Slides for
Design Methods for Reactive Systems:
Yourdon, Statemate and the UML

Notes for Teachers

Roel Wieringa
Department of Computer Science
University of Twente,
the Netherlands
roelw@cs.utwente.nl
www.cs.utwente.nl/∼roelw

Organizing the course.

- The first lesson must set the expectations of the students. I do this by means of examples: "This is the kind of system we are dealing with in this course; you will learn notartions and guidelines for designing these systems".

- Spend time on breaking through the traditional partitioning of systems into information systems, embedded/control systems, telecom systems. Use examples listed in the book to make clear that the distinctions among these systems are getting blurred.

The role of practical work. The course would not be effective without work assignments for the students. This is a hard statement to swallow for some students. But in five years of teaching this material I have never seen someone pass a test who did not do practical assignments first. The reason is this: The solutions often look simple; the difficulty is in finding them. Learning consists in experiencing and surmounting this difficulty.

The role of notations is, in the end, the same as the role of handwriting: As a child you spend time in practicing writing. This takes a lot of time and effort. But after you learned it, you can forget about it: It has become part of yourself, and writing does not get in the way between you and what you want to write down. The same for notations: Once you master them, they will not get in the way between the you and the design that you want to describe.

Organizing practical work.

- I usually form groups of two students. each.
- Each group does an assignment that roughly is the size of one of the appendices.
- Each group gets a supervisor. Each supervisor gets no more than 10 groups.
- Each group sees a supervisor two times. Each meeting lasts 30 minutes and needs about 30 minutes preparation by the supervisor.
- Before the first meeting, they hand in their draft solution. This is discussed at the first meeting.
- Before their second meeting, the final solution is handed in. This is discussed at the second meeting.
- If you partition the assignment into sub-assignments, you need more meetings.

If you have large numbers of students, and not much supervision capacity:

- Again work with groups of two.
- Each group invents an example case and writes a half-page description of it. The group will act as customer with respect to this case.
- Cases are published and then groups negotiate so that pairs of groups arise. In each group pair, each group is the customer with respect to the other group.
- Now pairs of groups start working for each other. In addition to each group being a customer for the other group in a pair, each group helps the other doing the assignment, by explaining the notations and how they should be used. In this way, groups help each other.

- Each group thus plays two roles: As customer the group provides information about the case as the need arises; as supervisor of the other group, they help them doing their work.
- Define deadlines by which certain deliverables must be finished. Collect those deliverables at those deadlines.
- When the assignment is done, groups in a pair give each other a mark. The teacher evaluates the deliverables plus these marks and based on this, each group gets a mark.

Addition to any form of group work:

- To avoid free-riding, each student can write a short self-evaluation to explain the contribution he or she made to the deliverables. Comparing the self-evaluations of group members gives a lot of information.

The line of argument for these three examples is as follows.

- The reactive systems we are interested in, are software systems. These communicate with their environment by exchanging symbols (symbol *occurrences* really).
- Each symbolic exchange has the properties that any speech act has, listed below. To be technology-independent, imagine that the system interacts with its environment by exchanging post-it notes. What can we say about these exchanges?
  - Each note has a *topic,* the part of the world that it is about. This is the subject domain of the message.
  - Each note has a *destination* (system to external entity or external entity to system).
  - Each note travels through a *communication channel,* which we call a connection domain.
  - Each note has a *purpose,* e.g. to inform external entities, to direct external entities, or to manipulate lexical entities. This is called its function.

The ETS decomposition hierarchy is a difficult one. There are really two kinds of decomposition relations represented in example 3:

- Ownership. E.g. traveler's PDA is owned by traveler.
- Software encapsulation. E.g. Rail network database is encapsulated within the ETS (in this example).

These are decompositions in the social world and in the software world, respectively.

There is a third kind of decomposition, which is physical decomposition: The composite is bigger than the component, and in three-dimensional space, the component is inside the composite. This physical component-composite relationship is used as a metaphor to talk about decomposition in the social and software worlds. This is so pervasive in our language and in our perception of the social and software worlds, that it takes some effort to realize that in the social and software worlds, there is no physical "inside".

For example, "inside" the traveler, we find a heart, a stomach , organs etc. Even if in the far future the traveler's PDA would be inside the

traveler in this way, that is irrelevant for our analysis. The relevant relationship represented by the decomposition tree is that the traveler is *accountable* for the PDA and this is a legal relationship. The traveler is accountable for the PDA because he or she owns it. Similarly, the railway company, which is a legal construct, is accountable for the ETS and the rail network, the conductor, etc. This accountability follows from the fact that these entities are owned by the company or work for the company.

In the software world, decomposition is best viewed as encapsulation, i.e. the component is only accessible through the interface of the composite. One complicating factor is that the interface of the composite is itself realized by some components—these provide access to the other components. All components deliver some service to the composite. See section 19.2 for an analysis of encapsulation.

- The term "nonfunctional property" is a misnomer and I avoid it. For many people it means "vaguely specified property" or worse, "property that I do not have to describe precisely".

- All property must be operationalized, i.e. we must give descriptions of ways in which they can be measured.

- Many operationalized specifications take the form of stimulus-response descriptions but some, such as security specifications, are negative (e.g. certain things should not be possible in certain circumstances) or statistical (e.g. on the long run, most users will learn to use the system in 1 hour).

The diagram says that

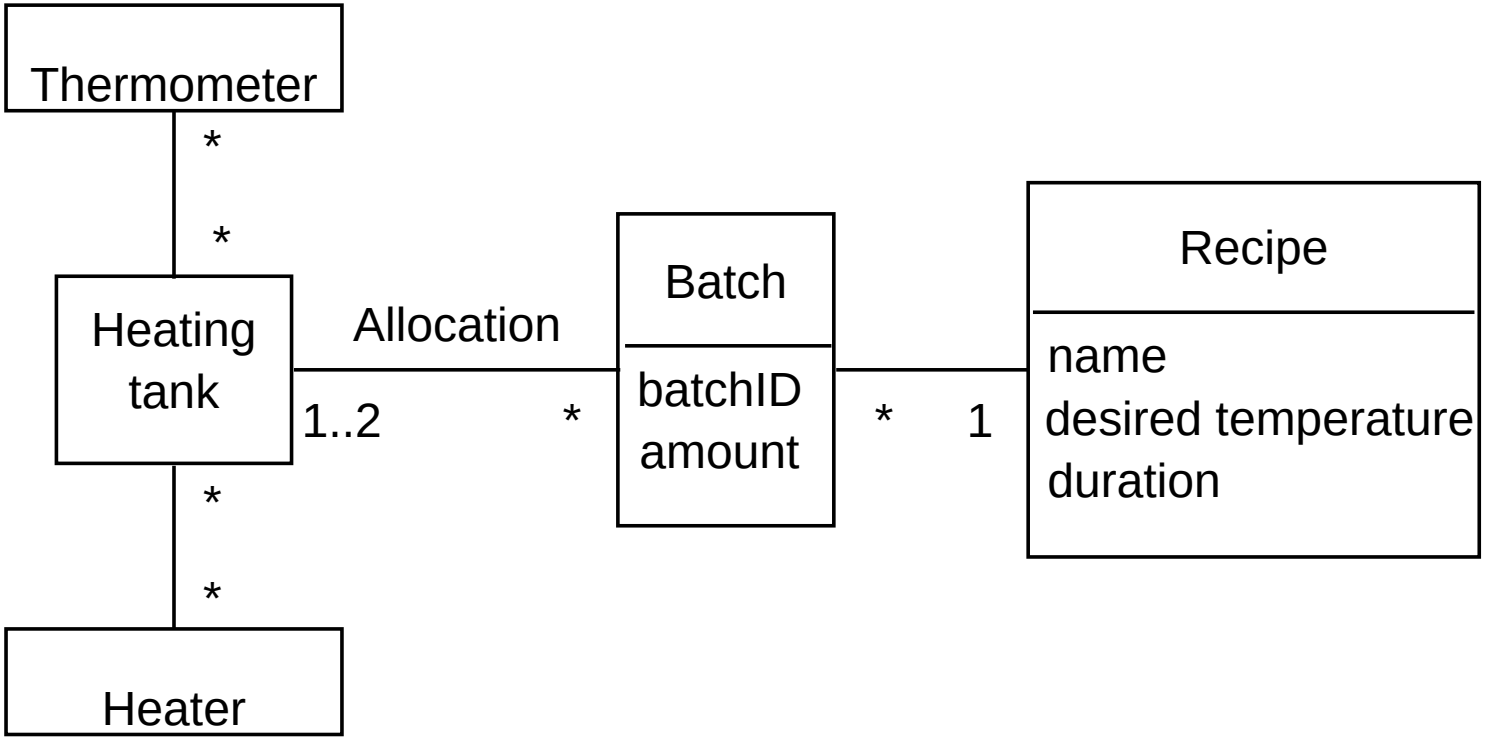
- Railway stations are discrete entities, i.e. each station has an identity different from any other station, and railway stations can be counted.
- It says that each railway station has a name.

The diagram says

- For each course offering there is exactly 1 teacher, 1 room and 1 course. (This follows from the fact that we have a ternary relationship.)
- For each existing (course, room) pair, there is at most one teacher. In other words, in each room there is at most one teacher for this course.
- For each existing (course, teacher) pair, there is at least one room. in other words, if a teacher teaches a course, then there is a room, and possibly more rooms, in which this teacher can teach the course.

The answer to both questions is Yes.

Here is the resulting diagram:



NB Gadgets are kinds of gadgets, not individuals.

Not equivalent.

The second diagram falls in the connection trap of replacing a ternary relationship by three binary relationships, from which the original information cannot be reconstructed.

The second diagram only represents that a customer ordered a gadget, that a gadget is stored in certain warehouses, and that a delivery from a warehouse to a customer is made. But it cannot represent that a gadget is delivered from a warehouse to a customer.

Not equivalent.

The second diagram gives each delivery of a gadget to a customer from a warehouse a different identifier. The first diagram cannot distinguish between different deliveries of the same kind of gadget to the same customer from the same warehouse.

- Station, Trip segment, Route.
- No, the system communicates about routes and tickets, not about passengers. It cannot determine the identity of a passenger.

If Employee is a subtype of Person (upper diagram), a person is either an employee or it is not. If it is a role of persons (lower diagram), then one person can play any number of employee roles (including 0) at the same time.

- S2 polls every 60 seconds. Switching heater on or off may be up to 60 seconds too late.
- There is a margin of -5 to +5 degrees, making delays even larger.

This is acceptable if we assume certain domain properties concerning the rate of change of temperature in the fluid, and the effect of high or low temperatures on the pasteurization process.

The usual connection domain assumptions:

- Keyboard and screen are functioning,
- Registration personnel is not lying, and
- makes no typing mistakes.
- A connection with personnel information system exists.

Other assumptions about external entities:

- Personnel data is reasonably up-to-date.
- Personnel information system can deal with incorrectly spelled names.
- Registration desk personnel is working according to workflow.
- Closing time of registration does not occur while someone is registering.

Classification of design decisions

Design decisions for the requirements-level architecture are made in terms of

- Functions
- External communication
  - Events
  - Devices
  - Users
- External behavior
- Subject domain structure