

A scenario is an instance of a use case; that is, a use case specifies all possible scenarios for a given piece of functionality. A use case is initiated by an actor. After its initiation, a use case may interact with other actors, as well. A use case represents a complete flow of events through the system in the sense that it describes a series of related interactions that result from its initiation.

Figure 4-7 depicts the use case ReportEmergency of which the scenario warehouseOnFire (see Figure 4-6) is an instance. The FieldOfficer actor initiates this use case by activating the "Report Emergency" function of FRIEND. The use case completes when the FieldOfficer actor receives an acknowledgment that an incident has been created. The steps in the flow of events are indented to denote who initiates the step. Steps 1 and 3 are initiated by the actor, while steps

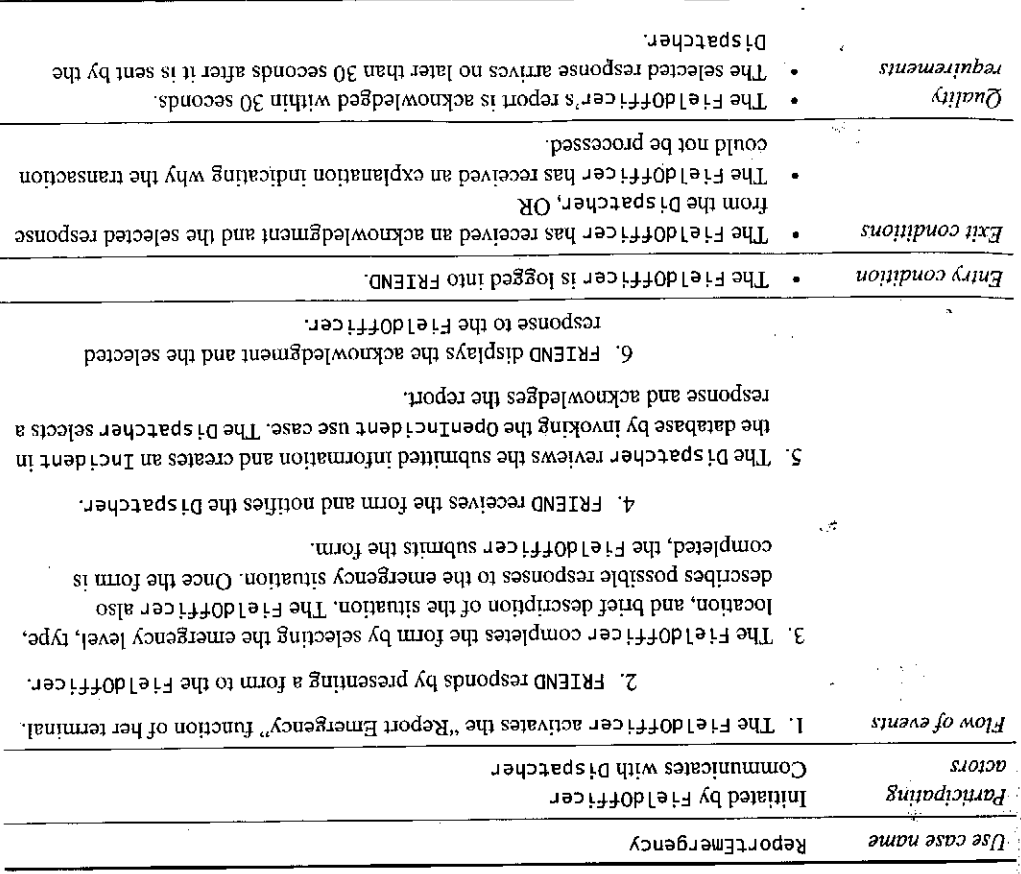


Figure 4-7 An example of a use case, ReportEmergency. Under ReportEmergency, the left column denotes actor actions, and the right column denotes system responses.

Requirements Elicitation Concepts

In this section, we describe the main requirements elicitation concepts used in this chapter. In particular, we describe

- Functional Requirements (Section 4.3.1)
- Nonfunctional Requirements (Section 4.3.2)
- Completeness, Consistency, Clarity, and Correctness (Section 4.3.3)
- Realism, Verifiability, and Tracability (Section 4.3.4)
- Greenfield Engineering, Reengineering, and Interface Engineering (Section 4.3.5).

We describe the requirements elicitation activities in Section 4.4.

3.1 Functional Requirements

Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts. For example, Figure 4-2 is an example of functional requirements for SatWatch, a watch that resets itself without user intervention:

SatWatch is a wrist watch that displays the time based on its current location. SatWatch uses GPS satellites (Global Positioning System) to determine its location and internal data structures to convert this location into a time zone.

The information stored in SatWatch and its accuracy measuring time is such that the watch owner never needs to reset the time. SatWatch adjusts the time and date displayed as the watch owner crosses time zones and political boundaries. For this reason, SatWatch has no buttons or controls available to the user. SatWatch determines its location using GPS satellites and, as such, suffers from the same limitations as all other GPS devices (e.g., inability to determine location at certain times of the day in mountainous regions). During blackout periods, SatWatch assumes that it does not cross a time zone or a political boundary. SatWatch corrects its time zone as soon as a blackout period ends.

SatWatch has a two-line display showing, on the top line, the time (hour, minute, second, time zone) and on the bottom line, the date (day, date, month, year). The display technology used is such that the watch owner can see the time and date even under poor light conditions.

When political boundaries change, the watch owner may upgrade the software of the watch using the WebifyWatch device (provided with the watch) and a personal computer connected to the Internet.

Figure 4-2 Functional requirements for SatWatch.

The above functional requirements focus only on the possible interactions between SatWatch and its external world (i.e., the watch owner, GPS, and WebifyWatch). The above description does not focus on any of the implementation details (e.g., processor, language, display technology).

4.3.2 Nonfunctional Requirements

Nonfunctional requirements describe aspects of the system that are not directly related to the functional behavior of the system. Nonfunctional requirements include a broad variety of requirements that apply to many different aspects of the system, from usability to performance. The FURPS+ model² used by the United Process [Jacobson et al., 1999] provides the following categories of nonfunctional requirements:

- **Usability** is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. Usability requirements include, for example, conventions adopted by the user interface, the scope of online help, and the level of user documentation. Often, clients address usability issues by requiring the developer to follow user interface guidelines on color schemes, logos, and fonts.
- **Reliability** is the ability of a system or component to perform its required functions under stated conditions for a specified period of time. Reliability requirements include, for example, an acceptable mean time to failure and the ability to detect specified faults, or to withstand specified security attacks. More recently, this category is often replaced by **dependability**, which is the property of a computer system such that reliance can justifiably be placed on the service it delivers. Dependability includes reliability, **robustness** (the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions), and **safety** (a measure of the absence of catastrophic consequences to the environment).
- **Performance** requirements are concerned with quantifiable attributes of the system, such as **response time** (how quickly the system reacts to a user input), **throughput** (how much work the system can accomplish within a specified amount of time), **availability** (the degree to which a system or component is operational and accessible when required for use), and **accuracy**.
- **Supportability** requirements are concerned with the ease of changes to the system after deployment, including for example, **adaptability** (the ability to change the system to deal with additional application domain concepts), **maintainability** (the ability to change the system to deal with new technology or to fix defects), and internationalization (the ability to change the system to deal with additional international conventions, such as languages, units, and number formats). The ISO 9126 standard on software quality [ISO Std. 9126], similar to the FURPS+ model, replaces this category with two categories: **maintainability** and **portability** (the ease with which a system or component can be transferred from one hardware or software environment to another).

² FURPS+ is an acronym using the first letter of the requirements categories: Functionality, Usability, Reliability, Performance, and Supportability. The + indicates the additional categories. The FURPS model was originally

Example questions

Category	Example questions
Usability	<ul style="list-style-type: none"> • What is the level of expertise of the user? • What user interface standards are familiar to the user? • What documentation should be provided to the user?
Reliability <i>(including robustness, safety, and security)</i>	<ul style="list-style-type: none"> • How reliable, available, and robust should the system be? • Is restarting the system acceptable in the event of a failure? • How much data can the system lose? • How should the system handle exceptions? • Are there safety requirements of the system? • Are there security requirements of the system?
Performance	<ul style="list-style-type: none"> • How responsive should the system be? • Are any user tasks time critical? • How many concurrent users should it support? • How large is a typical data store for comparable systems? • What is the worst latency that is acceptable to users?
Supportability <i>(including maintainability and portability)</i>	<ul style="list-style-type: none"> • What are the foreseen extensions to the system? • Who maintains the system? • Are there plans to port the system to different software or hardware environments?
Implementation	<ul style="list-style-type: none"> • Are there constraints on the hardware platform? • Are constraints imposed by the maintenance team? • Are constraints imposed by the testing team?
Interface	<ul style="list-style-type: none"> • Should the system interact with any existing systems? • How are data exported/imported into the system? • What standards in use by the client should be supported by the system?
Operation	<ul style="list-style-type: none"> • Who manages the running system?
Packaging	<ul style="list-style-type: none"> • Who installs the system? • How many installations are foreseen? • Are there time constraints on the installation?
Legal	<ul style="list-style-type: none"> • How should the system be licensed? • Are any liability issues associated with system failures? • Are any royalties or licensing fees incurred by using specific algorithms or components?

ents,
To
two
it is
l for
ding
, the
f the
er of
it to
must
. To
ex to
other
only
rains
their
term
es. A
n the
s are
rface
to its
non
ites