

Methodology for Design of IT Architectures

ARVE MEISINGSET



Arve Meisingset is Senior Research Scientist in Telenor Group Business Development and Research

All IT architects depict boxes with lines between them, but most architects depict different categories of things, and often different categories in the same graph. Hence, it is not enough to request an IT architecture; you have to know exactly what you want and why. This paper tells you what you need.

Domains

In the Introduction to Information Systems Architecture, *Teletronikk*, vol 94, No. 1, 1998, the following architectural domains are identified:

- *Systems planning*
Identification and delimitation of what systems should exist within an organization, and identification of relationships, eg. reference points, between the systems.
- *Interworking/interoperation planning*
Identification and standardisation of communication flow, contents and means.
- *Evolution planning*
Planning of transition from current systems and interfaces to planned systems and interfaces.
- *Systems framework*
This is the analysis and design framework (eg. Reference Model for Open Distributed Processing (RM-ODP)) provided to the designer, within which he undertakes his work.
- *System architecture*
Structuring of each system as seen by the system developer as the primary user of the architecture.
- *System development*
Survey, analysis, design, implementation, testing and installation of an individual system or component of a system.
- *Change management*
Management of change requests, analysis of change effects, management of changes and provision of the changed system.
- *Infrastructure*
The total environment in which a system is running and supported.
- *Platform*
The hardware, software and middleware on which the system is running.

- *Migration planning*

Planning of transformation from current infrastructure to new infrastructure.

- *Software portfolio planning*

Planning of software components to be provided to some market, and which can be variously configured in customer systems.

We realise that these words are defined from an IT architecture perspective in order to identify the subset covered by this document. Other architects, such as network architects, organisation or market developers, may use different terminologies.

In this paper, we will focus on Systems and Interworking planning of a telecom operating company.

Methodology

The methodology proposed in this paper is based on the Urd method (2) for systems planning, ie. the basic thinking is as in Urd. But the context of systems planning may often be short in terms of available calendar time and limited resources, which will not allow such a detailed study as proposed in the Urd method. Therefore, essential simplifications are made in this paper.

We propose an informal approach to design systems and interworking plans. However, we will use a detailed and formal background knowledge in designing the proposals. The informal approach will consist of the following steps:

- Informal definition of scope
- Systems planning
- Application planning
- Interworking planning
- Evaluation

Typically, the application domain provided in the Informal definition of scope should comprise *a set of organisation units* within an operating company. This will ensure that the planning becomes relevant for these organisation units. The reader should note that abstract frameworks like eTOM and TAM from the

TeleManagement Forum may neither fit with organisation boundaries nor with interfaces between systems.

Existing boundaries between organisation units may also be misleading, as

- The systems planning may lead to redesign of these boundaries;
- Any organisation unit may contain data and tasks that are not formalised, and the boundary between formalised and not formalised data is fluid.

As to the second point; a start up company may have little formalised data, while an incumbent has formalised more. Operating companies may distinguish themselves by different levels of formalisation, ie. they focus on different aspects.

A systems plan is an *idealistic design*. Some aspects of the plan may not be implemented until 10 years into the future; some aspects may never be implemented. But implementation of some aspects of the plan should start immediately. Therefore, we typically say that a systems plan has a time horizon of 3-5 years, but in reality implementation will often be slower in an incumbent company. In order to harvest benefits as soon as possible, we have introduced corrective design as a separate phase in the Urd method, but has left out this phase in the steps proposed above.

Systems Planning

Systems planning is about IT architecture of a telecom operating company. The operating company is understood as an *organisation of users* of IT systems. This perspective is different from the perspective of a software vendor who develops software components that can be variously configured and tailored to the user organisation's needs.

Both scope and contents of IT architectures may be defined using a set of different approaches. We may identify:

- Data
- Processes
- Functions
- Systems
- Interfaces
- Business organisation

Different architects use different approaches and produce different results. A bad architect does a little of each and then thinks that by doing so, he has done it all. A knowledgeable architect knows exactly what to do to achieve which result. He can focus on one issue to achieve a particular goal, he may discard other

work, but may as well do a little of each to accomplish the total solution. Knowledge of what works and foreseeing the consequences of the design work are essential.

We recommend that the IT architect of a telecom operating company identifies systems to appear in the final solution. So what is a system? A system is a set of data that is enforced as a consistent whole!

In terms of relational mathematics, the data of a system is a set of n -tuples, where $n \geq 2$. Each n -tuple states an association between two or more terms.

A system class is defined as a set of n -tuple classes, ie. as a set of record classes. The system classes make up a vertical partitioning of data.

A system instance is a set of n -tuple instances, ie. a set of record instances. The system instances make up a horizontal partitioning of data.

When identifying the data classes that will make up a system class, we will first have to design the data classes. ITU-T Recommendations Z.351-Z.352 contain some guidelines for good data design, and ITU-T Recommendations M.1401-M.1405 make concrete proposals for such classes for network management, service management and order management of a telecom operator. This kind of data class definitions make up the starting point for systems planning.

Within one system class we want to enforce the functional dependencies between data. Hence, we need to identify sets of data classes that are functionally dependent on each other. Such a set of data classes we call a data subject.

Often, no final design of the data classes exists when doing systems planning. Then we will need to make a best guess of data subjects and take this as the starting point for systems planning.

When end users want to carry out their tasks, they will prefer to access one system only to carry out one task. Hence, we group data subjects such that in most cases a user organisation can carry out all functions related to one task by accessing one system only.

The described goal of system design requires that we also consider redesign of the manual organisation. We move work to front offices, such that most tasks may be finalised there, and each front office needs only access one system to carry out the task.

The result of the system planning is a list of system classes with a sub-list of data subjects of each system class!

Note that two data subjects may share the same attribute class, eg. identifier attribute, without this being redundancy. Non-overlapping data subjects will not share the same relationships. And non-overlapping system classes will not share the same relationships either.

But if two system classes communicate with each other, one system will receive the data that are sent from the other. Hence, they will share common relationship data. So all communication between systems will result in overlap of relationship data. The same relationship data class will have to be handled in two systems. This does not mean that these data are physically stored in both systems. Storage of data is out of scope of the systems planning. We consider only data as perceived over the human-computer interfaces of the systems. Systems planning is defined for these human-computer interaction data, ie. for *what is perceived by the human user* within the organisation.

Application Planning

Application planning is about identifying horizontal partitions of data. Two regional units that do not share data, may not need to share any system instance. Hence, the system class may be split into two system instances.

At other occasions, we may have to accept horizontal partitioning even when this is not beneficial from a usage point of view. The reason may be use of different technologies for different parts of the network, with accompanying separate management systems.

We realise that some data of a corporation may be centralised, some may be regionalised. Sometimes, one system – eg. a service platform – may need to handle data from different corporations. Therefore, it is essential to design the data such that the owner of the data is clearly identified.

We observe that use of horizontal partitioning of data may be due to organisational boundaries, technological boundaries and capacity/performance boundaries.

Interworking Planning

The *purpose of interworking planning* for an organisation is to identify what communication should exist between systems classes. Design of details of the interworking will be postponed to the design of each individual system, that is not addressed in this paper.

Interworking planning is about communication between the system instances. The communication will be characterised and implemented at class level, ie. implemented in and defined between system classes.

Some communication will be between systems of different classes. Other communication may be between systems of the *same class*. Interworking planning will identify all these communication lines, characterise each, and identify the contents of each.

The objective of Systems planning and Application planning is to identify data that need tight interaction and to merge these within the scope of one system instance. We are trying to design system classes and system instances in such a way that as little interaction as possible is left for interworking between system instances.

When interworking between system instances is needed, we look for solutions that allow each system to operate autonomously without strong interaction with other system instances. This leaves the greatest independence and flexibility during operation and during evolution and replacement of systems. Note then that technology vendors – of middleware and service oriented architecture – may have the opposite motive, to link everything to everything. A real architect may try to make houses independent of each other, while an engineer may want to create tunnels and corridors everywhere. For the overall design you should listen to the real architect and not to the engineer.

Manual interaction between systems, eg. by copy-and-paste of data from one system to the other, is outside the scope of Interworking planning. As an example, such manual interaction may be needed when prognosis data are copied into a planning system, or planning data are copied into inventories, and when the source system may not have proper identifiers while the sink system does.

The designer should make the following priorities:

1. Try to avoid interworking between system instances
2. Try to defer interworking to manual interactions, eg. copy-and-paste
3. Investigate if links by references between systems can be used rather than flow of data
4. Investigate if batch exchange of data is sufficient
5. As a last resort, design real-time interactions between systems.

It is recommended to keep workflow as a function between data within existing systems, such as to include orders in an inventory, and not to implement workflow as a separate system. However, if workflow and brokering are separated, then this adds to the Systems and Application plan. Note that trouble ticketing may be considered to be such a separated workflow system and therefore we recommend to consider integrating this with work orders within the inventory systems.

It should be noted that vendors of middleware, enterprise integration architectures and workflow engines typically advocate using their tools as the solution to every integration problem. This paper recommends the opposite. We recommend to decouple systems and to avoid strong links between them.

The interworking between systems may be specified at different levels of detail:

- a. *Reference points.* A system class is defined to consist of data subjects. If one object instance appears in more than one system instance, then as a minimum, the object identifier needs to be exchanged between these system instances. Therefore, an object class may be depicted as an n -ary reference point between system classes. The reference points say nothing about communication sequences.
- b. *Busses.* If data needs to be exchanged between two or more systems, this may be depicted as branching communication lines between the systems. The bus design does not give the direction of the communication, but the accompanying text may indicate so. We observe that use of busses may indicate a rudimentary interworking design. The interworking planner may make a design decision on whether the communication is from all (relevant) systems to all (other relevant) systems, if some systems are used as intermediaries, or if some are used as masters.
- c. *Interfaces.* Sink and source systems of data are identified, and the message classes to be communicated may informally be identified. The message classes may be complex files, or they may be object based, ie. communicating one object instance at a time. More than one message class may be communicated over one interface: All communication over an interface should have the same direction.
- d. *Message classes.* The definition of message classes involves definition of permissible operations, control scopes for executions within the message and name bindings of data. This may involve a need to specify communication of human-computer interaction data, and a head-on definition of implemen-

tations in XML may be inappropriate. The definition of message classes is not explored in this document.

- e. *Implementations.* This involves definition of ports, contracts, services, encodings, middleware, etc.

In this methodology, we constrain the definition of interworking to identification of directed interfaces, ie. bullet c, followed by an explanation in natural language text. See an example of this in the paper on OSS Architecture (in this *Teletronikk* issue).

Communication between two system instances can be direct or indirect via other system instances. If we had a complete systems map *à la* the Urd method, we would already have reference points for these communications. These we will not have in all interworking planning projects, so we will have to invent the communication lines. Each line is a binary relationship, as we consider bus technology with multiple end points to be outside the scope of Interworking planning. We will focus on the definition of directed interfaces, which means that the relationships will be defined as ordered pairs from source to sink systems.

Ideally, we should have one line for each exchange of a message between two systems. However, this will lead to a too detailed design when trying to develop the overall architecture. And we do not have the needed knowledge at this juncture. We will therefore most often have only one communication line between two systems for all communication of the same characteristics. See this in the next section.

The implication of the restriction to have only one communication line between two systems is that there will be many message classes associated with this line. Some message classes over the same or different interfaces may be related, such as questions and answers, while others may be unrelated. In Interworking planning, we will not identify each message class and therefore not identify relationships or dialogues between them.

The scope of the application domain puts limits to the Interworking planning. If OSSes communicate with BSSes, we have to know each BSS in order to design communication between the OSSes and BSSes. But when designing the OSSes, the BSSes may be unknown. This leads to a two-step approach of designing communication lines:

- Design of communication lines within the application domain;

- Design of communication lines out of the application domain.

The communication lines will be dependent on the use of direct or indirect communication. Order management systems provide transaction handling of long transactions between systems. In this case, the communication is indirect from the source system, via the order management system or function to the sink system. Work flow engines and brokers have similar roles. When putting in middleware that transforms between different data structures, we have a similar situation. The solutions, their use and implementations can become much simplified if order management systems, workflow engines, brokers and separate middleware systems can be avoided. However, sometimes these systems will provide extra and much wanted functionality, eg. when communicating with alternative outsourced installation companies. Therefore, the Interworking designer should be very conscious about his choices. To use the most complex solution – eg. work flow engines – for every problem is not a good choice.

The interworking between two systems may be by

- Real-time communication;
- Batch communication;
- References.

In order to distinguish these means of communication between two systems, we may need several lines between two systems. Real-time communication is indicated by a solid line, batch communication by a dashed line, and references by a dotted line.

Evaluation

When having developed a draft Systems, Application and Interworking plan, it is good practice to review, evaluate and revise the result.

Summary

For each system class in the Systems plan, we will have a statement of intention with the system, and a list of data subjects covered by the system class. For each system instance in the Application plan, we may have a statement of purpose and identification of what populations are covered by the application. The Application planning may in case of replication lead to revision of what data should be covered by a system class. Hence, the Application planning may lead to a revision of the Systems plan.

For each communication line of the Interworking plan, we will have a statement of intention of this communication, and we will have an indication of what communication means is used. For each communication line we may have a list of data subjects that the communication is about. During Interworking planning we will not identify message classes, their contents or message interactions.

References

- 1 Meisingset, A. Introduction to Information Systems Architecture. *Teletronikk*, 94 (1), 3-11, 1998
- 2 Meisingset, A. The Urd Systems Planning Method. *Teletronikk*, 94 (1), 12-21, 1998.
- 3 ITU. *Data oriented human-machine interface specification technique – Scope, approach and reference model*. Geneva, International Telecommunication Union, 1993. (ITU-T Rec. Z.352)
- 4 ITU. *Designations for interconnections among operators' networks*. Geneva, International Telecommunication Union, 2004. (ITU-T rec. M.1400 series)

Arve Meisingset is Senior Research Scientist in Telenor Group Business Development and Research. He acts as a consultant on IT architecture in Telenor's operations abroad, and he is the Telenor ITU-T co-ordinator. He has held many positions in ITU-T and is now editor and associate rapporteur. He is also inventor and founder of a new company on virtual travels.

arve.meisingset@telenor.com