



The Unified Modeling Language

The Unified Modeling Language is a set of 13 different diagram types that may be used to model software systems. It emerged from work in the 1990s on object-oriented modeling where similar object-oriented notations were integrated to create the UML. A major revision (UML 2) was finalized in 2004. The UML is universally accepted as the standard approach for developing models of software systems. Variants have been proposed for more general system modeling.

<http://www.SoftwareEngineering-9.com/Web/UML/>

In the third case, where models are used as part of a model-based development process, the system models have to be both complete and correct. The reason for this is that they are used as a basis for generating the source code of the system. Therefore, you have to be very careful not to confuse similar symbols, such as stick and block arrowheads, that have different meanings.

5.1 Context models

At an early stage in the specification of a system, you should decide on the system boundaries. This involves working with system stakeholders to decide what functionality should be included in the system and what is provided by the system's environment. You may decide that automated support for some business processes should be implemented but others should be manual processes or supported by different systems. You should look at possible overlaps in functionality with existing systems and decide where new functionality should be implemented. These decisions should be made early in the process to limit the system costs and the time needed for understanding the system requirements and design.

In some cases, the boundary between a system and its environment is relatively clear. For example, where an automated system is replacing an existing manual or computerized system, the environment of the new system is usually the same as the existing system's environment. In other cases, there is more flexibility, and you decide what constitutes the boundary between the system and its environment during the requirements engineering process.

For example, say you are developing the specification for the patient information system for mental healthcare. This system is intended to manage information about patients attending mental health clinics and the treatments that have been prescribed. In developing the specification for this system, you have to decide whether the system should focus exclusively on collecting information about consultations (using other systems to collect personal information about patients) or whether it should also collect personal patient information. The advantage of relying on other systems for patient information is that you avoid duplicating data. The major disadvantage, however, is that using other systems may make it slower to access information. If these systems are unavailable, then the MHC-PMS cannot be used.

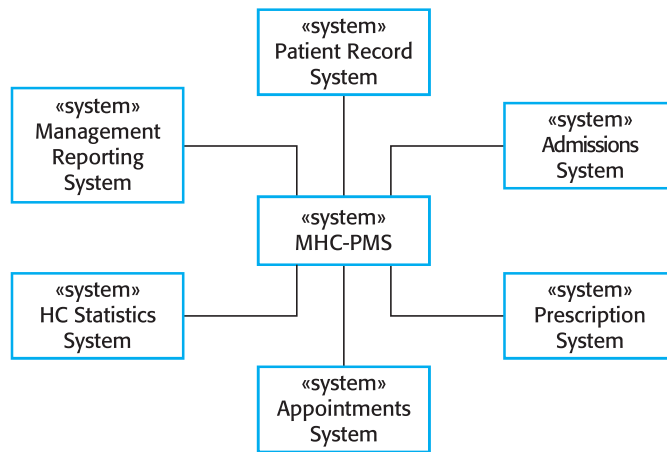


Figure 5.1 The context of the MHC-PMS

The definition of a system boundary is not a value-free judgment. Social and organizational concerns may mean that the position of a system boundary may be determined by non-technical factors. For example, a system boundary may be deliberately positioned so that the analysis process can all be carried out on one site; it may be chosen so that a particularly difficult manager need not be consulted; it may be positioned so that the system cost is increased and the system development division must therefore expand to design and implement the system.

Once some decisions on the boundaries of the system have been made, part of the analysis activity is the definition of that context and the dependencies that a system has on its environment. Normally, producing a simple architectural model is the first step in this activity.

Figure 5.1 is a simple context model that shows the patient information system and the other systems in its environment. From Figure 5.1, you can see that the MHC-PMS is connected to an appointments system and a more general patient record system with which it shares data. The system is also connected to systems for management reporting and hospital bed allocation and a statistics system that collects information for research. Finally, it makes use of a prescription system to generate prescriptions for patients' medication.

Context models normally show that the environment includes several other automated systems. However, they do not show the types of relationships between the systems in the environment and the system that is being specified. External systems might produce data for or consume data from the system. They might share data with the system, or they might be connected directly, through a network or not connected at all. They might be physically co-located or located in separate buildings. All of these relations may affect the requirements and design of the system being defined and must be taken into account.

Therefore, simple context models are used along with other models, such as business process models. These describe human and automated processes in which particular software systems are used.

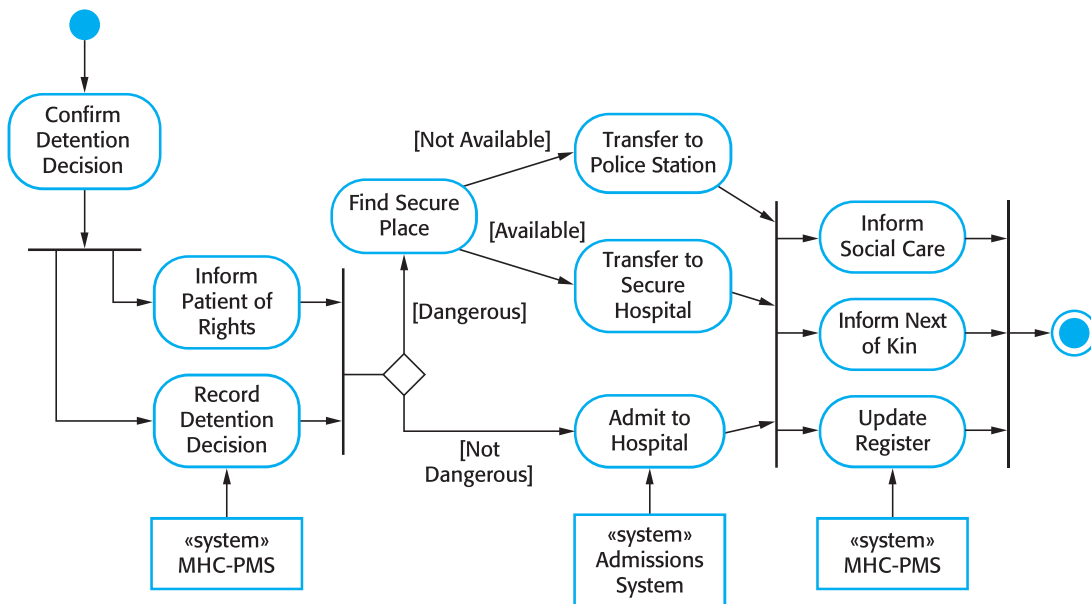


Figure 5.2 Process model of involuntary detention

Figure 5.2 is a model of an important system process that shows the processes in which the MHC-PMS is used. Sometimes, patients who are suffering from mental health problems may be a danger to others or to themselves. They may therefore have to be detained against their will in a hospital so that treatment can be administered. Such detention is subject to strict legal safeguards—for example, the decision to detain a patient must be regularly reviewed so that people are not held indefinitely without good reason. One of the functions of the MHC-PMS is to ensure that such safeguards are implemented.

Figure 5.2 is a UML activity diagram. Activity diagrams are intended to show the activities that make up a system process and the flow of control from one activity to another. The start of a process is indicated by a filled circle; the end by a filled circle inside another circle. Rectangles with round corners represent activities, that is, the specific sub-processes that must be carried out. You may include objects in activity charts. In Figure 5.2, I have shown the systems that are used to support different processes. I have indicated that these are separate systems using the UML stereotype feature.

In a UML activity diagram, arrows represent the flow of work from one activity to another. A solid bar is used to indicate activity coordination. When the flow from more than one activity leads to a solid bar then all of these activities must be complete before progress is possible. When the flow from a solid bar leads to a number of activities, these may be executed in parallel. Therefore, in Figure 5.2, the activities to inform social care and the patient's next of kin, and to update the detention register may be concurrent.

Arrows may be annotated with guards that indicate the condition when that flow is taken. In Figure 5.2, you can see guards showing the flows for patients who are