

Comments Question 1

The question examines the fundamentals of the course's curriculum and most students managed to answer it reasonably well. The main weakness is a tendency not to explain things as required by the question. For example, it is not enough to say that the idealized component life cycle consists of three phases. The meaning of these phases should also be explained. In addition, the fact that composition is possible in both the design and deployment phases is an important characteristic.

Question 2

This question examines the principles of encapsulated components and the details of the X-MAN component model. Most students gave a reasonably good answer, except for data channels. In the vending machine example, many students do not use the correct syntax. In X-MAN, connectors always form a hierarchy on top of (atomic) components; they cannot also appear below the components. Incorrect syntax results in wrong semantics and therefore a wrong system.

Question 3

This question examines the principles of objects as components and the details of EJBs. Most students answered the question reasonably well. A common mistake is not to explain the elements of EJB accurately. For example, to say that components in EJB are EJBs (or that components in JavaBeans are beans) is not meaningful at all. Also, many students did not explain the different kinds of EJBs: session, entity and message beans. The diagrams for the vending machine example were not good in general, often mistakenly showing EJBs as architectural units, instead of classes with methods.

Question 4

This question examines the principles of architectural units as components and the details of UML2.0. Again, most students answered the question reasonably well. One mistake is to define an UML2.0 component as a "modular unit ...". Such a definition is not specific to UML2.0 (even though UML gives this definition in UML1.0); neither is it meaningful since more precisely a UML2.0 component is an architectural unit. A common mistake in the vending machine example is not to have an interface (with required services) for coin insertion and drink selection.
