Agile product lifecycle management for service delivery frameworks: history, architecture and tools

N Georgalas and A Achilleos (BT), V Freskos (University of Patras, Greece) and D Economou (University of the Aegean, Greece)

As communication service providers strive to become leaner and more responsive to their customers, product lifecycle management will become a much more commonplace discipline. Already used extensively in other industries, the technique reduces both the costs of developing and launching new products and overall time to market. However, when applied to the service-oriented architectures being adopted by BT and other industry leaders, the process can become highly complex. Products and services may have a myriad of dependencies on the capabilities of constituent parts and other related business entities, all of which must be understood and addressed.

This paper shows that the key to fast adoption of product lifecycle management is simplification. Communication service providers can achieve this by introducing consistent, enterprise-wide lifecycle management frameworks that apply a common management pattern to the lifecycle of products, services, resources and other business entities, building standard management features into business capabilities and service delivery frameworks and using end-to-end model-driven tools.

The paper describes prototypes of the tools required.

1. Introduction

To get new products and services to market faster and in higher volumes, the manufacturing and other industries have come to rely on automated Product Lifecycle Management (PLM) systems. Typically, such systems bring together all the organisational domains and engineering disciplines involved. Furthermore, they help companies manage the complexity of the new product development process.

Until recently, however, PLM was relatively rarely used in the communications business. This is set to change. The industry has been transformed by convergence and the advent of technologies such as 3G and IP, and companies from other industry sectors have begun to enter the market. Products have become much more complex and are increasingly assembled from components whose origins and ownership are diverse. To respond, Communication Service Providers (CSPs) need to change the way they develop, deploy and offer new products [1], increasing throughput and reducing time to market.

The success of PLM in other industries has encouraged CSPs to adopt the technique, adapting it to meet their specific needs.

2. General survey of PLM and associated tools in various industries

As illustrated in figure 1, PLM is the process of managing the entire lifecycle of a product from its conception, through design and development, to operation and retirement.

The technique integrates people, data, processes and business systems and provides a product information backbone for companies and their extended enterprise. It is a mature and highly applied discipline that is widely used in the automotive, aerospace and hi-tech industries, so there is a wealth of experience on which new adopters, such as the communications industry, can call.

This section explores the history and evolution of PLM and the lessons that can be learned from its application in other industries.

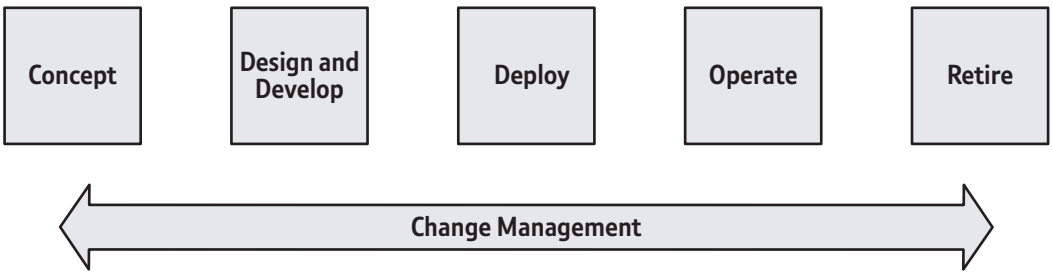


Figure 1. Product lifecycle management process steps

2.1 Evolution of PLM tools

According to CIMdata, a PLM consultancy, the origins of PLM lie in Computer-Aided Design (CAD) [2]. Initially, Product Data Management (PDM) systems were developed to manage the numerous CAD files associated with product designs. Later, functions were added to address product structure, change control, configuration management and other issues, and technologies such as visualisation were introduced to enhance the capabilities and value of the tools.

In response to the requirements of their customers, PLM tools suppliers are continuing to enhance their products. Much more emphasis is being placed on supporting collaboration between the parties involved in the product lifecycle and on the management of product definitions, rather than on just the management of product data. In addition, suppliers are developing pre-packaged solutions that offer generic functionality usable across different industries in addition to tools that meet industry-specific requirements. Web and internet-based technologies and operating paradigms are being incorporated to create solutions that enable increasing numbers of users and types of stakeholders to participate in the definition and use of product and plant information. Both the scope of the lifecycle addressed and the levels of support offered to businesses have increased as new technologies have been applied and new vendors have entered the market.

In outline, the history of PLM tools from their inception to the present day proceeded as follows [3]:

- Invention of various types of CAD tools – the CAD stage
- Integration of CAD data in Computer Aided Manufacturing – the CAD/CAM stage
- Extension of CAD tools – the CAD environment stage
- Integration of a repository and change management mechanism into CAD tools resulting in multiple competing environments for storage and change management
- Realisation that a comprehensive approach to product data is necessary – the PDM stage
- Adoption of enterprise-wide product data repositories supporting heterogeneous product data and tools
- Adoption of enterprise-wide engineering change processes
- Integration and rationalisation of product development processes
- Integration of extended enterprise and outside suppliers into the PDM environment – the collaborative PDM (cPDM) stage
- Visualisation and collaboration
- Extensions to address the full lifecycle through integration with pre-design, production system design and after-sale lifecycle support.

As this shows, PLM and PDM evolved principally in parallel to CAD tools. Taken together, increasing product complexity and the need to boost productivity gains through the use of CAD tools created the need for central corporate repositories of product data.

The communications industry does not have the rich set of CAD tools that exists in other industries, however. In essence, CAD tools build hierarchical models of products that detail the components they use and how they are assembled. Of the tools used in the communications business today, those most similar to the CAD tools used elsewhere include:

- network design tools (fixed and wireless);
- software design tools (IDEs, UML tools, etc.);
- product definition tools (product catalogues, etc.); and
- emerging tools for product and service assembly and Software Development Kits (SDKs) that provide access to CSP capabilities [4].

2.2 Typical PLM tools architectural principles

Commercial PLM tools have been applied to some of the world's largest and most complex projects – projects involving millions of objects and tens of thousands of simultaneous users. In some cases, they have been used to manage engineering data on almost every aspect of a product at almost every stage, from the capture of abstract requirements to the virtual simulation of the production process.

The underlying repository for commercial PLM tools is typically an object-oriented database. To accomplish their job, PLM backbones manage the storage of data from a heterogeneous collection of design tools. Sometimes, they store it as a file tree. Alternatively, it is converted into database objects.

PLM tools do not have a single unifying data model, except for the notion of Bill of Materials[1] (BOM), in which the product is viewed as a composition of parts. In addition to a BOM, tools may also provide object models representing suppliers, issues, requirements and so on.

PLM tools may also allow their underlying data model to be extended for different purposes. To allow users to work with data from a variety of sources, they typically offer Application Programming Interfaces (APIs) to support their integration with other design tools and enterprise applications. PLM tools also allow the creation of general-purpose associative links between stored data and various searching mechanisms, such as full text search or searching on attribute values. Such associative links can be used to loosely connect the engineering data created by various design tools, for example.

Finally, commercial PLM tools are often integrated with sets of general-purpose engineering management applications to support tasks such as project planning, system engineering, visualisation and mark up, issue tracking and engineering change management. Such applications work with whatever data has been stored in the engineering repository.

2.3 Practical issues

To meet the needs of their users and achieve fast and wide adoption, PLM system suppliers have based the designs of their products on a number of practical assumptions.

From their viewpoint, the ideal PLM environment would resemble an Application Lifecycle Management (ALM) tool suite. ALM tools assist in the creation and maintenance of software products and are characterised by their tightly-integrated development environment. In a similar way, an ideal generic PLM environment would provide a tightly-integrated suite of tools from a single vendor that spans most, if not all, aspects of product lifecycle.

In practice, though, PLM suppliers have found the need to support multiple engineering disciplines and multiple vendors' CAD tools. Different engineering disciplines use different tools and organisations often prefer to mix and match 'best of breed' design tools in their development processes. In such situations, PLM tools bridge the outputs of different CAD tools and support generalised system engineering activities. The BOM is used as the organising concept and domain-agnostic system engineering models are used to tie together the work done by different engineering disciplines, such as physical and electronics design.

Beyond the above needs, PLM systems have been required to embrace many forms of product definition, including less-structured forms. The models they must accommodate can be incomplete or inadequate at one extreme, and too formalised or too technical at the other. PLM systems must therefore be able to connect and manage heterogeneous information from sources, including Microsoft Office documents and emails.

2.4 Lessons from other industries

A number of lessons can be learnt from other industries' experience of developing and deploying PLM tools:

- **A single data model can never support every tool.** Industrial experience has shown the futility of the quest for a single universal data model to cover all aspects of product design. As mentioned above, design involves a variety of tools with their own, unique underlying data model. Lifecycle information exported from these tools is, therefore, heterogeneous.

- **PLM information should be held in a 'one-truth' common repository.** The original drive for PDM arose from the need to manage the data created by different CAD tools that may or may not maintain their own data repositories. One of the central value propositions of the PLM systems from Teamcenter and MatrixOne, for example, is their ability to bring all the data about a product together in one single repository overcoming the heterogeneity problem noted in the previous point. Generally, it is important for tools to support a one-truth common repository – a single point of contact regarding product lifecycle data and information.

[1] A Bill of Material (BOM) is a list of all the components and subassemblies that go into a parent assembly. It is the subset of a structure that includes only the physical items. A BOM includes the child items at a particular location in an assembly, their quantity and unit of measure, and other related information such as physical location.

- **PLM tools should support complex webs of relationships between heterogeneous artefacts.** Like Documentum and other document management systems, product data management systems must not only store a wide variety of information, they must provide facilities to interlink it in various ways. Among other uses, the resulting links allow the origins and relationships among product data hosted and managed in different tools to be traced. At the most basic level, linking can be achieved by providing simple pointers between related objects, but many more-sophisticated approaches can also be employed.

- **PLM tools should support enterprise-scale configuration and change management.** As PDM systems grew into maturity and the management of product data was extended to support a fuller collection of product data, it became increasingly important for enterprises to have change management processes that could span their operations. Mature systems now support large communities of users and allow them to participate in the modification of the product data.

- **PLM tools should support extended enterprises and supplier collaboration.** In today's business environment, products are rarely developed by a single company. PDM systems must therefore be able to support collaboration between the members of extended value chains. This becomes more imperative with the advent of Web 2.0, which makes it easy to create complex ecosystems of suppliers and consumers.

3. PLM in the communications industry

The market for communications services is much more dynamic and complex than it was in the past. CSPs can no longer afford to offer limited portfolios of regulated telephony and data services, or for it to take a long time to get new products into their customers' hands. Instead, they must operate in a market characterised by:

- **Software-based services** that can be designed and deployed without the need for substantial infrastructure deployments, and are therefore much easier to adapt as the market and consumer preferences change.

- **New competitors** that, enabled by deregulation, are pressurising margins on traditional products. For instance, among those competing strongly with traditional telcos to sell fixed or mobile telephony packages are the super-markets – the organisations that dominate the retail market.

- **Broadband communications based on IP standards**, which are dramatically lowering the cost of providing phone and multimedia services to customer premises.

- **Soft telcos**. The high market penetration of broadband allows service-centric soft telcos to deliver services without having to invest in physical access infrastructure.

- **Maturing product portfolios** that attract lower margins and are being replaced by new alternatives.

In the light of these trends, the key goal for CSPs is to reduce the time it takes to get new and updated products to market. To achieve this, they must improve and automate their strategy, infrastructure and product portfolio management processes. These are situated at higher layers than operational processes that were the targets of performance-improvement initiatives in the past. CSPs must also acquire the organisational agility to get products to market quickly – for example, by basing products on reusable capabilities and introducing data-driven configuration of operational and business support systems.

3.1 Intra-enterprise product lifecycle

In the case of communications products, end-to-end lifecycles can be complex and difficult to manage. Their complexity results from two factors:

- their structural/compositional complexity (figure 2); and

- their organisational complexity (figure 4).

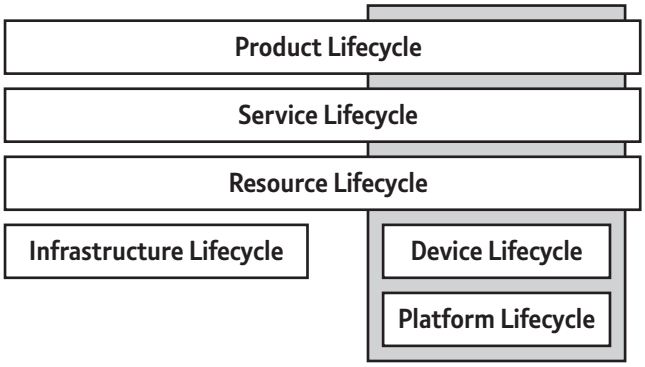
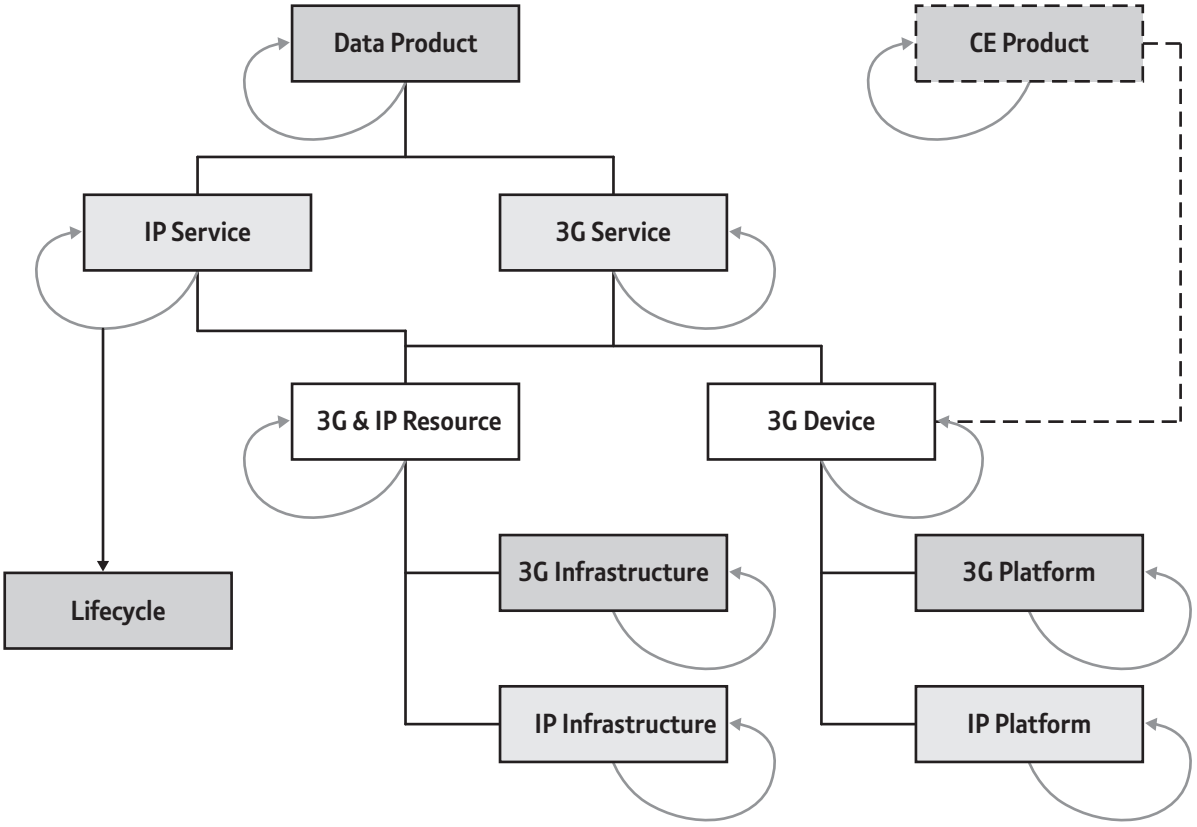


Figure 2. Compositional complexity

3.1.1 Structural/compositional complexity

A product offering is not a single artefact with an independent lifecycle. Under the wraps, numerous other artefacts – services in particular – are brought together to compose the product. For each different product offering, the constituent services will have a customised configuration and each service will be characterised by its own lifecycle, which again is not quite independent. In the different stages of their lifecycle, services engage a plethora of physical resources (routers and devices, for example) and logical resources, such as Commercial Off-The-Shelf (COTS) components. These resources are purchased, configured, deployed and operated



CE – Customer Equipment, 3G – Third Generation, IP – Internet Protocol

Figure 3. Example product composition[2]

as part of the CSP's infrastructure and have their own lifecycle. As a result, end-to-end PLM typically involves the management of numerous nested lifecycles.

Figure 3 illustrates the hierarchical composition of communications products from services and resources. Consider, for example, a product called 'Email on Move' that is to be made available on customer devices, such as mobile phones, connected by IP over 3G or WiFi networks. The product will depend on a number of components of the CSP's infrastructure as well as the customers' devices. Each component will have its lifecycle, consisting of multiple phases, from concept development and design through to deployment, delivery, in-life operation and retirement. The characteristics of the phases – the timescales, costs, design disciplines, delivery modes and so on – will differ vastly from component to component. However, all the components must be brought together in a timely, quality and cost-effective manner if the product is to be successfully launched and sold.

[2] The figure uses straight lines to show dependencies among the different components and circular lines to indicate that each component has its own lifecycle. It also uses a dotted rectangle with an associated dotted line to refer to the generic case of Customer Equipment (CE) products users can enjoy on third-generation (3G) mobile devices.

3.1.2 Organisational complexity

The organisational complexity of the communications product lifecycle is a result of the variety of information, processes, stakeholders and tools involved (see figure 4).

The product lifecycle is characterised by the states products traverse as they move from concept to retirement (from 'cradle' to 'grave'). Each state is described by a set of data, so PLM is essentially about managing product data as the product moves between states. PLM is further complicated because, at each state of the product lifecycle, different stakeholders are involved. As they tackle their tasks,

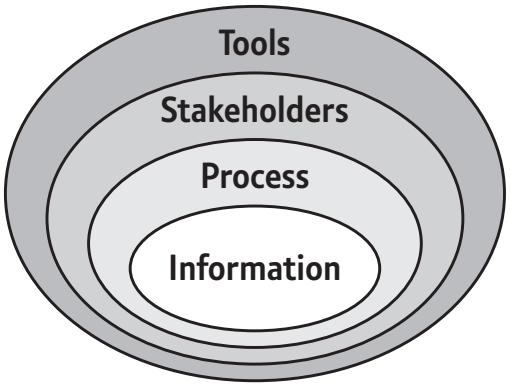


Figure 4. Organisational complexity

these stakeholders collaborate using tools and systems to capture the product data and manage the product lifecycle aspects for which they are responsible. To achieve this, they follow certain business processes. An agile PLM approach should provide high levels of automatic instrumentation to achieve integration across the organisation and facilitate stakeholder collaboration.

3.1.3 The challenges for PLM

Following from the above, the primary challenges for PLM are:

- **The lack of a common industry product model.** Each COTS vendor has developed its own view of a product model from its specific functional perspective. None of them align with the product model used in telcos.

- **The lack of an industry end-to-end product lifecycle.** COTS vendors restrict the data model to their functional area and provide local capabilities to update and maintain their databases.Telcos often find it difficult to connect and integrate the islands of product data that result. There is no industry API standard they can call on to automate the synchronisation and updating of product databases.

- **Performance and non-functional aspects.** The volume of data needed to support products is substantial. As a result, it is not feasible to hold it all in a single database to which all applications direct their queries.

- **The divergence of existing processes.** Traditionally, Operational and Business Support Systems (OSSs and BSSs) have been developed product-by-product in isolated 'silos'. For PLM to operate effectively, the complex array of systems that has resulted must be replaced by a single integrated infrastructure of reusable OSSs and BSSs [5].

3.2 Inter-enterprise product lifecycle

To develop, deliver and manage its products, a CSP may depend on multiple business-to-business relationships as shown in figure 5. Each counterpart plays a specific role in the product delivery process and therefore manages certain aspects of the product lifecycle. The CSP, for instance, will employ tool vendors and OSS suppliers to build the infrastructure it needs to support product development and delivery. Additionally, the CSP may require the services of system integrators to assemble and configure the infrastructure upon which the product will function. To keep costs down, CSPs may make the strategic decision to outsource or conduct offshore all or part of the product development process. Furthermore, they may join forces with other organisations to deliver products to market, sharing the revenues generated.

The products that result may be sold through SMEs, and they may prefer to offer them to customers under their own brand. Alternatively, CSPs may use Software-as-a-Service (SaaS) infrastructure operated by organisations such as BT and Google, either to make SaaS functionality available as part of a product offering or to use the functionality on demand during the product development process.

In all such situations, the control of the product lifecycle crosses enterprise boundaries and flows over the value chain. This results in a set of additional challenges for PLM implementations to address:

- **Data sharing** – subject to security precautions and access-control measures, organisations should be able to manage and share product data flexibly using commonly-understood formats.

- **Collaboration** – processes must operate smoothly across business boundaries complete with all necessary automated support.

- **Project management across organisations** – stakeholders from different organisations should be able to co-ordinate actions and collaborate to deliver project tasks.

- **Quality and performance** – the biggest challenge is the preservation of high levels of performance and the quality of the end result.

3.3 Lifecycle pattern

Analysing the discussion in sections 3.1 and 3.2, the following observations can be made:

- Starting from CSP products and traversing the dependencies that link them to entities such as services and resources (figure 2), it can be seen that lifecycle management is a generic and common concern for an organisation – one that involves business entities other than just products.

- The aspects that are most important when it comes to the successful management of the lifecycles of different business entities are processes, stakeholders, information and systems/tools (see figure 4).

- Where the management of an end-to-end lifecycle involves complex business-to-business relationships, commonly-agreed structures are needed that characterise the entity's lifecycle across all the organisations involved, facilitate the interaction of their activities and simplify the complexity of the lifecycle management process over organisational boundaries.

Based on the first and last observations above, we introduce the concept of a generic lifecycle pattern. This pattern

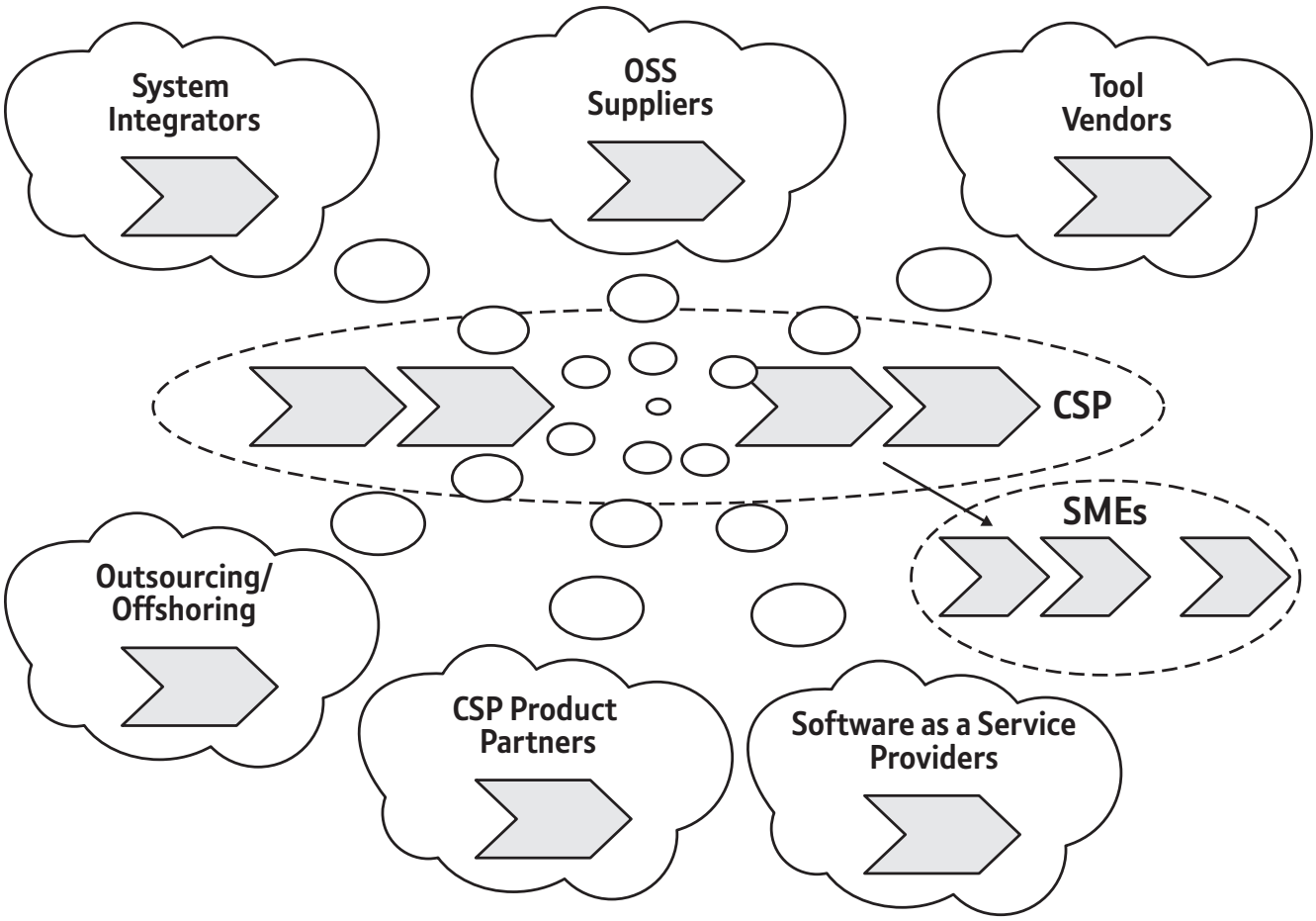


Figure 5. Product lifecycle complexity and value chains

describes the aspects of a lifecycle that are common across different business entities. The use of such a pattern would allow CSPs to address the lifecycle management problem by building solutions that concentrated only on a handful of common concerns. Where complex business-to-business relationships are involved, such patterns would establish a basis of common understanding between the interacting organisations.

The proposed generic lifecycle pattern is illustrated in figure 6. Its structure originates mainly in observation 2 above and comprises the following aspects:

- **Lifecycle management processes**, which specify the sequence of steps undertaken in managing an entity's lifecycle.
- **Roles/actors**, which identify various parties and their responsibilities in the management of an entity's lifecycle, those parties being business stakeholders and/or systems.
- **Information/metadata**, which describes the different states an entity goes through during its lifetime. The entity meta-data is typically classified in four main categories, as illustrated in figure 6, namely:
  - o configuration, which indicates the entity's specification data customised for its adoption in a particular context of use;
  - o usage, which captures performance data regarding how the entity is currently used;
  - o health, which shows whether the entity requires repairing or recovery measures; and
  - o Service Level Agreements (SLAs), which set the thresholds of the entity's acceptable performance levels.
- **Management capabilities**, which characterise the capabilities required to manage an entity's lifecycle. Provided either by stakeholders (manual) or by systems (automatic), they allow them to manage the entity's metadata in the context of lifecycle management process tasks.

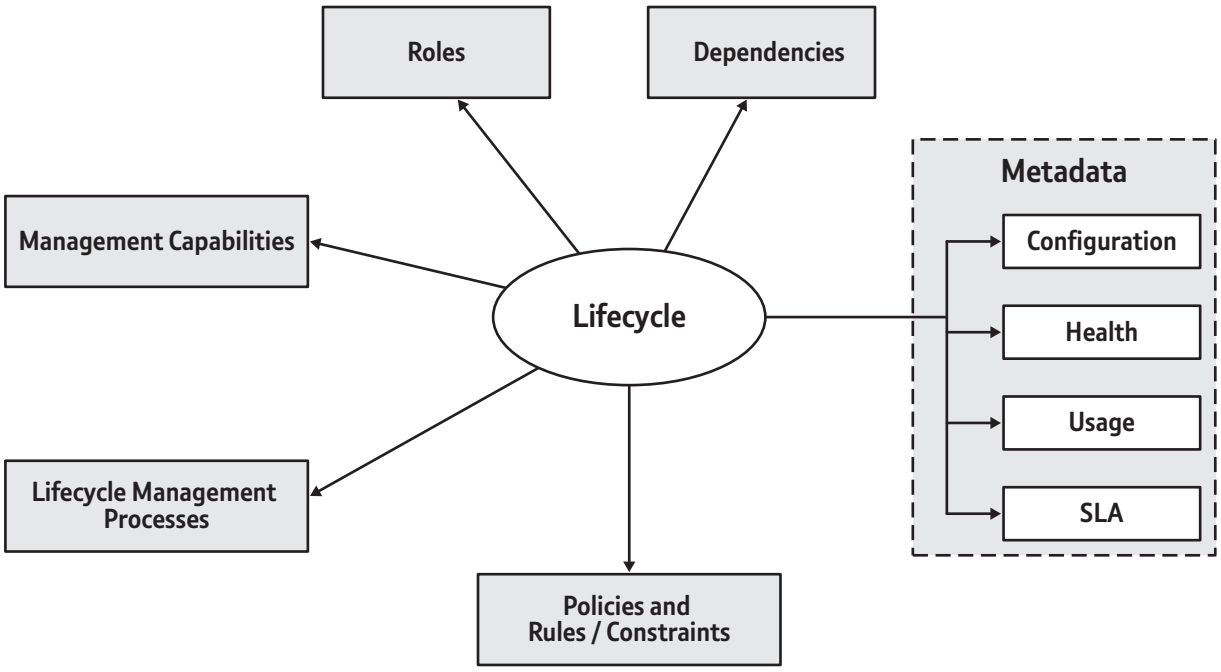


Figure 6. Lifecycle pattern

- **Dependencies**, which identify the set of business entities on which a particular entity's lifecycle depends. This provides traceability among the dependent entity lifecycles, which is very important when changes are introduced.

- **Policies and rules/constraints**, which specify behaviour and/or restrictions applied on an entity at a certain stage of its lifecycle.

3.4 Holistic lifecycle management framework

The lifecycle pattern described in the previous section generalises the issue of lifecycle management by abstracting specific details and concentrating on the concerns that characterise the lifecycles of business entities such as products, services and resources. This lays the ground for the introduction of a holistic framework that, through use of a generic set of management facilities, can manage end-to-end the lifecycles of business entities regardless of their type.

A process-based view of the holistic lifecycle management framework's scope is shown in figure 7. The bottom layer describes the Concept-to-Market (C2M) part, where an entity such as a product or service is initially perceived as a concept and then driven to market through a number of subsequent steps, including design, development, testing and launch (deployment and activation).

The middle layer of the figure shows the in-life stage of the lifecycle, in which entities are instantiated to fulfil customers' orders, operated, maintained and upgraded until eventually they are retired. This stage includes the high-level customer-facing processes BT refers to as Lead-to-Cash (L2C) and Trouble-to-Resolve (T2R) [6].

C2M and in-life stages form a loop in figure 7, which indicates that the holistic lifecycle management framework:

- provides a unified, end-to-end view over the entire entity lifecycle;

- integrates all historic data of the entity's lifecycle gathered from the individual process steps and allows, if required, any decision-making task to access and consider this data in order to draw more rounded conclusions involving the entity;

- enables feedback loops so that changes incurred by, or events influencing, an evolving entity at a later lifecycle stage could feedback into earlier stages and drive changes on aspects of this or other dependent entities.

With regard to feedback loops, an example would be when a customer contacts a customer support centre to provide feedback on a product that is in its in-life stage, recommending improvements to certain product features. Such input might be fed back either to the C2M design and development stage (for implementation of such improvements, leading eventually to a product update) or to the C2M concept stage (in which case, the requirements would be captured as inputs that could lead to new product offerings).

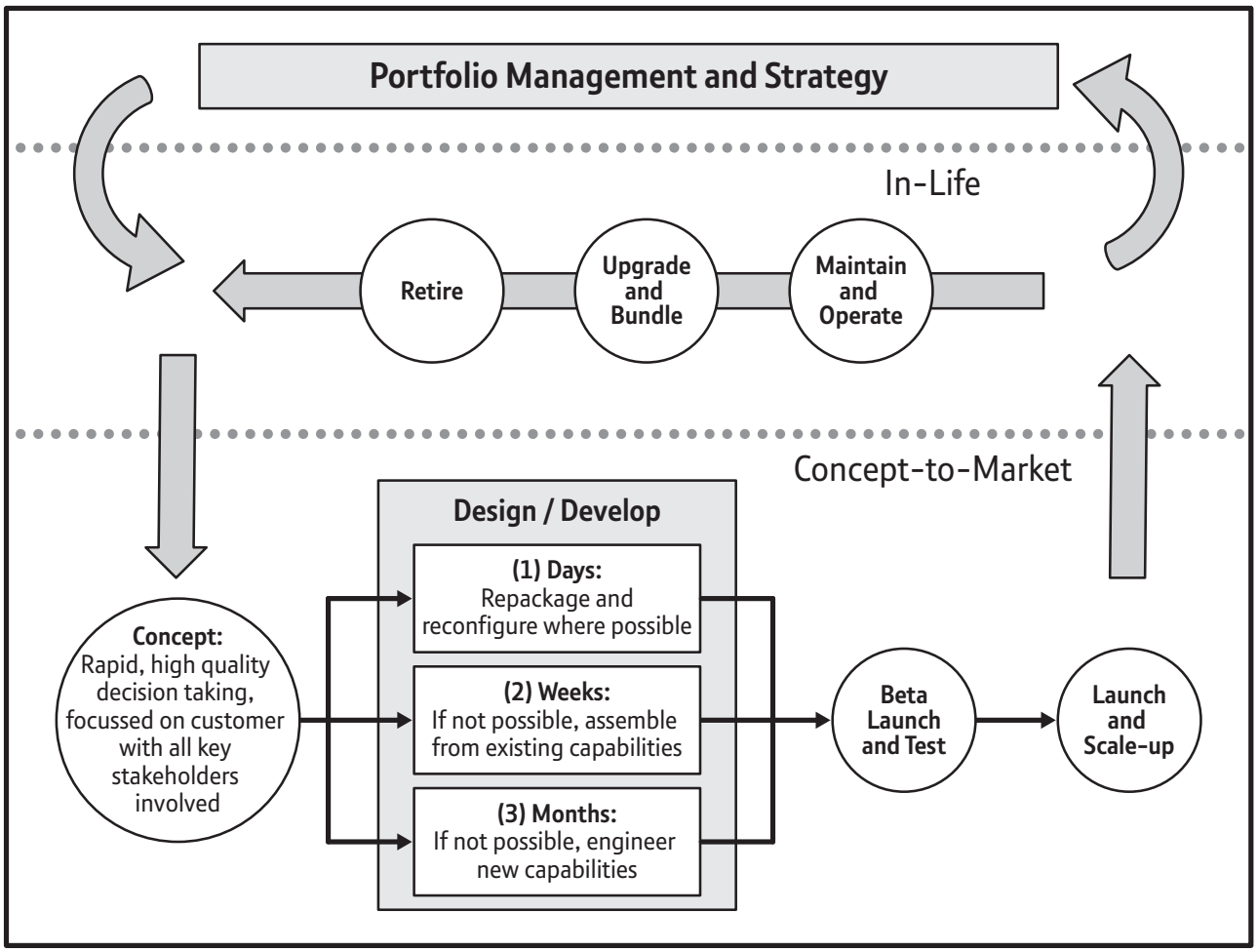


Figure 7. Holistic lifecycle management framework for CSPs

Portfolio management and strategy are shown as the top layer in figure 7. The link to the two lower layers – C2M and in-life – indicates that the proposed holistic lifecycle management framework provides facilities that help decision-makers at this level hook into low-level operational systems and data stores. Such direct feeds from raw entity lifecycle data can significantly enhance the agility with which CSPs' strategies and portfolios are fine-tuned and optimised.

All layers are governed by rigorously-defined change management processes and supporting automation, which both underpin the holistic lifecycle management framework and constitute an integral part of it. Change management guarantees that any changes incurred by a composite entity during its lifetime will be properly carried out and propagated down the entity's dependency chain to all the component entities that must be involved in its implementation. It also ensures that changes are traceable.

A key aspect of the holistic lifecycle management framework's C2M layer lies in the agility with which new entities (products and services) are constructed and driven to market. Best practice suggests that, to achieve such agility, the 70-20-10 rule of thumb should be applied. Introduced in 2005 [7], the 70-20-10 business resource management model indicates the desired distribution of effort over three possible modes of generating new products:

- **Configure (70 per cent).** New products should be specified by configuration of existing products and features. The process is data-driven – that is, no new code is required – and should be sufficiently simple for customer service agents and product managers to complete. This would provide maximum possible reuse as all relevant systems and capabilities are already in place. The configuration should be rapid, at most days – preferably hours.

- **Assemble (20 per cent).** If reconfiguration is not possible, new products should be built by assembling reusable capabilities. This mode would require more complex COTS package configurations for the definition of new rules and processes. It would also involve

assembly of software, network and hardware to enable, for instance, the reuse of an existing capability that needs to be configured in the new context, or the deployment of new hardware to increase capacity. Little, if any, new code would be necessary and the cycle time would be in the order of weeks.

- **Engineer (10 per cent)**. In a minority of cases – for example, where entirely new capabilities are required – more significant engineering will be required. This mode may require new software development, new hardware configurations and/or the building of new network capabilities. Cycle times could be in the order of months.

BT is currently tuning its systems, processes and practices towards implementation of the 70-20-10 rule.

At the heart of a holistic lifecycle management framework reside a set of key management capabilities aimed at co-ordinating the lifecycle aspects of a business entity such as a product, service or resource as it mutates and evolves over its lifetime. These capabilities originate from the meta-model illustrated in figure 8. In this figure:

- **Entity** refers to the key artefact managed by the holistic lifecycle management framework which mutates during its lifecycle, i.e. it gets created, accessed, reused, changed or transformed.

- **Process workflow** refers to the automation that drives the overarching process or processes controlling the change and evolution of an entity.

- **Access control** refers to the set of capabilities controlling access of processes, stakeholders and systems to the business entity state data.

- **Logic** refers to business logic that is owned by the framework and governs the entity or that defines the way a business entity interacts with others around it.

- **State (or lifecycle state)** refers to the states a changing business entity goes through; a minimum of one state is required. There may be an association between state and revision/version capability (see below), which shows that state transitions can be traced.

- **Revision/version** refers to management capabilities of the framework that allow snapshots of the entity's transformation to be captured. Audit trails can be created if this is required by the change management process.

- **Associations** refer to the relationships an entity develops with the others of which it is composed (compositional associations) and with the entities with which it communicates (interactional associations).

Based on the meta-model of figure 8, a holistic lifecycle management framework should provide the following key management capabilities:

- **Repository management.** To ensure a single view that spans the full lifecycle, all the data about a business entity must be held at the same logical place. Repository management polices the 'one-truth' information model and supports associative linking between information items.

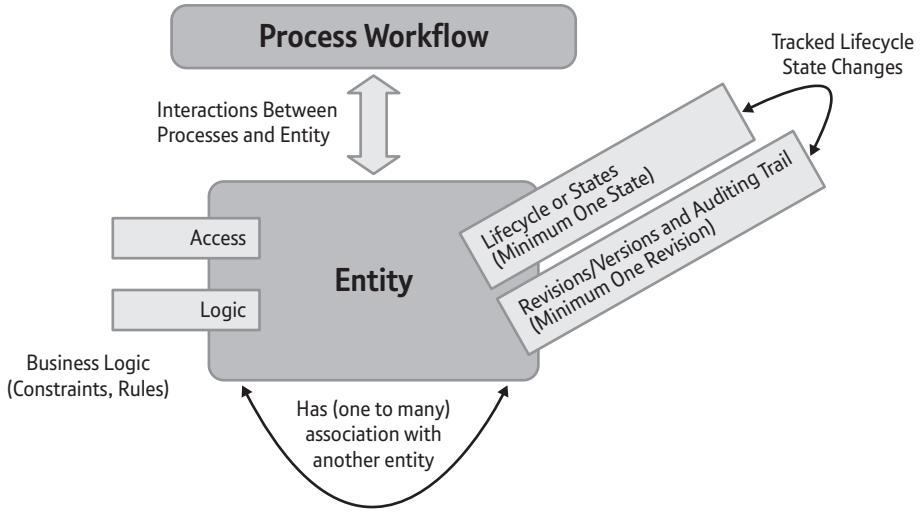


Figure 8. Meta-model for lifecycle management capabilities

- **Configuration management.** This involves both change management and version control. As noted above, change management is a core lifecycle management framework feature enabling the evolution of a business entity throughout its lifetime. Versioning enables the management of multiple revisions of the same entity and can result in creating a history or audit trail of all entity revisions.

- **Information sharing.** This involves two aspects:

    - Searching and visualisation of data – that is, providing querying facilities to obtain the desired data sets and visualisation mechanisms that present the data in ways it can be easily understood and consumed.

    - Access control and data security – that is, providing elaborate security capabilities and ways to partition different levels of authorisation rights to create, access and change data.

- **Process management and collaboration.** This involves two aspects:

    - Workflow, which enables the automation of processes by introducing a controlled way of managing individual process tasks, in sequence or in parallel.

    - Notification, which is a generic mechanism for facilitating collaboration between the human stakeholders or automated tools involved in the management of an entity's lifecycle. Notification can involve the exchange of simple emails or more complex synchronous or asynchronous messaging. To make collaboration effective, notifications should carry a payload of meta-data providing information that helps the relevant process to progress. For instance, a notification informing a stakeholder of the need to review a document could include in its payload a link to the document, a review delivery deadline and additional assisting meta-data.

- **Management of business logic.** This plays a key role in the governance and management of the business entity lifecycle. It manifests itself in the following ways:

    - Functionality – Business logic may be captured as a block of functionality that executes certain behaviour. This functionality implements a software capability, such as management or analysis, which can be invoked at certain stages in an entity's lifecycle.

    - Constraints – The entity should comply with several constraints imposed by its constituent parts, by supporting functionality or by the context within which the entity operates. It is important to ensure that certain validation procedures are in place to guarantee constraint satisfaction in a timely manner and, in case of constraint conflicts, to provide alternative strategies for resolution.

    - Policies/rules – An entity can be governed by certain rules and policies that define expected behaviour in certain circumstances. There are several types of rules. Some are relevant to the processes that manage the entity lifecycle while others are associated with validation, access control and OSS.

- **Tools integration.** A range of tools may be used in the process of managing an entity through the various stages of its lifecycle. For the holistic lifecycle management framework to provide seamless end-to-end management, the tools should interact and be integrated. Integration can be achieved in one of three ways:

    - Function-driven integration, in which tools are integrated through functional capabilities that are requested by one tool and provided by another. Traditionally, there are two ways of achieving functional integration. The first is tight coupling, in which the tool requesting a functional capability is strongly dependent on that which provides it. If changes are made to the tool providing the capability, the tool that uses it will also need to be updated. The second is loose coupling, in which functional capabilities are provided as services which are invoked with calls on a contractually-specified service interface. In general, changes made to individual tools do not have knock-on consequences.

    - Data-driven integration, in which tools are connected through shared product data. To do this, tools must conform to a common information model that ensures a common semantic understanding of data items involved. Shared data may be collected and managed in a common repository in which case the integrated tools will have to check data in and out of the repository as they use it to avoid the chance that more than one tool might attempt to change it at the same time. Alternatively, tools can be integrated as a result of exchanging data directly among themselves. In this case, the tools concerned must be able to parse and understand a common data exchange format.

- **Process-driven integration,** in which the process provides the context within which tools must synthesise and link their outputs.

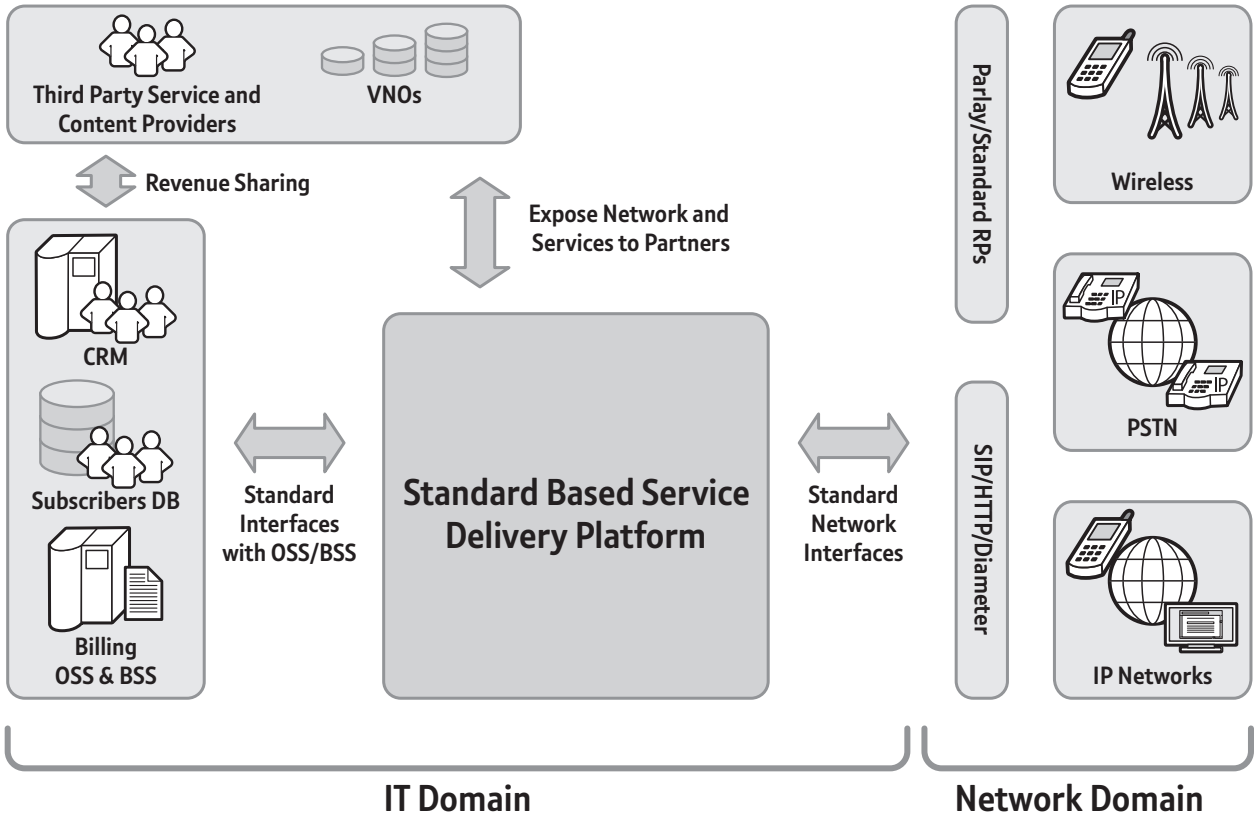


Figure 9. Service delivery platforms

A more thorough examination of the aforementioned lifecycle management capabilities[3] and the key dimensions underpinning holistic PLM frameworks is provided in [8], which was developed by the TeleManagement Forum (TMF) product lifecycle management team [9]. Established and led by BT, the team brings together CSPs and tool vendors from across the world.

The team has also developed a more detailed analysis of the holistic lifecycle management framework's process-based view (figure 7) [10] but, at the time of writing, this was still a work in progress.

The description of the holistic lifecycle management framework provided in this paper relates it to the generic lifecycle pattern presented in section 3.3. More specifically, the process-based view of the framework (figure 7) links to the lifecycle management processes noted in the pattern, whilst the lifecycle management capabilities view (figure 8) links to the pattern's management capabilities. All other pattern items – roles, policies and rules/constraints, meta-data and dependencies – serve as sources of requirements that drive the definition of the specific list of the framework's generic management capabilities covered above. This tight association highlights the key role the lifecycle pattern plays in the holistic lifecycle management framework.

4. Service delivery framework

Leading CSPs are shifting their portfolios from traditional communications to more software-orientated products and services. The blurring of boundaries between networks, ICT and applications allows their capabilities to be integrated to create new-wave services. Service Delivery Platforms (SDPs) [11] facilitate this integration.

More specifically, SDPs enable the rapid development and deployment of new converged multimedia services. These services are composed of telecoms and IT capabilities. As illustrated in figure 9, an SDP sits in the middle, bridging different sources of service capability. Examples of such capability include telephony, wireless, IP, content, OSS and third-party services. Capabilities are exposed through standard functional interfaces. SDPs typically provide a service control environment, a service creation/assembly environment, a service orchestration and execution environment and abstractions for media control, presence/location, integration and other low-level communications capabilities. They are used for the composition of both consumer and business applications. To make them carrier grade (that is, sufficiently reliable and

[3] The term "meta-capabilities" is used in [TMF TR137] when referring to the management capabilities of the holistic LM framework, which is due to the capabilities' generic nature and horizontal treatment of different business entity types.

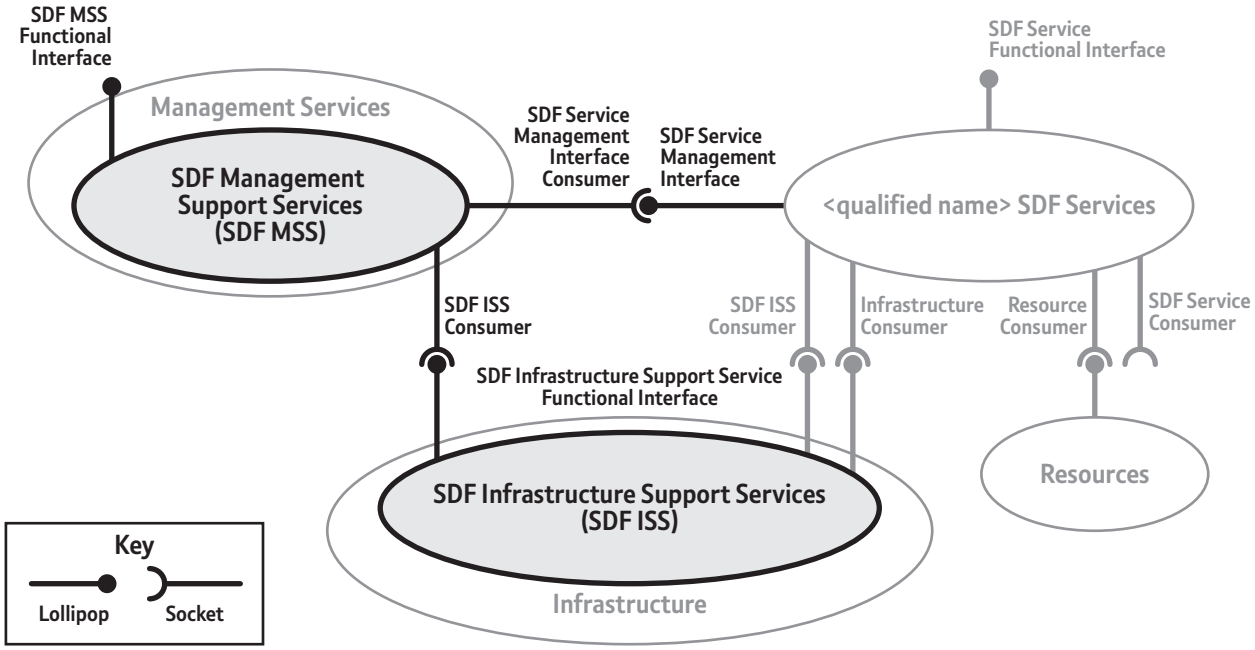


Figure 10. TMF service delivery framework reference model
(Note: The elements shown in grey are part of the reference model but fall outside the scope of the service delivery framework)

scalable for CSPs to adopt them), best practice requires the principles of Service-Oriented Architectures (SOAs) to be applied to the exposure of service capabilities [12]. An SOA-based SDP will generally use web service or other SOA technology standards to integrate services and compose applications.

The SDPs available today are optimised for the delivery of services in a given technological or network domain (examples include web, IMS, IPTV, Mobile TV, etc.). Few standards exist, which is why the TMF's Service Delivery Framework (SDF) programme was created. In the context of SDF, services are defined as components that expose their functionality via one or more functional interfaces. A service becomes an SDF service when it exposes one or more operational interfaces whose role is to manage the service lifecycle. (These operational interfaces are later defined as SDF service management interfaces.) The SDF programme focuses on the management of the SDF service lifecycle.

Figure 10 shows the reference model for the TMF service delivery framework [13]. A typical SDF service is shown top right. SDF services expose functional capability through Service Functional Interfaces (SFIs), which are shown in the figure as 'lollipops' (see key). SDF Service Management Interfaces (SDF SMIs) are special types of SFI that contain the lifecycle management capabilities of the SDF service. Such capabilities include, but are not limited to, configuration, performance management, retirement, fault handling, versioning, monitoring and usage. Finally, an SDF service may itself rely on capabilities exposed by other services. Such SDF service consumers are shown by 'sockets' in the figure (again, see key).

The rest of figure 10 illustrates the SDF reference model comprising:

- **SDF Management Support Services (SDF MSSs)** which are responsible for the end-to-end SDF service lifecycle management. They support both operational (e.g. provisioning, installation, update/activation, monitoring capabilities) and business process automation. SDF MSS capabilities may either invoke SDF SMI capabilities, in which case they receive SDF service management metadata, or invoke other support services from the infrastructure or the management services domains of figure 10.

- **Infrastructure and SDF Infrastructure Support Services (SDF ISSs).** The infrastructure domain provides specific capabilities to management processes or SDF MSSs that are usable across all SDF services and facilitate their lifecycle management. These capabilities constitute the SDF ISSs. Examples of SDF ISSs include SDF service catalogues, metadata repositories, profile (specific information for subscribers or other actors), resource management capabilities and charging capabilities.

- **Resources** which are capabilities that can be used by SDF services and are exposed by network, IT infrastructure, OSS/BSS applications, or services on the internet[4]. They can exist anywhere, either within or outside the CSP's domain, and offer their capability to SDF services through their functional interfaces.

[4] If we look back in figure 8, all capabilities exposed to an SDP from all surrounding domains (3rd party, wireless, telephony etc) over the standard functional interfaces can be considered as SDF resources.

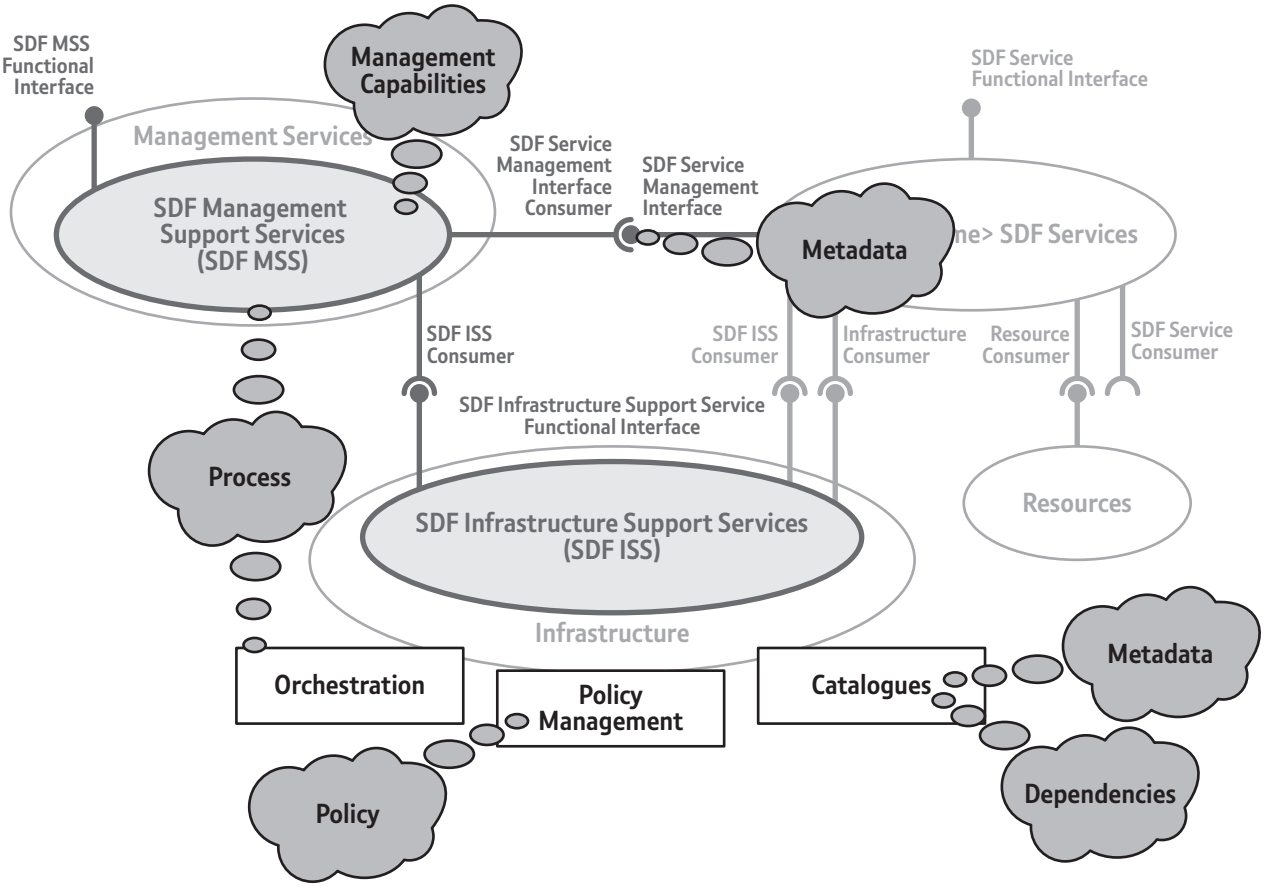


Figure 11. Mapping the lifecycle pattern on SDF

5. Mapping lifecycle pattern on SDF

This section applies the lifecycle pattern, described in section 3.3, to the service delivery framework. The mapping is illustrated in figure 11, which shows how the different lifecycle pattern aspects are distributed around the various parts of the standard framework. The mapping is specific to the study of the lifecycle pattern in the context of SDF services.

At any one point in time, the state of an SDF service is reflected in the set values of the service lifecycle metadata. Some of this metadata is performance-related, illustrating the usage and health of the service. It changes as the service is used and can be accessed through the SDF SMI, which is the interaction point for lifecycle checks as the service operates. The piece of service lifecycle metadata relating to service configurations and SLAs changes less frequently. It is hosted on service catalogues and inventories that constitute part of the SDF supporting infrastructure. Access to this metadata is provided through SDF ISS interfaces. The actual logic, which invokes the SMI and ISS interface so that lifecycle metadata can be processed and the service be managed, lies in the SDF MSSs. These reflect the management capabilities encountered in the lifecycle pattern. Some of the SDF MSSs may implement business process automation that reflects the lifecycle management process item of the lifecycle pattern. Such processes may be specified in the form of workflows or orchestrated service invocations executed in orchestration mechanisms provided by the SDF infrastructure.

The SDF infrastructure may host further supporting facilities for the remaining lifecycle pattern aspects. It can provide policy-based management facilities in order to handle service policies and rules, another important aspect of the lifecycle pattern. It can also use the service catalogue to capture dependencies of a service onto other component artefacts. Finally, it can offer authentication and authorisation mechanisms for role-based access control in order to define roles, materialised by human stakeholders or systems, and assign privileges to those roles regarding the scope of operations they can conduct within the SDF or define management capabilities they bear and can make available to the SDF.

6. Tools

Due to the key role data plays in managing the lifecycle, the data management discipline sits at the heart of PLM. This section focuses on describing current practice and tools in capturing/specifying, managing and sharing product data in the CSP estate of OSS/BSS, service capabilities and platforms. The section is divided in four subsections which, in

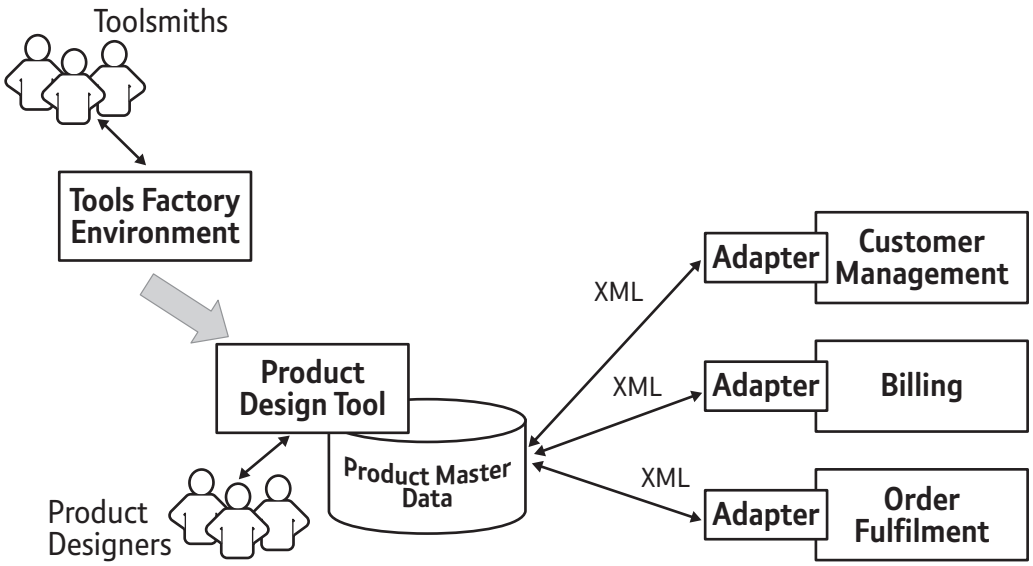


Figure 12. Master data management and PLM tooling driving OSS/BSS platforms

sequence, present master data management, current practice for defining new products, cutting edge research work focusing on a factory-based approach for the production of product specification tools and a general discussion around the proposed approach.

6.1 Master data management

Any data that is considered key in the core operation of a business is called master data. Master data may include data about products, customers, employees, inventory, suppliers, analytics and more. It is typically shared by multiple stakeholders (users and groups) across an organisation. In particular, because OSS and BSS are so deeply involved in the management of the product lifecycle, it is important that a CSP's OSS and BSS systems access and manipulate product master data in a consistent and seamless manner.

Master data may be stored on different systems that manage different aspects of the lifecycle. When this is the case, however, frequent exchanges of data will occur as tools interact with these hosts to provide a seamless end-to-end lifecycle view.

In all, the activity of master data management includes the technology, tools, and processes required to create and maintain consistent and accurate lists of master data. It is used by business and IT to ensure uniformity, stewardship and accountability of shared information assets. The role of master data management ties neatly into the key management capabilities of the holistic lifecycle management framework reviewed in section 3.4 and, particularly, to repository management, which has the job of maintaining a single-view one-truth information base for the artefacts involved in the product lifecycle.

BT has recently adopted a master data management approach. Traditionally, when the company introduced a new product, it would document the product description manually before working out which OSSs and/or BSSs would require development to provide the necessary service wrap. OSS/BSS development was a manual process involving stakeholders such as developers, system integrators and even vendors. The process was also time-consuming, costly and error-prone, and contributed to product master data being dispersed across numerous OSS platforms [14]. As BT seeks to reduce product introduction time, an agile and automated way of configuring OSSs and BSSs is required to replace the manual activities of the past. For this reason, BT is currently implementing an independent Master Data Management Platform (MDMP) as a component of its Matrix architecture [15][5].

A simple view of BT's master data management architecture is shown in figure 12. The MDMP builds and delivers product master data sources and executes the data governance function for the BT systems estate. To do this, it maintains an enterprise-wide data model and oversees the operation of data integrity and quality tools and processes. MDMP exposes a set of capabilities that not only allow master data management functionality to be used by humans and systems but also invokes capabilities on other OSS platforms to distribute product master data. These capabilities allow MDMP to collate product description data captured by product modellers/managers, store it using a commonly accepted enterprise-wide data model, and drive the configuration of OSS and BSS to support new product

[5] In the referenced paper, the master data management platform is referred to as the portfolio management platform.

introduction. Changes may be required to the design and implementation of OSSs and BSSs to facilitate data-driven configuration. The systems must offer a new XML-based access interface in the form of an adapter through which product master data can be communicated. The adapter's role is to transform product data between XML and the support system's native format as it flows to and from the MDMP. Streams of product data exported from the master data store in XML format will populate data structures internal to the target OSS platforms once their adapters have transformed it into their native data format. If product data needs to be exchanged between two or more OSS/BSS platforms, it will first be transformed into XML by the adapter of the originating OSS and then into the respective OSS internal formats by the receiving OSS adapters.

MDMP provides the data foundation for SOA capabilities across BT's Matrix architecture. It makes OSS and BSS platforms data driven, which removes hard coding and maximises data reuse. This is a very important step in the adoption of an agile PLM framework.

6.2 Current practice

As shown in figure 12, tools can help product designers specify and feed an MDMP with valid product data sets. Two key business stakeholders act as product designers:

- **Product modellers** who start with a new product concept described in marketing and product requirements documents and articulate a specification of the product's structure, features and configurability. The product specification will eventually generate change requests on the OSS/BSS platforms to build in the necessary modifications to support the new product. Typical timescales for implementing such an OSS/BSS engineering job range from six months to a year. Current practice for product modellers is to capture the product specification in text documents and use general-purpose drawing tools for modelling. This documentation will be shared with the OSS/BSS developers and the stakeholders who will eventually introduce the product data into the MDMP.

- **Product managers** who enable changes to the configuration of an existing product or construct new products using product templates. Product templates are reusable and configurable product structures that are specified by product modellers and are already implemented in OSSs and BSSs. Because changes only require in-life product configuration data updates to be made to the systems, the time to complete the work is dramatically reduced. Current practice for product managers is to specify products using platform-specific form-based user interfaces provided by the MDMP.

Both groups specify products using tools that are either general-purpose or tied to a particular MDMP. For instance, while using drawing tools such as MS Visio gives product modellers the flexibility to tailor product model diagrams to their requirements, the diagrams must be described explicitly in accompanying documentation. The information will drive the job of the OSS/BSS developers who implement system changes and the individuals updating MDMP product master data stores. This setting bears an element of risk as the slightest ambiguity in the documentation leaves room for misinterpretation by stakeholders. Agility also suffers in that costly iteration may be required in order to remove vagueness and avoid error in the MDMP updating and OSS/BSS configuration process. Furthermore, the first time product data is captured in a formally structured way is when it enters the MDMP, rather than from the start of the concept definition process. The data is organised according to the data model introduced by the MDMP. This data model is MDM platform-specific and, as mentioned above, is widely adopted across all OSS/BSS platforms (through the adapters).

Thanks to widespread support among OSS and BSS suppliers, the MDMP model has become the preferred and almost established way for product data modelling among the CSP product design community. This, however, does not favour the adoption of a higher-level enterprise-wide product information model that relates more directly to an organisation's internal business capabilities (an example of such model is BT's Common Capabilities Model, which is explained later in this section). One reason for this may be the lack of tools that can make a high-level information model usable for stakeholders. Even where such tools do exist, they may not integrate with the underlying platforms in a way that allows product data to be fed automatically into master data management and OSS/BSS. In such cases, product designers may prefer to work with platform-specific data models that ensure product data will be recognised directly by the underlying systems.

6.3 Model-driven factory for PLM tooling

Unfortunately, current practice does not:

- automatically drive the PLM process from product concept inception all the way through to OSS/BSS configuration;

- minimise the effort expended in iterative interactions of stakeholders involved; and

- maintain enterprise-wide information stewardship through a high-level product information model.

Our research work introduces an alternative approach that does deliver these benefits. It proposes the adoption of an end-to-end product design tool chain that allows

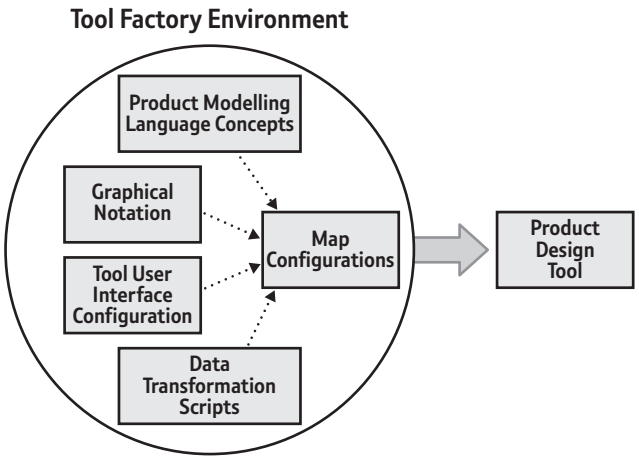


Figure 13. Product design tool factory environment

designers to specify products graphically in an unambiguous way, share their specifications with other stakeholders and exchange product data among different systems in different formats. The product design tools provide a product modelling language with a specific graphical notation. The product modelling language is an information model containing product-specific concepts a designer can use to articulate knowledge regarding a product. Such concepts include product offering, product specification (features and component parts), pricing information and rules. An example of product modelling language is the product domain of TMF's Shared Information and Data model (SID). A similar and more specific example is the portfolio package of BT's Common Capability Model (CCM). The CCM describes common capabilities of BT's Matrix architecture, and its portfolio package is a Unified Modelling Language (UML) class diagram defining product specification concepts.

The approach employs the service of a product design tools factory, which is a production environment 'manufacturing' tools for product modelling. The factory follows a generic process of constructing domain-specific tools which uses principles of the Model Driven Architecture (MDA) and is defined in our prior work [16]. In the context of product design, the process is characterised by five tasks, as illustrated in figure 13:

- **Definition of the product modelling language concepts.** The elements of the product specification language are captured in the form of a meta-model. An example of such a meta-model is shown in figure 14. It was developed in Borland's Together 2007 environment and presents in UML notation a model of concepts and relationships that are part of BT's CCM Portfolio package. This gives the abstract syntax of the product definition language. If necessary, the language abstract concepts can be augmented with constraints imposing special restrictions on the way they are used.

- **Definition of the language graphical notation.** The language's graphical notation is defined. At this stage, all graphical objects are defined, such as rectangles, boxes and

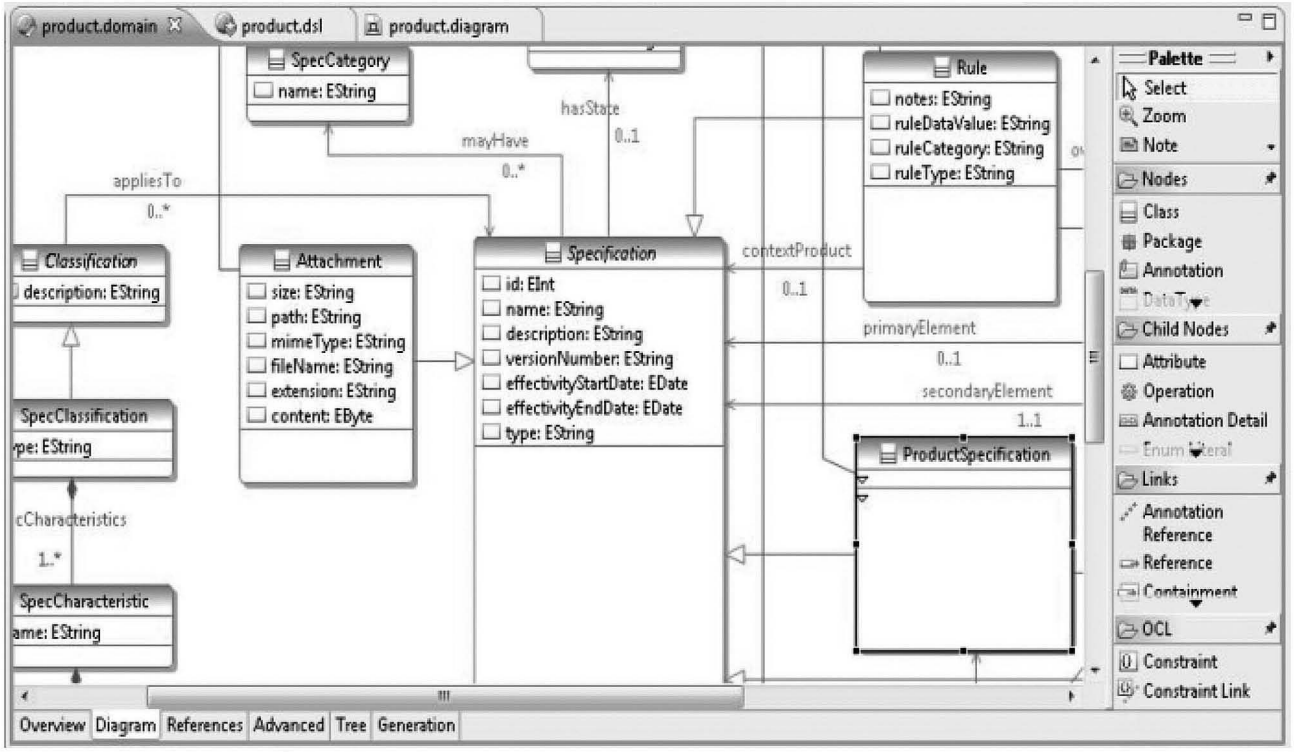


Figure 14. Tool factory environment defining the product modelling language

connectors, which will be used to visually represent the language concepts in a diagram. The only concern involved in the exercise is the capture of a visual objects library. The factory environment provides a default set of geometrical objects with which to populate the library as well as facilities to import icons or other objects constructed externally in drawing tools. Mapping the visual objects to language concepts is part of the mapping and tool generation task.

- **Configuration of tool user interface features.** Presentation characteristics of the generated tool are configured. This step organises individual modelling language concepts in separate groups on a palette. The palette is made available in the generated tool. In the process of modelling a product, product designers access the language concepts through this palette and drag-and-drop them on the drawing panel to gradually construct the model diagram. An example of such a palette is shown in figure 15 in the right-hand side of the depicted tool.

- **Development of data transformation scripts.** Data transformation templates are constructed. Their role is to convert product specifications defined in the product modelling language supported by the generated tool (for example, BT's CCM) into other data formats (such as XML). The templates are programmed in a scripting language provided by the factory environment. An example of such a script written in the Xpand language is shown in figure 16.

- **Overall mapping and tool generation.** The results of all the above tasks get associated. The visual objects of the library constructed when the language graphical notation was defined get mapped onto the abstract language concepts created when the first task was completed. Every language element now has a visual representation. As the palette is mapped onto the language elements, a place is reserved for each language concept which provides references to the concept and its visual aspect. Finally, the data transformation scripts are pulled in to be provided as an add-on utility in the modelling tool. With these mappings in place, the factory has all the necessary knowledge to manufacture the desired product modelling tool. The result is a standalone graphical editor, such as the example illustrated in figure 15, strictly dedicated to the semantics of the product specification language engaged.

The example product modelling tool illustrated in figure 15 uses BT's CCM product portfolio package as the modelling

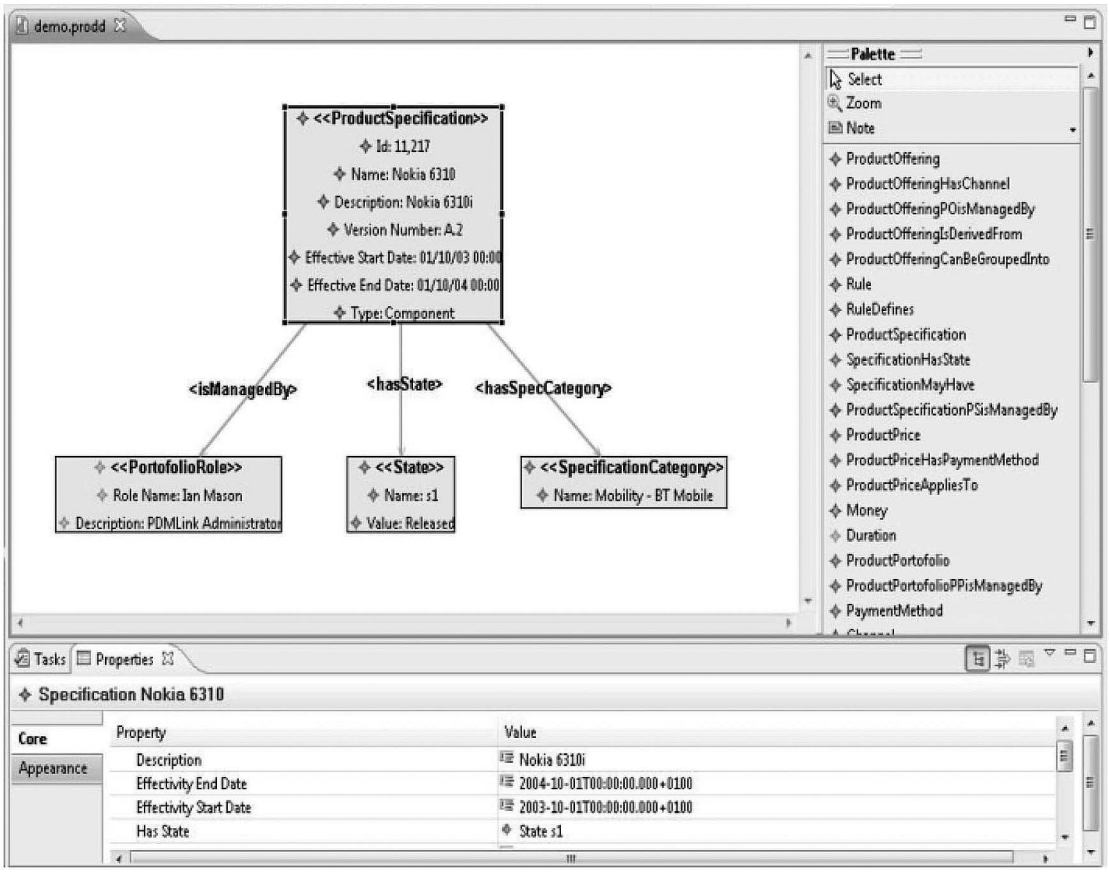


Figure 15. Generated tool with product model example

Figure 16. Code snippet for CCM-to-XML transformation template

language to specify products. It is developed by applying the aforementioned process steps in the domain specific language utility of Borland's Together 2007 shown in figure 14, which served as the tools factory environment. The layout of the produced tool is that of a quite standard graphical editor. It mainly comprises a drawing panel where the product model is to be specified. Product specification language constructs can be drag-and-dropped on the panel from the palette on the right-hand side to gradually build the product model. The palette in figure 15 shows the modelling entities of BT's CCM. Finally, the area at the bottom of the figure is dedicated to editing properties of each class in the product model diagram. For instance, this area shows the properties of the *ProductSpecification* class named 'Nokia 6310', which is part of the illustrated product model example. The tool has been configured so that each class in the product model is stereotyped with its meta-type – that is, with the type of model language construct of which it is an instance. For instance, <<ProductSpecification>> denotes that class 'Nokia 6310' is of type *ProductSpecification*, a product modelling language construct defined as part of


```
<?xml version="1.0" encoding="UTF-8" ?>
- <ProductSpecification>
    <Id>"11217"</Id>
    <Name>"Nokia 6310"</Name>
    <Description>"Nokia 6310i"</Description>
    <VersionNumber>"A.2"</VersionNumber>
    <EffectiveStartDate>"Wed Oct 01 00:00:00 BST 2003"</EffectiveStartDate>
    <EffectiveEndDate>"Fri Oct 01 00:00:00 BST 2004"</EffectiveEndDate>
    <Type>"Component"</Type>
  - <PortofolioRole>
      <RoleName>"Ian Mason"</RoleName>
      <Description>"PDMLink Administrator"</Description>
    </PortofolioRole>
  - <State>
      <Name>"s1"</Name>
      <Value>"Released"</Value>
    </State>
  - <SpecCategory>
      <Name>"Mobility - BT Mobile"</Name>
    </SpecCategory>
</ProductSpecification>
```


Figure 17. Snippet of product data in XML format

CCM in the first task of the tool generation process. A similar example involves all relationships in the model, which are also stereotyped.

Data transformation templates, developed when the data transformation task of the tool generation process is completed, accompany the produced tool as an add-on utility. These utilities enable the tools to convert product models into various data formats. The script shown in figure 16 is an example of a CCM-to-XML data transformation template. It ensures all product specifications created in the tool of figure 15 by means of the CCM modelling language can be transformed into XML format and be serialised in an export file. An example of such XML output is shown in figure 17. It is produced by the template of figure 16 and represents in XML the product model of figure 15. One use of the XML-formatted product data could be the configuration of the OSS, similarly as MDMP does in figure 12. In this scenario our approach substitutes the role of MDMP. Another use of the XML data could be to automatically populate the product master data store in MDMP, removing the need for manual data updates by product managers. In order to facilitate the updates, MDMP should provide an XML adapter interface. Alternatively, transformation templates could be developed in the tools that convert CCM-specified product data directly into MDMP data model format, in which case the product master data store can be updated over MDMP's native access interfaces. From the above two scenarios, the latter indicates a more preferable use of our approach in combination with MDMP because it utilises benefits of both camps whilst keeping enterprise custom-configured PLM tooling and MDMP loosely-coupled and complementary.

6.4 Discussion

Incorporating the proposed product design tools factory in the environment of a CSP has a number of advantages over the established current practice:

- **Data accuracy, right-first-time and short cycle-time.** As noted earlier in this section, currently product designers prefer the use of general-purpose drawing packages, like MS Visio, for modelling new products. In order to make this collateral shareable for review and use by other stakeholders involved, documentation is required to thoroughly explain the diagrams. The inherent ambiguity of this approach results in an error-prone process characterised by costly iterations before stakeholders can establish common understanding and perform MDMP or OSS/BSS configuration updates. The tools factory approach takes manual effort out of the design process and improves accuracy. It engages stakeholders in using a formal methodology and tools with a rigorously defined product specification language. Product models are therefore unambiguously specified and commonly

understood by all stakeholders involved in the process. Additionally, the tools can automatically drive MDMP or OSS/BSS by updating master or system data stores with product data expressed in the appropriate formats. Consequently, the proposed approach reduces iterations and errors and improves the processing cycle time.

- **Enterprise-wide product information stewardship.** The factory manufactures tools, which introduce a common product information model as the specification language for products. This allows flexibility to adopt as such a language any high-level product information model developed and owned by the CSP. Such is the case of BT's CCM. CCM has been developed in BT, evolved over time and ties neatly in the organisation's recognised Matrix architecture of reusable capabilities. Therefore, it is the best choice of language to govern the way BT products are specified and to maintain high-level information stewardship across the organisation. Product specification tools, such as the one reviewed in this section, can be devised by the factory to embrace CCM and assist its BT-wide use. This will benefit current practice in product modelling which does not engage CCM but, as noted above, is rather driven by general-purpose drawing tools, product specification documents and the data model endorsed by the MDMP of choice.

- **Evolution.** Any changes in CCM, or any other product specification language, can propagate and materialise in the supporting tools by adjusting the meta-models in the tools factory environment. This way the approach maintains the toolset in line with the direction in which the CCM evolves, providing modelling consistency and data integrity. Additionally, the factory offers utilities for tools backward compatibility so that older product specification data would still be accessed and managed by the new toolset versions.

- **Fast adoption.** Incorporating the PLM tools factory in BT, or any other CSP environment, does not introduce significant integration tax or impose special requirements for change on the established PLM business process and associated infrastructure. Rather, the factory plays a complementary role as the generated tools mainly focus on automating parts of the PLM process and on feeding with product data the MDMP and OSS/BSS platforms.

Let us elaborate on the last point – that is, on the factory's fast adoption by a CSP and its influence on the PLM process and associated infrastructure.

First, the incorporation of the tools factory into the CSP adds a formal product modelling task to the C2M phase of the PLM process that enhances or replaces the informal one currently practised by product designers. During this task, designers use the generated product-specification tools and apply either CCM or any other tool-embedded high-level product information model to capture product data. Not only does this elevate the attention to CCM, which is currently under-utilised in favour of the MDMP data model and the informal Visio product model diagrams, but it also introduces a strict data capture method that is governed by a tool. Additionally, in the tasks of the PLM process that follow the C2M phase, stakeholders share product data using the tools that remove ambiguity, which enhances collaborative interactions and keeps design iterations to a minimum. Relevant stakeholders will, of course, have to be trained in the use of product modelling tools. However, because the graphical interfaces of these tools comply with the Eclipse Framework [17], which is common ground for designers and developers, stakeholders will often be able to teach themselves how to use them, eliminating the need for specialised training.

Second, the incorporation of the tools factory should not result in significant changes to CSP's infrastructures. Two factors make generated tools CSP-platform-independent:

- the modelling language used by the tools is based on a high-level product information model that is not tied to a particular technology; and

- tools are standalone and independent of the MDMP and OSS/BSS platforms.

Take for instance the product portfolio package of BT's CCM, which is used as the modelling language for the product modelling tool presented in this section. CCM is a model of BT's Common Capabilities, developed in UML. It does not bear any association with particular technologies or depend on the choice of MDMP (e.g. Oracle's product information management hub) or OSS/BSS (e.g. Siebel). The tools integrate with platforms through special data transformation templates that are programmed in the factory and accompany the tools as add-on utilities. Product data captured in the tools could then transform either into MDMP and OSS/BSS platform native data formats or into XML and update platform data stores respectively over either native platform APIs or dedicated XML adapters. Again, as noted above and illustrated in figure 12, the recommended use of the factory and generated modelling tools is to complement the MDMP by data driving its product master data stores rather than to substitute the MDMP's role by directly data-driving the OSS/BSS platforms.

The job performed by the modelling tools' data transformation templates resembles that of the OSS adapters. The former are engineered to convert product data from CCM to another data format, such as XML. The

latter accommodate data exchanges between MDMP and OSS and are programmed to transform data from a common data exchange format, usually XML, into OSS native formats and vice versa. An interesting idea would be to look at whether custom data transformation templates can be developed in the tools factory environment to play the role of OSS adapters. In other words, OSS adapters are developed in the factory as templates programmed to achieve the same transformations. In the very frequent case of new product features or of completely new products being introduced, the required OSS development includes effort to implement additional mappings in the existing adapters. These map the new product data to their respective and newly developed OSS internal representations. This is a very costly and complex job which potentially could be significantly simplified if the adapters are implemented as templates within the integrated tools factory environment. The idea is pending further justification through future research work.

The introduction of the tools factory creates an associated set of new CSP stakeholder roles, primarily that of 'toolsmith' (see figure 12). Toolsmiths are responsible for the process of manufacturing product modelling tools using the factory (see figure 13). Tools are developed in line with the CSP's authorised product information model, such as CCM, which constitutes the product modelling language. Therefore, toolsmiths have close interactions with the CCM modellers and owners in order to co-ordinate their activities in synchrony. Any change in CCM should drive a respective change in the modelling tools. The number of individuals performing the toolsmith role should be kept low in order to restrict exposure of the tools factory environment. This way, all the complexity of content and technical mechanisms driving the tools factory production process remains encapsulated away from the eyes of the wider business community and exposed only to those few who are specialised and trained for the job. It is expected that tools users will significantly outnumber the toolsmiths. Beyond the production process lie tools maintenance tasks including deployment and distribution, installation, local configuration and patch upgrading. Because these are typical software maintenance tasks, tools could equally be handled by the respective processes already established in the company.

7. Conclusions

With increasing market pressure, CSPs are streamlining their processes to expedite new product introduction at reduced cost and time-to-market. In this context, PLM is seen as a key discipline for the communications industry, although it is still under-adopted in comparison to other industries. The need to accelerate PLM adoption is further increasing due to CSPs changing their product development approach from the traditional telecoms-orientated to one of a more software-orientated nature.

While shifting rapidly to a more SOA-based delivery of its capabilities, BT is incorporating a PLM approach in conjunction with its Matrix architecture. First steps in this direction include the inclusion of a master data management platform in the architecture. The challenge of implementing just this one move is vast. All of BT's OSS and BSS platforms need to interface to the MDMP and to interact with each other through it to complete all necessary data exchanges. Further steps include more agile lifecycle management processes and operations and more automation that would remove manual effort, long cycles and cost from product development.

CSPs should increasingly support standardisation work that specifies cross-industry PLM process transformation guidelines, lifecycle management interfaces for communication services and interfaces/capabilities for telecoms-orientated lifecycle management systems. Such effort is currently ongoing in various teams of the TMF, and this paper has shown how the lifecycle pattern relates to the TMF's service delivery framework as a baseline for the direction of PLM standards.

It is equally important that CSPs support, through enterprise-wide programmes, the construction of integrated lifecycle management environments that adopt an end-to-end tools-orientated approach and deliver cross-enterprise PLM processes. Such environments should provide the tools that unify the C2M, L2C and T2R phases of the lifecycle.

The tracing of complex dependency chains that involve nested business entity lifecycles with intra- or inter-enterprise spans is important to the success of lifecycle management. Agile management of such complex lifecycle chains will only be possible through full PLM process automation and appropriate tooling support.

Acknowledgements

The main author would like to express his gratitude to Keith McKnight, Tommy Loughlin and Helen Hepburn of BT's MDMP team for their support of the work on the PLM tools factory.

Special thanks are also extended to Jim Hutton for the very fruitful discussions and the unreserved sharing of his hands-on experience on BT's product design processes and tools.

We also recognise the significant contributions members of the TMF's PLM team have made to the ideas we have presented, through the development of PLM as a discipline in the industry, the development of standards and by encouraging the exchange of in-depth knowledge between subject matter experts.

References

1 Ward A and Smith C, 'Service orientation: impact on BT's product portfolio', BT Technology Journal, vol.26, no.2, April 2009, pp.13-23

2 CIMdata Inc, 'PDM to PLM: growth of an industry', http://www.edstechnologies.com/download/pdm-plm-growth-of-industry.pdf

3 Elliot L, 'Does one size fit all?', Desktop Engineering, February 2006, http://www.deskeng.com/articles/aaabzm.htm

4 Bruce G, Naughton B, Trew D, Parsons M and Robson P, 'Streamlining the telco production line', BT Technology Journal, ibid

5 Glass WG, 'BT's matrix architecture', BT Technology Journal, vol.26, no.1, September 2008, pp.86-96

6 Glass WG, 'BT's matrix architecture', BT Technology Journal, vol.26, no.1, September 2008, pp.86-96

7 Batelle J,'The 70 percent solution', Business 2.0 Magazine, December 2005, http://money.cnn.com/magazines/business2/business2_archive/2005/12/01/8364616/index.htm

8 The TMF Product Lifecycle Management Team, 'Holistic product lifecycle management: a framework for PLM in the communications industry', TeleManagement Forum, TMF TR137, September 2007

9 The TMF Product Lifecycle Management Team, 'Team activities', TeleManagement Forum, http://tmforum.org/BestPracticesStandards/TeamActivities/5277/Home.html

10 The TMF Product Lifecycle Management Team, 'Lifecycle management process specification', TeleManagement Forum, TMF TR142, January 2009

11 Bruce G, Naughton B, Trew D, Parsons M and Robson P, 'Streamlining the telco production line', BT Technology Journal, vol.26, no.2, April 2009, pp.35-50

12 Silva N et al, 'Service oriented architectures for convergent service delivery platforms', EURESCOM Project P1652, Deliverable 2, December 2006, http://www.eurescom.de/~pub/deliverables/documents/P1600-series/P1652/D2/P1652-D2.pdf

13 Service Delivery Framework Programme, 'Service delivery framework overview', TeleManagement Forum, TMF TR139, September 2008

14 Muschamp P, 'Service innovation (in a software world)', BT Technology Journal, vol.26, no.1, September 2008, pp.97-111

15 Glass WG, 'BT's matrix architecture', BT Technology Journal, vol.26, no.1, September 2008, pp.86-96

16 Achilleos A, Georgalas N and Yang K, 'An open source domain-specific tools framework to support model driven development of OSS', Proceedings of the Third European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2007), Haifa, Israel, June 2007

17 The Eclipse Framework, http://www.eclipse.org

Nektarios Georgalas holds a Diploma in Electrical and Computer Engineering from the University of Patras, Greece, an MPhil in Computation from UMIST and a PhD in Computer Science from the University of London. He joined BT in 1998 and is now a principal researcher in the company's Centre for Information and Security Systems Research. During his career with BT, he has participated in and managed research projects in areas including active networks, market-driven data management systems, policy-based management, distributed information systems, service-oriented architectures and web services. His research is currently focused on product lifecycle management and rapid service assembly. Nektarios has led numerous international collaborations on the application of model-driven architecture and New Generation Operations Systems and Software (NGOSS) standards in telecoms operational support systems and has both led and contributed to the work of the TeleManagement Forum. He holds five patents, has authored more than 30 papers and has frequently been invited to speak at international conferences.

Achilleas Achilleos is currently a PhD student in the School of Computer Science and Electronic Engineering at the University of Essex. He received his MSc from the same department and a BSc from the Budapest University of Technology and Economics in Hungary. His research interests centre on model-driven development, pervasive service creation, context-modelling and mobile computing. Currently employed by BT as a research contractor, his research work is co-funded by BT and the UK Engineering and Physical Sciences Research Council (EPSRC).

Vangelis Freskos holds a Diploma in Electrical and Computer Engineering from the University of Patras, Greece. He specialises in VLSI design. In 2008, he worked as a research contractor at BT under the EU's ERASMUS programme, working on the construction of PLM tools based on model-driven development principles.

Daphne Economou holds a degree in Graphic Arts Technology from the School of Graphic and Arts Technological Educational Institute in Athens, an MA in Design for Interactive Media (Multimedia) from Middlesex University and a PhD in Computer Science from Manchester Metropolitan University. Since January 2004, she has been a lecturer in interactive multimedia and hypermedia at the University of the Aegean. Her research interests include collaborative virtual reality environments for learning and archaeology, human computer interaction and multimedia application design for mobile devices.