Chapter 1

Reactive Systems

Answers for Chapter 1 (Reactive Systems)

2. The controller is treated as a nonterminating process that is always able to respond to the arrival of tea bags. It does not interact intensively with people, but it is highly interrupt driven. Its response to the arrival of a tea bag depends upon its current state, such as the current value of the maximum tea bag count. Its response is highly environment-oriented, because it tries to maintain a desirable state of affairs in the tea bag boxing cell. There is some parallelism, because it must be able to respond to input from sensors whatever other processes are currently executing. However, the number of simultaneous inputs is small because the cell does not contain so many entities that can generate simultaneous inputs for the controller. The controller has strict real-time requirements.

3. The bug fixing database is a nonterminating process that is supposed to be always switched on. It interacts with programmers and managers and is driven by the arrival of queries and updates from those entities. However, these are not treated as interrupts. Its response to queries and updates depends upon its current state. The response consists of providing information to the environment and/or updating its own state. It must be able to respond to any query or update request, even even when it is currently working on other requests as well. There are no strict real-time requirements.

Chapter 2

Answers for Chapter 2 (The Environment)

2. Yes, but it will always be a *part* of the subject domain of the entire system.

3. Yes. The heater and thermometer of a heating tank are a case in point. See figure **??**.

4. Yes. For example, a financial information system may send and receive messages about salaries and business trips. The salary administration department will exchange information with the system about the first salary domain, and the travel office will exchange information about the travel domain. Employees are part of both domains.

5. The system can exchange messages *about* lexical items, such as text or graphs. These lexical items are therefore part of its subject domain. The system can store these messages as part of its state, and so these lexical items become part of the system.

6. (a) Tea bag boxing: Pure directive function.
   (b) Bug fixing support: Information provision. Informing programmers and the manager of a change of state can be viewed as a directive function, because it tells these people what to do next. A pure directive function that it can provide is refusing to accept an event in the life of a bug that deviates from the prescribed workflow. Whether or not this function is provided is not clear from the case description, that talks of "tracking". The system has a manipulative function because it stores bug descriptions, fixes and test reports.
   (c) Chemical tracking system: Information provision. Ordering chemical containers is a manipulative function because the orders are stored and manipulated by the system.

7. When functions are added or deleted, the set of interactions between the system and its environment shrinks or grows, and therefore the set of subjects referred to by these interactions can shrink or grow. For example, a system for reserving theater seats may currently have as functions (1) show list of performances, (2) select performance (3) show free seats for selected performance, (4) book one of the free seats. Its subject domain consists of performances, seats, prices of seats per performance, and people who reserved a seat. An extra function could be to show a map of the theater indicating the free seats for the selected performances. This would extend the subject domain with rooms and their physical layout.

8. (a) Elevator control system: It depends upon how ambitious you define its functions. Assuming that the purpose of the control system is to respond properly to events like a

passenger pushing a button and an elevator arriving at a floor, entities like buttons, elevator cages, and motors are part of the subject domain. The communication channel connecting the system with this subject domain consists of the wires that connect these entities to the controller. If, however, the purpose of the control system is viewed as responding to the wish by people to go to another floor, then the subject domain would be the wishes of the user; but that domain would be unobservable by the control system because there is no reliable communication channel linking these events to the system. This channel would contain the intended passenger, the buttons and lights of the elevator, and the wires connecting these to the controller. Only the part from the buttons etc. to the system is reliable, and so we take these buttons etc. as the subject domain of the controller and keep the purpose of the controller correspondingly feasible: Respond to button pushes and not to decisions of a would-be passenger.

(b) Cruise control system in a car. Again, this depends upon how ambitious you define its functions. If the mission is to control the speed of the car as requested by the user, then the subject domain would consist of the buttons used by the driver to indicate her wishes, the speed of the car and the state of the engine and gear of the car. However, the speed of the car —presumably with respect to the road— is not observable. Axle rotation, by contrast, is observable. Assuming that the car is on the road, this indicates car speed. And it is not the axle that is controlled by the cruise control, but the engine throttle. So we may want to scale down the mission of the cruise control to controlling the engine as indicated by the user. The subject domain now consists of the buttons used by the driver to indicate her wishes, the axle rotation and the state of the engine and gear of the car. The corresponding connection domain would consist of the wires connecting the driver buttons to the cruise control, wires conveying the state of the engine and gear box to the control, and a sensor measuring axle rotation plus the wires connecting it to the control.

(c) Personnel information system. The subject domain consists of the people working for the company, and the employment relationships they have with the company. If we take the PC's and workstations used by the personnel department to be part of the Personnel Information System, then the connection domain consists of the people interacting with these devices and talking with the employees. (Assuming that these people are themselves also employees, then the connection domain is part of the subject domain.) Alternatively, if these devices are not viewed as part of the Personnel Information System, then they are part of the connection domain.

(d) Word processing system. The subject domain consists of the text edited by the system including its markup and its presentation to the user. Observe that the text itself is an abstract, conceptual entity of which there can be many physical representations. For example, one and the same text can be represented on a piece of paper, on a screen, in computer memory, and on a disk. The user communicates with the word processor about the text by communicating about the representation of the text on the screen. So the subject domain of the word processor consists of the text (an abstract conceptual entity) and some of its physical representations (on the screen, in memory, on disk). These physical representation are accessible by the computer, because they exist in the word processing system. There is no connection domain between the word processor and the text as abstract conceptual entity, because it is not *possible* to connect to that abstract entity; and there is no distinct connection domain between the word processor and the physical representations of the text in the word processor, because it is not *necessary* to have such a connection.

(e) Your bank. The bank is an information processing organization implemented in people, machines, procedures and software. The interactions with you as customer have as

subject domain you and your bank account. The connection domain that connects the bank to you consists of the communication channels used by the bank to communicate with you about your bank account: the post office, telephone, fax, newspapers, TV, radio, the Internet. The bank account is an abstract entity of which the bank contains physical representations, say on paper, disk, and tape. There is no connection domain that connects the bank computer to the abstract bank account, because it is not possible to have such a connection; and there is no connection domain between the bank and the physical representations of this account, because it is not necessary to have such a connection. These physical representations are part of the bank.

(f) Account database of your bank. Same subject domain as your bank. The corresponding connection domain is the same as that of the bank, plus the bank personnel involved in bringing information from the database to you and back again.

(g) On-line bookstore. The subject domain consists of books available, users accessing the web page of the store, click stream of these users, transactions actually performed, web pages containing links to this bookstore, etc. It is connected to this subject domain primarily by means of the Internet but also by more traditional technology like telephone and fax.

(h) A supermarket manipulates goods and processes customers, and manipulates information about both. Its subject domain therefore consists of goods and their suppliers and transporters, the goods sold, and the people buying it. Information about these entities is obtained through many different channels, including communication channels with suppliers and transporters and familiar devices such as a customer card. The supermarket may also have a web presence that allows it to collect customer data through the web.

(i) The EDI system interacts with the supermarket itself and with dairy product suppliers. The subject of these interactions consists of dairy products and their suppliers and transporters; and so those entities make up its subject domain. The system receives information about these entities, and outputs messages about them, by means of employees in both companies handling these entities, and devices such as bar code scanners. These employees and devices communicate with the system through network technology and data entry and display devices.

(j) The subject domain of a web search engine consists of all pages on the web and their contents. The engine is connected to its subject domain through the Internet.

9. Yes, see the example of the account database of your bank, and of an on-line book store above.

10. (a) Subject domain of the DBS is the people who have insurance policies with the company, the policies, and the claims they ever made on this policy.

    (b) The subject domain of the WFMS consists of the claims currently being processed by the organization, the people handling those claims, and the relevant databases and applications needed to handle the claims.

    (c) The subject domain of the insurance company includes the previous two as well as the potential market for insurances, including potential customers and competitors.

11. A data dictionary is a system that represents information about a database (DB). Interactions with the data dictionary are all about the DB, and therefore the DB is the subject domain of the data dictionary. But we can store the data dictionary in DB itself, making the data dictionary part of its own subject domain. The dictionary typically contains parts that represent themselves (e.g. a table representing all tables in the database).

12. Software is a symbolic state of a physical computer. Each each point in time, the parts of the computer are in a physical state. They have a direction of magnetization or an electrical potential. We interpret some of these states symbolically as "0" and others as "1". By this *meaning convention,* these states have turned into lexical items. Additional meaning conventions then interpret patterns of zeroes and ones as more complex lexical items. A lexical item exists in a computer if we can interpret its physical state according to this set of meaning conventions.

Chapter 3

Answers for Chapter 3 (Stimulus-Response Behavior)

3. Temporal events are not observable. Clocks do not observe anything but they try to tick in synchrony with the passage of time.

4. (a) (i) Event: An up button on floor 1 has been pushed. (ii) Subject domain entity in whose life this occurs: An up button on floor 1. (iii) Observer: The wire connecting the button to the controller. (iv) Desired action: The motor in elevator in shaft 3 starts rolling down its cable. (v) Subject domain entity in whose life the action occurs: The motor. (vi) Actor: the wire connecting the controller to the motor. (vii) Assumptions: The button has been pushed by a user with the intention of entering the lift, the cable connects to the elevator cabin, the cabin can move through the shaft, the shaft is connected to floor 1 and there is room for one more passenger in the cabin. The controller does not verify this, because it cannot verify it. Only under these assumptions, the response is a desired response. In this example, we assume there is currently no elevator at floor 1.

   (b) (i) Event: The Delete button was pushed. (ii) Subject domain entity: The Delete button. (iii) Observer: The hardware that registers this event and the operating system software that informs the word processor of it. (iv) Desired action: The cursor moves left, erasing the character at the position that it moves to. (v) Subject domain entity in whose life the action occurs: The edited text. (vi) Actor: None. This part of the subject domain (a physical representation of the text) is manipulated as part of the word processor. (vii) Assumptions: The Delete button was pushed by a user who wanted to delete the character left of the cursor. The word processor does not verify this, and no one expects it to.

   (c) (i) Event: A request to update the bank account was issued. (ii) Subject domain entities: The owner of the bank account. (iii) Observer: There are many ways in which information about the event can reach the database. One possible way is the following: The owner writes the request on a paper form and mails this to the bank. Upon reception, the bank routes this to an employee who is authorized to update bank accounts. This employee reads the paper and enters the data in a terminal connected to the database. In this scenario, the observers (i.e. entities that play a role in communicating the subject domain event to the database) are the post office, the employee entering the data, the terminal and the connection between the terminal and the database. Banks communicate with their customer through multiple communication channels, and this is only one possible communication route. The bank includes the bank account owner in the subject domain because it will not accept an update request from someone else.

For security reasons, it will probably also monitor parts of the connection domain, such as the bank employee and the terminal used to enter the update request, which thereby are included in its subject domain. So the subject domain event issue an account update request is expected to cause other subject domain events, such as employee 007 enters an update request in terminal 999, and if these other events do not occur, the update will not take place. (iv) Desired action: 567 is added to the balance field of the bank account record of account 345 and a confirmation is sent to the port connected to terminal 999. Let's say that this is port 888. (v) Subject domain entity: This action occurs in the life of bank account 345. (vi) Actor: None for the bank account. The domain of bank accounts is virtual. The bank account is considered to be updated if the bank account *record* is updated. The actor for the confirmation is the line connecting the database to terminal 999, and the terminal itself. (vii) Assumptions: The update request was issued by a bank account owner who can taken to be accountable for his or her actions. We also assume that port 888 is connected to terminal 999.

(d) (i) Event: The number of milk packages becomes less than 100. (ii) Subject domain entities: The milk packages in the shop. (iii) Observer: A program that watches the number of milk packages in store, as recorded by the information system. (iv) Desired action: An order is placed with the milk package supplier for 300 packages of milk. (v) Subject domain entities: Milk package (a type of product of which there can be many instances in store), supplier. (vi) Actor: The network that connects the supermarket information system with the production information system of the supplier. (vii) Assumptions: The information system accurately represents the number of milk packages in store at each point in time. The truth of this assumption must be guaranteed by devices that update the information system as soon as the number of milk packages in store changes. This is done by connecting point of sale terminals to the information system and by placing a point of entry in the warehouse of the store, and implementing appropriate organizational procedures. The assumption cannot be fully guaranteed though, due to events such as theft and breakage in the shop not being reported to the system.

5. (a) A car cruise control. The car starts driving faster than the maximum allowable speed. This is currently not observable by the cruise control. What it *can* observe, using a sensor connected to the axle of the car, is the number of rotations per second of an axle. It can thus respond to the event that the number of rotations becomes higher than a certain pre-set maximum. Note that even the current speed of the car is not observable by the cruise control. The car may be driving on ice, causing its wheels to rotate very fast while it is moving very slowly.

(b) A library document circulation information system. What cannot be observed by the system is the event that a document is stolen from the library. What *can* be observed is the set of books currently in store. But this requires a major operation by library personnel, who have to take stock of the entire collection. A more feasibly observable event is the event that no one can find a certain document; upon which it is registered as lost or stolen. The observers in both cases are library personnel.

(c) An elevator control. Unobservable event: The decision by some user to take the elevator. What *can* be observed is the fact that a button contact has closed. The observer is the wire connecting the control to the button. But note that this may happen because someone pushed it, or because two people are fighting and they bumped against the button, or because someone shot a gun and hit the button, or because their is a short circuit in the button, or because there is static electricity in the air, etc.

(d) A workflow system. Unobservable event: A customer places an order. What *can* be observed is that a customer order is received. The observer is an employee, who

registers this information in some workstation, plus the workstation and network used to communicate this information to the workflow system.

(e) An email system. Unobservable event: Someone receives an important message. What *can* be observed is that someone receives a message in which some priority or urgency flag has been set. The observer is the email system itself.

6. (a) Patient monitoring system: Cause the nurse to attend to the patient whenever vital data indicate that the patient is not well. What the system *can* do is sound audible alarms and show information on a screen when certain events occur. This is done using the nurse's workstation and attached devices. But if the nurse is away, or asleep, or is attending to another patient, then the nurse will not attend to the patient.

   (b) A library document circulation information system: Cause users to return documents in time. What it *can* do is help the library send out reminders to return documents. This is done using a clock that counts the indicated time, and printing devices.

   (c) The information system of a supermarket: Cause stock to be replenished when it falls below a threshold. What it *can* do is print a report that stock is low; or it can cause an EDI system to reorder stock. This is done by an interface to the printing facilities or an interface to the EDI system. It is not under the control of the system whether these actions lead to the desired effect of replenishing stock.

   (d) A web shop: Cause people to buy their products. The only thing it can do is offer easy catalog access and show advertisements on the screen, using standard web servers.

   (e) An email system: Inform a receiver that an important email has been sent. The only thing it can do is set a priority flag or urgency flag in an email that is sent. It depends on the receiver's email system what is done with this.

7. It is true in the sense that an actor (observer) connects the system with a subject domain; it cannot be part of that subject domain. It is false in the sense that any actor (observer) can itself be part of another subject domain; for example, the system may observe the actor (observer) to see whether it functions correctly.

8. An external event is an event in the environment to which the system is expected to respond. Some event in the environment may however trigger an observer, which causes a stimulus, even if the system is not expected to respond to the event. (An event recognition process performed by the SuD may filter out this irrelevant stimulus.) Or the observer may be broken and cause a stimulus even if there is no event at all. In both cases, the stimulus is not caused by a relevant external event. And of course, a stimulus may be a tick caused by a clock, which is not caused by a temporal event.

Chapter 4

Answers for Chapter 4 (Software Specifications)

5.
- Functions: The function of the machine is to make coffee from ground coffee and water.
- Behavior: Enter coffee or water (in any order); switch on; take out pot or switch off (in any order).
- Communication events: Switch on, switch off, enter coffee, enter water, take out pot. Each of these events involve several entities and is therefore a communication event.

6. Representing the interests of the user, I would argue as follows: We only know that the assumption-requirements specification is not *applicable* in this environment. If the environment does not satisfy the assumption, then the manufacturer *cannot know* whether or not the system still must satisfy the requirement. In order to find out, he should have asked. Before selling you the system, he should have verified that you know what his assumptions are.

7. (i) The system shall register borrow, extend and return events. This can be operationalized by operationalizing the borrow, extend and return events. Is taking a document to a separate reading room a borrow event? Is it a return event if some else than the borrower returns the document? And so on.

   (ii) The system shall respond to any update request within 2 seconds. This can be operationalized by stating whether this is an average or a worst case value, stating under what work load the system should be during the period of measurement, and for how long the measurement must take place.

   (iii) The system shall be easy to learn. One possible operationalization is making a list of functions that should be easy to learn, characterizing the user group that should learn these functions, and characterizing how long they should take to learn these functions. There are other indicators, such as those given by the ISO (JTC1/SC7/WG6 1995*a*, JTC1/SC7/WG6 1995*b*).

   (iv) The system shall be interoperable. This is hardly operationalizable. With which systems should the system "interoperate"? What does it mean to "interoperate"? Perhaps we can interpose a broker between the systems, that does all the necessary translation. One possible operationalizable property is giving a list of named interface standards that the system should conform to.

   (v) The system shall only use well-understood concepts at its user-interface. Another one that can hardly be operationalized. It would be better to refer to a dictionary of terms

that is in use by the owner of the system, and requiring that the user interface use only words from that dictionary.

8. No, it is not the same. An operationalized requirement merely states the observable criteria that the system has to satisfy. A test, on the other hand, specifies a test set-up, including the environment conditions in which the test is to be run and a one or more sequences of particular input values and the output values expected for those inputs. An operational requirement describes operations to be performed but does not specify the other elements of a test.

9. No, R' does not give enough information to design the system. R' specifies the desired properties and these can be realized in many different way. The reason why we had to replace R by R' is that we have no way of ascertaining whether a designed system satisfied R, but we do have a way of ascertaining whether a system satisfies R'.

Chapter 5

Answers for Chapter 5 (Mission Statement)

1. (b) The goal tree of the business is mirrored by the tree of responsibilities of the product.

2. (a) The business goal tree is shown in figure 5.1.
   (b) The business activities to be supported by the Chemical Tracking System are
   - Find a chemical
   - Purchase a chemical that is not available in the company itself
   - Provide a report about chemical usage, storage or disposal.

   (c) The purpose of the Chemical Tracking System is to support the acquisition of chemical containers.
   (d) Major responsibilities are to maintain information about vendors and about the location of containers, provide information about the location and usage of containers, and purchase containers if requested.
   (e) One possible exclusion is that the system will not maintain information about the identity of the users of chemical containers.

3. - **Name.** Teabag Boxing Robot Arm Controller (TBRAC).
   - **Purpose.** To control a robot arm that boxes teabags.
   - **Composition.** The TBRAC consists of software in the operator's workstation and in the robot arm processor.
   - **Responsibilities.**
     - Setting the weight interval and maximum teabag count.

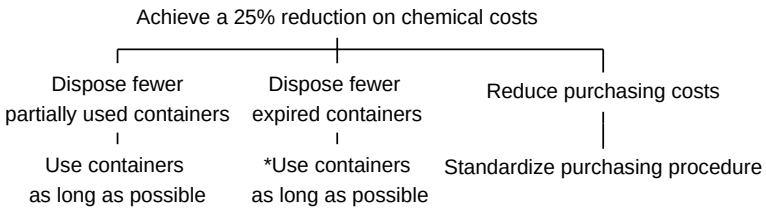


Figure 5.1: Business goals to be contributed to by the chemical tracking system.

  - Placing teabags in a box.
  - Replacing a full box with an empty box.
- **Exclusions.**
  - The controller will not deal with starting and stopping of the packaging cell.
  - In particular, safety management is not part of TBRAC's responsibility.
  - The controller will not deal with situations in which there are no empty boxes available or in which there is no room lefte to place full boxes.
  - The controller will not deal with situations in which the weight interval and teabag count entered by the operator are incorrect.

4.
- **Name.** Bugfix.
- **Purpose.** To support the workflow of bug fixing.
- **Composition.** A single program accessible through the company's network.
- **Responsibilities.**
  - Registering bug status.
  - Informing programmers of relevant bug status.
  - Maintaining a database of programmers allocated as fixer or tester to a bug.

5. (a) Mission statement of a sedan car:
   - **Name:** Whatever type of car you have in mind, e.g. Renault Clio.
   - **Purpose:** To provide mobility to people and express status.
   - **Responsibilities:**
     * Transport people
     * Carry luggage
     * Offer operating functions to the driver
     * Offer maintenance functions
     * Express status
   - **Exclusions:**
     * Mobility is restricted to road transport
     * Carrying heavy loads is excluded

   This mission statement assumes that we are talking about a type of car —something not stated in the exercise. Notice that naming the product forces us to think about the identity of the system.

   (b) A public coffee machine:
   - **Name:** Coffee machine type XYZ.
   - **Purpose:** Provide the coffee according to the taste of its users at a publicly accessible place.
   - **Responsibilities:**
     * Offer user functions, including selection of drinks, payment, and provision of drinks.
     * Offer Operator functions, including the replenishment of the coffee powder, collecting the payment and tuning the system.
     * Offer maintenance functions for diagnosing the system, replacing parts and tuning the system.

- **Exclusions:**
  - The system shall not offer any other drink but coffee.

(c) Mission of a word processing system:

- **Name:** My Word
- **Purpose:** Offer text processing capabilities
- **Responsibilities:**
  - Offer user functions, such as filing, editing, formatting, linking texts and displaying texts.
  - Offer operator functions, such as setting preferences.
  - Offer maintenance functions, such as diagnosis and installing patches.
- **Exclusions:**
  - No exclusions have been identified.

Chapter 6

Answers for Chapter 6 (Function Refinement Tree)

2. The list of functions is basically the list of controller responsibilities:

   - Allow operator to set required tea bag weight.
   - Allow operator to set maximum tea bag count.
   - Place tea bag in box or in waste container according to weight.
   - Replace full box by empty box.

3. This slightly refines the list of responsibilities:

   - Register assignment of bug to programmer and tester, and inform them by email.
   - Register bug fix and inform tester by email.
   - Register test report and inform programmer and bug manager by email.
   - Register bug release.

4. See figure 6.1.

5. (a) Figure 6.2 gives a function refinement tree of the sedan car. Notice that the parent of a node answers the question why the node is there. Why should the car collect passengers? To transport people. Etcetera.

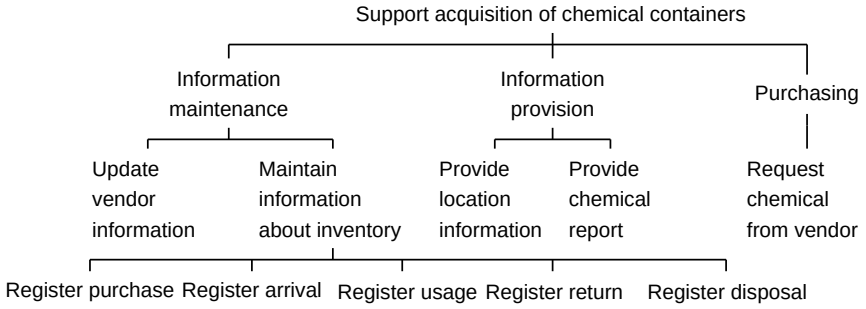


Figure 6.1: Function refinement tree of the chemical tracking system.

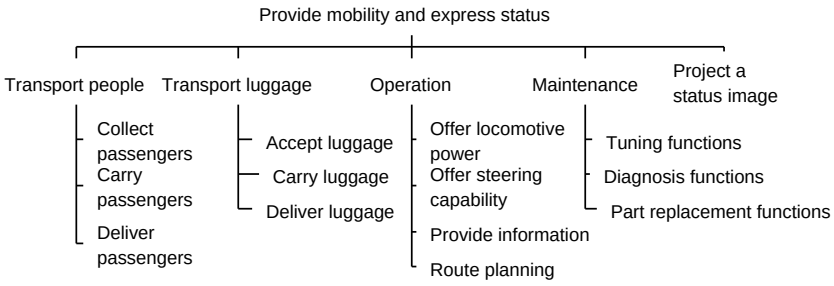


Figure 6.2: Major functions of a sedan car.

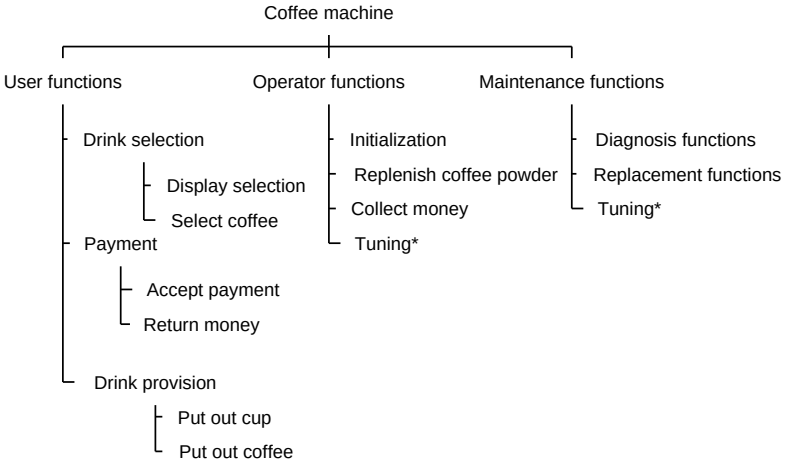


Figure 6.3: Major functions of a coffee machine.

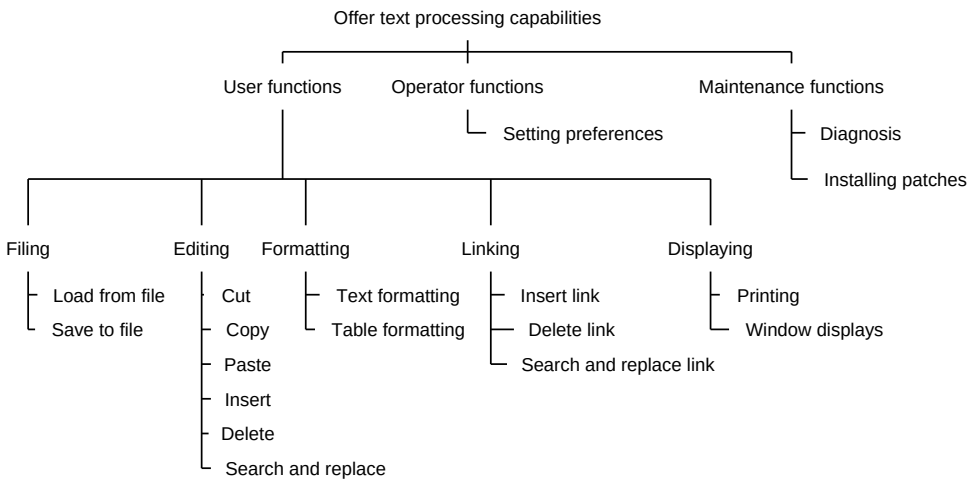


Figure 6.4: Major functions of a word processor.

(b) Figure 6.3 shows a function refinement tree of a coffee machine. Notice that we selected a means of payment: by entering money. Why not allow payment by debit card as well? And does money consist of coins or also of paper money? And which coffee is meant anyway? Decaffeinated? Expresso or not? Different strengths? And what about people who prefer to take their own cup? The tree raises many questions, which is exactly what it is intended to do: act as discussion framework with the customer.

The functions have been organized according to stakeholder. The tree illustrates that a function such as tuning may be relevant for several stakeholders.

(c) Figure 6.4 gives a function refinement tree of the word processor. Again, note that the reason for having a function in the tree is given by the parent node of the function. Note also that we made an assumption about the kind of text that is processed: It contains words, tables, and links. The function refinement tree thus causes us to make a first model of the subject domain.

6. Against: The decomposition dimension is independent from the external property dimension. The function refinement tree is merely a representation of the functional properties and belongs entirely to the external property dimension. See figure **??** on page **??**. In favor: You may choose to implement all function by system components. This gives you a functional decomposition of the system and as a result the function refinement tree also represents the decomposition structure.

Chapter 7

Answers for Chapter 7 (Service Descriptions)

3. The following description is slightly more precise and mentions the relevant assumptions:

   - **Name:** Remove teabag from balance.
   - **Triggering event:** Tea bag drops on balance.
   - **Delivered service:** The controller ensures that the teabag on the balance is placed in the box if the weight is OK and in the waste container if the weight is not OK. If the box becomes full, it is replaced by an empty one.
   - **Assumptions:**
     - There is a teabag on the balance.
     - The box and waste container can still accept teabags.
     - Teabag removal is fast enough for the robot arm to be able to remove the next teabag.

4.
   - **Name:** Register test report.
   - **Triggering event:** Tester enters his or her test report.
   - **Delivered service:** Bugfix stores the report and sends an email to the programmer and manager whether the test was or was not succesful.
   - **Assumptions:** The programmer and manager are reading their email regularly.

   The email system is mentioned because it is mentioned in the problem statement. If using the email system would have been a design choice made during the specification of the bug fixing support system, then it would have been inappropriate to mention it here.

5. All first elemements of the pairs are behavior-oriented, all second elements are value-oriented.

6. A value-oriented description tends to abstract from behavior, communication and data formats, and so lets you more freedom to design these once the value to be delivered is determined.

7. (a) Triggering event: Scientist requests chemical. Delivered service: The system offers the scientist a new or used container from stock or offers to order one from a vendor.
   (b) Interface information is dropped. (This can be added later when specifying a context diagram.)
   (c) The description does not state that a chemical can only be ordered from a vendor when it is not in stock.

Chapter 8

Answers for Chapter 8 (Entity-Relationship Diagrams)

5. See figure 8.1. Because Allocation is a relationship, two triples ⟨ t, r, c1 ⟩ and ⟨ t, r, c2 ⟩ will have different courses c1 and c2. These should therefore be given at different dates, as stated in the constraint.

   Modeling Allocation as an entity type is also possible, as has been done in figure 8.2. But this causes us to lose the information that for each (room, teacher, course offering) combination, there can exist at most one tuple, and so we must add this constraint in writing.

6. In the same order as they are listed in section **??**: Two teachers cannot use the same book for the same course. A teacher can use a book for any number of courses. A teacher uses at least one book per course.

7. Diagram (b) is wrong because it falls in the connection trap of replacing a ternary relationship by three binary relationships, from which the original information cannot be reconstructed. Diagram (b) 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. This is called the *connection trap*: Reducing the arity of relations too far, so that information is lost that cannot be regained by connecting the remaining relationships.

8. Figure 8.3 says that over time, a heating tank can have several heaters and thermometers but that each of these devices, after being attached to a heating tank, will not be attached to any other heating tank. In the course of time, a batch is at first not allocated at all, and can then be allocated to at most two heating tanks. In its life, a tank will be allocated many different batches. A batch will be treated according to exactly one recipe and this will not change over time. A recipe can be used for many batches.

9. - If heating tanks have always been grey and will alway be, then greyness is part of the intension of Heating tank. If they happen to be grey now but have had, or will have, other colors, then it is just an accidental property that is not part of the intension of Heating tank.
   - Each property P in intension(E2) is shared by all elements of extension(E2). Because each element of extension(E1) is also an element of extension(E2), all elements of extension(E1) have P. But then P is also in intension(E1). The reverse argument is similar.

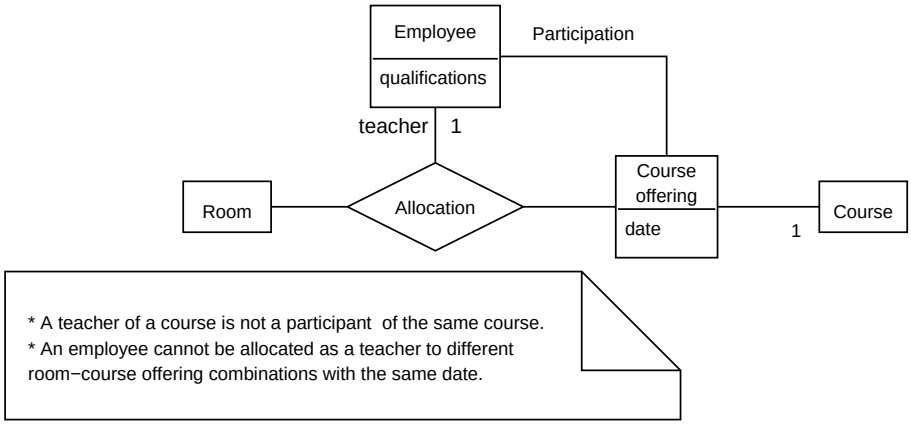


Figure 8.1: Catering for multiple rooms.

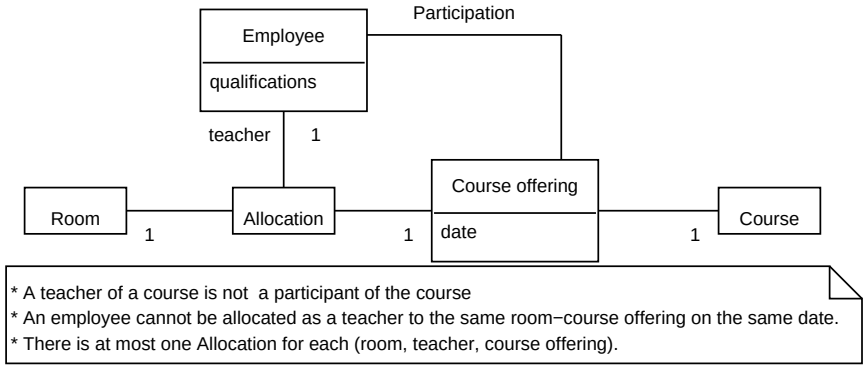


Figure 8.2: Trading a relationship for an entity type plus an extra constraint.

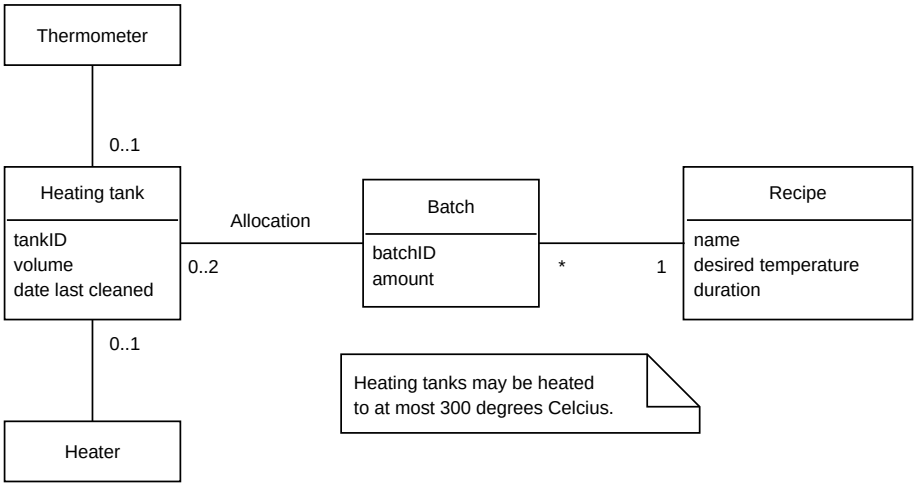
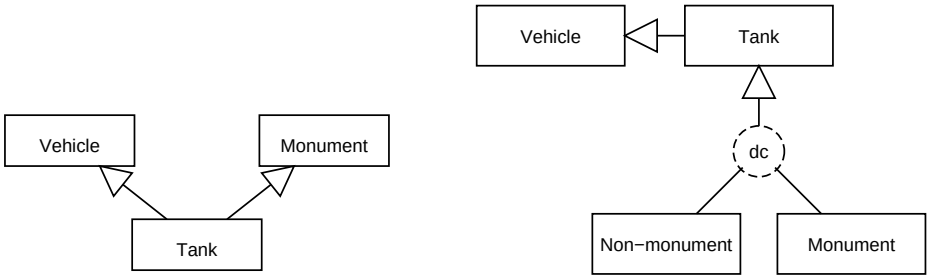


Figure 8.3: Historical cardinalities in the subject domain of the heating controller.



(a) (b)

Figure 8.4: Two incorrect models of tank, vehicles and monuments.

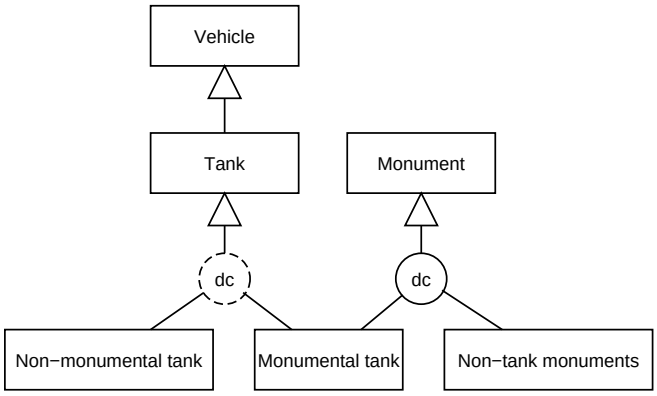


Figure 8.5: Another incorrect model.

10. Figure 8.4(a) is wrong, for it says that all tanks are monuments. Figure 8.4(b) allows vehicles to become monuments, which is close. But now all monuments are tanks, which is not what we want either.

    What we want is that tanks have two states, being-a-monument and not-being-a-monument. We also want to distinguish two kinds of monuments: Tanks in the state of being-a-monument, and other monuments. Something like figure 8.5? Now, when a tank becomes a monument, it does not change its identity but its state. The extension of Monumental tank is the extension of Tank. But figure 8.5 also says that the extension of Monumental tank is a subset of the extension of Monument. The implication is that all tanks are monuments! Again, this is not what we want.

    There is no model of the situation in terms of classification and specialization.

Chapter 9

Answers for Chapter 9 (ERD Modeling Guidelines)

3. The subtype extension is a subset of the supertype extension, so a subtype instance is at the same time also a supertype instance. They are identical and so have the same identification criterion. But the recognition criterion is different, because the subtype recognition criterion selects less entities than the supertype recognition criterion.

4. See figure 9.1. Note that the controller does not have to know the identity of teabags or boxes. If we would represent the required tea bag weight as an attribute of a metaclass TeaBagType, then all tea bags would have the same required weight, whereas the case description implies that the required weight is different for different kinds of tea bags. Note that figure 9.1 is *not* a database schema but merely helps to define the meaning of some subject domain terms. The controller is not required to maintain a database of tea bags or boxes.

5. See figure 9.2. The Assignment relationship must be ternary to avoid the connection trap. The cardinality property cannot be expressed by annotations in the diagram; but even if it could, it prevents misunderstanding to simply add the constraint as a comment.

6. See figure 9.3.

7. See figure 9.4. The pattern is that each time new information becomes available about a field, a new entity is created that contains this information and is uniquely related to the field. Sowing could not have been modeled as a relationship because we may sow the same seed for the same crop on the same field several times a year.

8. See figure 9.5. There are two type-instance relationships in the model: between Flight and Flight instance and between Hop and Hop instance. The relationship between hop and flight is one of aggregation: A flight consists of one or more hops. There is a corresponding

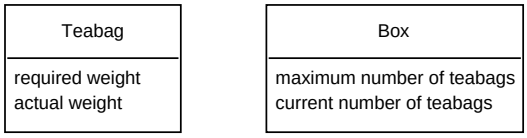


Figure 9.1: An ERD for the teabag boxing controller subject domain.

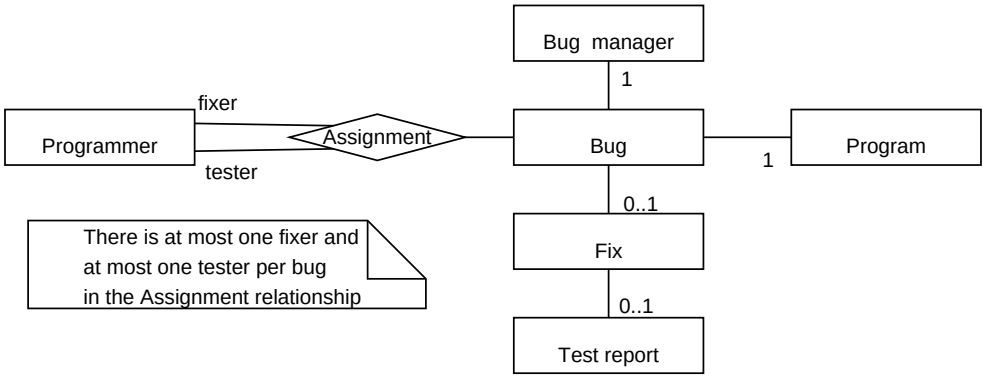


Figure 9.2: An ERD for the bug fixing support system.

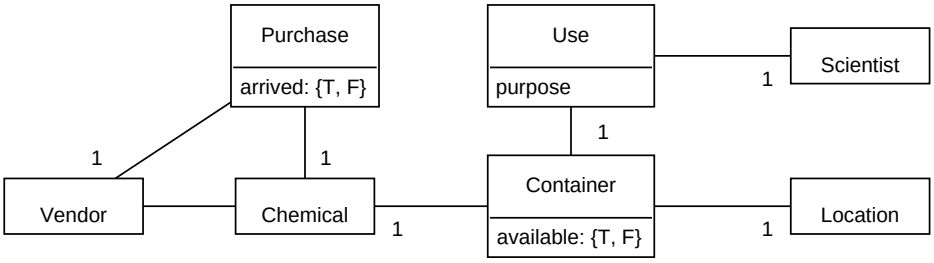


Figure 9.3: Subject domain ERD of the chemical tracking system.

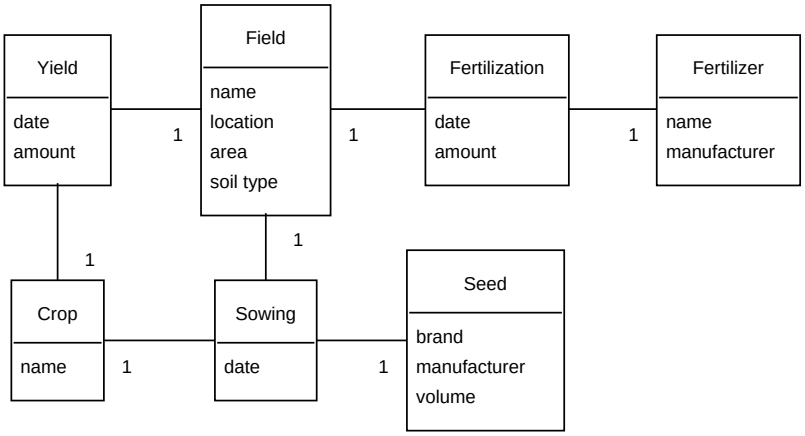


Figure 9.4: A model for agricultural fields.

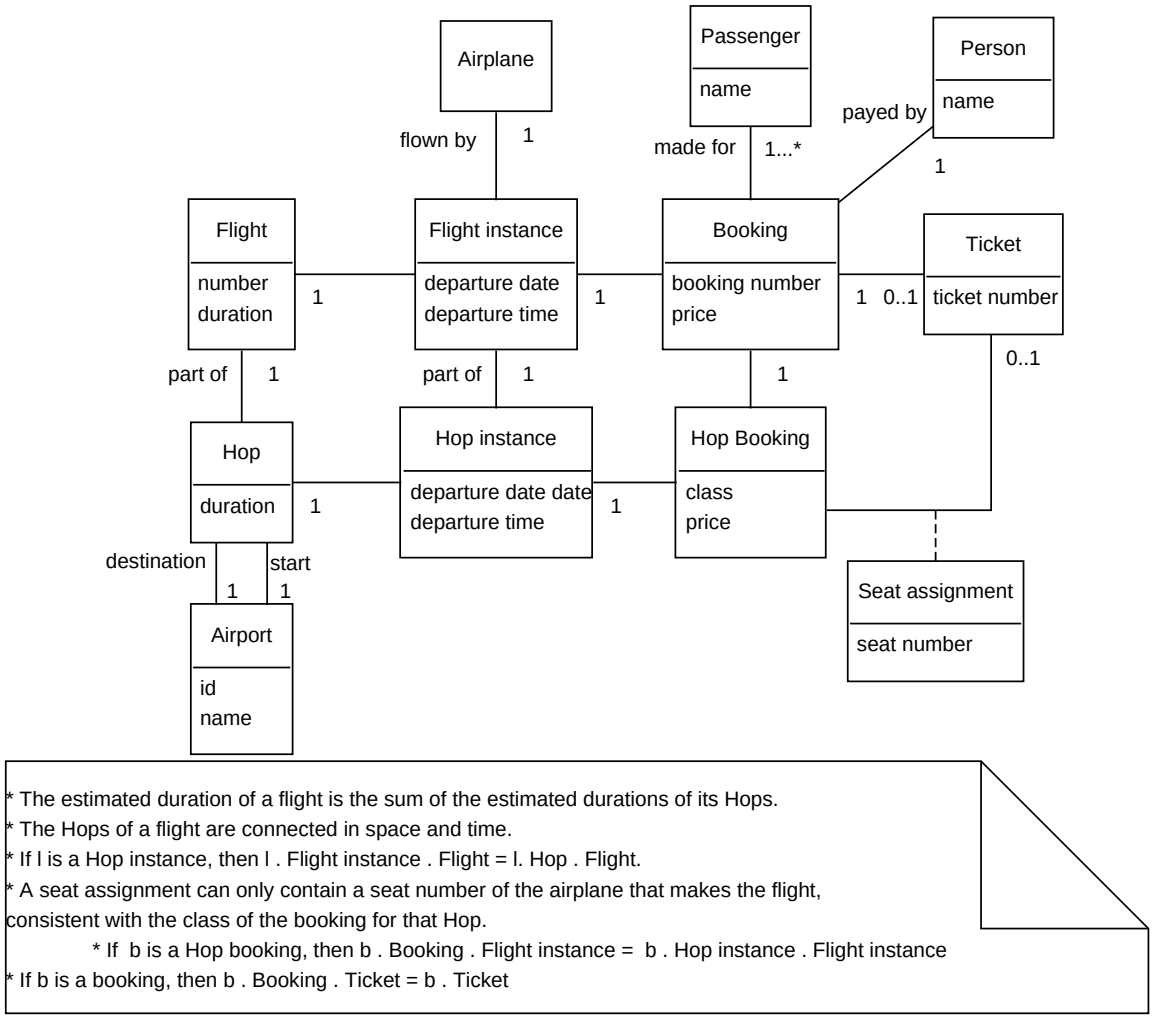


Figure 9.5: A model for booking flights.

aggregation relationship between flight instance and hop instances, and between booking and hop booking. A ticket is related to hop bookings by means of seat assignments.

9. The passenger is not observable. The conductor might be (by identifying himself to his PDA), but none of the system functions require information about the identity of the conductor.

10. (a) Railway station: In the subject domain, because the passenger must identify stations to the system. A station is observable by all railway employees (and travelers) and we consider the communication channel by which information about stations reaches the information system reliable, even though it consists of railway employees. (These are very reliable people.) Delays in transferring information through this channel are irrelevant. So we can consider railway stations to be observable by the information system.

    (b) Rail track: Not in the subject domain because the system need not know the identity of the tracks. The system needs only information about connectivity, not about the tracks.

    (c) Traveler: Very relevant, but nevertheless the system need not know the identify of a traveler. Registering individual travelers in a train would be useful for the traveler because it indicates how busy a particular connection is at different times of the day. It would be even more useful for the company because they can then start a personalized marketing campaign. But the current system does not provide this functionality and so travelers are outside its subject domain.

    (d) Railway connection: In the subject domain. It is observable by the people who initialize the database and this, as we saw, is a reliable connection. And the system must be able to identify connections. (They are identified by their end points.)

    (e) Rail car. Not relevant. The system does not need to know the identity of a rail car.

11. In figure **??**(a), a floor is identified by its level. In figure **??**(b), the level of a floor can be changed without changing the identity of the floor. Can this happen?

    Well, there may be a door in the corridor which partitions the floor into two, and that is locked at some times and unlocked at other times. We may need an elevator in each part of the corridor. Servicing one part is not equivalent to servicing another part. If both parts of the corridor are serviced by different elevator systems, controlled independently from each other, then there is no problem. But if they are serviced by the same system, then the controller must know in which part a button was pushed. And in that case, it is not sufficient to identify a floor by its level, and we should use model (b).

    So it all depends on what we want to do with the controller. Of course, we can represent the part of the floor (east, west) where a button is located as button attribute. But then we would be mixing up floor and button attributes, which is not a good idea.

    Notice how the decision is guided by consideration of the counting criterion of entities. When are entities (floors in this case) the same and when are they different?

12. (a) In figure **??**(a), Participation is a relationship between students and exams. That means that each Participation instance is identified by a pair $\langle \mathsf{s}, \mathsf{e} \rangle$, where s is a student and e an exam. Since any extent of Participation is a *set* (which cannot contain duplicate elements), there can at any moment be at most one participation per student per exam.

    If this is not what we want, then the model in figure **??**(b) is better. It can represent any number of participations of one student in an exam.

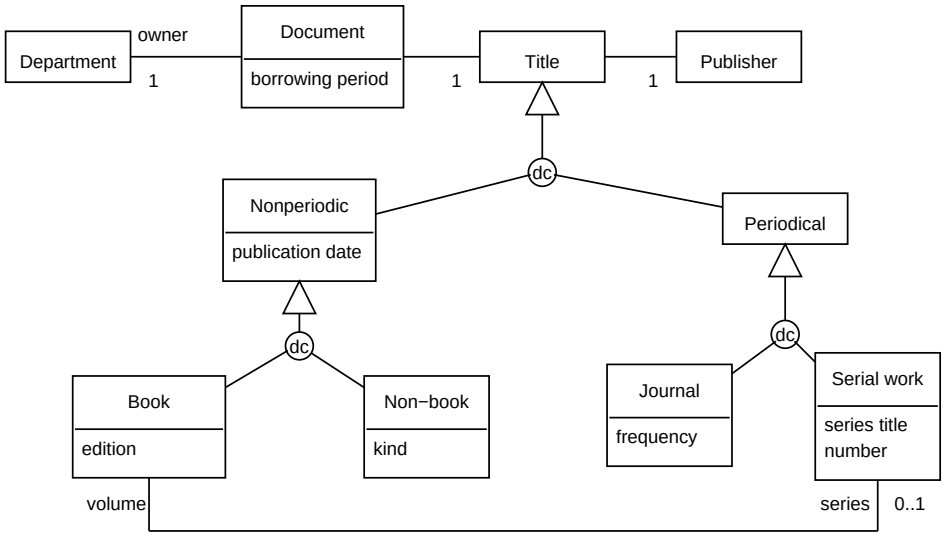


Figure 9.6: Improved document taxonomy.

(b) Figures **??**(a) and (b) distinguish the different instances of an exam. In figure **??**(a), each student can only participate in each exam offering once. This is an improvement (from the student's point of view) upon figure **??**(a), where a student can partake in only one offering of an exam.

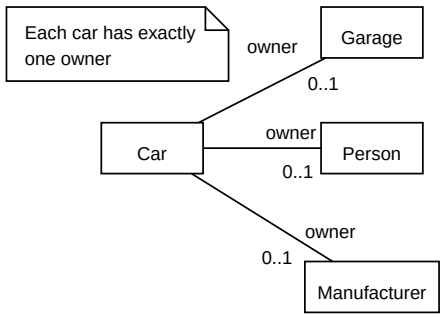
In figure **??**(b), each student can participate any number of times in one exam offering. This is nonsense and should be excluded. The difference with figure **??**(a) is that participations have their own identifier. It is very unlikely that there is a need for this.

Notice how the modeling decision is again guided by consideration of the counting criterion of relationships. How many instances of a type can there be and how are they counted?

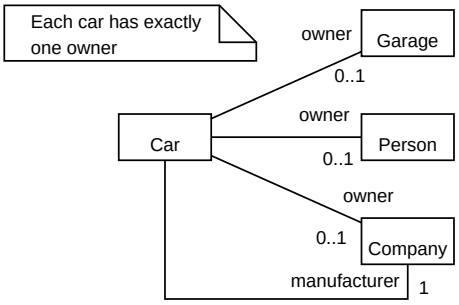
13. Figure 9.6 gives one possible improvement. It distinguishes documents, which are physical copies, from titles. One title may have many copies, each of which may have a different borrowing period, that may be any natural number including 0 (not borrowable). Titles appear once or with a period. All non-book titles, such as rare prints, maps and globes, are like this. Periodicals are either journals or serial works. An example of a serial work is Springer Lecture Notes in Computer Science. A book may be a volume in a serial work.

14. (a) Owner is a role that some entity (e.g. a person) is playing with respect to some other entity (a car). Person is *not* a role. A person entity is a person in itself, rather than with respect to something else. Manufacturer, Car and Garage are borderline cases. They are artificial entities manufactured with a particular functionality in mind, namely to manufacture cars, to transport people and to sell and maintain cars. This functionality is provided to users and customers; but these functions are not roles played by these cars and garages with respect to users and customers. Rather, cars and garages essentially have these functions. They cannot exist without these functions. If they would not have these functions anymore, they would cease to be cars or garages.

(b) The model is wrong because not all garages, cars and manufacturers are car owners.



(a) An improved model of car ownership.



(b) Another improvement.

Figure 9.7: Models of ownership.

(c) Figure 9.7(a) gives an improved model. The constraint cannot be expressed in the diagram, so we add it as a note. The model has been improved further in figure 9.7(b), by including the player of which manufacturer is a role.

15. (a) In **??**(a), there are no companies. In models **??**(b) and **??**(d), a supplier is identical to a company.

    (b) In **??**(c) and **??**(d), a supplier can have additional attributes in addition to those of companies.

    (c) In **??**(c), a supplier be identified independently from identifying a company.

    Model **??**(a) is never preferred because a role is modeled as an entity type.

    Model **??**(b) is useful if supplier is a role played by companies with respect to parts.

    If a supplier can have attributes in addition to those of companies, then **??**(c) or **??**(d) must be used. For example, we may want to distinguish trusted suppliers from risky suppliers. This is not an attribute of company, because one company can be a trusted supplier of one part and a risky supplier of another part. It is a property of the role a supplier plays with respect to a part, which is why **??**(c) or **??**(d) should be used.

    Model **??**(c) decouples the identification of a supplier from that of a company. This is useful if one company incarnates several suppliers, each with their own registration number in the Chamber of Commerce, and we still want to represent the fact that these "really" are the same company, i.e. if these are the same people in different disguise.

16. It depends. Maybe we want to count the number of wrecked cars that we currently own. And maybe we count wrecked cars by counting the number of cars that have been wrecked. In that case, Wrecked car is a dynamic subtype of Car. But we may count the number of wrecks that look like cars. We then may count two wrecks as different, where they actually originated from one car. In this case, Wrecked car is not even a subtype of Car, because it has a different identification criterion.

Chapter 10

Answers for Chapter 10 (The Dictionary)

3. There is a one-one relationship between an entity and its identifier. That means that an identifier, once used for an entity, is never reused for another entity.

   (a) A key of a database table. Not an identifier: The key value of a tuple may be updated and even reused, as long as in any state of the database, it is unique in the table.

   (b) People regularly get new passports with new numbers, so passport numbers cannot be identifiers of people. But it does identify the passport itself uniquely.

   (c) If a personnel number in a company is never changed or reused, it identifies the employee, but not the person. One person can quit his job and join the company later, getting a different employee number.

   (d) We would like a social security number to identify all tax payers. But in the Netherlands there are tax payers without social security number (*sofi* number) and currently the government considers extending its use to students, who mostly do not pay tax at all. In addition, of course, there is fraud, causing some people to use several social security numbers.

   (e) There is an unlimited and growing set of Unix processes that have been created; at some point in time, the process numbers are reused. So it is not an identifier of the process.

   (f) There is a world-wide authority that allocates finite sets of unused Ethernet addresses to manufacturers. There is however no agreement among manufacturers about what it is that these addresses identify. Some manufacturers use it to identify Ethernet boards. Others use it to identify the machine in which an Ethernet board is placed. Because one machine can contain several Ethernet boards, and also one Ethernet board can be taken out of one machine and placed into another machine, Ethernet addresses are not identifiers.

   (g) An Internet domain name is a proper name used to identify clusters of nodes in the Internet. It contains information, such as the name of a country, the name of a local network within the country and the name of a host in this local network. It is not an identifier, because these names can change even though the identified host does not change. For example, the union of West with East Germany, and the division of Czecho-Slovakia, caused some domain names to change.

4. (a) Chair. This has open texture. Chairs may have zero, one, two, three or more legs, they may have a flat surface to sit on but it may also be a perfectly round ball, it may

stand or hang from the ceiling, etc. The intensional definition "A device to be used for sitting" comes close, but since I may use devices not intended to be used this way as a chair, this is not yet good enough. So I regard this to be an open-textured term. The list of examples just given can be used as extensional definition. There is no single authority that determines what is or is not a chair. We all do.

(b) Assembler. A program that translate symbolic assembly code into a machine language program.

(c) Employee. It depends upon our subject domain model how we define this. If we view Employee as subtype of Person, then the definition is "A person who has an employment contract with a company." If we view Employee as a distinct identifiable role of Person, then the definition is "A role played by persons, in which the player has an employment contract with a company."

(d) Car. According to *Webster's*, "A vehicle moving on wheels."

Chapter 11

Answers for Chapter 11 (State Transition Lists and Tables)

1. A scenario consists of a set of possible states and state transitions. Each of these transitions can be described by a transactional list entry.

2. Extract enough of the condition on the current state to distinguish between the different effects that the event can have.

4. Each column corresponds to a combination of truth values of conditions for one event in a stateless transformation table.

5. See figure 11.1.

6. See figure 11.2. Note that the current state involves the actual event parameter.

Initially		Bug identified
Event	**Current bug state**	**Next bug state**
assign programmers	Bug identified	Fixing bug
propose fix	Fixing bug	Fix proposed
start testing	Fix proposed	Testing
report problem	Testing	Fixing bug
report fix	Testing	Fixed
release	Fixed	Relased

Figure 11.1: State transition table of bugs.

required: Rational
current, max: Natural

Initially		current := 0	Ready to receive tea bag.
Stimulus	**Current controller state**	**Controller response**	**Next controller state**
tea bag arrives(w)	Ready to receive tea bag, w = required	put in box, current := current +1	Waiting for tea bag to be put in box
	Ready to receive tea bag, w ≠ required	remove teabag	Waiting for tea bag to be put in container
teabag removed	Waiting for tea bag to be put in box, current ≥ max	stop belt, replace box	Waiting for box to be replaced
	Waiting for tea bag to be put in box, current < max		Ready to receive tea bag
	Waiting for tea bag to be put in container		Ready to receive tea bag
box replaced	Waiting for box to be replaced	start belt, current := 0	Ready to receive tea bag

Figure 11.2: State transition table for the box removal function.

Chapter 12

Answers for Chapter 12 (State Transition Diagrams)

3. If the transition leaves state S and has guard g, then the two events that can trigger the transition are: entry of S and a change of truth value of g from false to true.

4. If an action a performed in a transition affects a condition g that is used as guard along a transition, then we need a decision state to sequence the performance of a and the testing of g.

6. A state reaction does not involve exit from or entry into a state. A state transition has priority over a state reaction.

9. When the end time occurs and the controller is in state Not heating, this would cause the heater to switch on.

12. (a) See figure 12.1. The important semantic difference between this statechart and the Mealy diagram is that the statechart can respond to a flipx and a flipy event at the same time.

    (b) See figure 12.1. This diagram represents almost the same behavior as that of figure **??**. The only difference is that it does not indicate the initial value of the variables.

13. (a) See figure 12.3. The extend loop brings the book back to the initial state of Checked out, regardless with state of Checked out it is currently in.

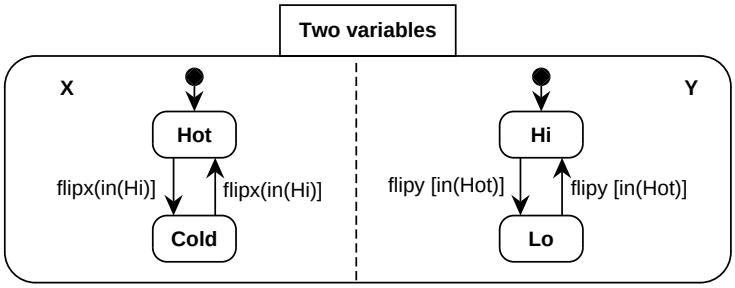


Figure 12.1: An STD with two explicit variables.

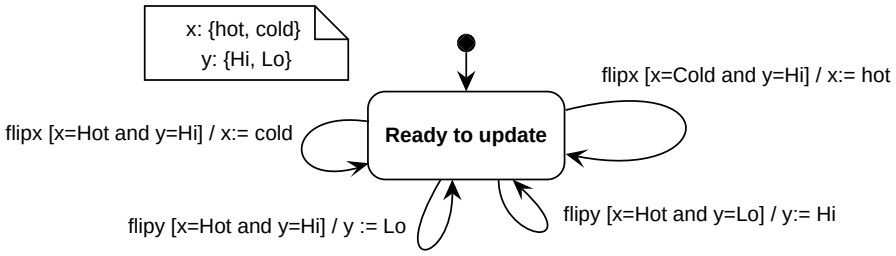


Figure 12.2: A trivial STD.

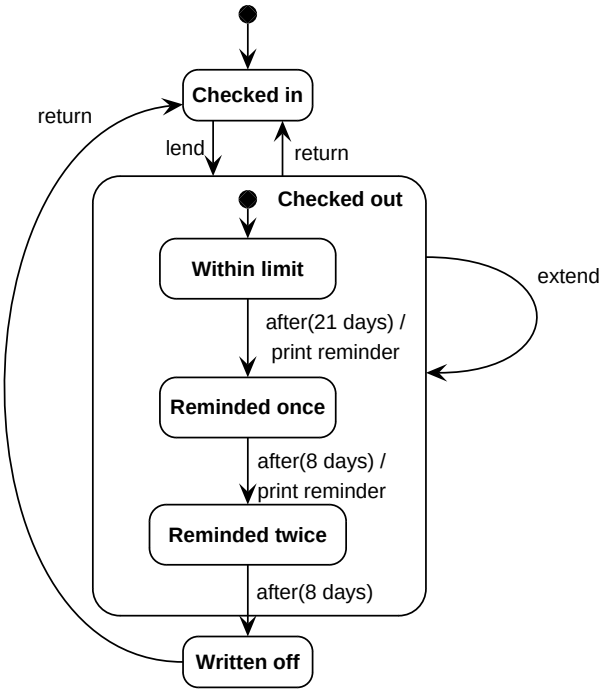


Figure 12.3: Introducing hierarchy.

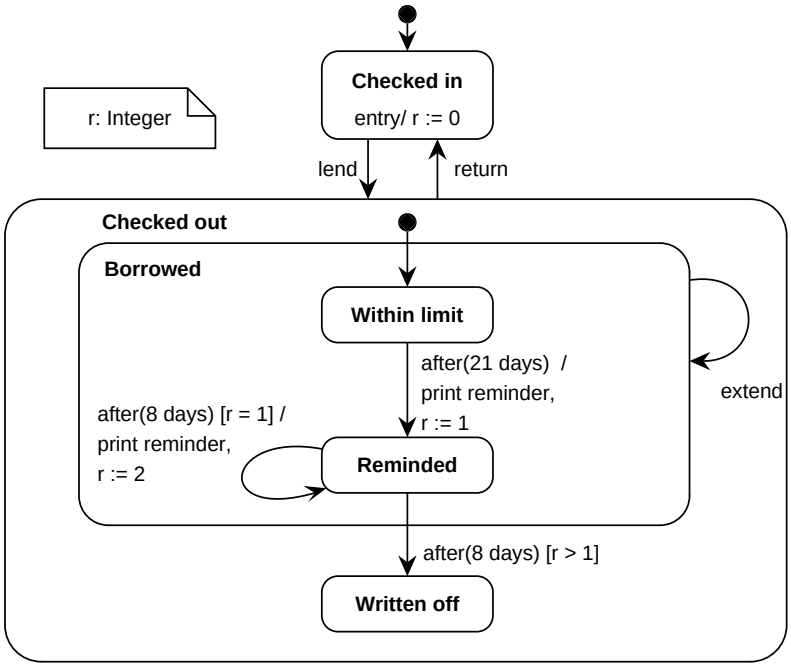


Figure 12.4: Using a local variable.

(b) See figure 12.4.

(c) See figure 12.5. The two processes coordinate by means of the in(state) predicate. See the transitions triggered by reserve and extend.

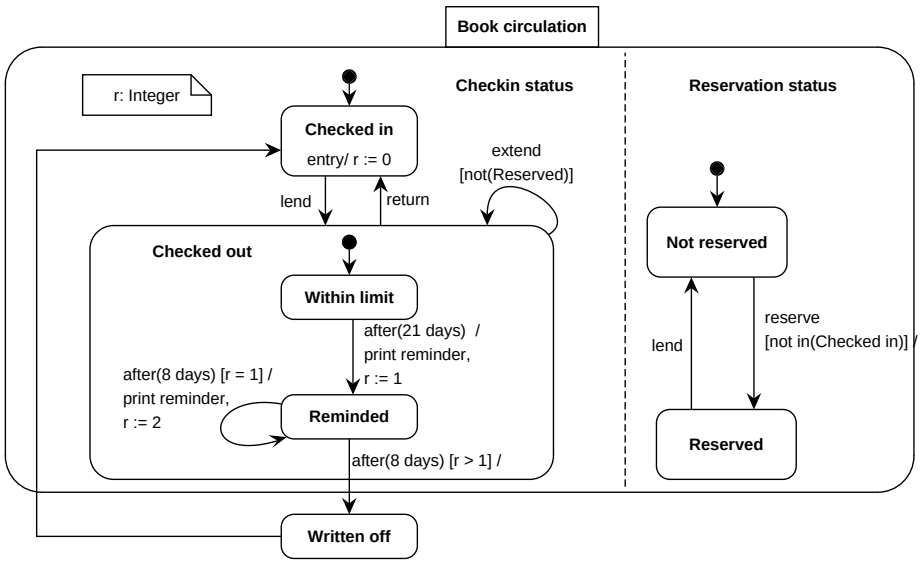


Figure 12.5: Adding a parallel process.

Chapter 13

Answers for Chapter 13 (Behavioral Semantics)

5. The behavior must be fast enough to process the next event, and all actions along a transition can use the same time value.

7. S1 → S, S2 → S, P → S1, T → S1, P ↔ not T, Q → S2, R → S2, S → S2, Q ↔ not R Q ↔ not S R ↔ not S.

8. The superstep response has already been shown in chapter 12. It is this:

Event	Current basic configuration	Action	Next basic configuration
e1	S11, S21, S31	e2	S12, S21, S31
e2	S12, S21, S31	e3	S12, S22, S31
e3	S12, S22, S31	e4	S12, S22, S32
e4	S12, S22, S32		S11, S22, S32

   If e5 occurs at any time during this response, it will be handled in basic configuration S11, S22, S32, which means that it will be ignored (assuming the Ignore semantics). In the step semantics, if e5 occurs during the first transition, then the response is nondeterministic. One possible response is to choose e2 and ignore e5:

Event	Current basic configuration	Action	Next basic configuration
e1	S11, S21, S31	e2	S12, S21, S31
e2, e5	S12, S21, S31	e3	S12, S22, S31
e3	S12, S22, S31	e4	S12, S22, S32
e4	S12, S22, S32		S11, S22, S32

   Another response is to choose e5 and ignore e2:

Event	Current basic configuration	Action	Next basic configuration
e1	S11, S21, S31	e2	S12, S21, S31
e2, e5	S12, S21, S31		S11, S22, S31

9. (a) Yes.

   (b) The S3 node in both diagrams does not represent the same state because from that node on, the two machines show different behavior. Machine (a) accepts either a b or a c but not both, whereas machine (b) accepts both.

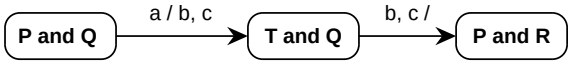


Figure 13.1: The superstep response.

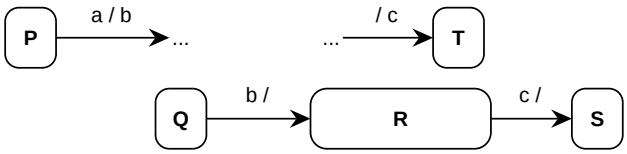


Figure 13.2: The nested step response.

10. In the superstep semantics, first b, c are generated and then the step to R is taken. The final basic configuration is P, R. See figure 13.1.

    In the nested step semantics, first b is generated, which triggers transition Q → R, and then c is generated, which triggers the transition R → S. The final basic configuration is T, S.

Chapter 14

Answers for Chapter 14 (Behavior Modeling and Design Guidelines)

1. All superstates are activity states. All states whose name start with a verbal noun in *-ing* form are activity states too, even though there is no entry action that starts the activity. They are activity states because the described behavior is not waiting for an event to occur but performing an activity.

2. E4 can occur jointly with E2 or E3. E2 and E3 exclude each other. E4 can occur jointly with E1 if the duration of a recipe is 0.

4. See figure 14.1.

5. (a) See figure 14.2.

   (b) See figure 14.3. This is not equivalent to figure 14.2, because it fropm configuration {Waiting for tea bag to be put in box, Ready to replace box}, the event teabag removed will cause a transition to configuration {Reade to receive tea bag, Ready to replace box}. This transition is not possible in figure 14.2.

6. (a) See figure 14.4.

   (b) The chemical and its containers are two different entities at different levels of aggregation. The chemical can be contained in various containers.

7. (a) Acquire chemicals, Purchase chemicals, Report on chemicals. Maintenance of information, which is an important function of the system, is a secondary service that is needed to support the other functions, which are more basic. Maintenance of information is not an activity to be supported by CTS but an activity needed to support the functioning of CTS.

   (b) The leaves of the function refinement tree are transactions.

   (c) The two solution goals are to use containers as long as possible and to use a standardized purchasing procedure. The system contributes to this if the following assumptions are satisfied.

   - Containers are not purchased if there is still sufficient chemical in inventory.
   - The only way to purchase chemicals is through CTS.
   - The CTS purchasing procedure is cheaper than a manual procedure.

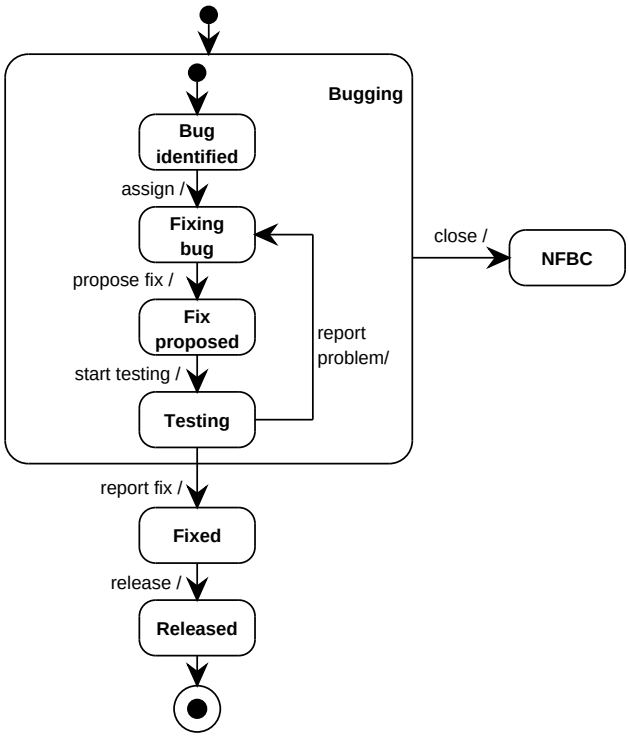


Figure 14.1: Using hierarchy in a statechart.

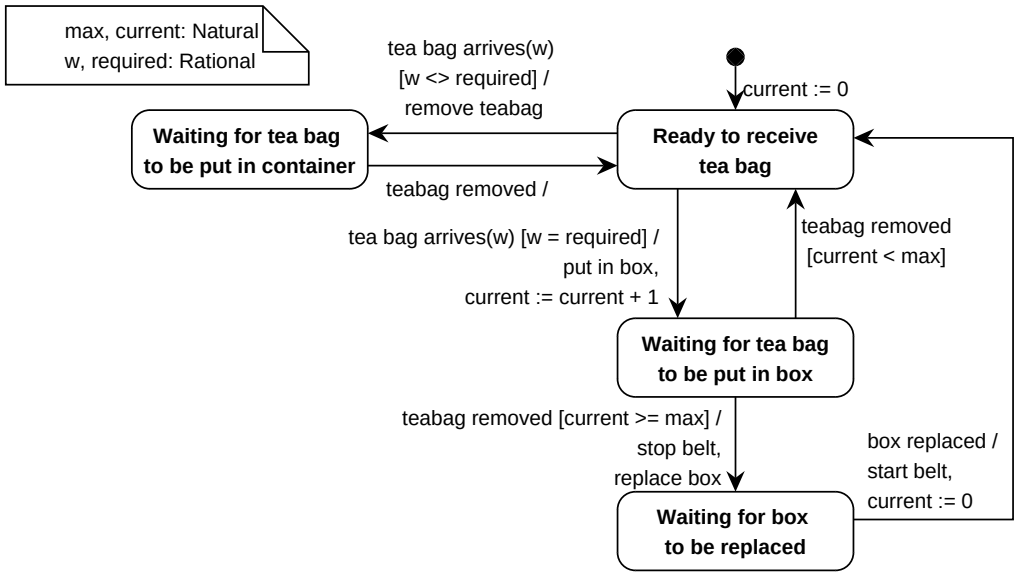


Figure 14.2: STD for removal of tea bags.

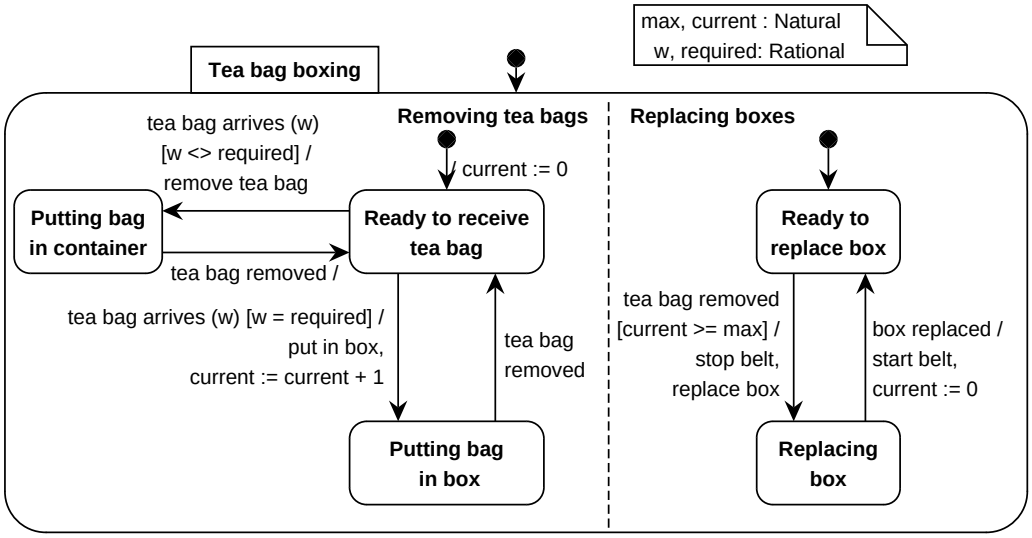


Figure 14.3: STD for removal of tea bags with parallel processes.

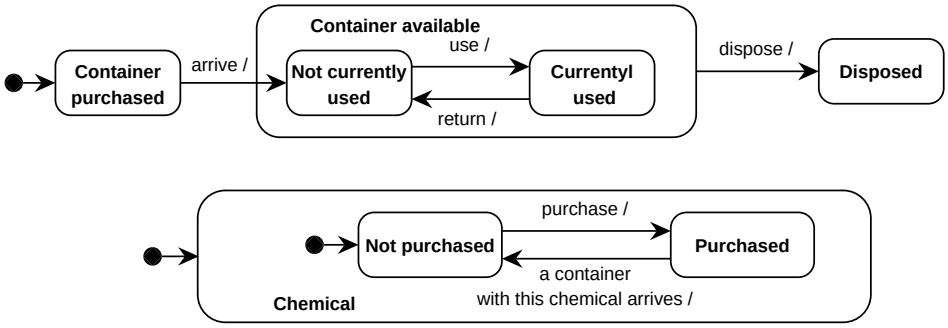


Figure 14.4: Two STDs for the life cycle of a chemical and of a container.

This leads to an extra desired property of CTS:

– CTS refuses to purchase a chemical that is still available from inventory.

Chapter 15

Answers for Chapter 15 (Data Flow Diagrams)

2. (a) No, because events are instantaneous.

   (b) Yes, because the value dimension is orthogonal to the time-dimension.

5. (a) Local Variable: current. Tests: w = required and current ¿= max. Updates: current := current +1 and current := 0.

   (b) Figure 15.1 shows the DFD and figure 15.2 shows the STD. Note that we need a decision state to wait on the result of the test. We also need to split the event flow tea bag arrives(w) into a parameterless event flow tea bag arrives and a time-continuous data flow w. The data flow must be time-continuous to make the data available at the moment that the control process asks Test weight to compare w with required.

6. Figure 15.3 shows the DFD of a monitoring system. Each function corresponds to a data process. The process Bug life cycle is specified by the STT of figure 15.4. It does the same as the STT of figure **??** but in addition refuses to do anything else.

7. A process is some system activity. An activity state is a state in which the system does something, i.e. in which the system performs a process. The difference is in the use of these concepts. A process is represented in a DFD to represent its interfaces. An activity state is included in a statechart to represent the behavioral structure of the activity.

8. It is stateless when it only contains stateless processes.

9. Yes. When disabled, the process would not respond to triggers and when enabled, it would respond.

10. (a) Material items have a mass, size and location but data items, although physical, are relevant only for their meaning. For example, there is a meaning convention by which we know that the data item "2", the black dot of ink that you now see, has a meaning, namely the number 2. The dot of ink has a mass, size and location, but this is not relevant. What is relevant is that it has a meaning. There is another meaning convention in which we can assign the same meaning to a certain pattern of magnetization on a disk, and this makes that pattern of magnetization an instance of the same data. Consider this as the difference between the way an elephant sees the sign board "Do not tread on the grass" and the way we see this.

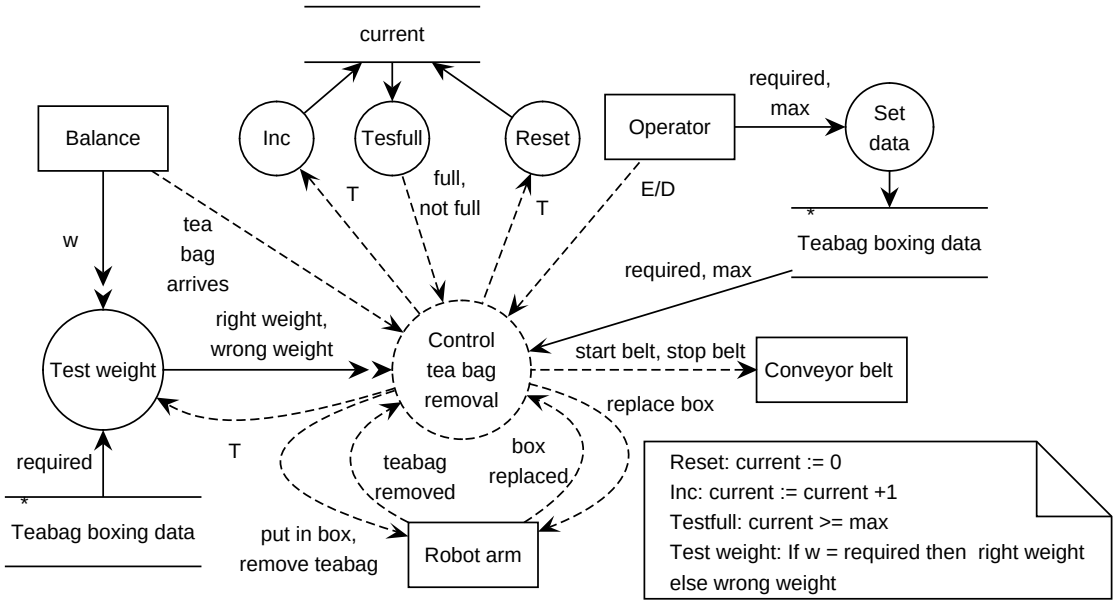


Figure 15.1: DFD of tea bag boxing using an STD without variables.

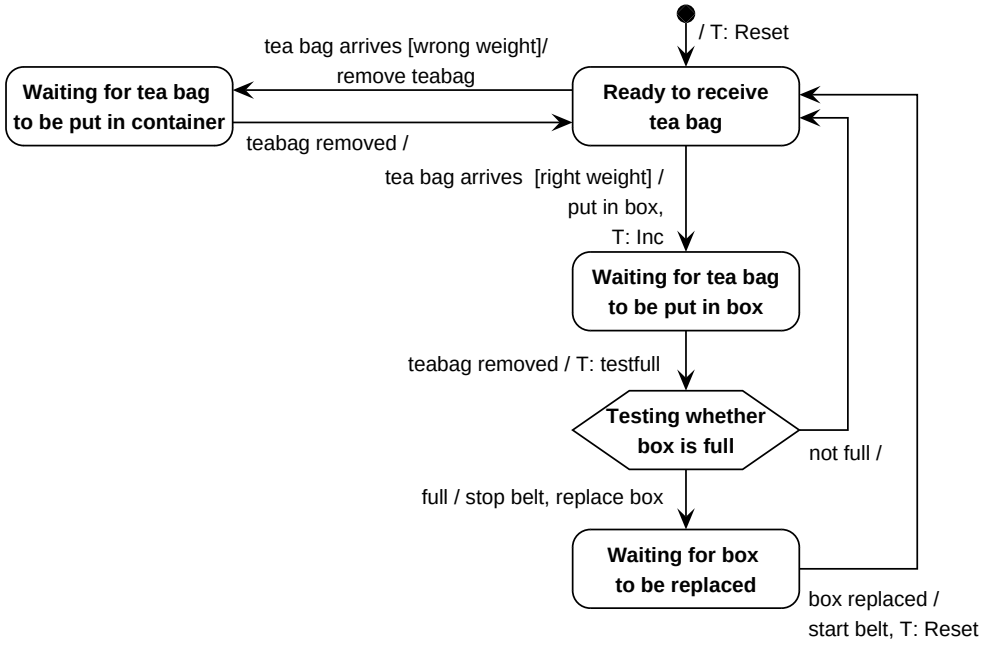


Figure 15.2: STD of control process in figure 15.1.

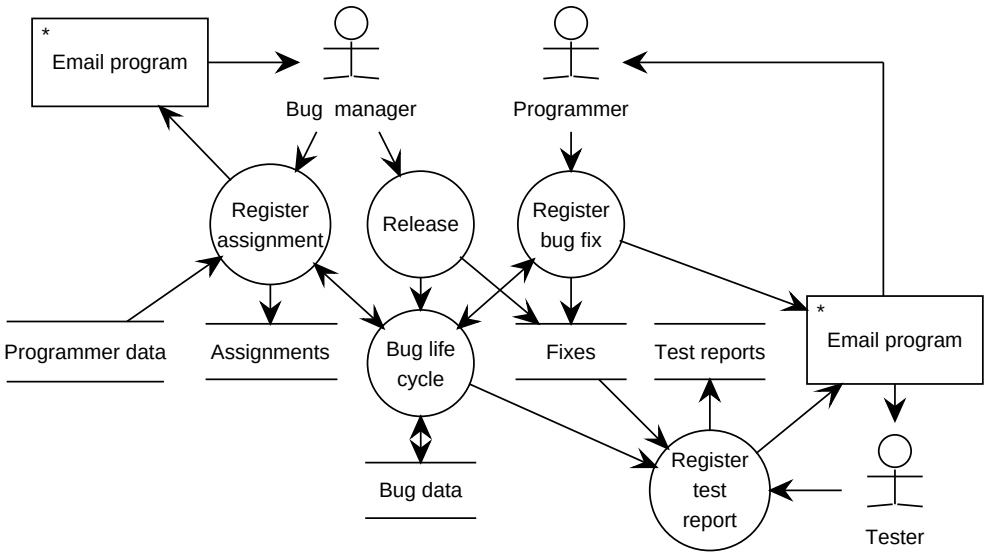


Figure 15.3: DFD of the bugfix support system.

Initially		Create bug data	Bug identified
Event	**Current bug state**	**Action**	**Next bug state**
assign to programmers	Bug identified		Fixing bug
	Other	error message	unchanged
propose fix	Fixing bug		Fix proposed
	Other	error message	unchanged
report problem	Fix proposed		Fixing bug
	Other	error message	unchanged
report fix	Fix proposed		Fixed
	Other	error message	unchanged
release	Fixed		Relased
	Other	error message	unchanged

Figure 15.4: State transition table of bugs. This is the specification of the "Bug life cycle" process of figure 15.3.

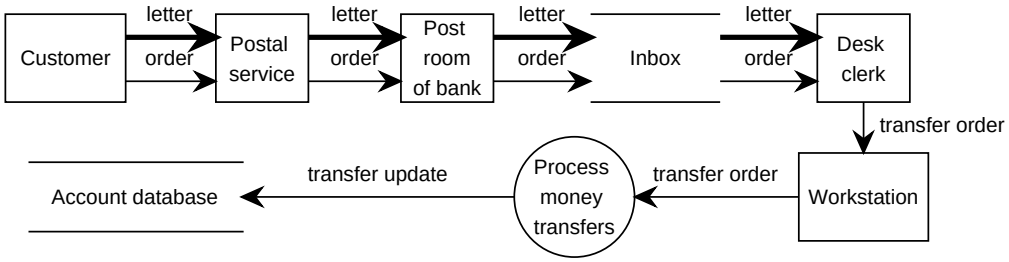


Figure 15.5: Material flows.

(b) A material flow represents the movement of physical items through space, i.e. from one place to another place. A data flow just represents the availability of input or output data to a process.

(c) The operations on a material store differ from those of a data store. A material store has the following operations:

- Add an item to the store. This is nondestructive, for the items already present are not changed. This is similar to creating a new record in a data store.
- Remove an item from a store. This is destructive for the store contents, for the removed item is no longer present in the store. This differs from data store read, which is not destructive.

(d) See figure 15.5. The material flow from the customer to the desk clerk consists of a piece of paper in an envelope. The model does not show where the envelope goes to or where the letter goes to after the clerk has read it. The letter is the carrier for a data flow, containing the order.

Note that we stop representing material flows at the desk clerk. In reality, the data flow from the desk clerk to the workstation is implemented by a material flow. Some physical objects move to realize this flow. The workstation then sends electrons — physical objects too— to the processor doing the computations and finally to the disk that holds the database. The very idea of a DFD is to abstract from this material flow, and therefore we did so in the lower part of figure 15.5.

Chapter 16

Answers for Chapter 16 (Communication Diagrams)

3. Figure 16.1 gives the communication diagram and figure 16.2 gives the allocation table.

4. Figure 16.3 shows the communication diagram and figure 16.4 shows the allocation table.

5. An E/D prompt switches a component on and off, and this means that all its components are switched on and off. A trigger on the other hand gives particular components a kick to do something useful.

6. If B always causes e2 when prompted by e1 and does nothing else, then it acts as a communication channel and can be removed. To be a useful component, B should have to make a decision or introduce a delay.

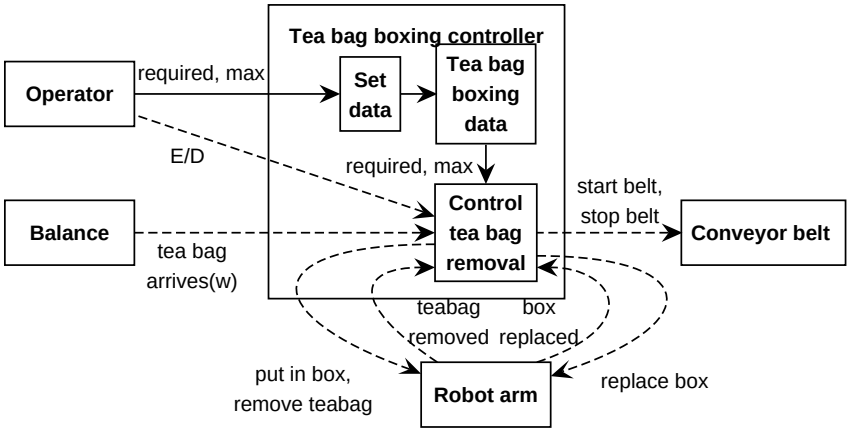


Figure 16.1: Communication diagram of the tea bag boxing controller.

	Set weight	Set maximum count	Place in box	Replace box
Set data	X	X		
Tea bag boxing data	X	X	X	X
Control tea bag removal			X	X

Figure 16.2: Allocation of tea bag boxing controller functions to components.

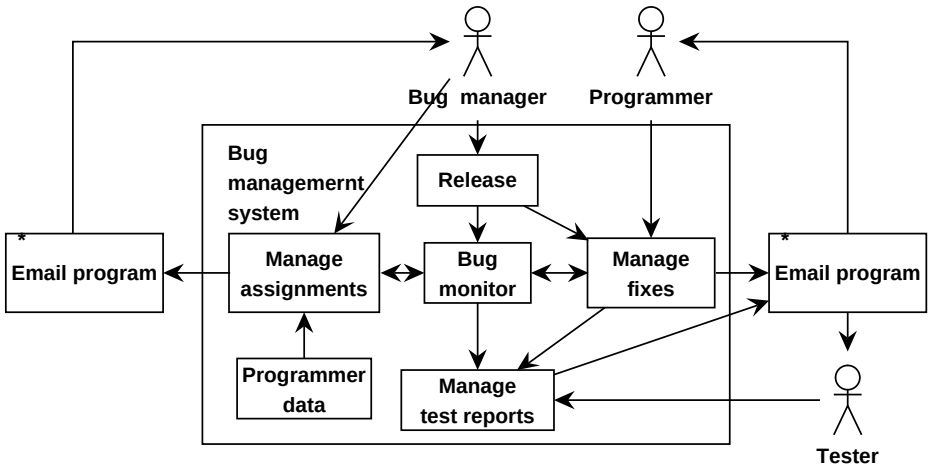


Figure 16.3: Communication diagram of the bug fixing system. Each box collects some of the components of the DFD of figure 15.3.

	Register assignment	Register bug fix	Register test report	Register bug release
Manage assignments	X			
Programmer data	X			
Bug monitor	X	X	X	X
Manage fixes		X		
Manage test reports			X	
Release				X

Figure 16.4: Allocation of bug fixing support functions to components.

Chapter 17

Answers for Chapter 17 (Execution Semantics)

1. As described in this chapter, each component has an input buffer.

6. Eager read/lazy write: At the start of a step, it would accept all inputs, and so create, update and delete items in its memory. At the end of a step, it would write all outputs, and so provide all values asked for it in a read operation. Lazy read/eager write: It would process its inputs as late as possible. Because it has nothing else to do, that would be at the end of a step. It would write its outputs as early as possible. Because it has nothing else to do, that would be at the start of a step.

7. (a) Figure 17.1 shows the figure with numbered nodes. The breadth-first order of execution is as follows: {1}, {2}, {3, 8}, {7, 9}, {8, 10, 17}, {4, 7, 11, 16, 19}, {5, 8, 12, 12, 18}, {7, 17}, {10, 16}, {8, 11, 19}, {7, 12, 18}, {17}, {10, 16}, etc. Note that in the seventh step, 12 receives two events simultaneously.

   (b) If we take the path from 1, 2, 3, 9, 10, 11, 12 first, then we have saved some internal events for later processing. These will lead to later responses. For example, somewhat later we will go from 10 to 19 en 12. This leads to observably different behavior.

8. At the start of the step, the data store provides all values asked for in this step and at the end of the step, the data store accepts all updates sent to it. The difference with the read/write

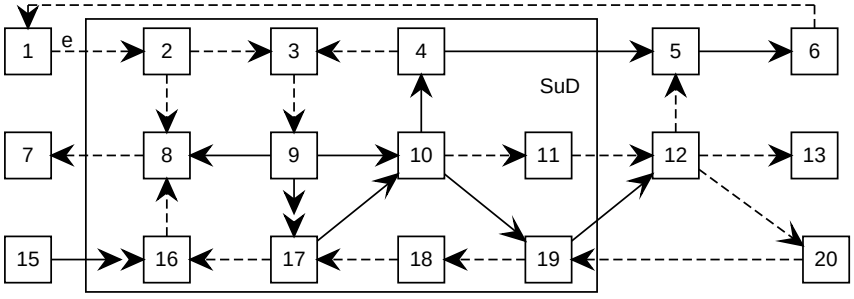


Figure 17.1: Triggering a response of the network.

semantics discussed in this chapter is that in those semantics, a data store responds to events generated in the previous step where here, we want the data store to respond to events generated by a process in the current step.

9. If there is perfect technology, clock-asynchronous behavior, and an environment that never offers more than one event simultaneously to one component, then we do not need an input buffer for that component.

10. There is always a value in a time-continuous channel, but any new value is present at the destination immediately.

11. In a nested step semantics, a component triggered by an output action is immediately executed. That is one particular way to do depth-first traversal.

12. Imperfect technology: use faster technology. Clock-synchronous semantics: Use smaller clock cycle. Finite buffer: Use larger buffer. Events occur within one "time point": use smaller time points that can discriminate their time of occurrence.

Chapter 18

Answers for Chapter 18 (Context Modeling Guidelines)

1. (a) Tea bag boxing system: This has pure directive functionality, with two feed back loops to the robot arm that regulate the placement of tea bags and the replacement of boxes.

   (b) Bug fixing support system: The manager, programmer and tester inform the system of changes in the subject domain. There are also information provision flows from the system to the manager, programmer and tester, that have a directive intent: They tell these people what to do next. If the system refuses to update the state of a bug if that would violate the workflow, the system also has directive functionality. And it has manipulative functionality because the lexical items in the subject domain are manipulated by the system itself.

2. See figure 18.1 for a context diagram.

   (a) The composite system crosses organizational boundaries because it includes parts of the company and also external vendors and the government. The emergent properties of the system are valuable for the company.

   (b) The system needs information about vendors, chemicals and containers, scientists and locations. See also the ERD (figure 9.3, page 28). Information about these entities reaches the system through the data processing department.

   (c) The desired effects of the system are that chemical containers are used more efficiently, purchasing takes place in a standard way, and standard reports are produced. These effects involve scientists, the stockroom, and vendors.

   (d) Users are scientists and the administrators who must produce reports.

   The context structure contains flows for information provision. Tracking and monitoring (which is directive) and manipulation (of purchase orders).

3. (a) The operator could perform the box replacing function, which makes the second function, setting the tea bag count, superfluous. He or she could in addition perform the function to put tea bags in a box, returning to a fully manual system.

   (b) Yes. The balance should then store the weight and the arm should store the maximum tea bag count. Based on the weight, the balance tells the robot arm where to put a bag. Based on the current and maximum tea bag count, the robot arm then decides whether to replace a box.

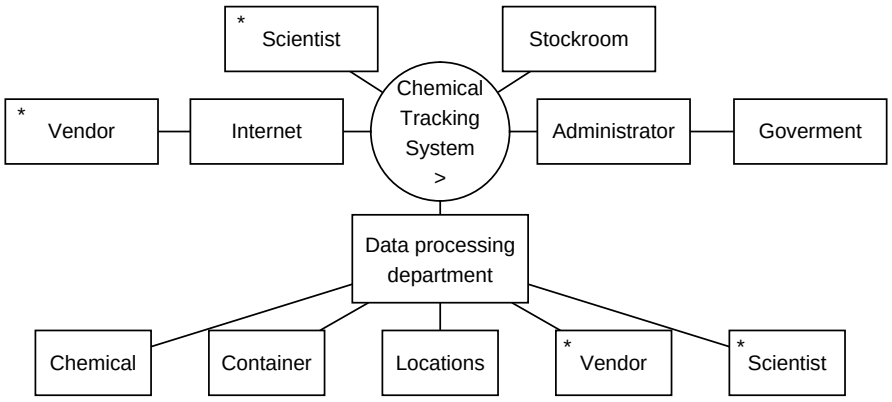


Figure 18.1: Context diagram of the Chemical Tracking System.

4. (a) They should be included because they are part of the effects of the system relevant for the creation of the desired emergent properties of the composite system (efficient and effective management of introductory courses).

   (b) They should not be included because they are not effects of the system. The SuD produces the messages to be communicated, but the communications place only because external entities decide to do so.

   I prefer (b) because the resulting diagram is simpler.

5. (a) Only lexical entities can be stored and manipulated by the SuD. The lexical entities are tickets and stamps and the description of the system functions tells us that the system must store and manipulate them.

   (b) The relationship between a ticket and the right of passages is a meaning relationship, not an interaction relationship. Manipulation of a ticket does have an effect on the right of passage, but there is no communication channel through which this effect takes place. It takes place because the legal context defines it to take place. Since a context diagram does not represent meaning relationships, we do not have to include these effects in the diagram.

Chapter 19

Answers for Chapter 19 (Requirements Decomposition Guidelines)

1. An architecture tells us how system-level properties emerge from component properties. A style is aset of constraints on architectures.

3. (a) The Von Neumann style and its refinement into the blackboard style, and the object-oriented style with two versions: one with destination addressing and one with dynamic channel addressing.

   (b) For a functionality that requires storage of sets of data (information provision and manipulation), a Von Neumann style should be considered. For directive functionality, encapsulation of behavior into objects should be considered.

6. See figure 20.1. Update vendor information is a function triggered by a temporal event (every night). The contents of the data stores is defined by the ERD you made earlier in chapter 9. (See page 28 for the ERD.)

7. See figure 20.2. The component Control tea bag is function-oriented and device-oriented. The data store is subject-oriented and the data transformation is function-oriented and event-oriented.

8. Location indication: Functional, subject-oriented, device-oriented. Direction indication: Functional, subject-oriented, device-oriented. Arrival sensing: Subject-oriented, event-oriented, device based. Movement control: Functional, subject-oriented, device-oriented, behavior-oriented. Door sensing: Subject-oriented. Door control: Subject-oriented, behavior-oriented. Allocation and button control: Subject-oriented, device-oriented, behavior-oriented.

9. They are all events in the life of a bug and should therefore be encapsulated in the Bug object. Figure 20.3 gives one possible solution. Another solution would be to enhance the transformations to objects that encapsulate a dialog with the user.

10. See figure 20.4. Figure 20.5 shows the life cycle of a ticket. Any event not conforming to this life cycle is simply ignored.

11. Figure 20.6 shows one possible solution. It encapsulates the heater and thermometer software objects inside the tank behavior objects.

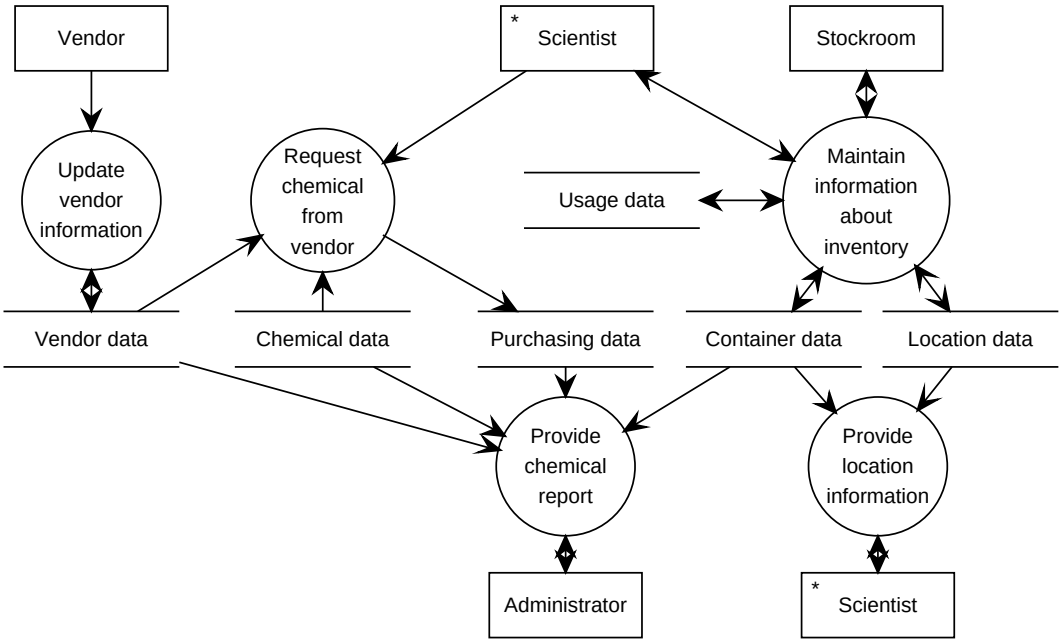


Figure 19.1: Functional decomposition of the Chemical Tracking System.

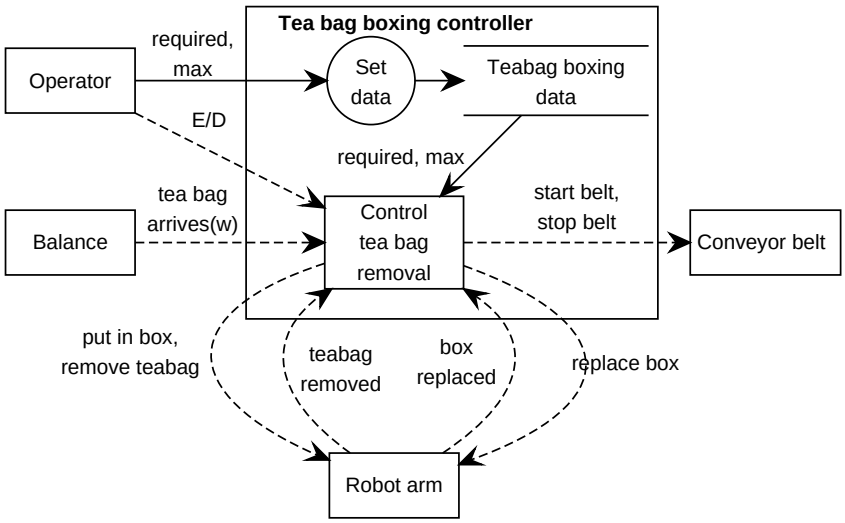


Figure 19.2: Decomposition of tea bag boxing using a hybrid style.

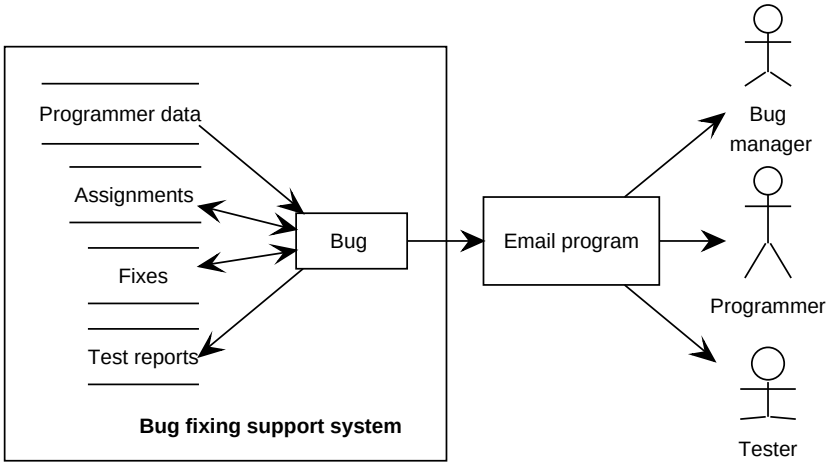


Figure 19.3: Encapsulating events into the bug object—hybrid style.

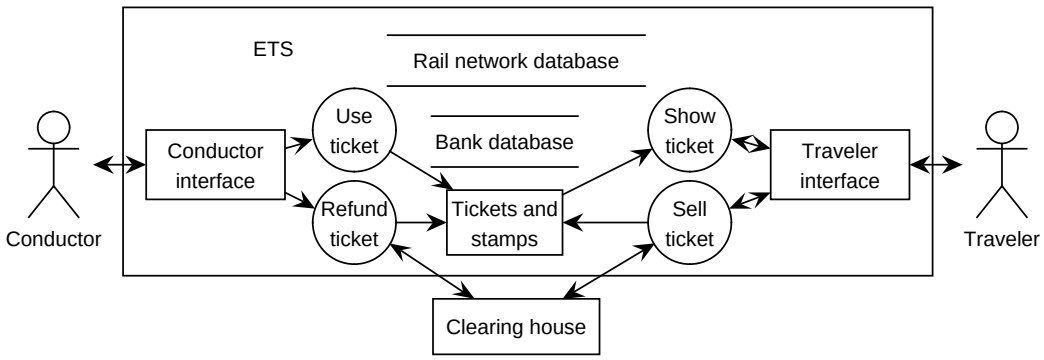


Figure 19.4: Requirements-level ETS architecture—hybrid style.

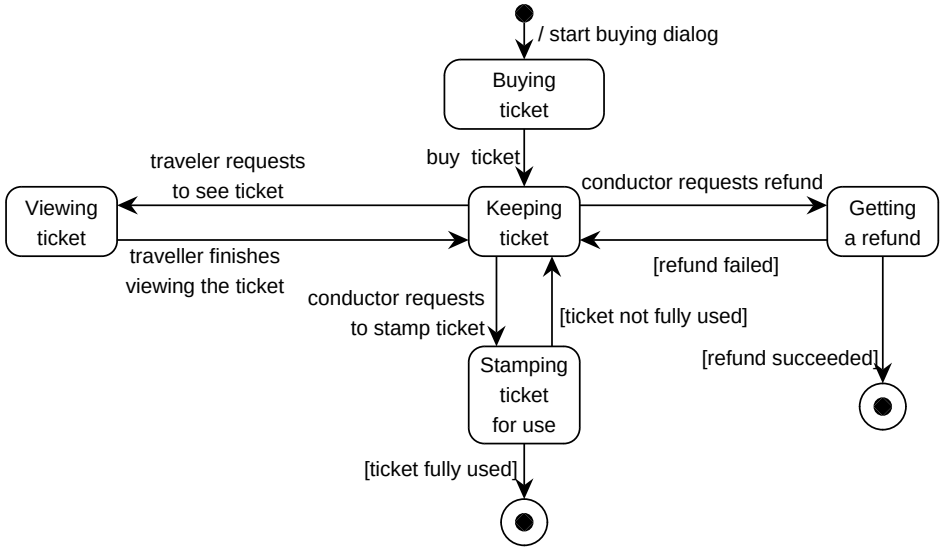


Figure 19.5: Ticket life cycle.

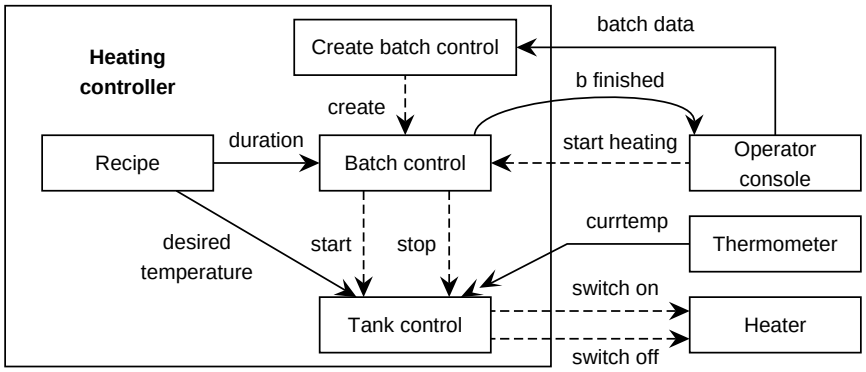


Figure 19.6: Requirements-level subject-oriented decomposition of the heating controller—object-oriented style.

Chapter 20

Answers for Chapter 19 (Requirements Decomposition Guidelines)

1. An architecture tells us how system-level properties emerge from component properties. A style is aset of constraints on architectures.

3. (a) The Von Neumann style and its refinement into the blackboard style, and the object-oriented style with two versions: one with destination addressing and one with dynamic channel addressing.

   (b) For a functionality that requires storage of sets of data (information provision and manipulation), a Von Neumann style should be considered. For directive functionality, encapsulation of behavior into objects should be considered.

6. See figure 20.1. Update vendor information is a function triggered by a temporal event (every night). The contents of the data stores is defined by the ERD you made earlier in chapter 9. (See page 28 for the ERD.)

7. See figure 20.2. The component Control tea bag is function-oriented and device-oriented. The data store is subject-oriented and the data transformation is function-oriented and event-oriented.

8. Location indication: Functional, subject-oriented, device-oriented. Direction indication: Functional, subject-oriented, device-oriented. Arrival sensing: Subject-oriented, event-oriented, device based. Movement control: Functional, subject-oriented, device-oriented, behavior-oriented. Door sensing: Subject-oriented. Door control: Subject-oriented, behavior-oriented. Allocation and button control: Subject-oriented, device-oriented, behavior-oriented.

9. They are all events in the life of a bug and should therefore be encapsulated in the Bug object. Figure 20.3 gives one possible solution. Another solution would be to enhance the transformations to objects that encapsulate a dialog with the user.

10. See figure 20.4. Figure 20.5 shows the life cycle of a ticket. Any event not conforming to this life cycle is simply ignored.

11. Figure 20.6 shows one possible solution. It encapsulates the heater and thermometer software objects inside the tank behavior objects.

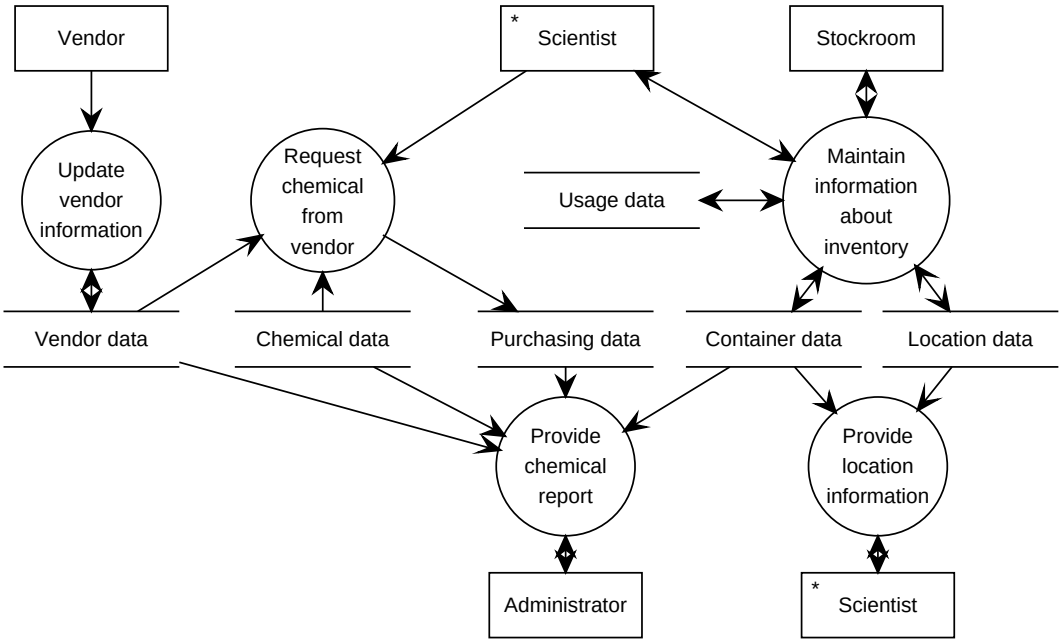


Figure 20.1: Functional decomposition of the Chemical Tracking System.

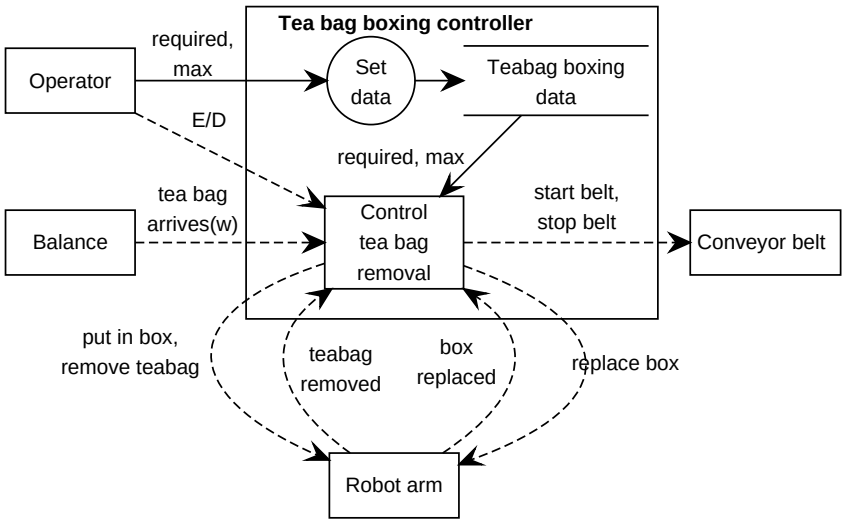


Figure 20.2: Decomposition of tea bag boxing using a hybrid style.

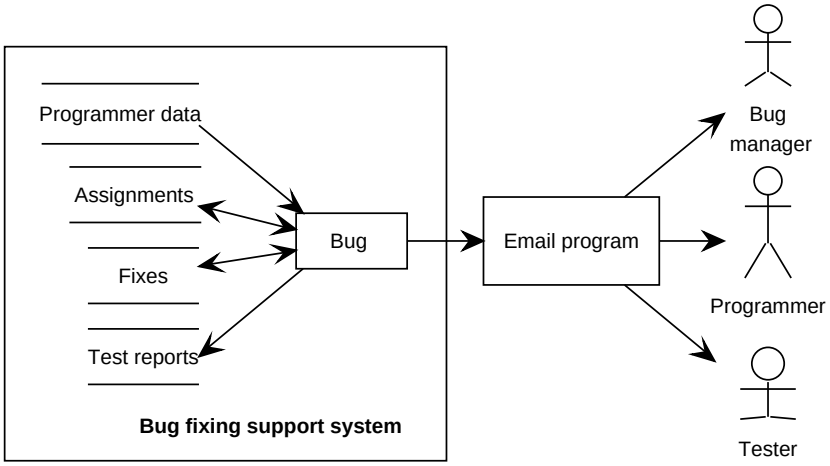


Figure 20.3: Encapsulating events into the bug object—hybrid style.

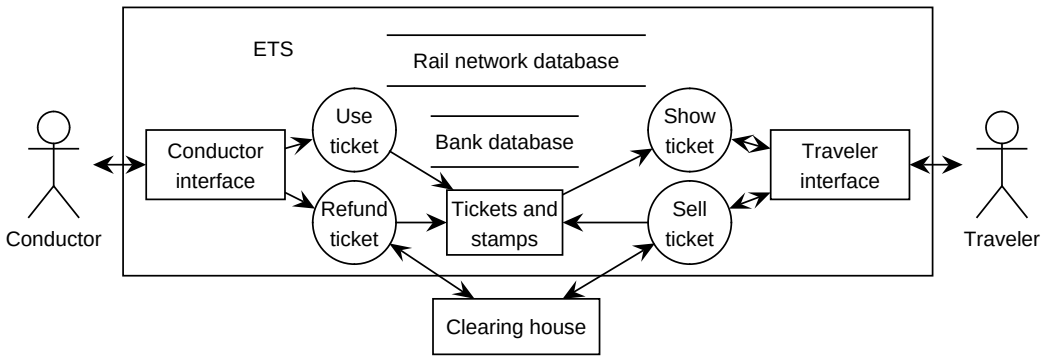


Figure 20.4: Requirements-level ETS architecture—hybrid style.

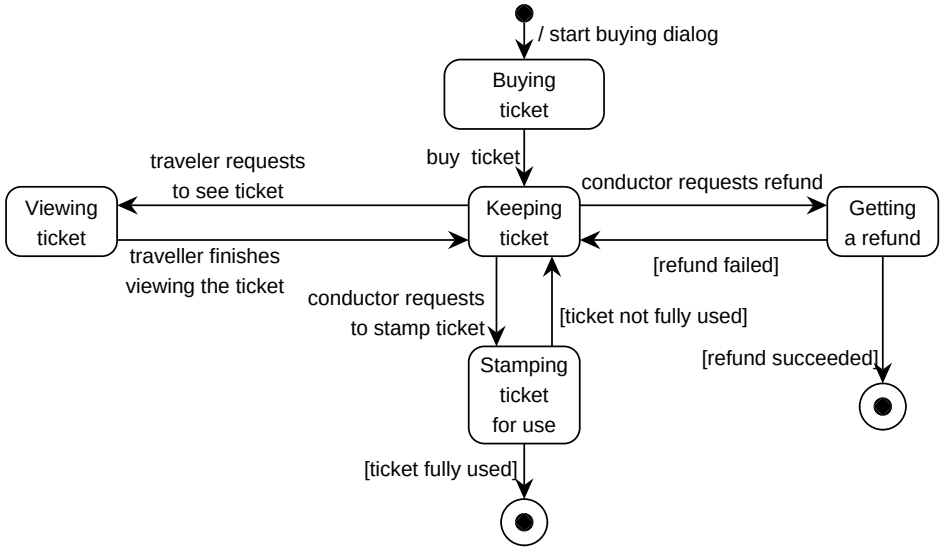


Figure 20.5: Ticket life cycle.

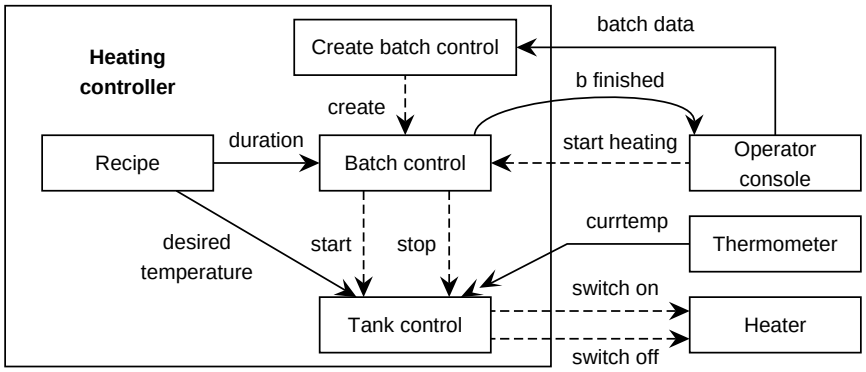


Figure 20.6: Requirements-level subject-oriented decomposition of the heating controller—object-oriented style.

Chapter 21

Answers for Chapter 21 (Statemate)

1. Because A contains a control activity then, as explained in section **??**, starting A consists of SC1. This means that SC1 enters state S1 and this implies, according to clause 1(c) of the algorithm, that the entry action of this state is performed. Executing derived actions recursively in the same step, performing the entry action implies starting B, which consists of entering S11. That finishes the initialization step. The resulting configuration is {S1, S11}, reached in one step just as in figure **??**.

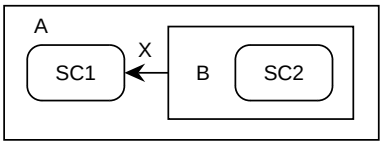
2. See figure 21.1. The transition leaving S12 triggered by e4 is replaced by a transition leaving S1 triggered by e4, that can only be enabled when the control activity in B is in state S12. The step behavior of both models is the same.

3. Starting state: {Ready to receive teabag, Ready to replace box} and current = max − 1.

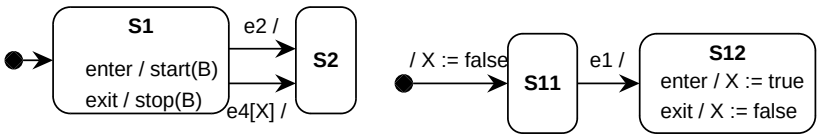
   E1 Step: Transition to Putting bag in box. Next state: {Putting bag in box, Ready to replace box} and current = max.

   E2 Step: transitions to Ready to receive bag and Replacing box. Next state: {Ready to receive tea bag, Replacing box} and current = max.

   E3 Step: Transition to Ready to replace box. Next state: {Ready to receive tea bag, Ready to replace box} and current = 0.



(a) Activity chart.



(b) SC1

(c) SC2

Figure 21.1: Moving the activity in S1 to another statechart.

Chapter 22

Answers for Chapter 22 (Unified Modeling Language)

1. (b) No.

2. (f) An object is an entity that offers services to its environment. Because it is an entity, it has identity and state.

5. (b) The only action that does not lead to a message is the terminate action.

   (c) There are no messages that cause time events, change events and completion events.

6. (a) See figure 22.1. The create and destroy messages are not visible in the statechart.

   (b) See figure 22.2. The multiplicities are not particularly interesting. Remember that they characterize snapshots, not histories. The interface for putting and getting the attributes of Tea bag boxing data is not shown.

7. (b) See figure 22.3.

   (c) See figure 22.4.

8. - *Cars and engines.* In both diagrams, a car cannot exist when it is not related to an engine. So in both diagrams,
     - when a car is created, a link to an existing engine must be created at the same time, and
     - when an engine is destroyed, the car must immediately get another engine or it must be destroyed too.

     The additional meaning conveyed by the diamond seems to be that
     - A car physically contains an engine and
     - An engine delivers a service to a car.
   - *Car surrogates and engine surrogates.* Surrogates are software objects that represent real-world objects. The multiplicity properties are identical to those for cars and engines. What could the diamond mean?
     - A car software object physically contains an engine software object. This is an implementation instruction, for it says that the physical executable code of one object must contain the physical executable code of some other object. Even if this would be a useful instruction, in this book we avoid implementation instructions in an SSD.

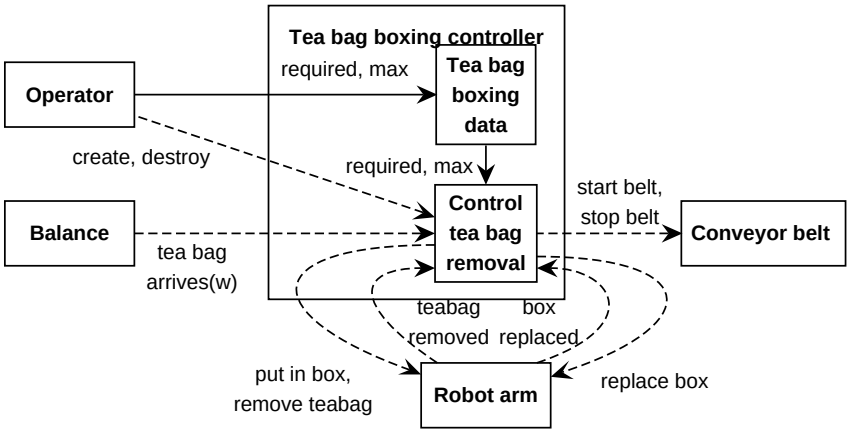


Figure 22.1: Communication diagram of an object-oriented architecture of the tea bag boxing controller.

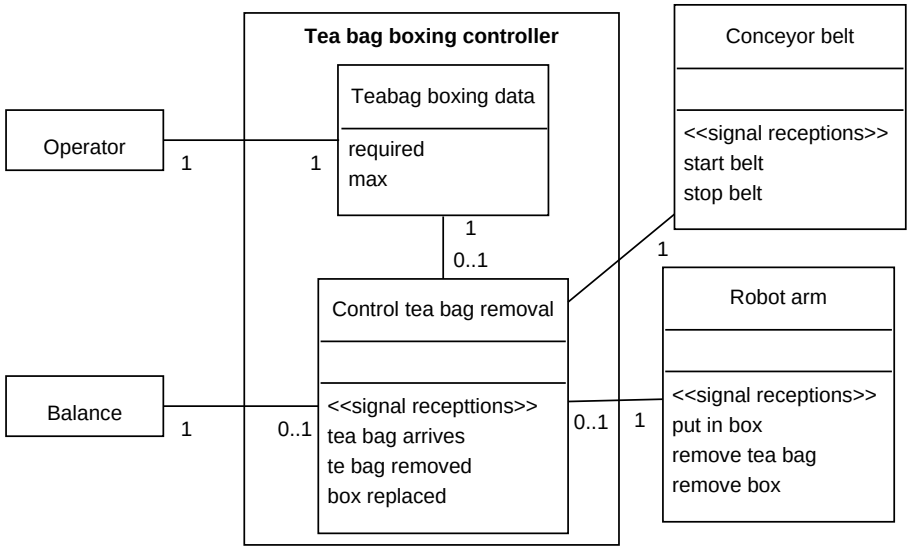


Figure 22.2: SSD of the tea bag boxing controller architecture in figure 22.1.

Entry stimulus rec
– When pass(e), then pass doors(c: Elevator cage) where e = c▪doors▪entry_sensor.
Door action comp
– When close doors(c), then close(c▪doors).
– When open doors(c), then open(c▪doors).

Figure 22.3: Transformation specifications for the elevator controller.

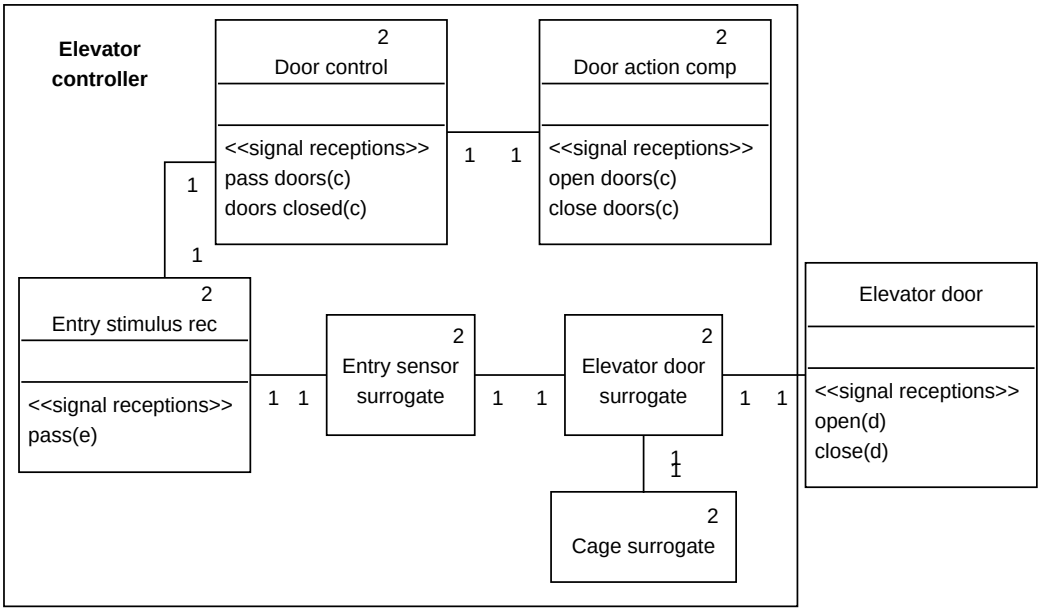


Figure 22.4: SSD showing the access paths needed by "Entry stimulus rec", "Door action comp" and "Door control".

  - An engine software object delivers a service to a car software object. Whatever this service is, it is not the same service that engines provide to cars (providing locomotive power). In fact, I cannot think of a service that an engine software object delivers to a car software object.

- *Company and department.* The multiplicity property in figure **??**(c) and (d) and in figure **??**(c) is that a department can only exist in relationship to an existing company. This means that
  - When a department is created, a link to an existing company must be created at the same time and
  - When a company is destroyed, all its departments go with it or else they immediately get a link to an existing company.

  This already provides the element of existence dependency conveyed by the black diamond. The additional meaning provided by the black diamond could be:
  - A company physically contains a department. This is nonsense. A company is not a physical object and therefore does not physically contain anything.
  - An department delivers a service to a company. This is true, but one wonders whether one needs a black diamond to express this.

- *Database and database table.* The multiplicity constraints are the same as for departments and companies. Any additional meaning conveyed by the diamond could be:
  - A database physically contains a database table. As for car and engine software objects, this is an implementation instruction. But in this case, it is an empty instruction, for a database is *defined* to be a collection of database tables.

– An database table delivers a service to a database. This is true, for a table provides part of the storage capacity of a database.

Surveying these examples, we see that the additional meaning provided by the hollow or black diamond over and above the multiplicity properties differs in different examples. My recommendation is therefore to avoid using the diamond. If you see an SSD drawn by someone else, containing a diamond, ask what the diamond means precisely.

22.1 Answers for Chapter ??

1. See section **??**.

5. One can represent the extent of a class by a compound process and one can represent individual objects by data stores accessed by the operations of the class. See figures **??** and **??**.

Bibliography

JTC1/SC7/WG6, I. (1995*a*), *ISO/IEC 9126-1 Information technology – Software Quality Characteristics and Metrics – Quality Characteristics and subcharacteristics*, International Organization for Standardization.

JTC1/SC7/WG6, I. (1995*b*), *ISO/IEC 9126-2 Information technology – Software Quality Characteristics and Metrics – External Metrics*, International Organization for Standardization.