Managing Software Development Information in Global Configuration Management Activities

Rafael Capilla, Juan C. Dueñas, and René Krikhaar

ABSTRACT

Software Configuration Management (SCM) techniques have been considered the entry point to rigorous software engineering, where multiple organizations cooperate in a decentralized mode to save resources, ensure the quality of the diversity of software products, and manage corporate information to get a better return of investment. The incessant trend of Global Software Development (GSD) and the complexity of implementing a correct SCM solution grow not only because of the changing circumstances, but also because of the interactions and the forces related to GSD activities. This paper addresses the role SCM plays in the development of commercial products and systems, and introduces a SCM reference model to describe the relationships between the different technical, organizational, and product concerns any software development company should support in the global market. © 2011 Wiley Periodicals, Inc. Syst Eng 15

Key words: software configuration management; global software development; collaboration; outsourcing; software product lines; product data management

1. INTRODUCTION

The disappearance of distances and geographic barriers due to the widespread usage of the Internet has led to an increasing globalization of software companies that employ resources located across the world. As reported in Osterweil et al. [2008zaq;1], "configuration management is a multibillion dollar industry that provides important support for software engineering practice" (e.g., commercial and open source software configuration management tools have generated a live marketplace [Estublier, 2000]). Software Configuration Management (SCM) has been defined "as the discipline of managing the evolution of large and complex software systems" [Tichy, 1988zaq;1]. It is a software engineering discipline that addresses many of the practical problems related to the identification, storage, control, definition, relation, usage, and change of the pieces of information, so-called Configuration Items (CIs), conforming to a software system at any stage of development and evolution. Any valid set of versions of related CIs represent the configuration to release or baseline, composed either by software or mechanical or electronic pieces, thus making part of the systems engineering management process.[1] Today, the concepts of SCM have been widely adopted by several technical management functions including systems engineering (SE) among others, as SCM is considered a closely related subfield for the SE process, and the *INCOSE Systems Engineering Handbook v3.2* is now aligned with the principles of ISO/IEC 15288:2008, *Systems and Software Engineering*.

SCM is supported by integrated product and process development activities. Hence, configuration decisions are needed to determine which CIs will be managed in a multidisciplinary environment. Because globalization increases the importance of managing all information produced during different stages of development, SCM techniques and tools must deal with multiple products and versions that can be produced anywhere. Multiple products, multiple projects running in parallel, multiple organizations distributed across the world at different locations, and multiple disciplines (e.g., hardware-software codesign) complicate the SCM daily operations. Moreover, the development and evolution activities carried out in a many-to-many scenario require the adoption of complex SCM solutions and tools to optimize the resources employed and to avoid the communication overhead between distributed teams.

In this context, we examine how SCM fulfils new requirements arising from emergent trends such as globalization, outsourcing/off-shoring, shorter time to market, highly customized products, and tight integration of hardware and software in embedded system development. Engineering multiple products under multiple projects and by multiple organizations in an integrated manner increases the complexity of SCM activities. Therefore, companies need new models to visualize and understand the aspects they need to put more resources in order to manage more efficiently the variations and the configuration issues of building multiple products in a decentralized manner. The remainder of this paper is organized as follows: Section 2 summarizes the related work. Section 3 outlines our SCM reference model for multiple product configuration and management. In Section 4 we discuss the role of the reference model on SCM activities, and Section 5 provides some industrial evidence of its importance. Finally, Section 6 outlines the conclusions.

2. RELATED WORK

Software Configuration Management (SCM) involves a variety of functions, methods, and techniques such as: version management, change management, build management, and release management. SCM builds upon the identification of CIs (any piece of knowledge or representation of physical element—usually supported as a file—produced or used during the project development, that must be controlled separately), baselines (sets of CIs that reproduce an approved system configuration in discrete time), and relationships between CIs.

Version management controls versions of existing assets to support producing a valid configuration of the target system. A repository is used to store CIs, versions for different software development activities. **Change management** supports decision-making and keeps records for all the changes made in a product during maintenance and development processes. During development it is related to new features to add, while in maintenance it is related to defects to be solved in the system. **Build management** concerns the creation of the final product by compiling and linking the right versions of the components. **Release Management** takes care of delivering and baselining the right components of a system that constitutes the product. All these SCM activities are strongly connected, and as such, these features are supported by existing tools, such as Telelogic CM/Synergy[2] or IBM ClearCase,[3] CVS,[4] its successor Subversion [Mason, 2005] and newbies as git[5] or github.[6] The key information that becomes critical for the organization is stored in appropriated repositories for the development teams to be retrieved in the future.

These activities focus on the control of changes and configurations for single products. It is usual to confine SCM activities to the last phase of software development—so CIs are mainly source code files— but there is already a sign of change which claims for the importance of SCM to create the appropriate links between software architecture and its implementations, as versioning constitutes a major issue in order to track the versions of multiple product configurations that have to be maintained or derived into the right instances [Nistor, Erenkrantz, and Hendrickson, 2005]. Engineering large and complex systems, often under time-to-market pressure, requires the participation of many developers working concurrently, and SCM allows concurrent access to software artefacts for supporting the necessary consistency of the changes made [Estublier and Garcia, 2005]. Such concurrent engineering activities require cooperative engineering for all the participants who have to be aware of the changes made and processes executed in parallel.

Nowadays, many companies are using software product lines[7] [Clements, 2002] in a competitive environment as a way to deal with multiple products at a time. Therefore, controlling the evolution of software components and final products in a product line context introduces a factor of complexity for SCM tools, as they must manage concurrent changes and multiple dependencies. Schäfer [1996] states the problem to deal with sophisticated SCM systems to enable user-friendly specification for release dependencies, possible release states, change request, or dependencies between patches among others in order to reduce the hand-coded effort needed to adjust the configurable options for each new target application developed using a product line approach. This multidimensional view impacts on product line configuration management activities as the assets can be used in parallel projects. Jaring and Bosch [2004] describe variants introduction and instantiation, and explain that, in case instantiation occurs before releasing, it may affect for instance build management or release management (configuring and packaging the right product variant). On the product side, SCM activities are aimed to produce a particular version of the target system that has to be configured according to the requirements. In Cle-

[2]http://www.telelogic.com/products/synergy
[3]http://www-306.ibm.com/software/awdtools/clearcase/
[4]http://www.nongnu.org/cvs/
[5]http://git-scm.com/
[6]https://github.com/
[7]http://www.softwareproductlines.com

ments and Northrup [2002], a list of capabilities to elaborate tools for product line configuration management activities is provided, and all these capabilities extend those traditional SCM ones to multiple products, teams, projects, and locations. Other related experiences and case studies can be found in Jaring, Krikhaar, and Bosch [2004], Maccari and Heie [2005], Raatikainen et al. [2005], Engelsma [2006], van der Linden, Schmid, and Rommes [2007], and Myllärniemi, Raatikainen, and Männistö [2006], where intercompany collaboration and outsourcing play an important role that affects SCM activities for composing different product configurations.

Also, software organizations that run parallel projects in different countries need of adequate methods and tools to achieve the following goals: (i) Release multiple products under the time to market condition; (ii) control the evolution and versions of the architecture, components, and products with suitable CM tools; and (iii) provide collaborative support for different distributed teams that have to interact during the life of the project. In order to address the different facets that may impact SCM activities, we propose a model to describe the forces and the interactions of several disciplines related to multiple products and projects that can be developed in different organizations in different sites, as previous works poorly deal with such number of disciplines at the same time.

3. A 5-AXIS REFERENCE MODEL FOR SOFTWARE CONFIGURATION MANAGEMENT

Regarding the different aspects that may influence SCM activities, we have identified the following five key facets of global software development with direct impact on SCM:

1. **Products:** Software Product Line Engineering (SPLE) develops multiple assets and products under the same software architecture. SPLE is often overseen, and therefore SCM tools provide little support to SPLE for handling multiple products and versions. In terms of SCM, the baseline configuration of a product line is composed of a set of common CIs, which are customized using the variation points defined in the product line architecture and implemented by the build management (building the requested product variant) or release management (releasing the appropriate set of components) activities.
2. **Disciplines:** Under the SCM view, the system configuration is composed by CIs that contain software code, as well as items that represent, model, or simulate a piece of hardware (electronic schemas or printed circuit board descriptions). New challenges appear [Krikhaar et al., 2009]: In particular, the integrated hardware/software configurations must be managed carefully, as the two disciplines, which have been traditionally separated, must cooperate closely on a regular base. This requires adequate SCM functionality for storing hardware and software sources, but also for interdisciplinary release management. Despite early observations [Dart, 1992], the hardware and software worlds seem to have evolved in isolation with regard to SCM.
3. **Projects:** Any medium-size software organization might be able to run multiple projects simultaneously according to their internal resources and budget. The role of SCM in this scenario is twofold. First, each new project constitutes a knowledge repository about development processes and CIs. Second, SCM acts as a bridge between the consecutive phases of development of each project. For example, a team in charge of acceptance testing knows when to start this activity and which CIs in the SCM form a system configuration. Also, SCM that implements accounting functions constitutes a key project management tool. Outsourcing and branching practices may split and organize projects in sub-projects. In some industries, 4–6 months is the regular length for software project development while in large consortiums, 4-year projects are possible.
4. **Organizations:** Products and projects can be supported by several organizations and suborganizations, in particular in companies with well-established product lines. Because companies collaborate in the execution of a project or development of a product, the configuration items (CIs) are the result of the producer organizations, and they are consumed as inputs to create new CIs, to be changed (versioned) or to build up the configuration baseline. Multiple organizations impose more requirements on SCM, such as:network access to the repository crossing organization boundaries and controlled visibility and change of CIs supporting interorganization operations. In a strongly outsourced scenario, we could even think about the federation of SCM systems, supporting the transitive access to items and distributed builds (e.g., the Maven[8] open source system seems to explore this path).
5. **Sites:** Projects can run distributed in multiple sites and products can be engineering at different locations (e.g., due to software production costs). Many software organizations employ distributed teams across a number of geographically distributed sites (e.g., for consortium projects). Different teams with different cultural and language factors [Ma et al., 2007] require a stronger project management to minimize the physical distance factor, such as the storage of asynchronous communications between different sites staff. These issues have been rarely considered by existing SCM tools. More frequent is a large company which has different physical branches (with different time zones), and SCM tools are used to coordinate the versions of the assets (e.g., a repository storing the assets is synchronized periodically).

3.1. Formulation of the Model

The situation with respect to the five facets described above must be clearly stated and visualized in order to estimate the SCM requirements. From a previous work [Krikhaar and Crnkovic, 2007], we extend the idea of a 5-axis model for SCM, and we represent it in the following manner.

[8]http://maven.apache.org/

(a) 5-axis model: The 5-axis model is defined by an aggregation of the five facets, *f1, f2, f3, f4, f5* defined before: *Products, Disciplines, Projects, Organizations*, and *Sites*. Each facet is represented graphically by an axis in the 5-axis model, which is not necessary orthogonal to the rest of the axes. The axes do not have or define a predefined order neither does one axis prevail over the rest.

(b) Values of the facets: Each facet is represented by a numerical value or scale defined by each organization, which indicates the number of active elements of that facet in a given timeframe. For instance, *projects* = 3 indicate that 3 parallel projects are running in a certain period of time. A tabular representation can be used to organize the values of each facet.

(c) Ranges and limits of the facets: The estimation of valid ranges of values for each axis is not as easy as it depends on: (i) the size of the company (e.g., capacity for running parallel projects), (ii) number and location of branches, (iii) the nature of outsourced elements (e.g., projects, people, maintenance), (iv) use of a SPL approach, or (v) other context information. Based on our experience, we suggest the following:

- The number of **products** a particular software company can develop depends on the amount of available resources, its size, and its market segment. Some large companies encompass a wide variety of both internal and external products. For example, companies like Nokia may have many product families, and a distinction must be made, for instance, between two mobile phones belonging to the same family.
- The number of **disciplines** is often low and varies from purely software to complex scenarios including software, hardware, and mechanics. In any case this number is low.
- The average number of parallel **projects** a company could run is often estimated based on the available resources for a given timeframe. However, in our model we do not distinguish between macro and micro parallelism for a given project, or if a particular organization uses different software development approaches (e.g., waterfall model versus agile practices).
- The maximum number of **organizations** involved in the development of a product or project is also difficult to estimate, as SCM tools, the Internet, and outsourcing practices makes the cooperation easier between several organizations.
- The number of **sites** in which a product can be built or a project runs affect the delocalization of software companies, where cultural barriers and development and shipping costs, mainly for hardware disciplines, have a strong influence.
- **Limits of the facets:** Each organization should define their limits of the facets based on the maximum allowed values for each facet and provide a consistent balance for the 5 axes. For instance, large companies may be able to develop 100 commercial hardware/software products while smaller ones no more than, for instance, 3. Regarding projects, the Microsoft project server degrades its capabilities when the number of projects inserted is more than 20 (e.g., data obtained from 2009), and in this work we do not consider companies involved in more than 20 projects simultaneously. Very large projects may involve more than 10 organizations, but in many cases this fact can be considered subcontractors of large organizations.

(d) Visual representation: For representing the axes used the technique known as radar charts (i.e., Kiviat diagrams), as it is a well-known visualization model for describing independent multivariate data as a 2-dimensional chart of three or more quantitative variables represented on axes starting from a common point. Hence, we used the values of the 5-axis model to depict a Kiviat diagram, and the area shows the theoretical situation of a particular company, which can be modified or not according to the additional context information. The shape and the area of the resultant Kiviat shows: (i) the relevant characteristics of the facets being compared and (ii) the SCM activities or solutions which become more relevant according to the facets.

(e) Relationship between axes: Because the modification of the value of an axis may impact on other axes, we use the following notation to describe the forces between axes: +++ means that two axes are highly coupled, ++ represents medium coupled, and + indicates a loose coupled relationship (i.e., strong, medium, and weak relationships). Hence, users will have an indication that the modification in the value of one axis in a tight relationship may imply a change in the value of the other axis.

(f) Context information: Sometimes it is difficult to make the right SCM decisions based only on the number of projects, products, or sites a particular company poses. Hence, it can be possible that the forces between facets can be similar for companies that need different SCM activities. Therefore, each company must use additional context information to discriminate such cases, so we propose the following contextual information:

- **Human resources (HR):** It affects to the number of parallel projects, as each company should know how much time a software engineer can dedicate to each project and at what cost. Representing this factor as a function of the axis, we have: HR = *f*(projects, organizations, sites).
- **Budget (*B*):** Budget strongly influences the number of concurrent projects, but it also affects the number of organizations and sites involved in a particular software development, and at which cost a product is developed. Because budget and human resources are closely linked, an increment or decrement or the budget will increase or decrease the number of human resources, including those companies that outsource TI or have too many sites that may complicate SCM activities. A reduction in the budget may decrease also the number or projects, organizations, and sites, but not all these axes must be reduced at the same time. Similarly, we define *B* = *f*(projects, products, organizations, sites).

- **Technological infrastructure (TI)**: The available hardware/software infrastructure may affect the disciplines and products we develop. Not all organizations can afford launching a software product line approach, while in other cases SCM activities such as distributed repositories make no sense (e.g., the case of a very small organization), and part of the budget must be allocated to support the cost of the TI infrastructure required. This factor affects the number of products a company develops, the number of projects a company runs, and the number of sites in case a distributed location does not count with the appropriate technological infrastructure. Hence TI is a function dependent on projects and products: TI = *f*(projects, products, sites).

We summarize in Table I the context information items affecting each facet of the 5-axis reference model, as each organization must define the amount in which the context influences the allowed values of the facets.

3.2. Guidelines of Use

As a guideline of practical usage where software engineers, business and project managers, and other SCM stakeholders would like to use our model, we provide some guidelines based on the information described before:

1. Extract and obtain for each target company or companies the numbers for each facet.
2. In those cases where historical data are available and properly recorded, obtain data about the context information regarding human resources, budget, and available technological infrastructure.
3. Using the data obtained in step 1, build a Kiviat diagram for each pair of facets we want to compare.
4. For the facets compared, build a table indicating those relevant characteristics using the information specified in Tables II, V, VI, and VII, and add the corresponding SCM activities. Also, add the degree of the relationship (+, ++, +++) between the facet compared and the others for the company analyzed.
5. In case we need to compare more than one company, use the contextual information of Table II and indicate which facets increase or decrease. Repeat steps 3 and 4 to update the Kiviat and the number of SCM activities to include or exclude.

Table I. Influence of Context Information Items on the 5-Axis Model

	Human resources	Budget	Technological Infrastructure
Products		X	X
Disciplines		X	X
Projects	X	X	
Organizations	X	X	
Sites	X	X	X

4. THE ROLE OF THE REFERENCE MODEL ON SCM ACTIVITIES

This section describes the role of the 5-axis model on SCM activities for each facet and the relationships between them.

4.1. Multiple Products

Software Product Lines (SPLs) deals with the creation and management of product families for building multiple products belonging to the same market segment, increasing productivity, and enabling rapid market entry and flexible response. The creation of multiple products can be developed under one or several projects, but the multidimensional and interdisciplinary characteristics of products and projects require the combination of assets from different natures and sources. This multidimensional characteristic affects the creation and use of product families when multiple SPL [van Ommering, 2002; Trujillo, Kästner, and Apel, 2007] run in parallel. The CM system manages the build process and multisite development and follows a multiorganization, multisite approach for the consumer electronics domain. The range of variations of product line products and components influence the complexity of SCM activities,[9] and, hence, this hinders SCM methods from being well established in the product line area. The role of having multiple products and its influence on SCM activities is described in Table II.

- The **product** and **project** axes are tightly coupled because each product is developed under one project. Therefore, when we have several products developed under several projects, we recommend starting an SPL approach and use variability modeling techniques to facilitate product configuration. Hence, there is a need to manage different product releases for each product family and build management procedures for product configuration tasks.
- The **product** and **organization** facets have a medium-high coupled relationship because if a single project runs under a particular organization, the relationship is very dependent of the organization, while in a decentralized scheme projects can be supported by different organizations. As organizations must cooperate closely to develop their products, releases must aggregate the components developed by each single organization and coordinate changes and CM audits.
- The relation between **product and discipline** axes can be considered highly coupled because each product is usually classified under a particular discipline, but it could happen that a particular artefact may not belong to any of the disciplines supported by the main organization in which SCM activities are defined (e.g., third-party software).
- Because **products** are not very dependent on the **site** where they are being developed (decentralized approaches), we categorized this relationship as loose coupled**.** On the other hand, strategic, technological, or economic factors make a single SPL become very de-

[9]http://www.mrtc.mdh.se/publications/0373.pdf

Table II. Characteristics and SCM Activities for Multiple Products

Facets	Relation with	Relevant characteristics	SCM activities	Description of SCM activities
Projects	+++	Product Line approach Product variation modeling Product configurations	Component baselines Component releases Build management Dedicated release management	Build management are all activities to build (compile,link) a system. Dedicated release management implies release management activities that can be also applied for internal deliveries (for example for testing purposes or beta site testing) is release management
Organizations	++/+++	Product cooperation	Aggregated releases Staged Change Control Staged CM auditing	Staged Change Control concerns change control divided over different organizational units. Often implemented with hierarchical structure of Change Control Boards. With staged CM auditing we mean CM auditing applied to parts of the final product.
Disciplines	++/+++	Hardware close to Software	Central repository	A central repository is a repository that is used by different people distributed over different sites. The repository is used as if every developer is located on the same site.
Sites	+/++	Cultural aspects Communication	Branching strategies Collaboration Distributed repositories	Branching is used to separate certain development from other development. In larger companies we see often a huge number of branches which mess up development. At a certain stage the branches should come together again! Branching strategies provides rules for creating branching, merging branches and finishing branches to keep this under control.

pendent on a specific site, and the relation becomes stronger.

We believe that SCM activities change when a decentralized scheme is used, and specific SCM activities are defined for versioning and change control for products, core assets, and for information management as well. In multiple product lines and multiple projects running in parallel, updated SCM activities become more relevant than for single and isolated projects, because a particular product may use other components and products, which are being developed by other running projects. Our approach does not deal with variability or product variants explicitly, as variants may lead to different product configurations (e.g., two products differ in the color setting). In other cases, if a mobile phone developer sells two mobile phones to the same family, these can be seen as two different products; but it depends also on how much variability makes one product different from another, and the distinction is more market-driven rather than a new product. Table III shows an example where two companies with different values in the facets show different forces in the axes and are represented using Kiviat.

Case 1: Company A develops 10 products in 5 projects, while the rest of the facets remain equal to 1. Figure 1 depicts a Kiviat diagram where a tight relationship is expected between these two facets regarding the size of the area and compared to the other axes.

Case 2: In this case Company A doubles the number of projects with respect to case 1. Hence, the relationship becomes much more coupled than in the previous case, and *context information* can be used to discriminate both cases and if it is possible for the company to move to such new scenario (see Fig. 2).

Table III. Example of Companies Supporting Multiple Products

Company	Products	Projects	Organizations	Disciplines	Sites
A	10	5	1	1	1
A	10	10	1	1	1
B	10	5	3	1	1

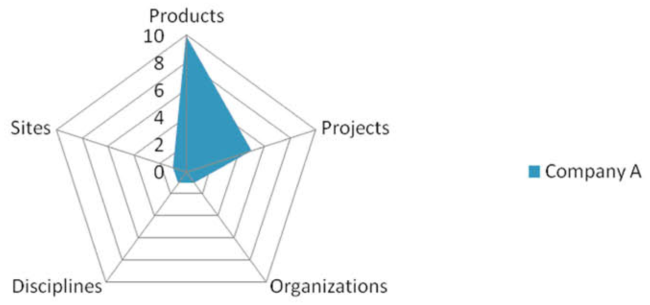

Figure 1. Relationship between multiple products and projects. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com]

According to Table I, an increment in the number of projects may impact, according to the number of available **human resources and budget**. Therefore, the organization must evaluate the possibility of having more projects according to the resources available. Also, new SCM activities, such as status accounting and CM auditing might be needed (see Table IV).

Case 3: Company B develops 10 products in 5 projects in 3 different sites. As the Kiviat diagram shows, the relationship between products and sites is weaker or less coupled than the relationship between products and projects, as described in Table II. When the available human resources decreases affecting one site, the development of a product can be reallocated to a different site, except if economic, technological, or other factors make it impossible (see Fig. 3).

4.2. Disciplines

Software and hardware engineering can be both considered as disciplines (i.e., the type of product a company produces), and each discipline encompasses its own product view or structure with related CIs, as changes in one domain must be propagated into the other domains (e.g., Do, Choi, and Song [2008] discuss propagation of engineering in multiple product views). According to Table V and for software-hardware development, the discipline axis is not very dependent of the organization, project, or site, but conversely it strongly depends of the type of product (e.g., embedded or nonembedded software). Otherwise, the marketing/sales discipline determines that a certain item shall no longer be sold, which means that the manufacturing discipline no longer has to produce the item, unless it is defined as a spare part by the service discipline. Hence, the discipline axis strengthens the relationship with the organizations where products are sold or manufactured. Sometimes, the site axis may increase its influence depending where products are produced (e.g., shipping costs for embedded systems). The relationship between **project** and **discipline** can be considered medium or highly coupled, as parallel projects may produce multidisciplinary products but not all the disciplines must be supported by a single project. For SCM, the structuring of the product according to the purpose of a discipline relies heavily on configuration management.

Table IV. Context Information for Company A, Case 2

	Human resources	Budget	Choices for additional resources	Additional SCM activities
Products		X	Same budget	
Disciplines				
Projects	X	X	+/- 5	Status accounting CM auditing
Organizations				
Sites				

4.3. Multiple Projects

Global software development fosters multisite project development, and configuration management activities play a key role in the success of decentralized organizations. For most companies, it is often better to run several projects concurrently, and CM plays an important role for controlling the multiple versions of artefacts and products. As mentioned in Palmer and Felsing [2002zaq;1], "configuration management systems may vary from the simple to the grotesquely complex." Table VI shows the relationships of our 5-axis model for multiple projects. Running multiple projects in different sites should not be affected in excess by the location in which the project is enacted, except when a delay turns risky for the

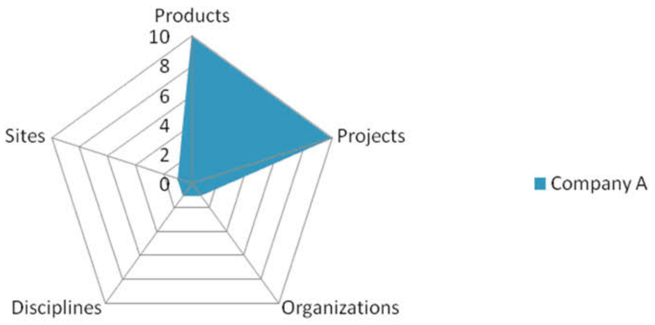

Figure 2. Increment in the number of projects for Company A. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com]

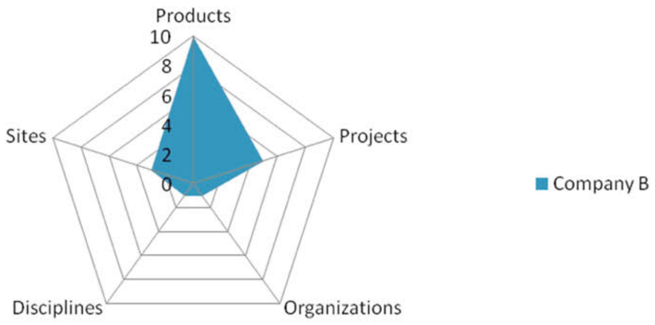

Figure 3. The number of sites has less influence than projects in relationship to the number of products. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com]

Table V. Characteristics and SCM Activities for Multiple Disciplines

Facets	Relation with	Relevant characteristics	SCM activities	Description of SCM activities
Projects	++/+++	Multidisciplinar projects	Status accounting Central repository CM auditing	Status accounting concerns measuring the status of development by inspecting the CM repository and databases. CM Auditing concerns all activities to verify the correctness and completeness of the system
Organizations	+/++	Multi-vendor HW Development	Baselining	Baselining is similar to labeling configuration items versions that belong together. At a later stage the right versions can be found to solve e.g. a problem in that baseline. Releases are always baselined, but also nightly build can be baselined.
Sites	+/++	Multi-site HW Development	Collaboration	Collaboration concerns all activities to work together with different teams.

whole organization. Hence our relationship for disciplines and sites can be defined as weak or medium. In addition, running projects in multiple sites should not be a problem, and the relationship is often considered weak or medium, except in those cases where cultural factors have a stronger influence for communication and work purposes. In such scenario, distributed SCM activities and auditing are often carried out for each site.

The impact on SCM activities when multiple projects run concurrently are as follows:

- The correct synchronization of changes (e.g., a change in an interface should be communicated across both sites of such an interface) must be supported by SCM tools and procedures to organize parallel work under a common workspace.
- The results of projects must be integrated in other projects to deliver the complete set of features, and eliminate the perception of users that the heartbeat of releases is faster than project length. In outsource organizations, large software development requires of specific policies to address the challenge of running parallel projects, as a way to reduce the communication overhead or updating software assets. This issue may become an important factor for selecting the right partners and the right sites according to their distance.
- The context under which the project is executed on and project management information must be represented by CIs, such as project scheduling, resources allocation, project control information, stakeholders' descriptions, and organizational deployment.

Table VI. Characteristics and SCM Activities for Multiple Projects

Facets	Relation with	Relevant characteristics	SCM activities	Description of SCM activities
Organizations	++/+++	Multi-Organizational Projects Quality aspects	Status Accounting CM Auditing	
Sites	+/++	Communication Cultural aspects	Distributed CM Staged CM auditing	We use distributed CM when the responsibility of various CM activities is distributed over different organizations or departments. With staged CM auditing, we mean CM auditing applied to parts of the final product. CM auditing concerns all activities to verify the correctness and completeness of the system.

- Changes in the organization are then mirrored by changes in the corresponding configuration item, so it makes it easier to recover the complete context of execution of the project. This feature of SCM systems will be supported in a short future by tools implementing advances in IT government.

4.4. Multiple Organizations

The growing trend since the mid-nineties towards outsourcing and offshoring makes the practice to purchase [Bergey, Fischer, and Jones, 1999], commissioning, and developing much more common than ever before; and technology-related outsourcing has grown rapidly (e.g., India's Wipro acquired a US infrastructure-management service for $600 million) [Leavitt, 2007]. By contrary, some risks may impede globalization multiply, such as loss of control, legal issues, coordination problems, hidden or unexpected costs, or training and cultural issues. Up to 20 major effects of offshore outsourcing reported by project managers can be found in Lacity and Rottman [2008], but one of the fundamental factors in determining the success and failure in globally distributed teams is trust [Moe and Smite, 2007]. Better practices in eSourcing (i.e., international sourcing of ICT products and services) are needed to crosscut geographical, social, and temporal barriers in GSD. As stated in Käkölä [2008zaq;1], "the globalization of the world economy is putting increased pressure on companies to leverage information and communication technology (ICT) in order to become more competitive." The trend that GSD involves several organizations for managing the knowledge of companies [Clerc, 2008], and decentralized organizations can cooperate with each other need to identify such best practices.

Specific to product line practices, segmented market analysis, domain scoping, and product line acquisition processes are key areas for launching and institutionalizing an outsourcing strategy involving several commercial organizations[10] [Berenbach, 2007]. If we take a look to the SPL strategy, core assets development can involve one or several organizations, in particular in the case of multiple product lines; but all of them share the same development platform and CM procedures. In our model, the **site** is very dependent on the **organization** (see Table VII) as in many cases subcontracting between organizations is very driven by cultural aspects and in other cases by cost.

4.5. Multiple Sites

Companies deploying their activity in multiple sites can refer to the tables discussed before to find the relationship between **sites** and the other facets. Distributed organizations and outsourcing practices in other countries are very driven in many cases by cultural and communication factors. Hence, distributed SCM activities strengthen the collaboration between branches or subsidiary. In those cases of branches located in the same country, cultural aspects do not influence SCM activities, but distributed repositories for managing the diversity of products might be needed.

Table VII. Characteristics and SCM Activities for Multiple Organizations

Facets	Relation with	Relevant characteristics	SCM activities
Sites	+++	Cultural aspects	Distributed CM

For instance, a European telecommunications operator, Telefónica,[11,12] serves to illustrate the role of the **site** axis, as one of the representative world largest telecom companies with 250,000 employees distributed across more than 25 countries, mainly in Europe (six countries) and in Central and South America (13 countries). The company has several branches (i.e., a division, an office, a subsidiary, or an external company of a large one in a particular area) geographically distributed in Europe to support different services and research. A corporative center is responsible for the global strategy and policies of the group, management and coordination of the different business units, to warranty the global vision of the company. Hence, multiple sites participate in their R&D activities, and outsourcing constitutes a characteristic of the business case of the company. Research and development projects are split across different locations with distributed teams and different time zones, involving several disciplines such as networking, electronics, or pure software.

5. INDUSTRIAL EVIDENCE OF THE SCM REFERENCE MODEL

So far, we have discussed the axes of our 5-axis model. We can conclude that the support given by SCM practices is of paramount importance for moving from the one to the many in project-product-organization-site-discipline. To achieve some confidence in this observation, we conducted a survey among the participants of the ICT NoviQ Configuration Management seminar and the Configuration Management workshop,[13] both held in 2008 in the Netherlands.[14] Thirty-two attendees participated in the survey, 23 of whom work in the embedded systems industry and 9 of whom develop information systems. Results of the survey were discussed in place, leading to the following advices in the application of the model:

(a) Projects: Only top-level projects leading to a final product should be accounted when dealing with this axis.

(b) Organizations: The number reflects the number of organizations involved which are (financial) independent of each other. Two departments within one organization count only for one.

[10]http://www.sei.cmu.edu/productlines/frame_report/devel.imp.AS.htm

[11]http://www.tid.es

[12]http://www.telefonica.es

[13]http://www.topic.nl/nl/cm-workshop/

[14]http://www.ictnoviq.nl/nl/14-mei-2008

(c) **Sites:** To derive the value of these facets, we focus on in-house development, not taking into account external components as operating systems or open source libraries outside the company.

(d) **Disciplines:** The number in this axis was filled out by counting whether if any of these disciplines fit in-tozaq;2 the following three categories: software, hardware, and mechanics.

(e) **Products:** The products that must be taken into account are those in which the diversity was explicitly handled during development and which had impact on managing it in software repositories.

Table VIII shows the results of the survey. The first column contains (in anonymity) company names, differentiating between embedded industry (E) and information systems (IS). Forces between the facets are not explicitly measured but can be estimated from the values. The table is sorted according to the sum of forces, starting with a low force and ending with a strong force (and so more complex to manage).

We observed that 11 (34%) companies are developing software in more than 5 parallel projects. To manage this amount of information during development, SCM provides so-called branches, to separate the information of different projects. After finishing the project, developed software has to be merged into a main branch. The difficulty of managing merging during the evolution of these projects is high. In the literature, different branching strategies are discussed to overcome problems in managing parallel projects. In practical terms, however, these strategies can be pragmatically implemented for many parallel projects with short development times.

Developing products with more organizations and more sites requires strict ways of working. The system may be sharply divided into subparts which can be developed and tested separately. Often, more collaboration is required to get the job done, as the case described in Oor and Krikhaar [2008].

From the survey we observed that, for 7 (22%) systems, more than 4 organizations are involved in 20 (63%) systems and development is performed in 3 or more sites (only 2 of them 6% develop systems on a single development site). This fact shows evidence that collaboration models play a key role in current and future development. This situation occurs more for embedded systems than for information systems, as, by definition, embedded systems involve more disciplines. Regarding the sites axis, Bird et al. [2009] distinguish development that is distributed globally within a single company from development with outsourcing, which involves multiple companies. They studied the Windows Vista code base and found that the physical distance did not contribute negatively to a large extent to the quality of the software products in term of postrelease defects. This result exhibits the differences in the organizational structure of companies [Nagappan, Murphy, and Basili, 2008], but it might change in an outsourcing context where several companies develop or maintain differ-

Table VIII. Industry Survey in Terms of the SCM Reference Model

Company	Projects	Organizations	Sites	Disciplines	Products
IS4	2	1	1	1	2
EC	1	1	2	1	2
IS5	1	2	1	1	3
IS1	3	1	1	1	3
E5	1	2	2	2	3
IS8	2	2	2	2	2
E9	3	1	4	1	3
EJ	1	2	2	2	6
ED	3	1	3	2	3
EE	3	1	4	2	3
EI	1	3	4	2	3
EM	5	2	1	3	3
IS3	5	2	1	1	10
IS9	5	2	2	1	5
E8	3	1	5	2	4
E1	4	1	4	2	4
IS6	2	2	4	2	4
E3	2	3	3	2	4
EF	4	2	2	3	4
IS2	10	2	1	1	10
E4	2	3	4	3	3
IS7	5	3	3	1	5
E6	3	1	3	3	10
E7	3	2	3	3	7
EL	4	2	6	2	5
EK	4	4	6	2	4
E2	12	2	3	3	4
EB	5	5	3	3	4
EG	10	4	3	3	4
EN	10	4	8	3	3
EA	10	8	7	2	5
EH	20	5	5	2	10

ent software products at different sites under one or more software projects. From our study and according to Table I, only one of the companies analyzed (E6) is powerful enough to support 3 distinct disciplines in a single organization but distributed in three different sites.

Also, in traditional SCM, a single group or person had the authority to decide on the organization of SCM activities. In the survey we see that only 8 (25%) systems were developed by a single organization, which means that there is only 25% that match with a traditional SCM strategy. We observed that 11 (34%) persons indicate that they develop more than 5 main products from a single development line, showing evidence for dedicated support for product lines on SCM systems.

Finally, as the number of projects can be larger than 20, some normalization should be done before averaging the values of the forces in order to avoid that the resultant forces can be always the same. This normalization must be defined by all the organizations cooperating in a common project or product development, but the definition of appropriate weights to normalize the results is out of the scope of this study.

5.1. SCM Solutions

Table II shows how facets are related to SCM activities. The implementation of the SCM activity heavily depends on the strategy of development. For example, the chosen technology for SPL development has impact on the way, e.g., build management is solved. Analysis of the forces provides insight in implementing the proper SCM solutions that support these forces.

From Tables II, V, VI, and VII we derive Table IX for the companies analyzed. In Table VIII, the first column numbers each particular case. Columns 2–6 refer to Projects (Pr), Organizations (Or), Sites (Si), Disciplines (Di), and Products (Pd). The next two columns describe the important characteristics required when adopting an SCM approach and those SCM solutions needed for implementing the characteristics. The last column indicates examples of the companies given in Table VII affected by the axis selected in columns 2–6.

Table IX provides insight of those SCM aspects that should get attention when a force exists between two axes. For each combination of forces, the best SCM implementation is given. For instance, regarding case 1 in Table IX and the particularities of companies IS2, IS3, and EH, the *build management* SCM activity should be implemented to support multiple products and projects, as, in industry, we have seen that wrong SCM decisions may lead to hamper software development if a bad approach or technological solution is chosen.

Based on the results of the survey and Table IX, which directs the focus on SCM solutions, we elaborated the first three cases:

Case 1: Companies IS2, IS3, and EH. In this case a medium/strong relationship appears between **projects** and **products** (e.g., due to a number of products developed under the SPL approach or projects running simultaneously), but these three companies provide different solutions to get multiple projects and multiple products under control. IS2 and IS3 develop components, which are separately released (requiring component baselines) and put a customer-specific system together by selecting the proper components (and ver-

Table IX. Solutions of the Industry Survey

Companies	Pr	Or	Si	Pd	Di	*Relevant characteristics*	*SCM activities*
IS2, IS3, EH	X			X		Product Line approach Product variation Modeling Product configurations	Component baselines Component releases Build management Dedicated release management
EB, EA		X		X		Product cooperation	Aggregated releases Staged Change Control Staged CM auditing
E2, EG	X				X	Multi disciplined projects	Status accounting Central repository CM auditing
E8, EL			X	X		Cultural aspects Communication	Branching strategies Collaboration (Mainline) Distributed repositories
E6, E7				X	X	Hardware close to Software	Central repository
EG, EN	X	X				Multi-Organizational Projects Quality aspects	Status Accounting CM Auditing
IS2, EH	X		X			Communication Cultural aspects	Distributed CM Staged CM auditing
EB		X			X	Multi-vendor HW Development	Baselining
EN			X		X	Multi-site HW Development	Collaboration
EK, EA		X	X			Cultural aspects	Distributed CM

sions) just before delivering to the customer. EH is a company that develops product lines by combining two approaches. First of all, in the architecture, measures are taken to select the proper components during run time (using a configuration file setting the right options). This method is combined with several build options which have to be set to create specific results for dedicated hardware parts in the system. This combination requires dedicated release management in order to release the product variants to the right customers. In summary, roughly two approaches were observed, one building components and putting a product together in a late stage of development. The other approach was more based on configuring (parts of) the system during building (compiling, linking) the system.

Case 2: Companies EG and EN. A strong force between **products** and **organizations** means that different organizations closely work together in product development (and business creation). A good release policy is extremely important to ensure that the proper products are released. In case of field problems, one should be able to trace back to the original source to resolve. As more organizations are involved, it should be hidden for the customer, but very clear for all organizations. The EG system showed us that they give special attention to auditing the status of the various product parts. As more organizations (and probably cultures) are involved, one should preserve software quality by doing CM auditing well. Are the right versions of the different developments put together? Are we testing the right version of the system? Is the right system released; are all elements included? Can we trace to the source in case a problem arises in a certain release? Answers to these questions become more difficult when more organizations are involved. The EG company experienced this, because due to an acquisition, different organizations had to work together strongly. Another topic which becomes more important is the Change Control Board. When more organizations work together, there is a chance that some problems are not resolved because they each think that the other organization will solve it. The EG company had to deal with this issue and experienced in the hard way that more control is required when changes are addressed.

Case 3: Companies E2 and EG. The strong relationship between **projects** and **disciplines** means that several projects run in parallel to produce a multidisciplinary product. As status accounting (recording the status by extracting information from the CM system) is one of the main reasons of control for a project leader, where special attention has to be paid to achieve commonality between the disciplines. A single source from which status is extracted helps to support this activity, so a single repository helps. Auditing is the verification of completeness, which is harder for multiple disciplines due to the fact that they produce artefacts from different nature. Company EG has a strong force between disciplines and projects. We found out that status accounting (knowing the status of all deliverables during development) was organized in such a way that the status of total development could be easily derived at any moment in time. By introducing a focus on status accounting the EG company tackled a serious problem of the past. For example, much time was spent on testing a system, while the wrong firmware was uploaded to the testing system. To simplify this (more or less) automated process, all system parts to be uploaded to a system were stored in a single central repository. Therefore it was easier to check the status and easier to have the right things done.

6. CONCLUSIONS

In this paper we have described a 5-axis reference model that uses Kiviat diagrams to describe the particular situation of companies from five different facets and how these facets interrelate with each other. We also describe how the model can serve as a guidance and assessment for SCM practices or activities based on a set of characteristics of each facet.

From the survey carried out, we observed that the diversity of products has a great impact on SCM solutions. The axis model helps to determine the SCM-related forces for software development. For instance, some of the companies in the case study, which are developing more products, are also involved in a large number of projects, while others build more products in less number of projects. In most of the cases, the multiple products problem was solved by applying various build scripts (i.e., build management). In the case of multiple projects, the build management was considered in a situation where different branches were implemented to serve multiple projects. As the night takes only 8 h, in some companies it became a problem to build all projects for all products. This resulted in a strategy where more projects are using a single branch. Another disadvantage of having multiple projects is that communication overhead increases.

With respect to the disciplines, only the 28% of the companies surveyed can afford having 3 disciplines at the same time. This may increase the cost and complexity of the structure of the organization and in many cases leads to run more projects or develop more products.

The 5-axis reference model provides a simple and straightforward way of understanding the internal structure of companies and to detect their strongest and weakest forces between axes as well. It offers a simple but effective way to communicate the situation of each particular company when the number of facets increase or decrease and to adopt the most suitable SCM solutions according the relationships and values of the facets. Also, the categorization used to determine the forces of the axes should be based on contextual information of each particular organization and for a given timeframe, as it is complex to provide a simple metric suitable for all companies and their circumstances. Such qualitative evaluation helps to make decisions when selecting a particular SCM approach, but also helps to prioritize the SCM implementations when improving an organization.

Regarding the correctness of our approach, two or more people using our model should get similar forces and use the same SCM recommended activities. Only different context information, specific to each company or case, may lead to different assessment of SCM activities based on the variety and degree of the relevant SCM characteristics used, as different combinations can be possible when each context information item is incremented or decremented according to the particular situation of the company. Because we tried first to validate the applicability of the 5-axis model, our initial conclusions are only derived from the comparison of the data provided by the companies in the industry survey, but not from a single company, from which further analysis and deeper conclusions we expect to derive using our model in any of the companies analyzed in the industry survey.

Finally, we suggest several areas of improvement. In particular, as our approach can be used to assess a particular company about the best practices or recommended SCM activities to use, the interdisciplinary systems engineering area must define more specifically which context information is more valuable to provide an accurate analysis of the SCM needed for each particular case, in particular when dealing with complex projects that involve different disciplines like control or industrial engineering, Also, the standard Configuration Management practice parallelize system engineering tasks, and our facets can be used to depict and assess for allocating the development items, auditing, and testing practices (i.e., verification engineering) to achieve the desired functionality. In addition, it would be interesting to explore a relationship between our 5-axis model and CM concepts implemented in well-known models and practices like CMMi, COBIT, or ITIL.

ACKNOWLEDGMENTS

We thank Niels Veerman for his effort, contribution, and reviews during the initial stage of this research.

REFERENCES

B. Berenbach, An introduction to global to global product line requirements engineering, Int Conf Global Software Eng (ICGSE 2007), IEEE CS, 2007.zaq;3

J. Bergey, M. Fischer, and L. Jones. The DoD acquisition environment and software product lines, Technical Report CMU/SEI-99-TN-004, ADA244787, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1999.

C. Bird, N. Nagappan, P.T. Devanbu, H. Gall, and B. Murphy, Does distributed development affect software quality? An empirical case study of Windows Vista, Int Conf Syst Eng, ICSE 2009, 2009, pp. 518–528.

J. Bosch, Design and use of software architectures, adopting and evolving a product-line approach, ACM Press/Addison-Wesley, Reading, MA, 2000.zaq;4

P. Clements and L. Northrop, Software product lines. Practices and patterns, Addison-Wesley, Reading, MA, 2002.

V. Clerc, Towards architectural knowledge management practices for global software development, Proc 3rd Workshop Sharing Reusing Architectural Knowledge (SHARK'08), ICSE Workshops, ACM DL, 2008, pp. 23–28.

S.A. Dart, Parallels in computer-aided design framework and software development environment efforts, Proc Third IFIP WG10.2/WG10.5 Workshop in Cooperation with GI/ITG FG 3.5.6/5.2.6 Electron Des Automat Frameworks, North-Holland, Amsterdam, 1992, pp. 175–189.

N. Do, I.J. Choi, and M. Song, Propagation of engineering changes to multiple product data views using history of product structure changes, Int J Comput Integr Manuf 21(1) (2008), 19–32.

E. Engelsma, "Incremental systems integration within multidisciplinary product line engineering using configuration item evolution diagrams," Software product lines: Research issues in engineering and management, T. Käkölä, and J.C. Dueñas (Editors), Springer, New York, 2006.zaq;3

J. Estublier, Software configuration management: A roadmap, Int Conf Software Eng, Proc Future Software Eng, Limerick, Ireland, 2000, pp. 279–289.

J. Estublier and S. Garcia, Process model and awareness in SCM, Proc 12th Int Workshop Software Configuration Management, ACM, Lisbon, Portugal, 2005, pp. 59–74.

R.D. Hamelin, D.D. Walden, and M.E. Krueger, INCOSE systems engineering handbook v3.2: Improving the process for SE practitioners, INL/CON-09-17286, INCOSE, Seattle, WA, 2010.zaq;4

M. Jaring and J. Bosch, A taxonomy and hierarchy of variability dependencies in software product family engineering, Proc 28th Annu IEEE Int Comput Software Appl Conf (CompSAC), Hong Kong, September 2004.zaq;3

M. Jaring, R.L. Krikhaar, and J. Bosch, Representing variability in a family of MRI scanners, Software Practice Experience 34(1) (2004), 69–100.

T. Käkölä. Best practices for international eSourcing of software products and services, Proc 41st Annu Hawaii Int Conf System Science (HICSS 2008), 2008, p. 17.

R.L. Krikhaar and I. Crnkovic. Software configuration management, Sci Comput Program 65(2) (2007), 215–221.

R. Krikhaar, W. Mosterman, N. Veerman, and C.Verhoef, Enabling system evolution through configuration management on the hardware/software boundary, Syst Eng 12(3) (2009), 233–264.

M. Lacity and J.W. Rottman, The impact of outsourcing on client project managers, IEEE Comput 41(1) (2008), 100–102.

N. Leavitt, The changing world of outsourcing, IEEE Comput 40(12) (2007), 13–16.

J. Ma, J. Li, W. Chen, R. Conradi, J. Ji, and C. Liu, "An industrial survey of software outsourcing in China," PROFES 2007, LNCS 4589, Springer-Verlag, Heidelberg, 2007, pp. 5–19.

A. Maccari and A. Heie, Managing infinite variability in mobile terminal software, Software Practice Experience 35(6) (2005), 513–537.

M. Mason, Pragmatic version control, The Pragmatic Programmers LLC, Lewisville, TX and Raleigh, NC, 2005.

N.B. Moe and D.Smite, "Understanding lacking trust in global software teams: A multi-case study," PROFES 2007, LNCS 4589, Springer-Verlag, Heidelberg, 2007, pp. 20–34.

V. Myllärniemi, M. Raatikainen, and T.Männistö, Inter-oganisational approach in rapid software software product family development—a case study, International Conference on Software Reuse (ICSR), Springer-Verlag, Heidelberg, 2006, pp. 73–86.

N. Nagappan, B. Murphy, and V.R. Basili, The influence of organizational structure on software quality: An empirical case study, Int Conf Syst Eng, ICSE 2008, 2008, pp. 521–530.

E.C. Nistor, J.R. Erenkrantz, S.A.S. Hendrickson, and A. van der Hoek, ArchEvol: Versioning architectural-implementation relationships, SCM, 2005, pp. 99–111.

P. Oor and R. Krikhaar, Balancing technology, organization and process in inner source, Dagstuhl Sem Proc Combining Advantages Prod Lines Open Source, April 2008.zaq;3

L.J. Osterweil, C. Ghezzi, J. Kramer, and A.L. Wolf, Determining the impact of software engineering research on practice, IEEE Software 41(3) (2008), 39–49.

S.R. Palmer and J.M. Felsing, A practical guide to feature-driven development, Prentice Hall PTR, Upper Saddle River, NJ, 2002.

M.Raatikainen, T. Soininen, T. Männistö, and A. Mattila, Characterizing configurable software product families and their derivation, Software Process Improvement Practice 10(1) (2005), 41–60.

W. Schäfer, Product-line development requires sophisticated software configuration management, 10th Int Software Process Workshop, Dijon, France, IEEE CS, 1996, pp. 15–16.

S. Trujillo, C. Kästner, and S. Apel, Product lines that support other product lines: A service-oriented approach, SOAPL Workshop, SPLC 2007, May 2007.zaq;3

W.F. Tichy, Tools for software configuration management, Proc Int Workshop Software Version Configuration Control, Grassau, Germany, 1988, pp. 1–20.

F. van der Linden, K. Schmid, and E. Rommes, Software product lines in action, Springer-Verlag, Heidelberg, 2007.

R. van Ommering, Building product populations with software components, Proc 24th Int Conf Software Engineering, ICSE 2002, 2002, pp. 255–265.

Rafael Capilla holds a Ph.D. in Computer Science from the Rey Juan Carlos University (URJC) and is a tenured Assistant Professor at URJC, Madrid, Spain. Currently, he leads the Software Architecture & Internet Technologies (SAIT) research group, and his research interests focus on software architectures, product line engineering, software variability,and Internet technologies. Collaborative architectural knowledge sharing is also a major topic of interest.

Juan C. Dueñas obtained a degree in Telecommunications Engineering from Universidad Politécnica de Madrid (UPM), Spain in 1990 and a Ph.D. from UPM in 1994. He is a Professor in Escuela Técnica Superior de Ingenieros de Telecomunicación at UPM, Madrid, Spain. He is president of the Spanish Chapter of the IEEE Computer Society. His research interests include: services engineering, Internet services, service-oriented architectures, and software engineering.

René Krikhaar is an Associate Professor at VU University, Amsterdam. His research topic concerns Collaborative Software Production. Software-intensive systems are more and more developed by different organizations (within a company, but also outside a company: outsourcing, offshoring, COTS), which requires special measures in software architecture, development process, and configuration management. Special attention will be taken on the evolution of systems that have already evolved over many years. Research is performed on these topics in close relation with industry.