TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica

Examination Architecture of Distributed Systems (2XI45)
on Friday October 29, 2010, 14.00h-15.30h

Work clearly. Read the entire exam before you start. Motivate each answer concisely and to-the-point. Maximal grades are given between parentheses. The total score sums up to 20 points.

1. Define / describe the following terms:
   a. (1) Architecture
      The fundamental organization of a system embodied by its components, their relationships to each other and to the environment and the principles guiding its design and evolution.
   b. (1) Architectural Description
      A collection of models of the system, organized in views. This includes models describing assumptions about the environment and models describing interactions
   c. (1) Architectural View
      A collection of models that address one or more concerns of a stakeholder; the view conforms to a viewpoint.
2. (2) Define what it means that a system is *scalable*.
   Given is a scaling parameter k (giving different scales at which the system can be considered), a metric $m(k)$ and a scalability criterion $S(k)$. The system is called scalable (with respect to m and S) if $m(k)/m(1) >= S(k)$ (or $m(1)/m(k) >= S(k)$, if m is decreasing).
   Alternatively, as a special case of this, we call the scalability of a system the extent in which architecture parameters can and must be adjusted to accommodate changes in usage parameters in order to sustain a given metric.
3. (1+1 corrected) Explain what Amdahl's law expresses and give an example.
   In the general formulation, Amdahl law expresses that improvements that address only part of an overall cost function can only cut down the cost function for that part and, hence, the benefit of improvements quickly becomes negligible. In addition, the overall improvement is bounded.
   For the case of speedup of a parallel computation: if the scalability metric is the time, then typically
   ($m(1,N) = $) $T(1,N) = seq(N) + par(N)$, hence $T(P,N) = seq(N) + par(N)/P$
   The speedup: $S(P,N) <= 1 + par(N)/seq(N)$
   Example: matrix multiplication on a shared bus gives
   $T(P,N) = (2fN^3)/P + (cN^2)$
   The latter term represents $seq(N)$.
4. (2) Describe similarities and differences of the Batch sequential and the Pipes & filter architectural styles (in total, at least three points).
   (*see Architectural Patterns Revisited – A Pattern Language*)
   Similarities:
   a. patterns are intended for data flow applications
   b. data driven structure of several sequential operations over the input data
   c. operations are implement in independent, standalone components (or stages) that do not share data.
   d. reconfigurability and modularity - dependence on neighboring stages only.

Differences

a. BS performs stages sequentially and completes each operation before starting the next one while in P&F all operations are (can be) in progress on the same input data. (Discrete steps versus constant flow.)
b. In F&S the connectors ('pipes') are explicitly part of the pattern (typically buffers, FIFOs).
c. F&S supports the creation of various structures using the (flexible) connectors, even feedback loops are allowed.

5. In the *Service Oriented Architecture* style there is a strong decoupling between functionality and implementation, throughout the application lifecycle. Applications are built by combining or connecting services. Explain why and how this architecture facilitates
   a. (1) reuse of existing software components;
   b. (1) independent development.

   Answer. In SOA, application components depend on each other only through the service interface. No details of an implementation may be used or be visible.

   (a) There are two elements of reuse. Once a service (interface) is defined, the only thing that needs to be done to use an existing piece of software is to map the service interface to this software, typically by building an adapter. Secondly, services can be reused in many different applications since they do not contain knowledge about their application context. One remark on the latter is in place: a service may use other services.

   (b) This relies on the same separation between service definition and implementation. By defining and using only the service interface, the implementation of the service becomes irrelevant for the service user. For example, by defining the UPnP interface, the implementation behind is it entirely invisible and can be developed by independent parties. One might also replace one implementation by the other.

6. Interactions and interaction styles have certain qualities (for example, whether a pending interaction is persistent).
   a. (1) discuss and explain at least 4 different quality aspects (quality categories) of interactions;

      Memory/storage

      i. *transient*: interaction requires sender and receiver to 'execute' at same time
      ii. *persistent*: interaction remains while sender and receiver disappear

      Synchronization

      iii. *asynchronous*: sender/caller does not wait or block;
      iv. *synchronous*: caller blocks till request acceptance
          1. several different synchronization points, see next slide
      v. *buffered*: limited difference between #calls and #responses

      Units of information:

      vi. *discrete*: structured unit, independent and complete
      vii. *continuous (streaming)*: basic units without message structure

      Connection

      viii. Connection oriented
      ix. Connection-less

      Reliability

      Time dependence

      x. temporal relationships typically with streaming
      xi. *synchronous*: bounded delay
      xii. *isochronous*: bound minimum and maximum delay (i.e., jitter)

b. (1) what are qualities of a *message queueing system* for these aspects?
persistent, asynchronous, (or: synchronized on handing off to the MQS) and buffered, discrete, reliable, connectionless, no temporal relationships
[one can argue somewhat on these points per queueing system; also, the queuing system itself is probably connection oriented; however, the sender/receiver pair is not]

7. (2) What is the difference between an object adapter and a proxy?
Proxy:
   - a component that acts on behalf of another component
   - capable of implementing filtering policies
   - typically, 'proxy' refers to a client side (representing the client); a '*reverse proxy*' is placed at a server side

   Object Adapter or Object Wrapper:
   - a component that relays calls to an object interface
   - typically implementing management policies for the object, e.g. creation policy, multi-threading, perhaps transient/persistent
   - converting between interfaces and possibly state holding.

8. (2) In the *Open Service Architecture for Sensors* (OSAS) system a number of styles are used, among which are the virtual machine style and the publish & subscribe style. Which extra-functional properties are achieved by this?
Answer. Note that there are general good properties of these two styles but the question is what they do for OSAS. Just mentioning a few ilities without explanation does not score points.
Virtual machine:
   a. portability (independence of OS or ISA)
   b. compact, domain-specific representation of programs
   c. flexibility / performance (of reprogramming): reprogrammable over the air in combination with content-based addressing.
   d. interoperability
   e. protection of the underlying system (controlled interpreter)

   Publish & Subscribe
   a. loose coupling: sender and receiver do not 'know' each other
   b. run-time changes of sender/receiver pairs
   c. allow third-party control [can send commands to couple senders and receivers]
   d. avoid polling: move communication triggering to the publisher.

   These are aspects of ease of deployment, dynamic change, performance.
   Maintainability also counts but less since the P and the S are mostly part of the same program code. So, maintainability is more an issue of having a network program.

9. (2) Discuss the problem of service discovery and its solutions.
Service discovery: meeting of two parties (provider and service user) that don't know each other. Solution consists of query and service advertisement, propagation and matching. These phases can be pushed to the client, to the server, or meet in the middle at a repository.
      i. basic form (bootstrapping): immediate, via broadcast or multicast (server initiated, client initiated, or both; matching done by either.
      ii. via a repository (which then must be known, or resolved via a resolution procedure that needs a closure)
      iii. via a known service access point

10. (1) Explain how service discovery (mail server, http server) is done within DNS.
    Discovery of a mail server for a domain goes through asking the MX query to the DNS server for that domain (in the general service case an SRV query is asked) that will return the hostname. This is further similar to asking an IP address of any host in that domain. For example, resolving the name of the mailserver of justaname@anywhere.nl:
    a. local DNS stub (of your SMTP server) asks MX? of anywhere.nl through a recursive query to its local nameserver;
    b. this local nameserver finds the name server for **anywhere.nl** through iterative queries, starting at the root servers;
    c. then it asks the anywhere.nl nameserver the MX? query and returns the result to the stub;
    d. through a similar procedure, the name of this MX server is resolved into the IP address of this server through an A query.
    e. Of course, caching occurs everywhere.