. A specific partner will need to allow access certain work products and resources. However, due to confidentiality issues, every partner needs to be in control of the accessibility of their work products and resources. Especially, when considering distributed development with different partners this is an important issue, as intellectual property or business interests, for instance, need to be protected. In our case study, for example, the source code of the product under test was not disclosed to the testing partner.

The role-based access rights mechanism in SoftFab currently defines three fixed roles. This is not sufficient for multi-partner collaborations, where a specific partner wants to control access to his local factory for each partner separately. In SkyFab access rights will be more fine-grained and will be managed per individual user. SkyFab uses a local SoftFab factory per partner. Each local Control Center serves as a kind of gatekeeper to the working environment at a specific site. Certain processes and work products can be shared with the other partners, while others remain private. It allows the other partners to execute operations on certain data and retrieve the results, without the need to access the data directly.

6.4. Heterogeneous environment support

Each partner has its own software development infrastructure. Some of the differences originate in the different role each partner plays in the collaboration, requiring different technical solutions. Homogenization of systems is not a solution: it would disrupt established ways of working and invalidate past investments, in addition to ignoring the fact that the diverse skills of the different partners are key to making a collaboration perform better than a single partner could. Thus, DSE tools should be based on an open architecture, allowing interaction with different systems from multiple vendors running on different platforms. This is especially important for MP-DSE.

SoftFab supports this by storing results in the native format of a tool. For example, test results are stored in the test report format of the specific test tool, typically a plain text, HTML or PDF. The result document is then accessible to other sites, for instance, via a web server. Other sites do not need to have licenses for these other tools but still are able to create and access the results. Furthermore, SoftFab enables a partner to execute test processes that use tools and equipment at a remote partners' site. One partner, for example, can test software in the test environment of other partners, without having the licenses for the involved tools.

6.5. Real-time status updating

Collaboration between sites and especially between partners requires a continuous insight in status of work and work products. In non-distributed settings this awareness is created by informal communication via email, telephone and in face-to-face meetings. DSE support systems should compensate for the lack of informal communication between remote locations and enable transparency in work carried out and the work products being produced, in order to increase developers' awareness. An example of tool-mediated awareness was presented in Section 4.

SoftFab supports this by providing on-line insight into the work carried out at other sites, showing status and result overviews of tasks carried out (see again Figure 2 and Figure 3). Depending on users' access rights (see also Sec. 6.3), they are able to access these overviews to find out what has been done and what the result are of the tasks executed. As developers can see for themselves what is going on, there is no need to consult other sites just to know the current status. They do not need to ask; SoftFab simply shows it.

Naturally, for more complex tasks, such as analyzing a difficult bug, direct communication between partners is still required. In such cases, SoftFab helps by providing a clearly labeled status overview of the tasks that were executed on all sites, and by providing access to all reports and log files to all developers involved. This avoids misunderstandings ("which version are we talking about?") and allows engineers to focus on the problem only.

6.6. Consistency and timeliness management

In our case study, the testing subcontractor needed to run its regression tests against the latest release of the product as well as against previous versions. As such, the subcontractor needs the correct versions of the application to be easily available.

If operations on software development data can be remotely executed, the need to physically distribute that data disappears. Sites are able to acquire work products when needed. Version checks will not be necessary, because the tested product is directly and remotely built from the configuration management system at the development site. Differences and delays due to development at different continents and time zones are overcome when a DSE support system is able to manage consistency and timeliness of work products automatically.

In SoftFab consistency is supported by ensuring that the actual developer or maintainer of a work product has it physically on its own site and allows sharing it with others. In case of sharing of resources, consistency and timeliness is managed by ensuring that the owning site also develops and maintains its resources. Additionally, the automation of tasks makes developers independent of the presence of people at other sites.

6.7. Knowledge transfer

The knowledge and experience of developers is incorporated in the work products they produce. The same applies to automated scripts used in a DSE support system. As such, the tasks that are incorporated in a DSE support system should hide complexity from their users. When this is done correctly, the transferability of work processes and sometimes of complete projects increases. When tasks are not automated, not archived and not documented in a standardized way, transferability of work products is only feasible face-to-face by teaching and handing over. By putting the knowledge in a standardized way into scripts, users can (remotely) control and execute these work processes without the need to understand their content. As such, complexity is hidden, increasing the transferability of work.

SoftFab supports this by standardized scripts (wrappers) that can be executed remotely, but are maintained locally.

7. Discussion

A set of features for MP-DSE has been introduced in Section 6 of this paper. The need for each individual feature differs for each application of MP-DSE. Naturally, the ability to share information is a necessary condition for any collaboration. However, many solutions offer that feature, a simple FTP-server would suffice. The other features are more specifically aimed at support for *distributed* software engineering.

For instance, the ability to share technical software development resources — together with work product sharing the most important feature of SoftFab — solves a number of practical problems often encountered in multi-site development project, such as the difficulty of replicating build and test environments on multiple sites. The ability to share both work products and resources, sets SoftFab apart from many other systems that can be used to support distributed software development, such as groupware systems.

Support for heterogeneous development environments is obviously only necessary in situations where the technical environments used at the different sites in a collaboration are actually different. A similar argument holds for fine-grained control of access to work products and resources, which is primarily relevant in cases were intellectual property is an issue.

Real-time status updates and consistent and timely access to work products are features that are not absolutely necessary, but they are very useful to increase the awareness of developers at different sites.

Finally, support for transfer of work processes makes it easier to hand-over a project to another site or to a customer, for instance, when a product has been delivered and enters the maintenance phase of the software life-cycle. Thus, indirectly such a feature improves its maintainability.

The need for the above features potentially exists in every phase of the software life-cycle, not only for building and testing. Currently, SoftFab mainly supports the build and test phases of a software project. Support for other phases is constrained by one limitation: if it cannot be controlled remotely (so at least partly automated), SoftFab cannot deal with it. Therefore, the applicability of SoftFab to other development phases strongly depends on the usage of automated tools in those phases. Previous research shows that in the early phases of the life-cycle, i.e., requirements engineering and architecture development, tool support is still limited [14]. During later phases usage of automated tools is more common. As SoftFab builds upon remote control and access, it largely builds upon automated tooling. So it seems that the applicability and benefit of SoftFab lies in later phases of the life-cycle.

However, more tools are expected to be used in earlier phases as well. For instance, if we consider the trend of model-driven development, and OMG's model-driven architecture (MDA) [15] in particular, we see that tool vendors develop more and more tools to be applied during architecture development. These tools automate development tasks, such as model transformation and code generation.

Also in the area of requirements engineering some software engineering activities are amenable for support by SoftFab. One of them is coverage analysis: determining the extent to which requirements are covered by other (downstream) work products. Tools for doing this automatically are being developed, e.g., [16]. By connecting such tools, a DSE support system could help to provide up-to-date status views of the requirements coverage of a software system under development. The underlying (potentially confidential) information does not have to be shown in a shared report, but it can be used for generating the coverage view. This way the actual progress of a project in terms of addressed requirements can constantly be monitored during the life-cycle.

SoftFab and SkyFab support such future developments by sharing the control and results of automated software engineering tasks. However, it should be remembered that even when the features discussed in this paper are all fully supported by a DSE support system, successful collaboration cannot be guaranteed. The success lies in the way the DSE support system is used, which is largely determined by the willingness of companies to collaborate, and their openness towards and confidence in each other.

8. Conclusions

DSE support is of large interest for today's industry. Software has an intrinsic power due to its relatively cheap reproduction and transportation cost. To fully benefit from this strength, DSE support systems need to be deployed, even across company borders. The SoftFab infrastructure discussed in this paper is based on industrial multi-site practice. It has been applied many times to full satisfaction of the involved partners. Furthermore, we discussed SkyFab based on the application of SoftFab as a MP-DSE support system.

Based on Philips' experience, using a system such as SoftFab to connect the development environments of partners that develop software in collaboration, while they maintain self-control, has great potentials to speed up software development. As such, we conclude that the road towards profitable, faster and more reliable software development lies in collaboration. DSE support systems are needed for that, preferably developed and build upon industrial best-practices.

We consider following to be our main contributions:

- We discussed SoftFab, an infrastructure for automating the build and test process.
- We illustrated the possibilities of SoftFab in a case study, revealing the strengths and the weakness of SoftFab in a multi-partner setting.
- We proposed an extended version of SoftFab, SkyFab, that addresses the issues revealed in the multi-partner case study, and identified three levels of collaboration that are relevant for MP-DSE support systems
- We introduced and analyzed seven features of a MP-DSE support system, giving a good overview of the requirements for developing or selecting such a support system.

In future research we will expand and implement the SkyFab concept. Currently, we are further evolving SoftFab into SkyFab. In this industrial case study we implement the features we identified to overcome the difficulties we encountered when applying SoftFab *as-is* in a multi-partner environment. After that we will investigate how to extend SkyFab to support other software development phases, as discussed in Section 7. We are already developing an application for analyzing and monitoring requirements in a distributed software development setting.

Acknowledgements

We would like to thank the ITEA organization and the national public authorities for enabling and supporting MERLIN (http://www.merlinproject.org/).

Partial support was obtained from NWO Jacquard, project Reconstructor.

References

[1] Erran Carmel. *Global Software Teams: Collaborating Across Borders and Time Zones.* Prentice Hall, 1999.

[2] James D. Herbsleb and Deependra Moitra. Global Software Development. *IEEE Software*, 18(2):16-20, March 2001.

[3] James D. Herbsleb and Audris Mockus. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Trans. Software Engineering*, 29(6):481-494, June 2003.

[4] Rebecca E. Grinter, James D. Herbsleb, and Dewayne E. Perry. The Geography of Coordination: Dealing with Distance in R&D Work. *Proc. Int'l Conf. Supporting Group Work (GROUP'99)*. ACM Press, 1999.

[5] James Chisan and Daniela Damian. Towards a Model of Awareness Support of Software Development in GSD. *Proc. 3rd Int'l Workshop Global Software Development (GSD2004)*, pp. 28-33, 2004.

[6] Thomas J. Allen. Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D Organization. MIT Press, 1977.

[7] Jerry Z. Gao, Fukao Itaru, and Y. Toyoshima. Managing Problems for Global Software Production – Experience and Lessons. *Information Technology and Management,* 3(1-2):85-112, January 2002.

[8] W.S. Humphrey. Managing the Software Process. Addison-Wesley, 1989.

[9] Melvin E. Conway. How Do Committees Invent? *Datamation,* 14(4):28-31, 1968.

[10] Jerry Z. Gao, Cris Chen, and David K. Leung. Engineering on the Internet for Global Software Production. *IEEE Computer*, 32(5):38-47, May 1999.

[11] F. Lanubile. A P2P Toolset for Distributed Requirements Elicitation. *Proc. 2nd Int'l Workshop Global Software Development (GSD 2003)*, pp. 12-15, 2003.

[12] Filippo Lanubile, Teresa Mallardo, and Fabio Calefato. Tool Support for Geographically Dispersed Inspection Teams. *Software Process Improvement and Practice*, 8(4):217-231, October 2003.

[13] Mary Beth Chrissis, Bart Broekman, Sandy Shrum, and Mike Konrad. CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley, 2003.

[14] Bas Graaf, Marco Lormans, and Hans Toetenel. Embedded Software Engineering: The State of the Practice. *IEEE Software*, 20(6):61-69, November 2003.

[15] OMG. MDA, http://www.omg.org/mda/, 2006.

[16] Marco Lormans and Arie van Deursen. Reconstructing Requirements Coverage Views from Design and Test using Traceability Recovery via LSI. *Proc. Int'l Workshop Traceability in Emerging Forms of Software Engineering (TEFSE'05),* November 2005.

TUD-SERG-2006-008
ISSN 1872-5392