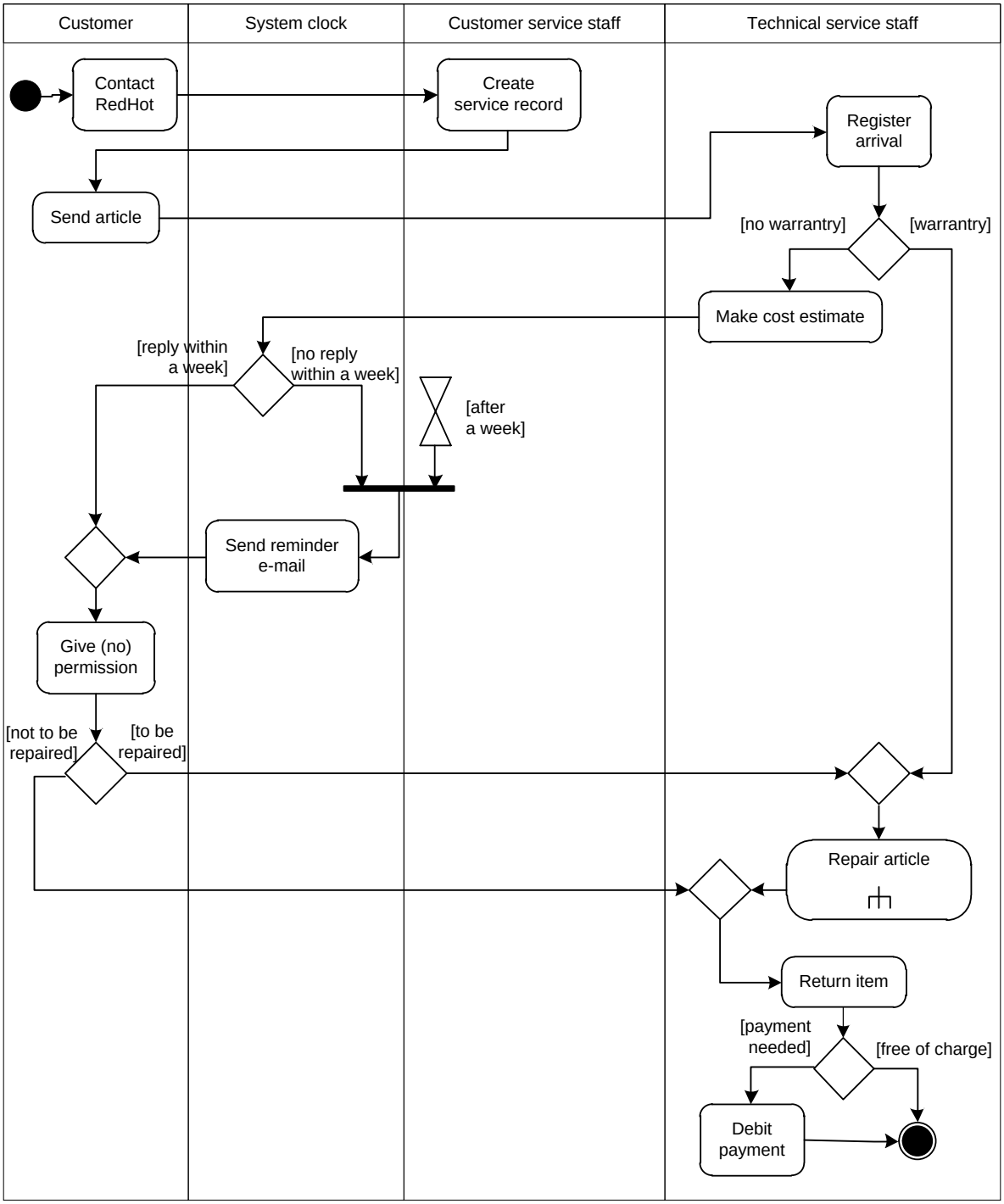
Design test 3 December 2014 – Solutions

1. Activity diagram

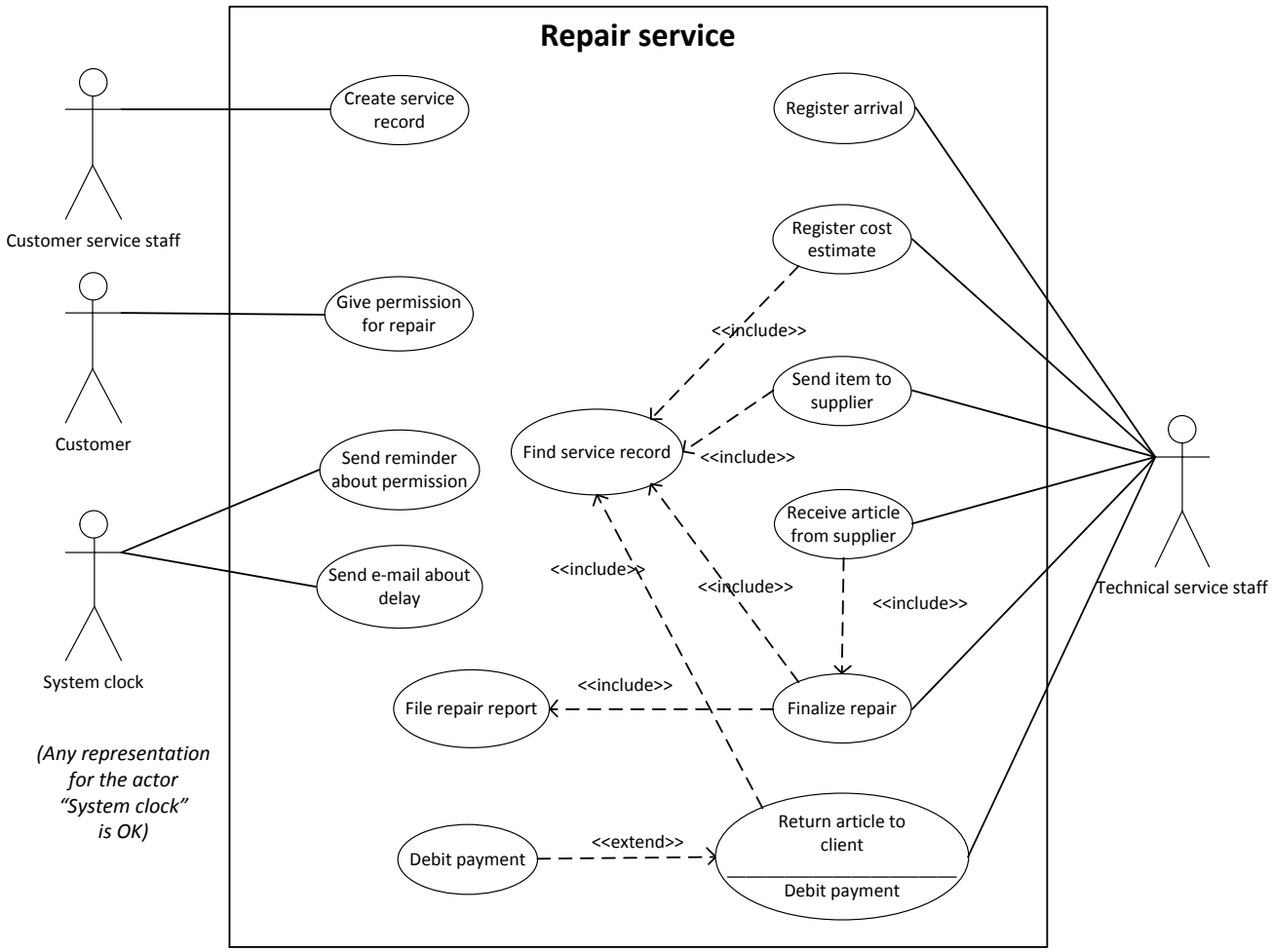


- The activity "make cost estimate" induces an automatic e-mail to the client (therefore, sending the e-mail isn't represented by a separate activity – but if you would have done that, this is OK).
  When the customer doesn't respond within a week, a reminder is sent. If the customer doesn't respond again, this can be interpreted as an implicit response from the customer to cancel the repair – so the text said. This means there is always a response, and we don't need another clock, etc.

2 Actor list and Use Case Diagram

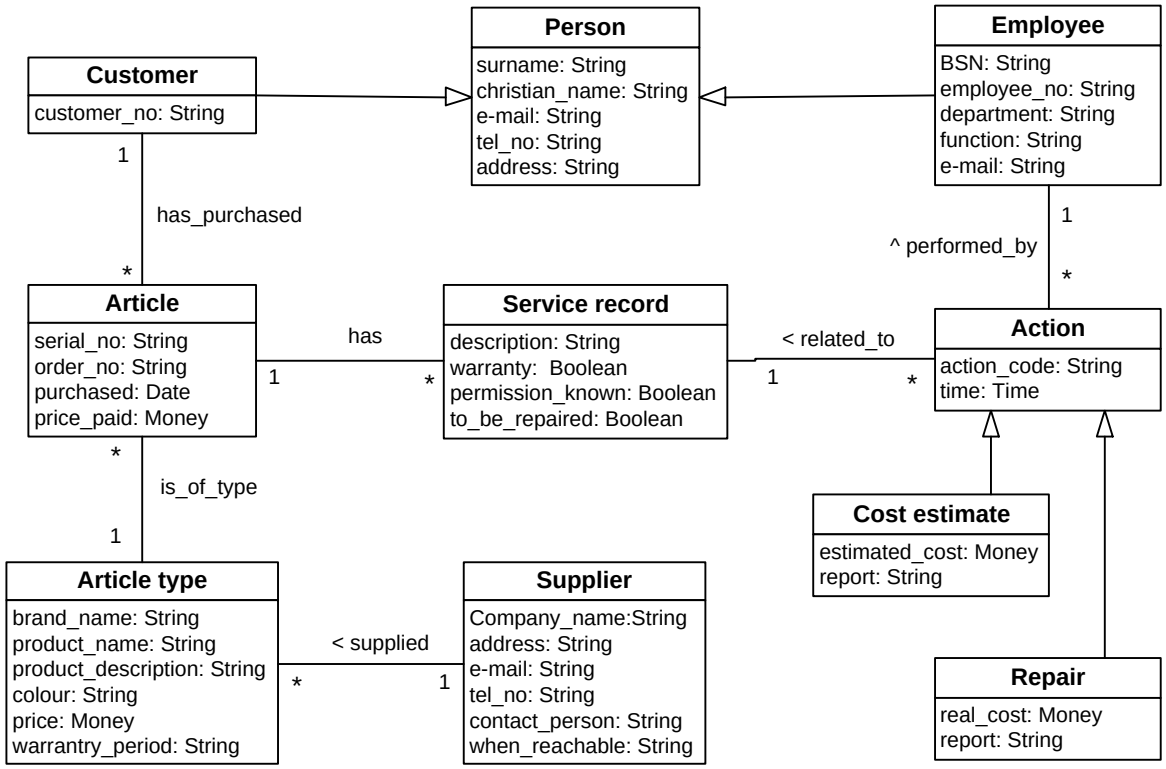
Actor	***Description***
Customer	Person who has purchased an article from RedHot (and wants it to be repaired)
Customer Service staff	First contact point for customers who want a repair; initiates the repair process.
Technical Service staff	Carries out the repair, possibly by sending it back to the suppliers
System clock	Triggers automatic e-mails after a specified period

The actor list should **not** contain the supplier, as the supplier has no direct contact with the system. See the use case diagram below. That the system clock is an actor can be inferred from Figure 2 in the test.



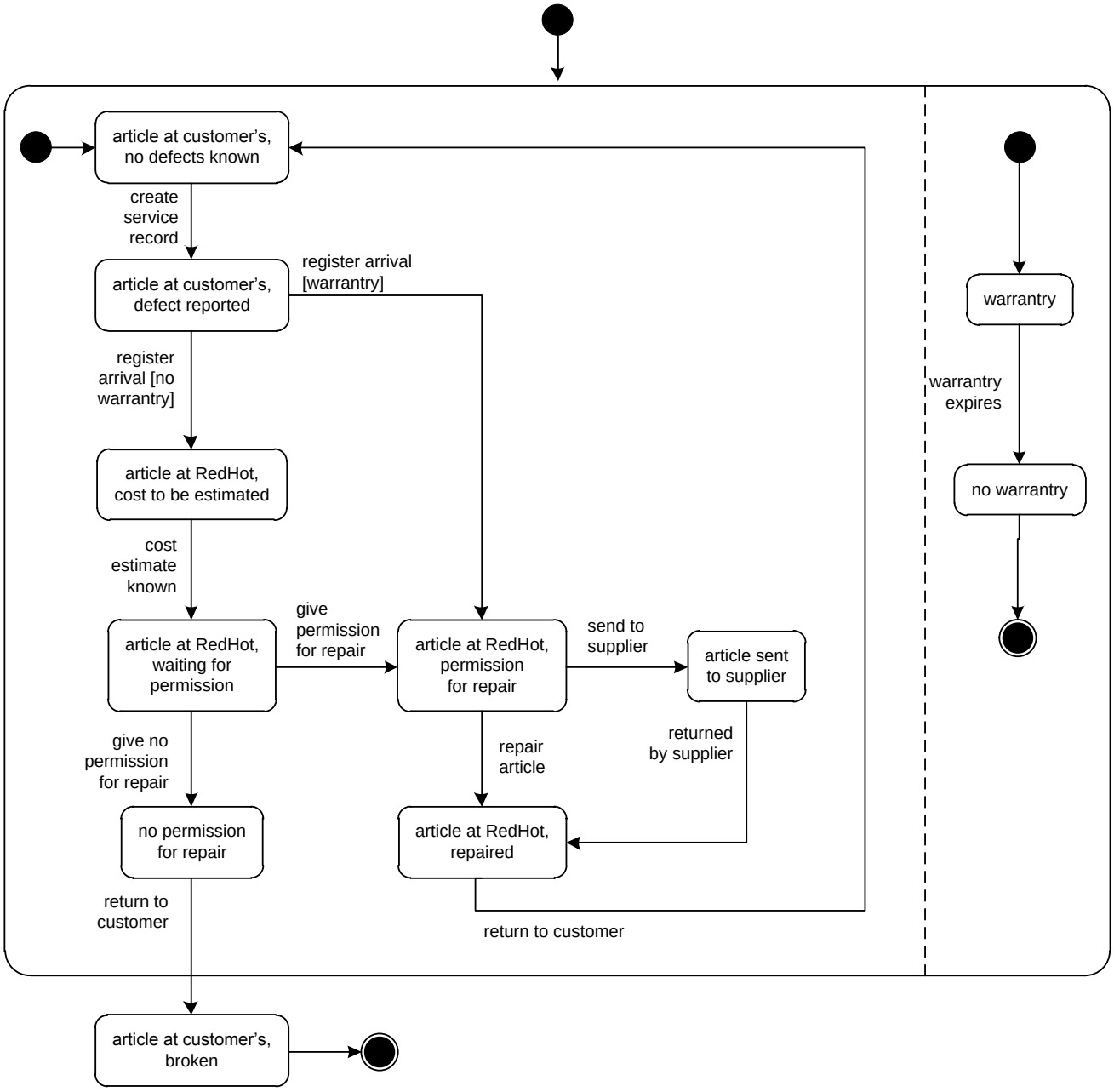
- Any action on the service record could include "find service record" as a use case – as was hinted at in the text. (Costs only very few point if you don't have it.)
  The idea is that the Technical Service employee first has to interact with the system to get the right record before they can add information to that record. (But for "Register arrival" the text explains that it is automatically linked to the right service record).
- "Return article to client" has an extension "Debit payment", because this is done only for repairs without warranty.
- The system clock is the actor for e-mails that are sent when a particular time interval has finished. (UML provides a different notation for non-person actors, but this is not in the slides nor in the book).

2. Class diagram



- Generalization of "Employee" and "Customer" to "Person" should be obvious.
- Generalization of "Cost estimate" and "Repair" to "Action" simplifies the model. "Cost estimate" and "Repair" both need associations with "Service record" and "Employee". The same holds for *any* action related to the service record that an employee carries out. "Cost estimate" and "Repair" can be regarded as actions with additional attributes, hence subclasses.
- "Article type" has not been explicitly mentioned in the text, but from the lecture and the lab session you should have learned that in natural language a homonym is used to refer to both. It is obvious that e.g. "product name" is an attribute of the article type, and "serial number" is an attribute of the individual article.

4. State machine



- It is tempting to add a transition from "article at customer's, no defects known" to a state where the article is thrown away by the customer, when it is no longer used, or perhaps broken and not repaired.
  This is what happens in reality. But the state machine only models states as they are known inside the system. Customers will not notify RedHot when they throw away articles, so all articles (except those for which a repair is cancelled) at some point will stay forever in "article at customer's, no defects known".
- "Waiting for permission" could be split into two states, if we strictly follow the use cases. Sending the reminder email does not change anything to the state (other than that a reminder has been sent). So nothing is gained by adding a state and a few transitions. (But if you have added it, fine.)
- Similarly, the email after 14 days (which could be modelled as a transition from "article sent to supplier" to "article sent to supplier") has not been included.