Pitfalls using UML in RUP (1)

Hans Admiraal

Summary

Did you ever follow the rules of the Rational Unified Process (RUP) in using UML? "Well, I tried!" is a commonly heard answer to that question. This is the first of two papers written for those that tried or are about to. I've tried it myself a number of times and yes, I failed. Following RUP blindly simply doesn't work, but fortunately, you don't have to say good-bye to the process altogether. Let me help you by pointing out some of the pitfalls and first of all, by providing an overview of the various models in RUP.

This first paper focuses on *business modeling* and *requirements specification*. I will recommend to drop the Business Use Case Model, to use the Domain Model in your use cases and to detail the use cases using activity diagrams.

The second paper describes the use of UML for *analysis and design*.

Models in RUP

A model is a description of a certain aspect of a software system or, in business modeling, an organisation. It is most often a combination of text and graphics. Figure 1 shows an overview of the models specified by RUP. For each project, it has to be decided which of those models add sufficient value, although RUP recommends at least the use case model and the design model.

Figure 1 is not a picture you will find in the official RUP product. The vast network of web pages does not give a clear overview of the relationships between the various UML models. Oh yes, there is a lot of information, but it is scattered around in artifact descriptions, guidelines etc. This creates a lot of confusion during projects: which UML diagrams are we going to draw and how are they interrelated?

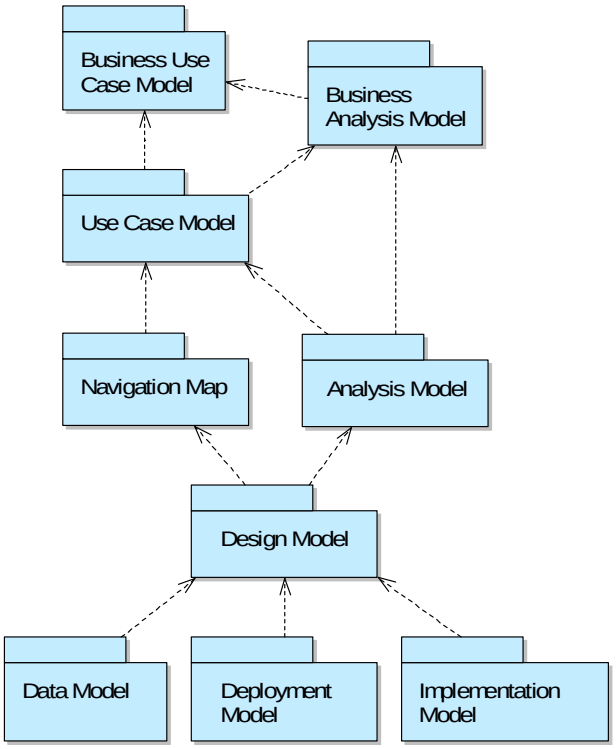

Figure 1. The RUP models and their dependencies.

RUP discerns three disciplines where UML is used:

- Business Modeling
- Requirements
- Analysis and Design

For each of these, I will summarize the view of RUP on UML modeling, the diagram types that can be used, their meaning and the relationships between elements in different models. Apart from that, I will give my personal opinion and suggestions about making practical decisions and dealing with the weak spots of RUP.

BUSINESS MODELING

Not all software development requires business modeling, but for administrative applications, you always need to know the business process to be supported. Large scale business modeling is sometimes performed outside the context of a particular IT project, so that RUP does not play a role. Too often, however, I am involved in software development projects where no proper business models exist. In those cases, if RUP is adopted, business modeling is also done RUP-wise.

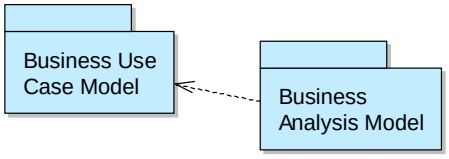

Figure 2. The models specified for business modeling.

RUP clearly separates two models:

- The Business Use Case Model, which describes the external interactions of the organization in terms of *business use cases*;
- The Business Analysis Model, which shows how the organization behaves internally to realize those interactions.

The dependency shown in Figure 2 reflects the fact that the Business Analysis Model contains references to the business use cases, because it provides an organization-internal realization of each business use case.

Business Use Case Model

The business use cases specify the interaction between the organization, regarded as a black box, and the outside world. The outside world consists of *business actors*, mostly customers and suppliers. A business use case is described from the actor's viewpoint. For business actor 'Customer', for instance, there might be a use case 'Order a product' and for 'Supplier' a use case 'Supply materials'. Table 1 shows the

UML diagram types that can be used to create this model. Apart from UML, business use cases and business actors are further described in regular text.

Package diagram	Large models are divided in packages. A package diagram provides an overview of these packages and their interrelationships.
Use case diagram	Shows the business use cases, the relationships among them and their relationships with the business actors.
Activity diagram	The flow of events during a business use case.

Table 1. Possible UML diagram types in the Business Use Case Model.

Business Analysis Model

The internal functioning of the organization, meant to realize the business use cases, is the subject of the Business Analysis Model. Here, the business processes are decomposed and the work flow is reveiled. You also model the organizational structure and the flow of information, as far as relevant.

Package diagram	An overview of the packages in the model.
Class diagram	The structure of the organization and the information. *Business workers* are active objects: employees and information systems. Passive objects like documents and products are called *business entities*.
Activity diagram	Work flow model, focusing on activities. Business workers are diagram partitions, containing actions. Business entities are the input and output of these actions.
Interaction diagram	Work flow model, focusing on message exchange between employees or teams.
State machine diagram	The life cycle of a single business entity (states and state transitions).

Table 2. Possible UML diagrams in the Business Analysis Model.

As an alternative, RUP presents the *domain model* as a kind of light-weight business model. The domain model only describes the business entities and their relationships and business rules.

Creating a domain model

Although RUP considers business modeling an optional activity, I would always at least make a domain model. In such a model, I define the concepts from the user's reality and draw them in a class diagram. A business glossary adds a clear definition of each business class.

Figure 3 shows a class model of an issue management domain, needed when you are designing an application for a help desk, for example. I will use this case throughout part 1 and part 2 of this paper.

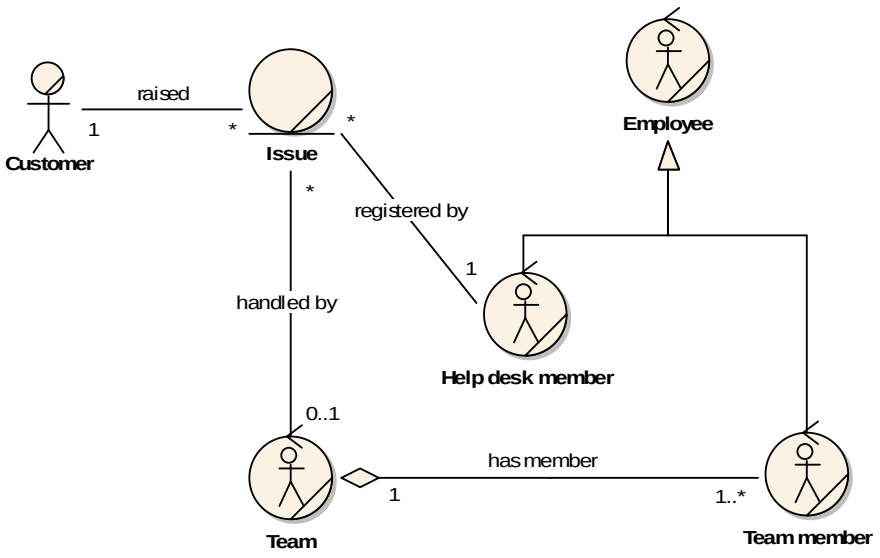

Figure 3. Class diagram showing business classes and their relationships.

The domain model is an important terminology framework for the specification of use cases by the requirements discipline. Don't skip it!

One final remark about this example. I didn't show any attributes or operations, but that does not mean they are not there. On the contrary, some domain concepts can best be modeled as attributes or operations, although business *entities* don't have any operations: they are passive.

More business analysis

If the application is supporting one or more business processes that are not well defined, you should do more business modeling. You could make a Business Use Case Model, but I suggest you just forget it. Instead, I replace it by a top-level activity diagram like Figure 4 in the Business Analysis Model. This diagram shows one UML action stereotyped as «business process», for each business process the application needs to support. By using the UML symbols *accept event action* and *send signal action* I show how the processes communicate with the world outside the organization. The 'Handle issue' symbol covers the complete process of handling a call from a customer, including anything that needs to be done to solve the issue.

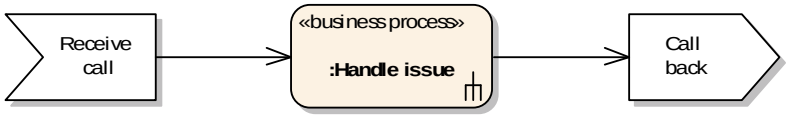

Figure 4. Top-level activity diagram.

Note the rake symbol in the bottom right corner. It indicates that the business process is decomposed in a lower level activity diagram, as you can see in Figure 5. There is a partition for each business worker that participates in the process.

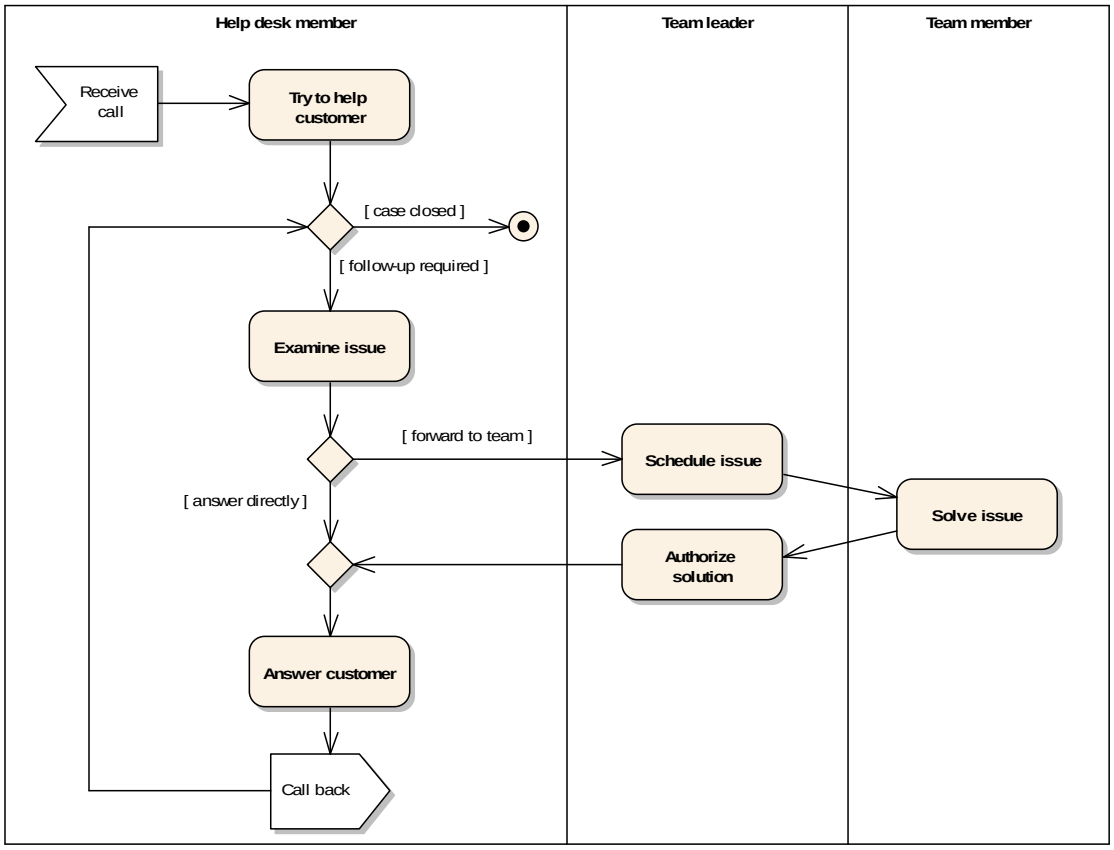

Figure 5. Second-level activity diagram.

Apart from the extra work required when you create business use cases, there may be business processes that cannot be expressed at all as business use cases, in case they are entirely internal to the organization: 'Check administration', 'Update web site'. Another pitfall is, that business use cases should be named from the business actor's viewpoint ('Let the organization handle my issue'), while business processes are usually named from the organization's viewpoint ('Handle the customer's issue'). This introduces a viewpoint shift when you compare the Business Use Case Model with the Business Analysis Model.

Now, what about the interaction diagrams? Well, I think that activity diagrams are sufficient and well-equipped to do all the process modeling. Interaction diagrams don't add much value and are usually less readable.

Creating state machines

Business entities that have a life cycle of going through particular states, get a state machine. When an issue is registered, for example, it first waits to be examined by a help desk member. If this person decides to forward it to a team, it enters the 'Forwarded' state, etc.

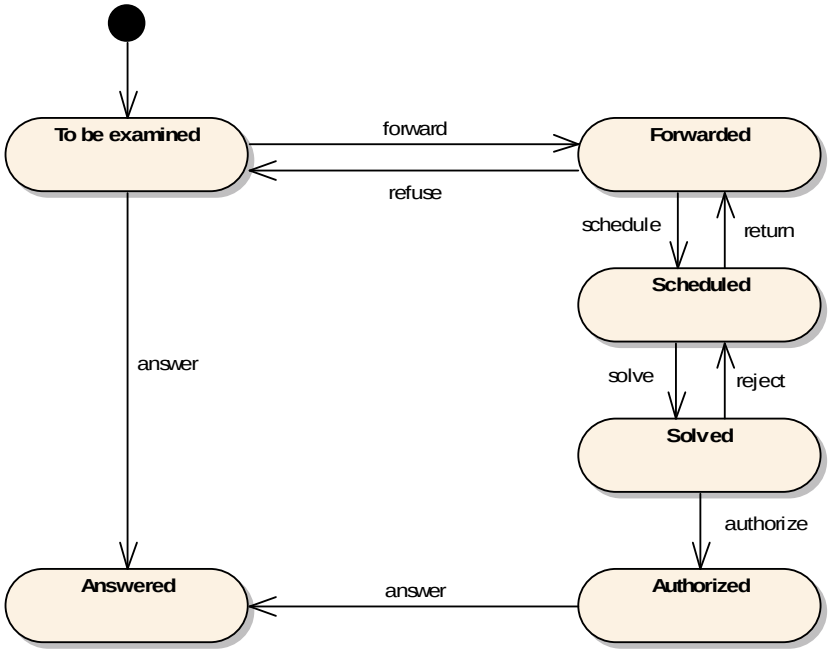

Figure 6. State machine diagram for business entity Issue.

A state machine is often another view on the work flow and therefore creates some redundancy in the Business Analysis Model. On the other hand, this second view brings a great opportunity to check and enhance the activity diagrams. Furthermore, the states can be very useful to refer to in the definition of business rules.

Business Modeling – Summary

Figure 7 shows the diagram types I use for business modeling, and their dependencies. They are not created in a particular order, instead, they evolve simultaneously.

- The class diagrams are always there and that's why they have got this blue color.
- The activity diagrams are used when the application supports a particular work flow. From these diagrams, you may refer to certain classes (business entities) in your class diagrams.
- State machine diagrams are used to model the life cycles of certain business entities. Consequently, they depend on the class diagrams. The transitions are triggered by actions of business workers, hence the dependency on the activity diagrams.
- Package diagrams are rare in business modeling, so I left them out from the figure. Large scale business modeling is usually not part of a RUP project.

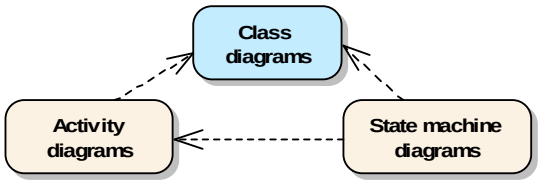

Figure 7. Typical set of diagram types for business modeling.

REQUIREMENTS

The man and women of the requirements discipline use the business models to work out the requirements for the application that should support that business. As far as UML is concerned, the "only" thing we have to do is to create a Use Case Model. This model depends on both business models, as shown in Figure 8. Although I ruthlessly unmasked the Business Use Case Model as a pitfall, earlier in this paper, the figure shows what RUP tells you.

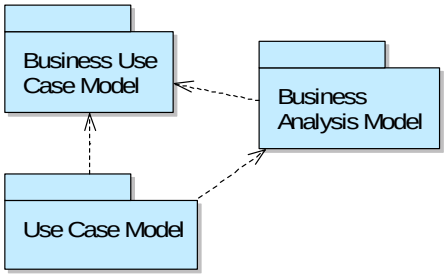

Figure 8. The requirements discipline adds the Use Case Model.

Use Case Model

This model is recommended for all RUP-projects. Basically, this model is shaped by iterating over three main activities:

1. First, you describe the *actors*: who actually work with the system, in terms of user roles.
2. Then, the *use cases* themselves are identified: what do the actors want to achieve by using the system? A use case diagram serves as an overview of these use cases (Figure 9).
3. Finally, for each use case, the interaction between the actor and the system is specified. This is a textual specification, which can be visualized in the form of an activity diagram (Figure 10).

Actors

The actors are often already identified in the Business Analysis Model as business workers. They are the ones that need the system to do their jobs. The actors in Figure 9 were the business workers I had casted in the domain model, remember? An actor could also correspond to a business actor, in case the business actor can access the system directly, through the internet for instance.

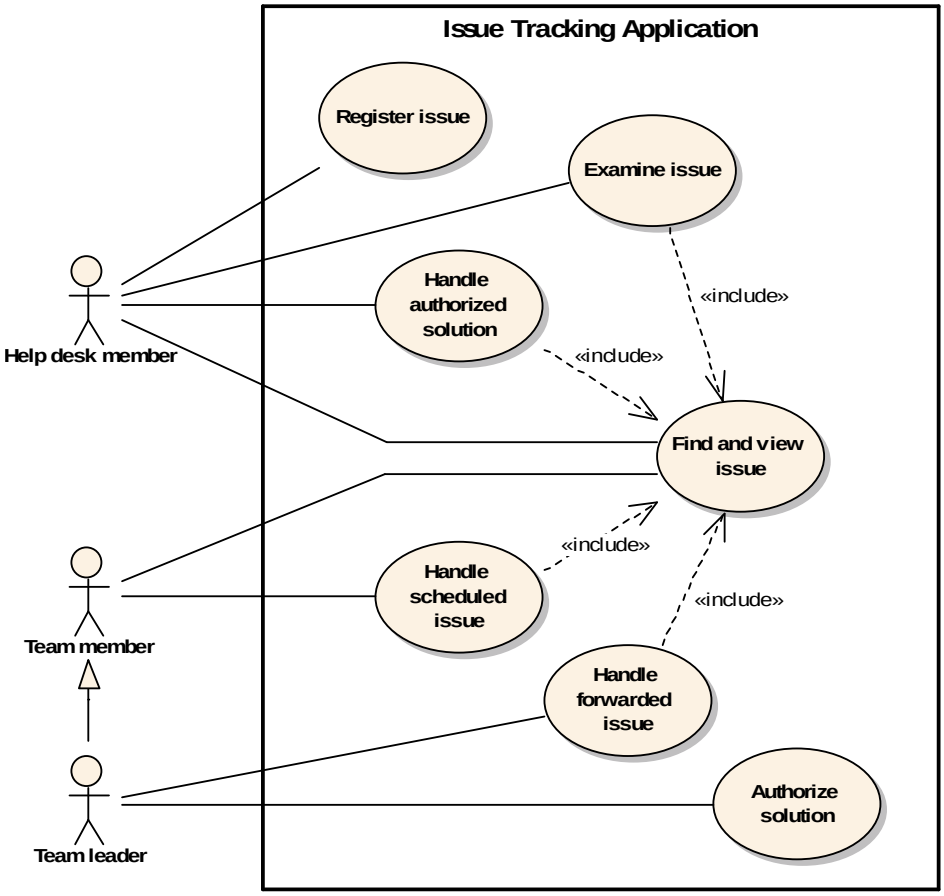

Figure 9. The use case diagram in the Use Case Model.

Use case identification

According to RUP, the use cases can be derived from business use cases, but this is only true if the initiating actor is derived from a business actor. In our example case, the actors are derived from the business workers, which is a more common situation, in which the use cases should be identified by looking at the actions defined for these

business workers. This does not mean that there is a one-to-one correpondence, as you will notice when you compare Figure 9 with the activity diagram in the Business Analysis Model. I recommed to document the relationships between the use cases and the work flow defined in the business model.

Use case specification

Most RUP practitioners write use case specifications only as structured text. That can work very well. It can also become a small disaster. How often did you renumber the steps in your flows? How often did your alternative flows look like a bunch of snakes, biting each other's tails? Those of you who start smiling at this point, may benefit from the option to use activity diagrams to lay down all alternative paths in a simple picture. Testers can greatly enjoy these pictures too, when doing a test paths analysis.

Figure 10 shows the internals of use case 'Examine issue'. The situation is, that a help desk member takes an issue from the pool of issues that are in the state 'To be examined' in order to either answer this issue immediately, or to forward it to the right team (second line help) for further analysis. I hope you can interpret the flow without further explanation, but in practice, when my diagrams have stabilized, I add a piece of text to my diagrams to help future readers. Even more important is the specification per action. Each action in the diagram should be specified either by text (maybe a one-liner, maybe a lot more), or by a separate diagram. I'm afraid I'll skip that for now. As you can see, the use case starts off with an included use case. The rake symbol in the bottom right corner indicates that there is a separate activity diagram for this action. Do you notice the references to my state machine in the business model?

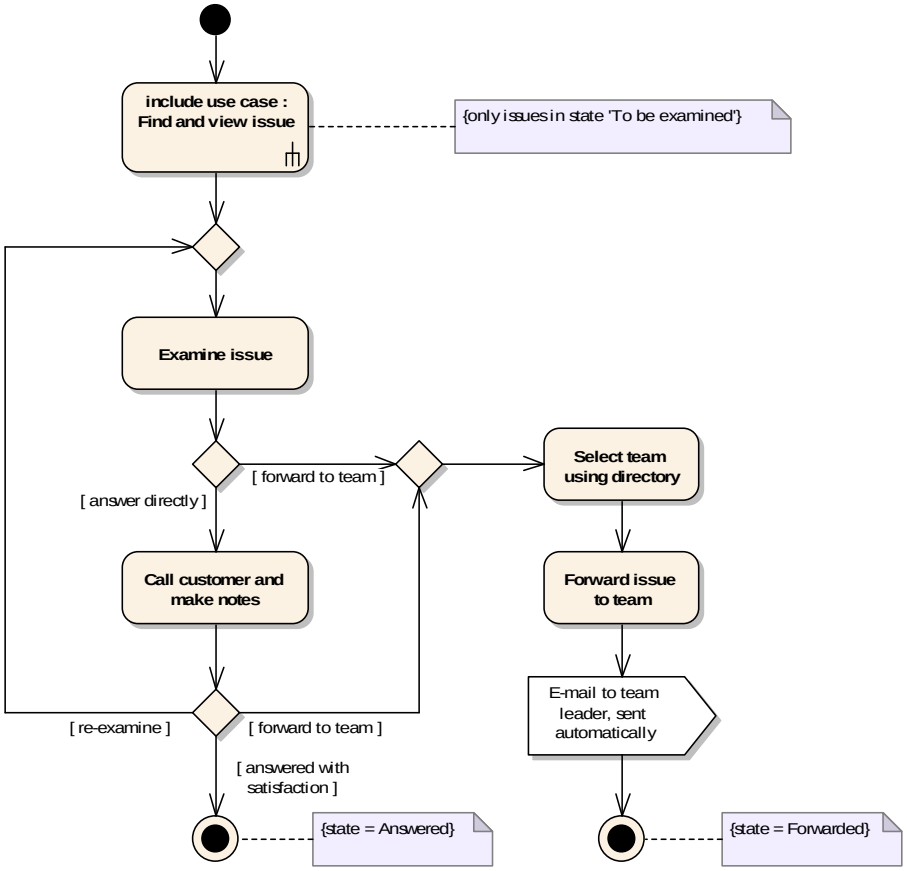

Figure 10. An activity diagram showing the flow during use case 'Examine issue'.

Requirements – Summary

Altogether, the following UML diagram types may be found in a Use Case Model.

Package diagram	Shows an overview of the packages, in case your system is large enough to require separate use case packages.
Use case diagram	Shows an overview of the use cases, their relationships and their relationships with the actors.
Activity diagram	A visualization of all possible flows of interaction between actor and system during the use case.

Table 3. Possible UML diagram types in the Use Case Model.

The complete picture of UML diagram types so far, looks like this. The class diagrams and the use case diagrams are mandatory in my opinion, even for small systems.

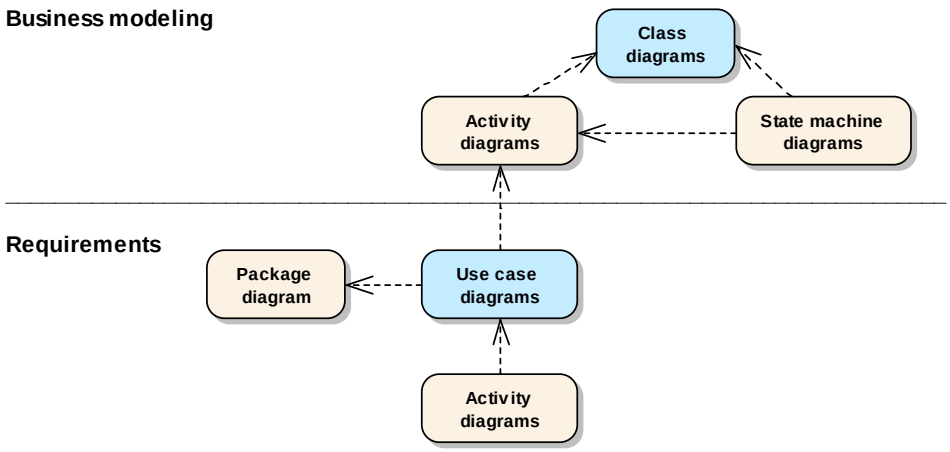

Figure 11. UML diagram types for the business modeling and requirements disciplines.

Remember, applying RUP is all about making the right selections and decisions based on the characteristics of an individual project. I don't have the silver bullet either, but I hope my writings will help you to organize the modeling efforts in your projects such, that a coherent set of diagrams emerges. If you have any feedback, please let me know!

Hans Admiraal,
IT architect at Ordina, an IT company based in The Netherlands.
hans.admiraal@ordina.nl

The diagrams shown in this paper are created using Enterprise Architect ®.