Designing the Enterprise
A hierarchy of Goals and Means
Trygve Reenskaug
August 22, 1999

Summary
In this position paper, I explore the relationship between enterprise needs and solutions at the upper levels and the possible information systems and solutions at the lower levels.

What I find is a goal-means hierarchy, where the specification of the computerized system can be considered as a fairly low level design decision. I try the following hierarchy:

1. *A recognized need for some change or improvement in the enterprise*
2. *Choice of solution environment/boundaries*
3. *Design of business procedures (distribution of responsibility, who does what and when).*
4. *Describe people's tasks.*
5. *Technology selection and system architecture*
6. *Detailed tool design. Determine use cases.*
7. *Specify missing information services*
8. *Design and implement computerized information services. (e.g. as Enterprise JavaBeans)*
9. *Design and implement user interfaces (tools) that support user operations.*

I like this approach. It offers formalisms if and where we want them, while we can keep other parts informal and even undocumented. It also lets us link information between the different levels wherever we choose to do so.

It would be fun to build a development environment based on these ideas.

1 There is a recognized need for some change or improvement in the enterprise.

This is a business need; something we want to change or improve or introduce because our business requires it.

Example:
We need to organize the handling of travel expense accounts, since we are not satisfied with the way it is currently done.

In a real case, we would say much more about what we wanted to achieve and why.

2 Choice of solution environment/boundaries

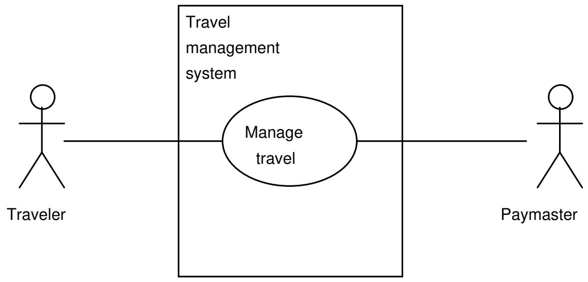
Definition:
A system is a part of the real world which we choose to regard as a whole, separated from the rest of the world during some period of consideration. A whole that we choose to consider as a collection of parts, each part being characterized by attributes and by actions which may involve itself and other parts.

For a given system, the environment is the set of all objects outside the system whose actions affect it or who are affected by the system, and also those objects outside the system whose attributes are changed by its actions.

So we describe the system seen as a black box, identifying the environment objects (actors) and the messages flowing to and from these objects.

Example:
We focus on travel management, and do not model details about why the journey was made, nor how the traveler is reimbursed for his expenses. Further, we do not want to change our current organization plan.

As a UML use case diagram:



Everything is still very open. We haven't decided if we want to be slipshod or very formalistic. Can a traveler demand that the casher pays him some money? Or does he have to submit receipts? Or an expense report on a special form? Should the form be authorized by some powerful person? Does the traveler need prior permission before traveling? Shall this permission be oral or formal?

And we have certainly not made any decision as to the application of a computerized system.

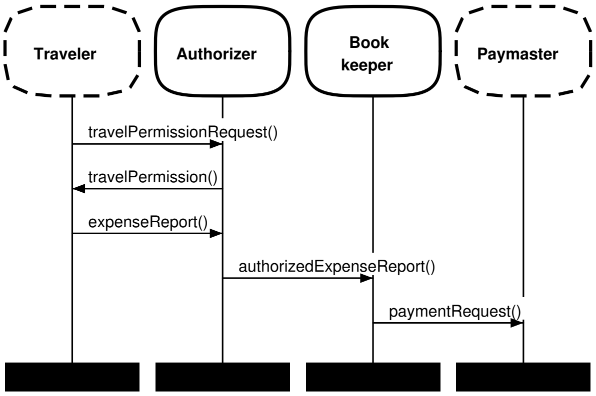
3 Design of business procedures (distribution of responsibility, who does what and when).

We now answer many of the questions that were left open on the previous level. We design a *business procedure* that implements a solution to our original need. But we are still not making any assumptions about computers or telecommunication!.

There is a significant difference between a business procedure and a computer procedure. A business procedure shows a baseline way of doing something, real instances are variants of this baseline. In contrast, a computer procedure embodies all possible executions, there cannot be any variants outside the procedure bounds.

We use role models (UML collaborations) to model business procedures. We have never found it worth while to model all possible variants. I suppose we would need some process modeling technique if we wanted to do so.

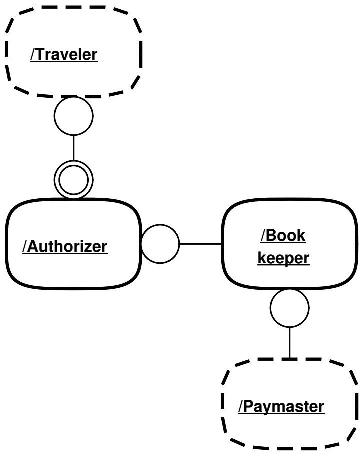
We decide we want a fairly formal procedure where an Authorizer shall authorize the trip beforehand and also the travel expense account after the trip. The following sequence diagram illustrates the proposed procedure:



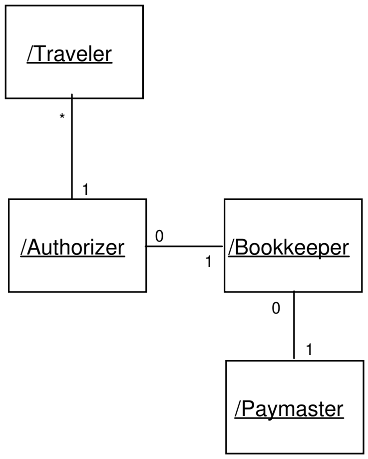
The collaboration diagram is shown below. An oval represents a (classifier) role. A small circle represents a *port,* that is a *reference* from the near role to the far role. A double circle indicates *many,* so an Authorizer knows any number of Travelers but only one BookKeeper. (UML calls it an *AssociationEndRole* and denotes it by an optional name and/or cardinality indication at the opposite end of the line from the role that holds the reference).

A role model (Collaboration) describes a particular Activity, which objects are involved in this Activity, and how these objects interact during this Activity.

A role (ClassifierRole) describes an object in a particular environment of other objects during a particular Activity.

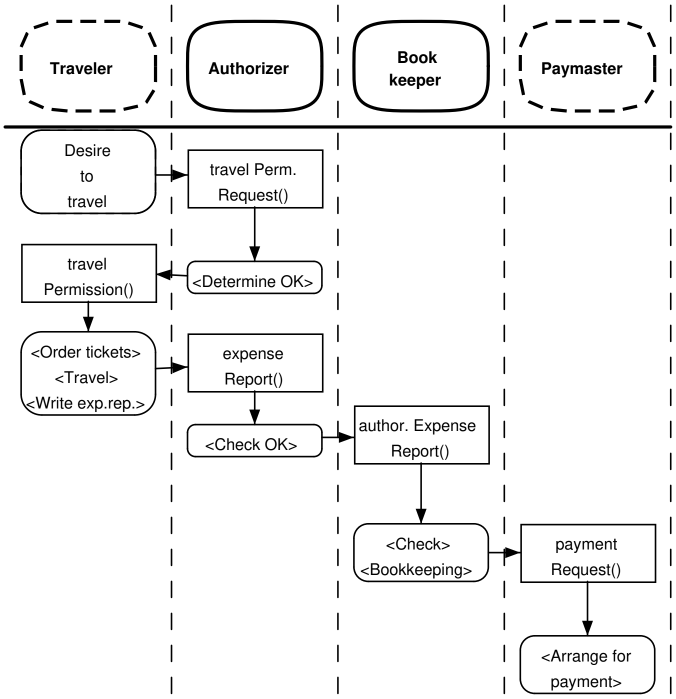


OOram role model notation



UML collaboration notation

Below is a kind of data flow diagram for the same procedure. There is one column ("swim lane") for each role. A rectangle denotes data transfer and a rounded rectangle denotes a task to be performed by the indicated role player.



Many decisions have now been made. We have decided who will be involved in managing a trip and their responsibilities. We have also determined typical triggers and data flow, and the tasks to be performed by each participant.

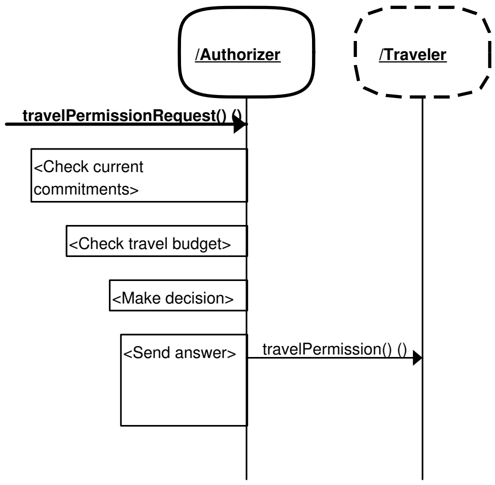
The model is quite concrete. We have neither considered classes nor (UML) class diagrams. We don't need them and we don't want them.

We still haven't considered the possible use of computers. This is a design decision that is postponed until level 5, where we choose technology.

4 Describe people's tasks.

We think in terms of decisions and deliverables without considering support technology.

For example, the Authorizer receiving a travelPermissionRequest could perform the following task (which is a method in the Authorizer object):



We still haven't decided on computerization. But we do have a fairly structured description of what we want to happen that helps us decide where to use computers and how.

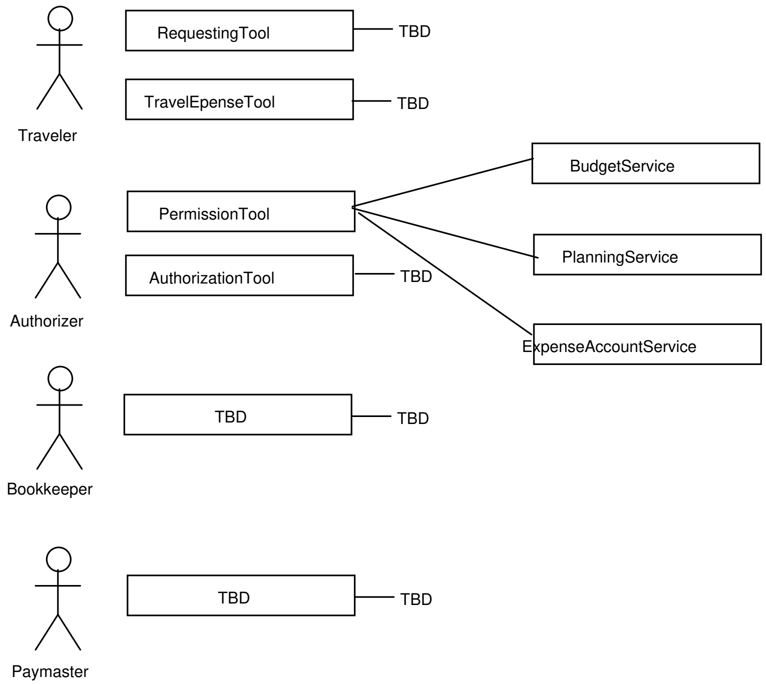
5 Technology selection and system architecture

We at long last get to the stage where we consider choice of technology. A wide spectrum of possibilities is open to us ranging from purely paper-based to purely computer-based operation.

We somewhat arbitrarily decide to support the complete procedure with a computer system. We also decide to furnish at least some of the participants with taylor-made tools to provide maximum support for their tasks.

A first consequence of this choice is that we need to store information about trips in an information storage service that must be available through the net. But looking through the task descriptions, we also find that our users need access to schedules and budgets.

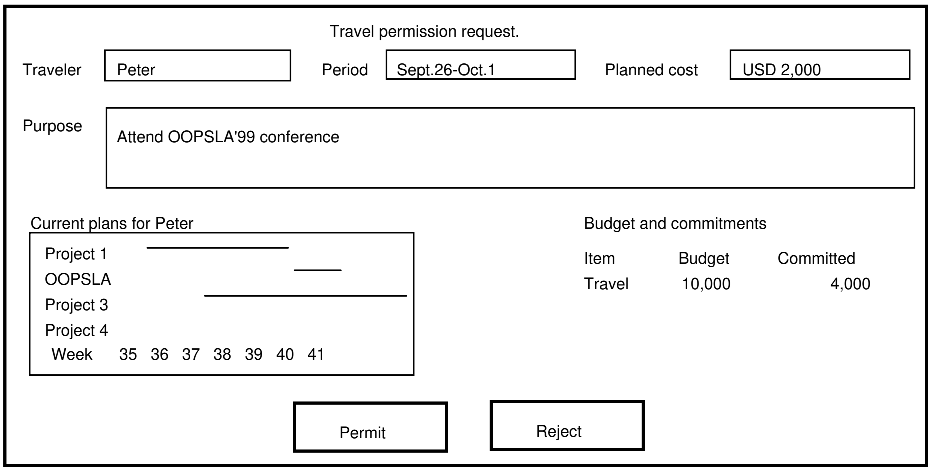
Let's assume we already have installed a Planning service component and a Budget service component. We check that they support the operations required by the tasks. Assuming this is OK, we end up with a system architecture. Some of it is illustrated here:



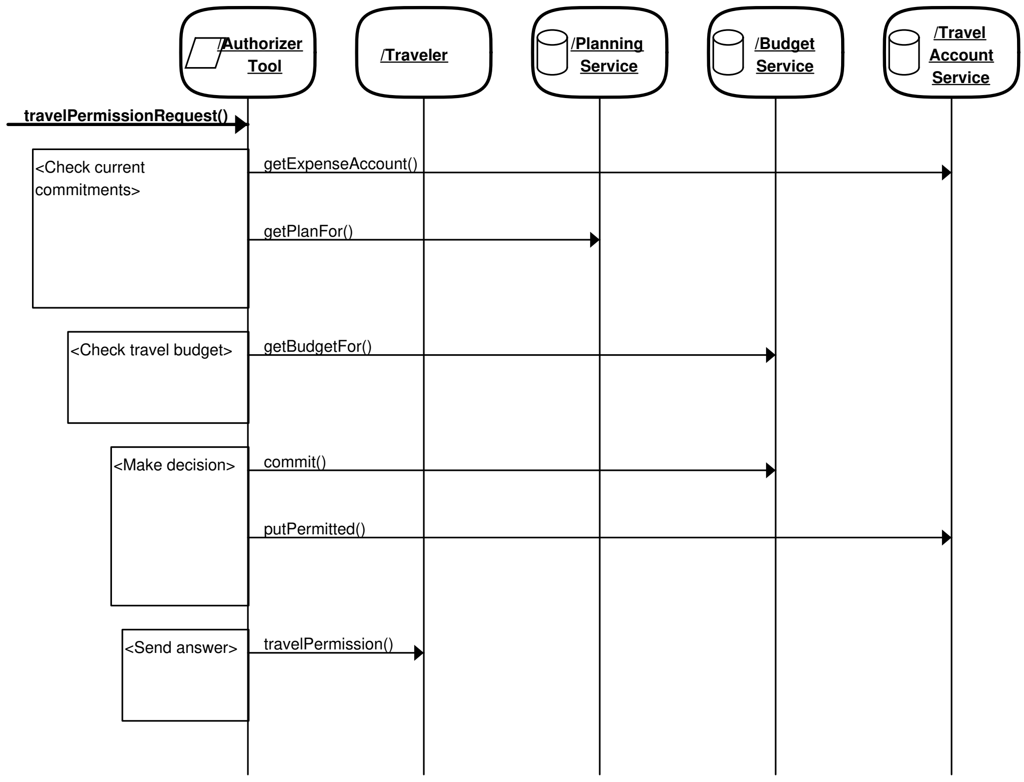
6 Detailed tool design. Determine use cases.

May be we can use standard tools for some of the operations. May be we can use the same tool for several operations. We clearly have to iterate between detailed tool design and architecture, trying to hit a good balance between cost and benefit. It will be a good idea to go trough a number of *studies,* each exploring one alternative.

The studies could profitably include tool prototyping. As an example, here is a possible tool for the authorizer considering to give a travel permission:



And this is an elaborated method diagram based on this tool:



7 Specify missing information services

We use the baseline task information requirements to specify the travel expense information model. We also consider information that will be needed in variations on the baseline tasks. Traditional database specification interviews with key participants further help us in this work.

We similarly consider the use cases that result from the task operations; again remembering that we have only described baseline cases and that variants may lead to additional use cases. These use cases specify what the users can do with the tools. This has to be filtered through the tools to get the service operations, and we get a first shot at the service interfaces.

We have probably decided long ago on a common technology for information services, e.g. Enterprise JavaBeans. This decision, which is outside the scope of an individual project like our example, will to some extent color our thoughts.

8 Design and implement computerized information services. (e.g. as Enterprise JavaBeans)

Use any applicable method.

9 Design and implement user interfaces (tools) that support user operations.

Use any applicable method.