TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica **TU/e**

Partial examination 2YI45 (Software Architecture) on Monday 10 November 2008, 10.30h–12.00h

Work clearly. Read the entire exam before you start. Motivate each answer concisely and to the point. Maximal scores per question are given between parentheses. The maximum total score is 20 points on 11 questions.

We provide a set of hints for answering the questions. These are not the only correct answers, and in most cases they are not fully written out.

1. (2) Motivate software architecture based on the economy of defects.

   Hint See slide 13 of Lecture 6. Compare development with and without designing and documenting the architecture. The cost of a defect increases with the duration between injection and detection. Defects detected (in requirements) or prevented (in the implementation) during architectural design (e.g. due to earlier or improved verifiability) give rise to high savings. Architectural design should facilitate analysis and measurements, allowing early conformance checking of a system against its requirements. Moreover, architectural design is expected to reduce complexity and to improve maintainability of the system, resulting in fewer defects.

   Discovering 'architectural' defects during implementation can be very costly. By writing down an architectural design and verifying it (instead of forgoing that), such defects need not reach the implementation phase.

   N.B. Relationship to software architecture must be made explicit, in particular, architectural design as a phase after requirements definition and before implementation. It concerns early detection of defects in requirements and prevention of defects in implementation.

2. (2) What do the metrics *Coupling* and *Cohesion* mean, and what is their role in architectural design?

   Hint These are metrics on a modular design.

   Coupling: number and nature of dependencies between components (entities)

   Cohesion: internal consistency of components (to what extent things in one entity are related)

   Role: quality metrics for architure, that quantify modifiability and verifiability of the system

3. (1) Explain why the following statement is a misconception: "Usability of a software product is not an architectural concern, because the user interface can be encapsulated in a separate module."

   Hint Usability is not only a matter of an external user interface for human users. There are also various cross-cutting issues that concern usability, such as system-wide consistent undo and cancel operations, and internationalization.

   Designing an extensible user interface through plug-ins is an architectural concern.

   N.B. Usability and utility (available functionality) are orthogonal concerns. Performance is also orthogonal.

4. (2) Explain the need for *Module Architecture Control*.

   Hint See Lecture 7 *Module Architecture Control using Relation Algebra*. The software architecture may be viewed as both a constraint on as well as documentation of a software system. The module architecture belongs to the development view, and is meant for development and maintenance, amongst others. When the "intended" module architecture and "derived" module architecture deviate, the system as implemented no longer satisfies the (original) architectural constraints. Moreover, the module architecture only partially serves as documentation, thereby complicating development and maintenance.

5. (1) What architectural aspects affect performance?

   Hint Distributed resource management (sharing), communication (volume, mapping), distribution of functionality

   N.B. The question is not about performance metrics, or other qualities that have trade-offs with performance.

6. (2) Present a general and a specific scalability requirement in the form of a *Quality Attribute Scenario*.

   Hint Example of a general scalability quality attribute scenario:

   Source system owner

   Stimulus request to accommodate more concurrent users (usage parameter)

   Artifact the system, incl. computing platforms

   Environment normal operation, design time, run time

   Response add extra memory to servers, add extra servers (architectural parameters)

   Response measure cost of additional hardware, change in performance

   A concrete scalability quality attribute scenario:

   Source system owner

   Stimulus request to accommodate five times more concurrent users

   Artifact the main server cluster

   Environment normal operation

   Response increase the number of servers no more than sixfold, without recompiling the software

   Response measure performance as measured by average number of typical requests processed per minute may not drop more than 10%

   Also see slide 8 of ADS Architecture Lecture (Block A). Do not confuse this with a (plain) performance QAS.

7. (2) What is the ATAM? Give three benefits of applying the ATAM, and also two weaknesses.

   Hint ATAM stands for Architecture Trade-off Analysis Method. It is a way of organizing and carrying out a qualitative evaluation of architural designs for software-intensive systems.

   Benefits, weaknesses: See ATAM slides 38–39 of Lecture 9.

   N.B. Cost and required experience/preparation are not ATAM weaknesses. These apply to any kind of evaluation method. Furthermore, it is the balance that matters: do the benefits outweigh the costs?

8. (2) In the context of ATAM, what is a *Utility Tree* and how is it used?

   Hint See ATAM slides 20–21 of Lecture 9.

9. (2) Discuss the similarities and differences between a *Component Model* and an architectural design description.

   Hint Similarities: interface definitions, components, composition

   Differences: component model is generic, involves late binding (after component production), aims at reuse as-is in multiple, not necessarily similar, systems; architectural design description is system specific, may involve design-time and compile-time binding

10. (2) Explain at least two benefits of *Component-Based Development* (CBD) over more traditional architectural development approaches. Give two examples of how CBD could negatively affect the quality of the final product.

    Hint See slide 15 of CBSE Lecture 9: productivity, quality, time-to-market, maintenance

    More generic components with run-time binding and language-independent interfaces can be slower than more specialized components with compile-time binding.

    If appropriate components are lacking, a workaround solution can be overly complex.

    More generic interfaces allow less static checking and hence are more error-prone.

    Using a third-party component reduces the control you have over its design and implementation.

    N.B. Using high-quality components does not necessarily lead to a high-quality system.

    N.B. We are looking for CBD-specific drawbacks (e.g. a low-quality component is also possible in a traditional approach).

11. (2) Describe general steps to extract architectural information from a given source code base, and indicate what information can be obtained in that way. Give at least two reasons why such reverse engineering would be interesting.

    Hint See slides of Lecture 10 (p. 9, 10): select desired models, classify nature of given code; extraction steps: Code $\rightarrow$ Data $\rightarrow$ Model $\rightarrow$ Information

    Models for each of Kruchten's 4+1 views can be extracted this way. Typically, logical view and process view, incl. execution scenarios, are addressed. A deployment view may be harder to extract.

    Interesting (a) because architectural documentation may be lacking, and is needed for maintenance; (b) in order to check whether an implementation-under-construction adheres to architectural decisions (cf. question 4); possibly also (c) to obtain quality metrics (especially when the architectural models are informal).