

Composition of Component Services

H. Kopetz
Technical University of Vienna
hk@vmars.tuwien.ac.at

Extended Abstract

With the increase in the size of a computer applications, a sequence of more abstract concepts to express the computational intentions of a user have been introduced over the years. At the beginning of the computer era the service of a computer was specified by presenting a list of *machine instructions* that must be executed by the physical machine[1]. To be able to abstract from the detailed hardware features of a particular machine, *assemblers* were introduced in the next phase to specify the instruction for a more general virtual machine. In the following years *assemblers* were replaced by *compilers* that translate a higher level representation of an algorithm into a form that can be executed by the computer hardware. *Objects* that encapsulate a computational method and the associated state into a single item were the next units of abstractions. At present we are introducing the notion of a *component* as a still higher unit of abstraction in the design of large computer systems and are investigating the issues related to the composition of sets of components.

In this short contribution we consider a *component* to be an encapsulated hardware-software unit that can be used as a building block in the construction of a larger system and that delivers a specified service across a component interface. The problem of composition of a set of components can be broken down into the following two sub-problems, the composition of the component services and the incorporation of the components into an integrated software/hardware architecture. In this contribution we focus on the problem of component service composition.

In a distributed system the *service* of such a component consists of the output messages that the component produces at the considered interface as a consequence of previous input messages and the progression of physical time. In a large system, the services of many components are integrated in order to generate, in addition to the existing (*prior*) services of the components, new *emergent* services. Such a composition of component services is simplified, if the set of components support the property of *compositionality*: that the emergent services of the system of components are determined by the meaning of the constituting component services and the dynamics of the interconnection.

We are not in the position to provide *sufficient* conditions for the *compositionality* for a set of components of a real-

time computer application. However, we can formulate an (incomplete) set of necessary conditions [2], such as

Independent development of the components: The independent development of a component requires a precise specification of the component interfaces, both in the domains of time and value. The meaning of the information processing that is performed by the component must be captured by an appropriate interface model.

Stability of prior services: The properties of the services of the components that were available prior to the composition must remain stable and may not be modified by the composition.

Performability of the communication service: The communication system that transports the message among the components must be performant, i.e., must meet the required temporal properties.

Interaction compatibility: The protocols that control the interactions and the data items that are exchanged among the components must be compatible with each other. If there is a *property mismatch* at the interface level, a connection system must be introduced to resolve this property mismatch[3].

Diagnosability: One of the motivations for building a system out of components is the possibility to replace a defect component, and not the complete system, in case of a fault. This requires that in case of a failure of the emergent service it must be possible to locate the component that is responsible for the failure.

Replica Determinism: In a system where fault-tolerance is achieved by replication and voting, the replicated component must support replica determinism.

We call a component framework that supports the enlisted necessary conditions for compositionality *composable*.

1. Kopetz, H. *The Future of Autonomous Decentralized Systems*. in *ISADS 2003*. 2003. Pisa: IEEE Press.
2. Kopetz, H. and N. Suri. *Compositional Design of Real-Time System: A Conceptual Basis for the Specification of Linking Interfaces*. in *ISORC 2003--The 6th International Symposium on Object Oriented Real-Time Computing*. 2003. Hakodate, Japan: IEEE Press.
3. Jones, C., et al., *DSOS Conceptual Model*. 2003: University of Newcastle upon Tyne, Techn. Report CS-TR-782, TU Vienna, Technical Report 54/2002, QinetiQ Technical Report TR030434, LAAS Technical Report. p. 1-122.