Solution Exam Software Engineering (400071)
27 March 2009

Part of this exam is based on the following case study[1]:

A software house wants to develop an application for planning and monitoring its software development projects.
This application must be integrated with an existing database component, which already stores all information about the employees (including professional roles like programmer, analyst and software architect) and about the projects (both past, currently running and planned).
For the **planning part**, the application must allow project managers to open a new project, define the development activities, their timing dependencies (e.g. 'prototyping' followed by 'end-user validation'), their planned inputs and outputs, and their individual duration. He or she must be able to search for employees according to their role and availability, assign employees to activities, and start the project when fully defined. Employees only have access to the activities they are assigned to, and can indicate if a certain activity is actually started, ongoing and eventually completed. He or she can also download the inputs needed, and upload the resulting outputs.
For the **monitoring part**, project managers can select the projects he or she is responsible for, visualize their status, and eventually modify any part of the project (e.g. reschedule an activity, assign different employees, add/remove inputs or outputs). The employees responsible for individual activities can open the project and visualize the status of their own activities, and of other activities directly related with their own.
The application must be available on-line to any previously authenticated person.

Questions about the theory

1. Draw the V-model representing one of the software life cycles we studied. What specific aspects of the software life cycle does it emphasize? [0.5 point]
It illustrates the different types of "testing" (V&V) and their link to phases
2. Select one requirements elicitation technique. Give a concise description of it. [1 point]

3. Give the definitions of 'procedural abstraction' and 'data abstraction' in software design. Explain how using one or the other influences the structure of the resulting design solution. [1 point]
... data abstraction usually leads to more compact design solutions (data entities are less in number than functionalities). 0.25x 2 definitions, 0.5 influences
4. Indicate which of the following statements about requirements engineering is true [0.5 point] (select one or more answers from the following possible answers):

- 4.a) Negotiation of the requirements with the various stakeholders is one of the four processes that make up requirements engineering.
- 4.b) Requirements engineering (including all its processes) typically consumes about 20% of the total effort spent for the whole software development process.
- 4.c) In the so-called "20-40-20 rule" about the relative effort spent on the various development activities, requirements engineering alone consumes about 40% of the total effort.

5. Indicate which of the following statements about design documentation is true? [0.5 point] (select one answer from the following possible answers):

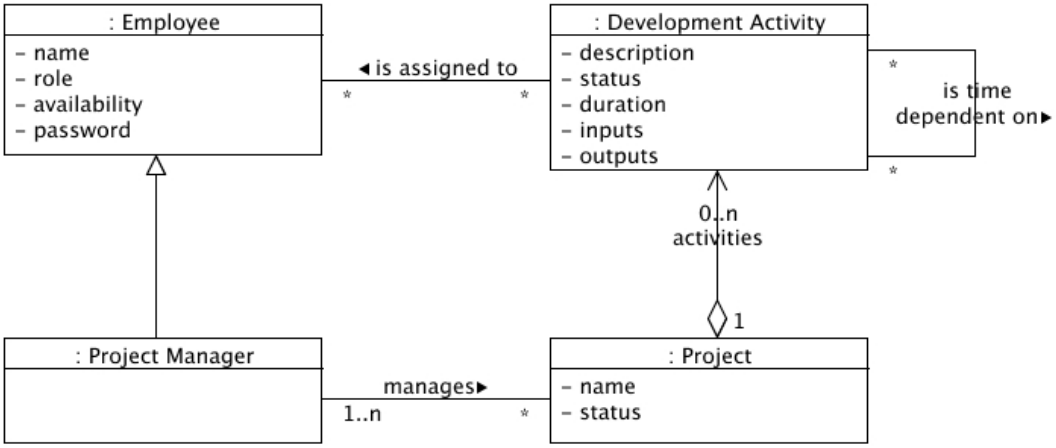
- 5.a) The goal of design documentation is to describe only what the targeted documentation readers need to know.
- 5.b) The goal of design documentation is to describe everything about the developed design solution.
- 5.c) The goal of design documentation is to describe only how the functional requirements have been fulfilled in a design solution.

6. Explain why having a good oracle is important in software testing. In your answer also describe the role that the oracle has in the testing process. [1 point]
The oracle is a program or other entity that you assume produces the correct output given a certain input. One compares the output of the oracle to the output of the test subject P. If the output of P disagrees with the output of the oracle we assume P to have produced the incorrect output. We need to be able to trust the output of the oracle otherwise we might look for a bug in P while we should be looking for a bug in the oracle. This is why having a good oracle is important.

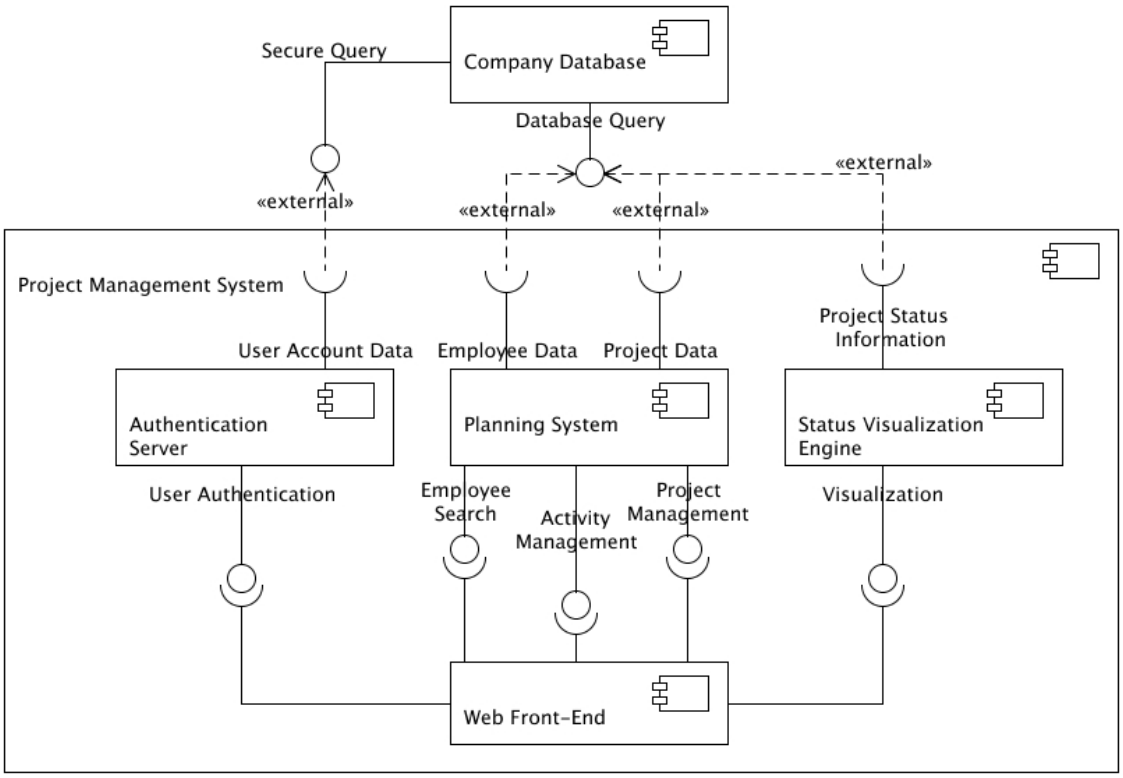
Questions related to the case study

7. Use a UML class diagram to describe the data model for the complete case study. [2 points].

1 This case study description is also translated in Dutch in the end of this exam text.



8. Use a UML component diagram to describe a possible design solution for the complete case study. Define in a clear way each operation offered by the interface(s) of all components. If needed, use additional text. [2.5 points].



9. Classify the components of your design solution as: (a) user interface management, (b) data management, (c) elaboration/business logics, (d) general mechanisms (like distributed communication, security, third-party interaction management). Explain for each component the rationale for your classification. [1 point]

Component 'Authentication server' is a general mechanism (can be reused across different domains/applications, provides generic security mechanisms).
Components 'Status Visualization engine' and 'Planning system' are both elaboration/business logics mechanisms, because they handle all the application-specific functionalities.
Component 'Company database' is clearly a data management mechanism (it manages all the data about employees and projects).
Component 'Web Front-End' is a user interface management mechanism, because it implements the presentation to and interaction with the user. If processing some application-specific logics, it could be classified as elaboration/business logics mechanism, too, but at this stage it is not clear yet.

Exam rules:
–No books or reference material.
–No calculator, mobile phones or other electronic device.

Translation of the “Case study description” in Dutch:

Een softwarebedrijf wil een applicatie ontwikkelen om haar software ontwikkelingsprojecten te plannen en de voortgang ervan te volgen.
De applicatie moet gebruik maken van een bestaande database component, waarin reeds informatie is opgeslagen over de medewerkers (inclusief hun functies zoals programmeur, analist en software architect) en de projecten (dit zijn zowel pas geplande projecten als projecten uit het heden en verleden).
In het **planningsonderdeel** van het systeem moeten project managers nieuwe projecten kunnen aanmaken en voor deze projecten ontwikkelactiviteiten kunnen definiëren. Binnen deze activiteiten kunnen tijdsafhankelijkheden (bijv. 'prototyping' wordt gevolgd door 'end-user validation'), geplande invoer en uitvoer (o.a. documenten) en de geplande duur worden gespecificeerd. Bovendien kan de project manager met behulp van de applicatie zoeken naar medewerkers op basis van functie en beschikbaarheid; activiteiten toewijzen aan de medewerkers en het project starten zodra het volledig gedefinieerd is. Medewerkers hebben alleen toegang tot activiteiten die aan hen zijn toegewezen en kunnen daarin aangeven of de activiteit net gestart, actief of voltooid is. De medewerker kan ook de benodigde invoer downloaden en later de voltooide uitvoer uploaden.
In het **voortgangsvolgsysteem** van de applicatie kan een projectmanager projecten waarvoor hij of zij verantwoordelijk is selecteren en een visualisatie van hun status opvragen. Daarnaast hebben de project managers de mogelijkheid het project of een onderdeel ervan aan te passen (bijv. de planning van een activiteit wijzigen, andere medewerkers toewijzen aan een activiteit of in- en/of uitvoer toevoegen en verwijderen). De medewerkers kunnen de projecten waarin zij actief zijn openen en de status van hun eigen en direct daaraan gekoppelde activiteiten laten visualiseren.
De applicatie moet on-line beschikbaar zijn voor geauthenticeerde gebruikers.

N.B.: deze probleemomschrijving is mogelijk ambigue en/of onvolledig. Bij het beantwoorden van de vragen mag je aanvullingen doen (indien nodig) en wordt je verzocht eventuele aannames kort te motiveren.