Domain, Process and Service Usage Models

Domain Maps and Models

For the purposes of current JISC Programmes, a domain is considered to be a coherent area of practice, as recognised by the practitioners within it and also by external stakeholders. It has its own goals or performs functions that are part of its identity. In carrying out these functions, it typically has its own expertise, practices and processes. These may be distributed across various domain roles. Practitioners in these roles employ them to provide products or services for others outside the domain and, in turn, they may be supplied with products and services by those working in other domains. There are usually information entities that are commonly used within the domain and a supporting ICT infrastructure. All these characteristics of a domain can be informally set out in Domain Maps or diagrams and more formally set out in separate Domain Models. Cross-links between these models create an integrated Domain Model.

Domain Maps/Models are important because:

a. When practitioners articulate and map out their areas of activity and their supporting systems, it enables them to reflect on them and then identify and prioritise opportunities and problem areas for further development.
b. These prioritised areas provide the starting points for developing Practice and Process Models (**P/PMs**) of what already exists and need to be improve, as well as for new ones identified for development.
c. They thus also provide the context for P/PMs which can then be placed at specific points within the Domain Model.
d. They bring together features that are common across P/PMs.
e. They tend to capture a domain's slower changing, more invariant functions and information entities that will in turn be captured in Domain Services. These can then be used by several or all P/PMs and applications, but each application in the domain must additionally capture the features that are unique to the opportunity or problem area that it addresses.

Setting out a domain can begin with a lightweight <u>informal</u> Domain Map, but this can be elaborated over time into a more <u>formal</u> Domain Model[1]. In the process, the Map should be maintained in line with the more detailed modelling as means of high level presentation and communication with those new to, or outside, the domain. In particular an important goal is for communicating with ICT developers and professionals that are supporting the domain. And a good way of elaborating a map into a model is through the development of applications that address high priority issues in the domain.

In the service oriented approach, the top level integration software should be relatively lightweight, relatively quick to construct and relatively easy to change. They can thus be modified in the light of improved practices, strategic change or changing external circumstances. The functionality to be supported by these applications is captured in P/PMs. Conversely, taking a bottom up approach, Roles, Information Entities, etc. that are found to occur in more than one P/PM can be moved to the higher level Domain Role and Information. Models and then simply referenced from other P/PMs. This is one way of detailing a Model.

The intent is to move the development of support more into the hands of domain experts. It promises to allow ICT support for processes to be mare easily developed and improved, rather than acting has a block to this. If it works, it would enable a much greater degree of process innovation, but, working through Development programmes and projects, we need to

explore and understand better where this can be most effectively applied. Many functions in the field of e-Learning, e-Research and are at early stages of integrating ICT support, and may not be sufficiently routine yet to formalise as processes, so it may initially have greater applicability in the area of e-Administration. However, if it lowers the cost of developing and changing ICT application support for processes, it should be possible to start making effective use of ICT earlier than it would otherwise. The big SOA promise being evaluated is that it enables an ICT infrastructure to become more flexible and better aligned with the needs of the organisation is it evolves and responds to its changing environment.

A Domain Map &/or Model can be elaborated as a set of interlinked sub-maps or models:

0. Boundaries of the Domain (before you start)

Practitioners define what is included and what is excluded in their domain. This may be tacit and, when made explicit, there may be disagreement over borderline cases, but otherwise there is agreement over what is included & excluded, which is often sufficient.

1. Other domains that are related to this one (Context)

Activities within a domain usually include exchanges with activities in other domains. They depend on others, and others depend on it, for products, services, information, knowledge, etc. This can be formalised as a Context Model which sets out the domain in focus in relation to these other domains, and indicates the exchanges and / or the provision of services between them.

2. Domain Stakeholders

Domain Stakeholders are here considered as external players, distinct from the roles within the domain which are considered separately. They are those directly or indirectly related to the domain and who can either make an impact on the domain or are impacted by it. This includes those who provide information, services or products used in the domain and those who use information, services or products produced in the domain.

3. Domain Roles (types of practitioner within the domain)

Different roles are usually recognised within a domain. These have their respective responsibilities, typical areas of expertise, and, where they can be articulated, their practices (See 7).

4. Domain Aims/ Purposes

Practitioners in a domain typically recognise common aims or purposes (cf. the Hippocratic Oath) which form part of the identity of the community. There may be a composition of purposes, where goals are divided into sub-goals.

5. Domain Top Level Functions (that fulfil the Aims/Purposes)

The purpose/s or aims/s of the domain are carried out by a set of ‘functions’. They may, at the top level, be mainly characterised by their outputs, i.e. <u>what</u> they produce or provide (product or service), rather than <u>how</u> these are achieved.

6. Domain Scenarios (ways of implementing the Functions)

Functions may be delivered by people, machines or, more often, some combination of both. A particular way of delivering a Function is here referred to as a Domain Scenario. A Scenario is typically one of a range of possibilities and maps out a high level, bird’s eye view of the Practices and/or Processes, the human and/or machine steps, which together form this particular way of carrying out the function. A Scenario is generally concrete rather than abstract. It can be presented in an informal, narrative and diagrammatic form. There are three types of Scenario in this context:

a. Current Scenarios that map current ways of implementing a Function ('**As Is**' Scenarios). These might take the form of case studies. If they are to form the basis for further improvement, they will be accompanied by some analysis of where the problems lie, their nature and causes.
b. Design Scenarios that map out future ways of implementing a Function ('**To Be**' Scenarios) because current scenarios all have limitations. Developing these is hard and involves imagining new ways of addressing the area as well as inventing the ICT solution[2]. If successfully implemented and tested, it could become a new Good Practice Scenario.
c. Good Practice Scenarios that are offered as summarising what is currently considered to work well by the community. This may emerge form informal sharing of experience and/or may be synthesised the best from a set of case studies.

7. Domain Practices & Process Models (are used in Scenarios)

Practices and Process Models supply the detail for a Scenario's informal narrative. Where a task is primarily performed by a human, it may be difficult to describe how this is to be done. This may be because the practices involved are not articulated by the practitioners themselves i.e. they are using tacit knowledge or skills. Good practices are those that have been articulated and are thus candidates for translating into formal processes. Practices typically precede processes and thus may used without formalised processes. Process enhancement may begin as newly evolving practices. Processes can be developed when there is a regular pattern of activities to handle a recurring requirement.

One proposed way of distinguishing them (Brown & Duguid)[3] is that Processes are the formalised and articulated ways of achieving the required outputs, but the process can never be described exhaustively and completely as there are too many anomalies and special cases which require a unique response based on an understanding of the problem, the task and thee broader context. Practices are thus the informal human ways of making the formal processes actually work, performing all the necessary things that are not articulated in the process. These practices and processes may be represented using a variety of modelling techniques.

One of the aims of the service oriented approach is to make it easier to move between the two, enabling good practices to be translated into processes and processes to be improved by trialling better ways of doing things and then incorporating them. To support this, tools and standards are emerging that enable processes to be model graphically (e.g. using OMG's BPMN standard) and then translated into executable process descriptions (e.g. using OASIS' BPEL or W3C's CDL).

The e-Framework Initiative is currently negotiating a license to use the tools provided by a large industry player and we hope that more will follow. Watch for further information on this.

8. Use Cases (An interaction with an IT System to achieve a goal)

In a Scenario, practitioners interact with ICT systems. Each such set of interactions with a system, *which delivers something of value in the Scenario,* is called a Use Case, i.e. it is the Use Case as defined in the UML standard. The collection of Uses Cases is referred to as a Use Case Model. The Actor in a Use Case may be a Practitioner/User or it may be one system acting as a client calling on another system. Use Cases, as defined in the UML standards, and their development and use, are described in many books.

Note the need to distinguish between this broader sense of a 'Domain Scenario' and 'Use Case Scenarios' which map out a particular, detailed pathway through a Use Case, typically dealing with optional actions and exception handling. Note also the need to distinguish between both of these senses of scenario and the very broad used in Scenario Planning which develops a range of possible future scenarios in the wider environment against which to test proposed organisational strategies.

9. Domain Information Models (are used in Practices and Processes)

Typically there are domain specific information entities that are used by people working in a domain. These differ from the knowledge about a domain, being captured in the domain model. The main focus of the Information Model is the entities about which information is commonly exchanged in the various applications and services that support the practices and processes. These entities can be defined and characterised and the relationships between them can also be modelled. At this level, it is important that the Domain Information Model defines the entities and their relationships in terms that practitioner understand. This information structure should characterise the more stable information entities and are captured in information oriented services. The processes they support are likely to be more variable, both across institutions and within them over time. Another way of looking at it is that the Information Model supports the domain's information standards which typically bind parts of it to some particular format (e.g. XML) and its purpose is to ensure interoperability across applications and services that are used within the domain.

10. Domain ICT System Models (that use the previous)

Domain ICT System Models (**ISMs**) characterise the ICT systems that are typically used to support the activities and processes in the domain. It characterises the functions of the ICT systems and also the information exchanges between them which in turn should to link to the Information Model. This model, along with practices and processes, is probably the most changeable because of the rapid changes taking place in the underlying ICT technologies.

Three Steps From Practice & Process Models to Service Usage Model/s

There are a number of steps involved in going from a Domain Map to one or more Service Usage Models. A 'Service Usage Model' (**SUM**) is used in the e-Framework to describe a set or composition of services, defined by their service interfaces, which work together to support an application which in turn supports a process that delivers a function. Where a set of services is reused in several applications to solve a recurring type of problem, the e-Framework refers to them as Service Patterns.

The Process, Use Case and the Information Models can, in principle, provide the basis for the specification of supporting applications. However for new systems it is hard to completely and confidently provide a complete specification in advance. Good practice in software development now takes an iterative and incremental approach to the development of systems, engaging users in each iteration to review working software to date, to review and if need be, re-prioritise remaining requirements and discuss proposed user interface and interaction designs. The emphasis on user engagement is increasing and studies have shown this to be a critical factor in successful software. Increasingly it is also being realised that software, and the new or revised practices and processes that it will support, need to be designed together.

The implication is that Design Scenarios (as above, but for 'to be' systems) must also be iteratively and incrementally co-developed with the supporting software. Therefore the 'steps' described here are not linear or sequential, even if they have to be set out that way.

1. Application Specification

All the above are provided by practitioners to help developers understand the domain they are developing software for. Process, Use Case and Information Models are all particularly helpful to developers. However, as just noted, they are aware that these cannot always be articulated for new systems in advance – the more innovative, the more difficult this becomes. User engagement in all phases of the development process is a key to successfully co-evolving practices/processes and software systems.

Also as noted above, new applications developed using a service oriented approach are integrated through relatively lightweight programmes that provide the UI and unique functionality, but call out to and coordinate the services that provide information management, security and other common functions, which no longer have to be redeveloped for every application. Such applications are sometimes called ‘composite applications’ as they compose, or ‘orchestrate’, a number of separate services. Identifying and factoring out services is an important step. Services may already exist, or be created either by providing service interfaces over existing monolithic applications or from scratch.

Portals that support portlets and so-called Rich Client Platforms for desktop applications are also making it easier and quicker to develop these lightweight applications by providing common UI functionality and a plug in format for modules. Co-ordination of (machine) services can either be done directly by a lightweight application, or an emerging alternative is to use Business Process coordination services such as are defined e.g. by BPEL (Business Process Execution Language) and others.

2. Service Usage Models

The set of services that are pulled together and coordinated by a service oriented application are referred to as a Service Usage Model. Calls on services can also be modelled with Use Cases, but here the ‘actor’ is always another system, rather than a person. The e-Framework site includes a Template for entering SUMs, but in general, preparation should include, or provide references to:

1. The Domain/ Function/ Process Model/ Application that it plays a role in (there could be more than one, especially of it is a low level SUM).
2. The task it performs as a whole, or, if a low level SUM, the use case/s it supports.
3. The use cases for each of the services included.
4. The particular services that are used, defined in terms of the (open) service interfaces they provide.
5. The structural relationship between them, if some services call on other services.
6. The process and information flows over time showing how they are ‘orchestrated’ (called on by a central orchestrator e.g. the lightweight application, a BPEL engine, etc.) and/or ‘choreographed’, where they call each other, together with the invocations from the service interfaces used.
7. The information model/s that they share when exchanging information.

3. Service Patterns and Service Pattern Languages

Where a Service Usage Model, or more likely a sub-set of a Service Usage Model, is reused to resolve the same recurring problem, it becomes a candidate for a Service Pattern. Hopefully, it will be possible to identify a number of such patterns and perhaps, in future, use them to build a Service Pattern Language.

However this is for exploration and discovery through the practical use of services.

---

[1] For more on Domain Modelling for (component) software development, see the early chapters in: Czarnecki, Krzysztof, 2000, *Generative Programming*, Addison Wesley.
(Compared to this book, in this briefing, the domain is somewhat higher level and the components are distributed as services, but many of the techniques and approaches of Domain Engineering apply.)

[2] Polikoff, Irene, Robert Coyne and Ralph Hodgson, 2005, *Capability Cases: A Solution Envisioning Approach* Addison Wesley

[3] Brown, John Seeley, and Paul Duguid, 2000, “Balancing Act: How to Capture Knowledge without Killing It”, in 2001, *HBR on Organizational Learning,* Harvard Business School Press.
(Also in May-June 2000, Harvard Business Review).