2312
513.

A common Product Life Cycle in global Software Development

Jozef De Man* and Christof Ebert
Alcatel, Department of the Chief Technology Officer
** Ghent University, Department of Information Technology*

Abstract

The telecommunication market increasingly demands complex solutions involving a combination of products and requiring co-ordinated development in multiple divisions of a company and even across different companies. Such developments present unique challenges to reach adequate product quality goals within budget and delivery time constraints, especially in a company that continuously evolves to adapt to changing market conditions and regularly needs to incorporate new acquisitions. In this contribution, we show how a formal process model plays a key role in meeting these demands. As examples of the application of this model, we discuss a tailorable Product Life Cycle, and, as an example of more detailed process support in this general framework, the federation of Defect Tracking Systems.

1. Introduction

In a large organization, product development requires co-ordinated activities in multiple, geographically distributed divisions and it is therefore necessary to establish a common way of working. However, this must happen without imposing unnecessary constraints that might cause inefficiencies and lead to resistance against the common rules.

Written procedures are not the most adequate means to implement this. It is difficult if not impossible to ensure that the tailored procedures remain consistent with the common ones. Written procedures require extensive training and continuous discipline for their application. Any change involves an unlearning effect which may take a long time to spread across the company.

The obvious solution is to cast the procedures into automated workflow support to ensure adherence to processes. Such a system guides its users through the successive steps of procedures and automates the administrative aspects of activities. For example, it presents a task list based on the role of the user, offers templates and user guides for each document type, and it links automatically to document management systems. This approach has its pitfalls too. Often, such a system is tailored to the larger and more mature business units of the organization and does not adequately support start-up units which value simplicity, flexibility and agility. Moreover, we rarely have the opportunity to deploy a completely new system across the entire organization. Almost always we need to evolve incrementally from an heterogeneous, only partially integrated collection of legacy systems.

A common Product Life Cycle (PLC) provides a framework within which these conflicting requirements can be reconciled. The PLC is simple so it can be used for developments with effort ranging from several hundreds of person-years to a few person-weeks. The PLC also comes with tailoring rules to guide adaptation to the requirements of each business unit [4]. This paper elaborates how a formal model of the PLC has played a key role in keeping the tailoring under control and enabling the deployment of a common support system. We also show how the same approach can be used to federate an initially unconnected collection of defect tracking systems without necessitating a sophisticated synchronization system.

2. A common Product Life Cycle

Alcatel has defined a corporate Product Life Cycle (PLC) with clearly defined Decision Reviews and associated entry/exit criteria to verify whether a phase can be concluded and a next phase can be started (figure 1). The PLC covers the entire life of a product release from idea to end-of-life. It provides the framework not only for engineering processes but for all product-related processes: third-party product acquisition, subcontracting, environmental processes, marketing, etc.

Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'04)
0-7695-2218-1/04 $20.00 © 2004 IEEE

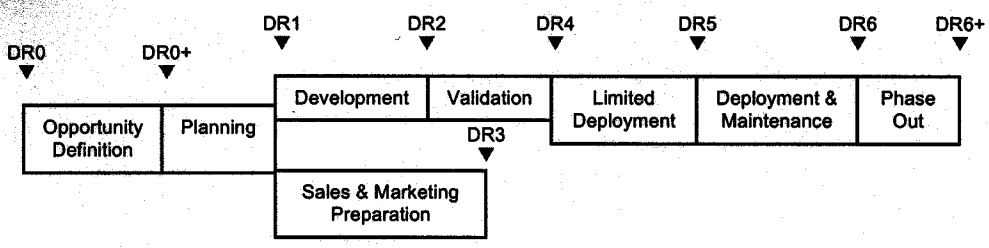

Figure 1. Product Life Cycle

The corporate PLC comes with tailoring rules for business units to maintain commonality where necessary and to enable at the same time adaptation to specific requirements of the business unit. The rules also apply to new acquisitions. The PLC is used for single product releases but also for composite (network) products, services and solutions. The common framework even encourages joint development of components following some of the principles of Open Source development within the community of a large corporation

Such a Product Life Cycle is very difficult to maintain by means of textual descriptions. Users want the PLC definition of a business unit to be self-contained and not expressed by means of a list of modification to the corporate standard. As the standard evolves, all derived documents must be carefully reviewed and updated, hopefully in time for the next version of the corporate standard.

To avoid the problems associated with a natural language description, we have defined the PLC using a simple entity-relationship model [1] with as basic entities: *milestones* (decision reviews), *work products*, *roles* and *activities* [2]. The model also involves

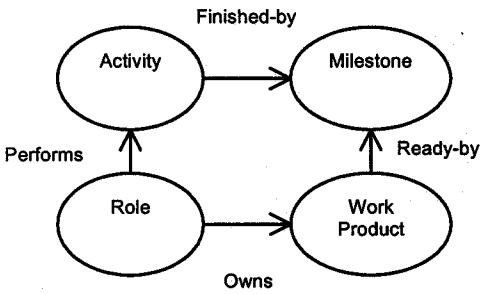

Figure 2. Process Entities and Relationships

relationships between entities (figure 2). For example, work products are the outcome of activities, they are owned by roles and they must be ready with a certain level of quality by a certain date (milestone). Attributes further define each instance of an entity. We have found *templates* for work products to be particularly useful to ensure consistent outcome of an activity, much more than detailed activity/process descriptions.

The modeling approach enables tailoring to be expressed formally by means of inheritance from the corporate PLC and constraints to be defined by specifying the mandatory/optional entities and attribute values of each type. Tailoring sometimes happens by defining additional entities (decision reviews, roles) but usually by specifying different attribute values (e.g. work product templates).

Our approach can be cast in the model hierarchy as defined by the Object Management Group [3]. It involves four layers of models:

M3	Entity-Relationship meta-metamodel
M2	Process metamode
M1	Process model
M0	Enactment in a project

Table 1. Process Model Hierarchy

- The meta-metamodel (M3) with entities, attributes defining properties of entities, and relationships between entities.
- The process metamodel (M2) where process-specific entities, attributes and relationships are defined. The basic entities are milestones (when?), work products (what?), roles (who?) and activities (how?).
- The process model of the PLC (M1) where instances of milestones, work products, roles and activities are defined. Examples of work products are product specifications, test lists, project plans, risk assessments. "Product Manager" and "Project Manager" are examples of roles. At this level, business units can define their particular tailoring.
- At the lowest level (M0), the process model is instantiated by assigning (originally planned) dates to milestones, specifying the names of actors

for the roles, and identifying the work products to be created. During the enactment of the project, other attribute values are dynamically defined: latest forecast / actual milestone dates, the content of the work products, etc. This level corresponds with the implementation of the PLC in the development of a particular product release.

Table 2 illustrates this model hierarchy: John Doe is an instance of a Project Manager, a Project Manager is an instance of a Role, a Role is an Entity in the meta-metamodel.

M3	Entity
M2	Role
M1	Project Manager
M0	John Doe

Table 2 Example of Process Model Hierachy

3. Automated Support

The model hierarchy enables automated support by means of a Product Release Portal (Figure 3) providing uniform web-access to product and project information. It provides automation for the concepts defined in the M3 and M2 models that are generally valid for all business units.

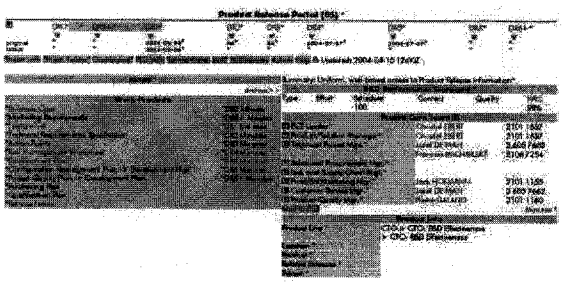

Figure 3. Product Release

The banner on top of the portal shows the (tailored) sequence of decision reviews with originally planned dates and latest forecast. A green bar shows progress of the project. The frame below gives direct access to the main work products and shows the members of the Project Core Team. The page template is created automatically from the tailored PLC definition. All the content manager needs to do is to click and select an actor from the directory services, enter the dates and submit documents through links embedded in the page.

The corporate PLC process is defined in M1 by means of simple tables with the common decision reviews of the PLC, the roles of the members of the Project Core Team and standard work products to be generated during a project. See table 3 for a few examples.

Project Manager	Owns	Development Plan
Project Manager	Part of	Project Core Team
Quality Manager	Owns	Quality Plan
Quality Manager	Part of	Project Core Team
Development Plan	Ready by	DR1
Quality Plan	Ready by	DR1

Table 3. Tabular PLC Definition

The tailoring of the business units are equally simply defined by modifications to these tables. For example, in Business Units where hardware is an important part of the development, the Hardware Manager may be part of the Project Core Team and owns the Hardware Development Plan. This is specified with some additional rows in the table (Table 4).

HW Manager	Owns	HW Dev. Plan
HW Manager	Part of	Project Core Team
HW Dev. Plan	Ready by	DR1

Table 4. Example of PLC tailoring

When the definition of the corporate PLC changes, all tailored PLCs inherit the modification without any editing. Possible inconsistencies that might result from the adaptations can be verified and flagged automatically.

Also the data associated with each development project (M0) are simple attribute values (e.g. dates) or unique references to documents in document management systems, or also references to personnel data in the corporate directory services.

The M0 and M1 models, and even part of M2 model are implemented in the same database, enabling flexible, yet controlled customization. For example, tailoring of the PLC (normally done in M1) can also be done at the level of individual projects (M0). It can even be adapted dynamically as the project proceeds. There are obviously constraints on the degree of flexibility that is acceptable in the context of good project management practices and these features should therefore not be made available without any limitation.

The system supports automatic notification, roadmap management, product and project portfolio management, metrics collection, aggregation and analysis, etc. It is also a foundation for federating configuration management systems and associated defect tracking and change management systems. This common support system ensures that the best practices in the area are available off-the-shelf to all business units and newly created ones in particular. Even more important, if allows aggregation of data to support senior management overview at each level of the organization (business unit, division, corporation).

Standard core metrics with quality targets for each decision review support co-ordination between decision reviews at the solution level with those at the product/component level. Distribution of defect detection effectiveness and efficiency across the main phases of the product life cycle can be used to assess quality in a uniform way. For example, there is a common Decision Review (DR2) to mark the end of any (incremental) development/integration and the start of validation. Such metrics are particularly important to monitor progress of all business units as a result of process improvement activities carried out at the level of the business units but co-ordinated at the corporate level following the SEI's Capability Maturity Model.

4. Federating Defect Tracking

Within the framework of the Product Life Cycle, more detailed processes can be modeled and integrated in the support environment. Defect tracking is an example. The common PLC and associated web-based support has as important side-effect that we have established a common repository and uniform identification of all product releases in the corporation. This global framework can be used to implement common support for several management functions, including defect tracking.

In our experience, the global deployment of a common defect tracking system is very difficult to achieve. Acquisitions bring their own defect tracking support into the corporation and migrating to the next generation of defect tracking systems may have to start before the preceding migration has been fully completed. We have therefore opted for a federation of multiple defect tracking systems based on a few simple rules that can be satisfied with modest investment.

The defect tracking system used for a product release is an attribute in the support system and this allows an incoming defect report to be routed automatically to the correct system through a uniform interface embedded in the product release portal. Routing within the defect tracking system must further be based on the product release identifier. These simple integration rules even support automatic routing of customer service requests. Combining the time-stamps of defect reports and the decision review dates in the product release portal we can automate collection of some of the core metrics mentioned above.

Also for the workflow of defect reports we can follow the 4-level model hierarchy. M3 and M2 are not affected by this extension. In this case we follow a bottom-up approach and accept the different implementations of defect tracking systems in business units as given (M0) and extract the common elements to define M1. This common model is as simple as possible but sufficiently rich to support features for which we need a global scope. The M1 model includes among others

- The defect reports as new work product,
- The originator and actionee of defect reports as roles,
- Affected product release, status code, and due date as attributes of the defect report.

The implementation must be more dynamic than in the case of the PLC where e.g. work products are statically owned by a role which is assigned to an actor, typically for the entire duration of a project. In defect tracking systems ownership shifts from one actor to another depending on the status changes (created, assigned, solved, closed).

Defect tracking systems have much more sophisticated models but it is fair to assume that almost any such model can be mapped to this simple one. Actions in the defect tracking system that affect the global model trigger an update in a common database that can be used to implement useful queries like: action lists of individual actors, list of open defect reports against a given work product, overdue defect reports, etc.

5. Implementation Issues

The four-level hierarchy of models proves to be a convenient framework to analyze and understand the process support in a complex organization. Perhaps somewhat surprisingly it also enables a very simple implementation of the support environment. Simple scripts on a standard web server are sufficient for the implementation. An application server would have increased complexity and cost and reduced response

times and flexibility. The advantages of web-enabled support are well-known:

- The system can be deployed instantly to the entire corporation, it can be accessed with a standard browser and does not require dedicated client software.
- The corporate network offers fast web access across the entire company and allows an implementation on a single server.
- Communication with other systems (corporate databases and "local" defect tracking systems) can use the same HTTP protocol as used for communication with web browsers.
- Consistent use of web interfaces encourages the creation of direct links from web pages across the company. People do not have to navigate from the top but use hyperlinks from their familiar home page to access tailored views directly.

Once a new acquisition is linked to the corporate intranet and its personnel is registered in the corporate directory services, the support environment becomes available without any additional investment.

A simple example may illustrate the power of this approach. The product release portal comes with a facility to publish short news items in a section of the portal. Using the organizational references of the portal, these news items can be included automatically in weblogs for the product line, business unit and business division.

6. Formal Modeling

Our use of formal methods is somewhat unconventional. With the entity-relationship model [1], we have not used a cutting-edge technology but one with an impressive track record. It offers the right level of abstraction for our intended purpose with an excellent balance between intuitive clarity and a formal foundation. Something which could not be matched by other formal methods that have been developed since (for an overview see [5]).

We do not use a separate, dedicated notation for the entity-relationship model (or any of the underlying models) such as class diagrams in UML [6]. Instead, all models are represented in the same scripting language. In this way, we may have lost the relative advantages of visual modeling and the formal verification possibly embedded in the support tools for the formal method but this is more than compensated by many advantages:

- Automatic translation to the web-based implementation can be expressed explicitly. It would not be realistic to expect such a translation to be supported by a tool for a general-purpose formal language.
- The definition of semantics at a higher level becomes available seamlessly at a lower level, either directly executable or as a data structure that can be used for scripts defined at lower levels. For example, forms to register entities and relationships between entities can be generically defined and be reused for each instance of the entities and relationships.
- We are not bound by the uniform presentation typically offered by process modeling tools. In practice we do not really need the sophisticated graphical capabilities of such tools which take up a lot of real estate on the screen. It is better to create compact, semi-graphical presentations matching the semantics of each entity, e.g. a timeline with dates to represent the milestones of a project (see figure 3).

Our approach combines common sense pragmatism with the advantages of formal methods avoiding notational complexity and overhead.

7. Deployment and future work

The Product Release Portal is implemented on a single web server accessible from anywhere in the corporation. Content management happens mainly at the level the product releases, often directly by members of the Project Core Team of the product release. System administration can be performed through a web interface and is delegated to the product line.

The navigation structure through business divisions, business units and product lines down to product releases is kept up-to-date automatically with the organizational structure maintained in the Organization Referential Database.

Product Release Portals are now operational in all divisions involved in product development. The project-level information systems that were already operational in many product lines are seamlessly linked into the system. Product lines without a local infrastructure are increasingly satisfied with the functionality of the corporate system and do not feel the need for implementing their own system. To avoid duplication of effort for data input, some dedicated interfaces have been developed to import information from major legacy systems automatically.

The environment serves a broad audience including senior management at all levels of the organization, project management, quality assurance, software engineering process groups and all individuals

involved in projects. It also serves as a co-ordination infrastructure for all parties involved in product development: marketing, product management, R&D, purchasing, industrialization and supply chain, customer service, quality assurance.

The flexible integration enables fast and incremental development of additional features. Some possible extensions are outlined below:

- *Integration with Management Systems.* A Product Release Portal immediately presents the tailored PLC of a product line in terms of its decision reviews and associated work products. This can be further extended with links to templates and associated process descriptions. It brings the Management System into the working environment and ensures transparent access to process documentation where and when necessary.
- *Tools Integration.* We have not yet fully exploited the possibilities for integration made possible by the common model hierarchy. For example, when change requests are routed to work products, they can be automatically assigned for approval to the Project Core Team and be assigned to the owner as defined by the PLC level-0 model.
- *Built-in Engineering Support Systems.* We have illustrated how the support environment can be used for federating engineering support system. To enable instant deployment it is necessary to offer also default implementations of such systems. We intend to achieve this by integrating Open Source Defect Tracking, Configuration Management and Project Management tools on the central server.
- *Support for very small projects.* Especially for projects of a few person-weeks it is important to have adequate support with minimal administrative overhead. One key-click should be sufficient to instantiate a portal with templates to be filled-out. We will ensure this by using the system also for small improvement projects, learning from our own experience to eliminate all unnecessary overhead.

8. Conclusion

We have shown how formal process modeling enables a pragmatic implementation of a common product life cycle with adequate tailoring in the different business units and also enables the implementation of a common, tailorable web-enabled support environment.

The environment is developed across the entire corporation and is quickly accepted by new acquisitions. It provides the initial framework for a gradual and incremental harmonization of practices across business units.

9. References

[1] Peter Chen: The Entity-Relationship Model - Towards a Unified View of Data. *ACM Transactions on Database Systems*, Vol. 1, no. 1, March 1976, pp. 9-36.

[2] Jozef De Man: A lightweight process-centered project support environment: motivation, implementation and experience. *Proceedings World Multiconference on Systemics, Cybernetics and Informatics.* Orlando, Florida, 2000 .

[3] OMG Object Management Group: *Software Process Engineering Metamodel Specification*, 2001.

[4] Christof Ebert, Jozef De Man, Fariba Schelenz: e-R&D - Effectively Managing and Using R&D Knowledge. *Managing Software Engineering Knowledge.* pp. 339-359, Springer, 2003.

[5] FME Formal Methods Europe. http://www.fmeurope.org/

[6] OMG Object Management Group: Unified Modeling Language. draft version 2.0, 2003.

Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'03) pp. 16-21

A Common Product Life Cycle in Global Software Development

Jozef De Man, Ghent University
Christof Ebert, Alcatel

Full Article Text: PDF BUY ARTICLE

DOI Bookmark:
http://doi.ieeecomputersociety.org/10.1109/STEP.2003.1

Abstract Contents:
Abstract
Citation

Abstract

The telecommunication market increasingly demands complex solutions involving a combination of products and requiring co-ordinated development in multiple divisions of a company and even across different companies. Such developments present unique challenges to reach adequate product quality goals within budget and delivery time constraints, especially in a company that continuously evolves to adapt to changing market conditions and regularly needs to incorporate new acquisitions. In this contribution, we show how a formal process model plays a key role in meeting these demands. As examples of the application of this model, we discuss a tailorable Product Life Cycle, and, as an example of more detailed process support in this general framework, the federation of Defect Tracking Systems.

Back to Top

Additional Information

Citation: Jozef De Man, Christof Ebert. "A Common Product Life Cycle in Global Software Development," *step*, vol. 00, no. , pp. 16-21, Eleventh 2003.

Eleventh Annual International Workshop on Software Technology and Engineering Practice
(STEP'03) pp. 16-21

A Common Product Life Cycle in Global Software Development

Jozef De Man, Ghent University
Christof Ebert, Alcatel

DOI Bookmark:
http://doi.ieeecomputersociety.org/10.1109/STEP.2003.1

Abstract

The telecommunication market increasingly demands complex solutions involving a combination of products and requiring co-ordinated development in multiple divisions of a company and even across different companies. Such developments present unique challenges to reach adequate product quality goals within budget and delivery time constraints, especially in a company that continuously evolves to adapt to changing market conditions and regularly needs to incorporate new acquisitions. In this contribution, we show how a formal process model plays a key role in meeting these demands. As examples of the application of this model, we discuss a tailorable Product Life Cycle, and, as an example of more detailed process support in this general framework, the federation of Defect Tracking Systems.

Back to Top

Additional Information

Citation: Jozef De Man, Christof Ebert. "A Common Product Life Cycle in Global Software Development," *step*, vol. 00, no. , pp. 16-21, Eleventh 2003.

Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'03)
September 19 - 21, 2003
Amsterdam, The Netherlands

pp. 16-21 • **A Common Product Life Cycle in Global Software Development**

Jozef De Man, **Ghent University**
Christof Ebert, **Alcatel**

The telecommunication market increasingly demands complex solutions involving a combination of products and requiring co-ordinated development in multiple divisions of a company and even across different companies. Such developments present unique challenges to reach adequate product quality goals within budget and delivery time constraints, especially in a company that continuously evolves to adapt to changing market conditions and regularly needs to incorporate new acquisitions. In this contribution, we show how a formal process model plays a key role in meeting these demands. As examples of the application of this model, we discuss a tailorable Product Life Cycle, and, as an example of more detailed process support in this general framework, the federation of Defect Tracking Systems.