Chapter Six

Putting together requirements gathering team

Chapter Outline

- *Fundamental requirements gathering techniques*
- *Essential Use Case Modeling*
- *Essential User Interface Prototyping*
- *Domain modeling with class responsibility collaborator (CRC) cards*
- *Developing a supplementary Specification*
- *Identifying Change Cases*

Fundamental requirements gathering techniques

- ***Individually interview people*** *informed about the operation and issues of the current system and future systems needs.*
- ***Interview groups*** *of people with diverse needs to fid interactions and contrasts among system requirements*
- ***Observe*** *workers at selected times to see how data are handled and what information people need to do their jobs.*
- ***Study business documents/document analysis*** *to discover reported issues, policies, rules, and directions as well as concrete examples of the use of data and information in the org.*
- *What can the analysis of documents tell you about the requirements for a new system?*

Cntd…

- ***In documents you can find information about the following:***
  - *Problems with existing systems (e.g., missing information or redundant steps*
  - ***Opportunities to meet new needs** if only certain information or information processing were available (e.g., analysis of sales based on customer type)*
  - *Organizational direction that can influence information system requirements (e.g., trying to link customers and suppliers more closely to the organization)*
  - ***Titles and names of key individuals who have an interest in relevant existing systems** (e.g., the name of a sales manager who led a study of the buying behavior of key customers)*
  - ***Values of the organization or individuals who can help determine priorities for different capabilities desired by different users** (e.g., maintaining market share even if it means lower short-term profits*

An Overview of Requirements Elicitation

- ***Requirements elicitation* focuses on describing the purpose of the system.**
- *The **client**, the developers, and the users identify a problem area and define a system that addresses the problem.*
- *Such a definition is called a **requirements specification** and serves as a contract between the client and the developers.*
- ***The requirements specification** is structured and formalized during analysis to produce an analysis model.*
- *Both requirements specification and analysis model represent the same information.*
- ***They differ only** in the **language and notation they use**;*
  - ✓ *the **requirements specification** is written in **natural language**,*
  - ✓ *whereas the **analysis model** is usually expressed in a formal or semiformal notation.*

Cntd…

- ***The requirements specification*** *supports the communication with the client and users.*
- ***The analysis model supports the communication among developers.***
- ***Both models represent the same aspects*** *of the system, requirements elicitation and analysis occur concurrently and iteratively.*
- ***Requirements elicitation and analysis focus only on the user's view of the system.***
- *For example, the system functionality, the interaction between the user and the system, the errors that the system can detect and handle, and the environmental conditions in which the system functions are part of the requirements.*
- *The* ***system structure****, the implementation technology selected to build the system, the system design,* ***the development methodology****, and other* ***aspects not directly*** *visible to the user* ***are not part*** *of the requirements.*

Cntd…

Requirements elicitation includes the following activities:

- ***Identifying actors.*** *During this activity, developers identify the different types of users the future system will support.*
- ***Identifying scenarios.*** *During this activity, developers observe users and develop a set of detailed scenarios for typical functionality provided by the future system. Scenarios are concrete examples of the future system in use. Developers use these scenarios to communicate with the user and deepen their understanding of the application domain.*
- ***Identifying use cases.*** *Once developers and users agree on a set of scenarios, developers derive from the scenarios a set of use cases that completely represent the future system.*
- ***Whereas scenarios are concrete examples illustrating a single case****, use cases area abstractions describing all possible cases.*
- *When describing use cases,* ***developers determine the scope of the system.***

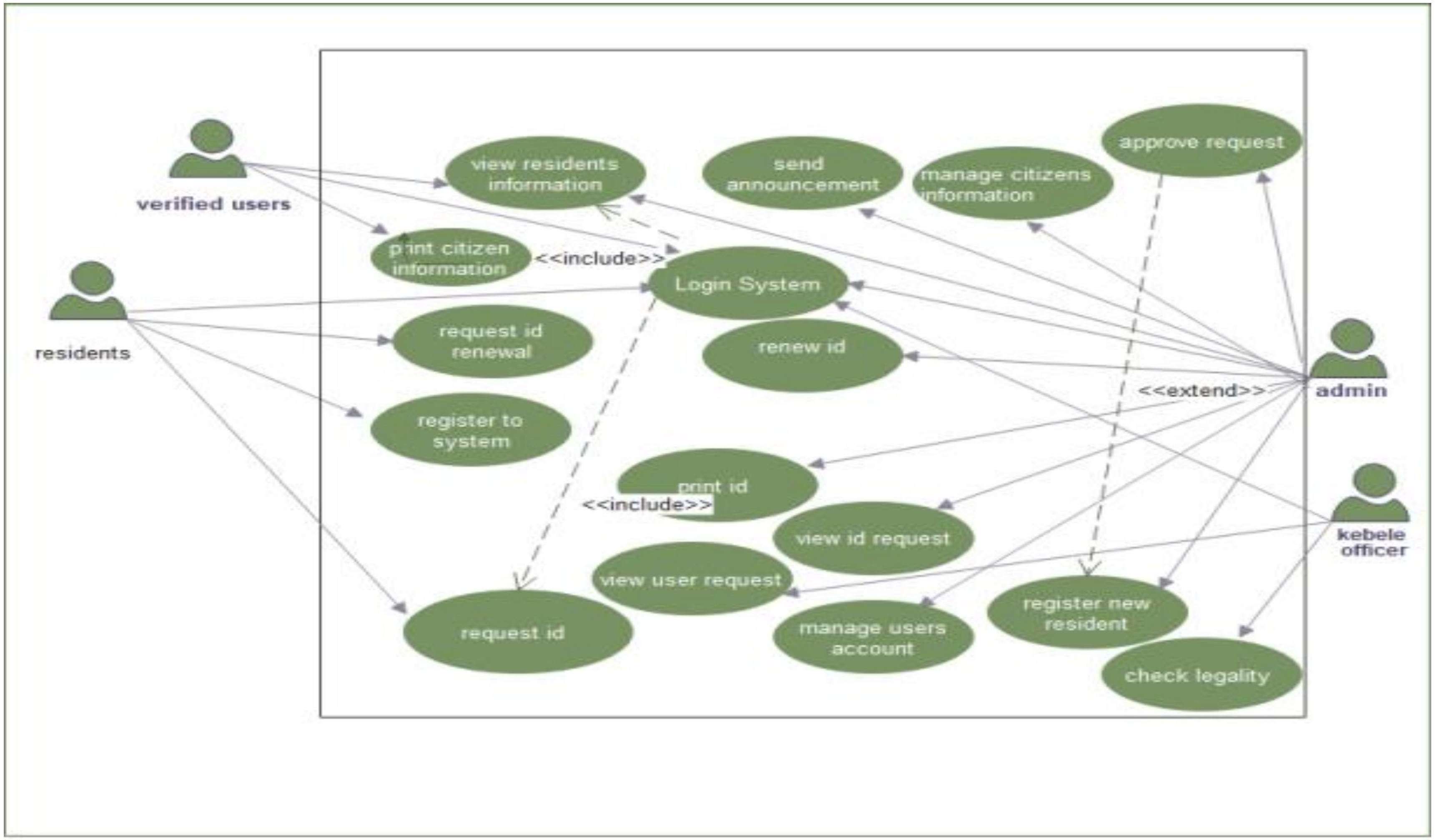
Cntd…

- ***Refining use cases.*** *During this activity, developers ensure that the requirements specification is complete by detailing each use case and describing the behavior of the system in* ***the presence of errors*** *and* ***exceptional conditions****.*
- ***Identifying relationships among use cases.*** *During this activity, developers identify dependencies among use cases. They also combine the use case model by factoring out common functionality.*
  - *This ensures that the requirements specification is consistent.*
- ***Identifying nonfunctional requirements.*** *During this activity, developers, users, and clients agree on aspects that are visible to the user,* ***but not directly related to functionality.***
  - ***These include*** *constraints on the performance of the system,* ***its documentation****, the* ***resources it consumes****,* ***its security****, and* ***its quality.***

Cntd…

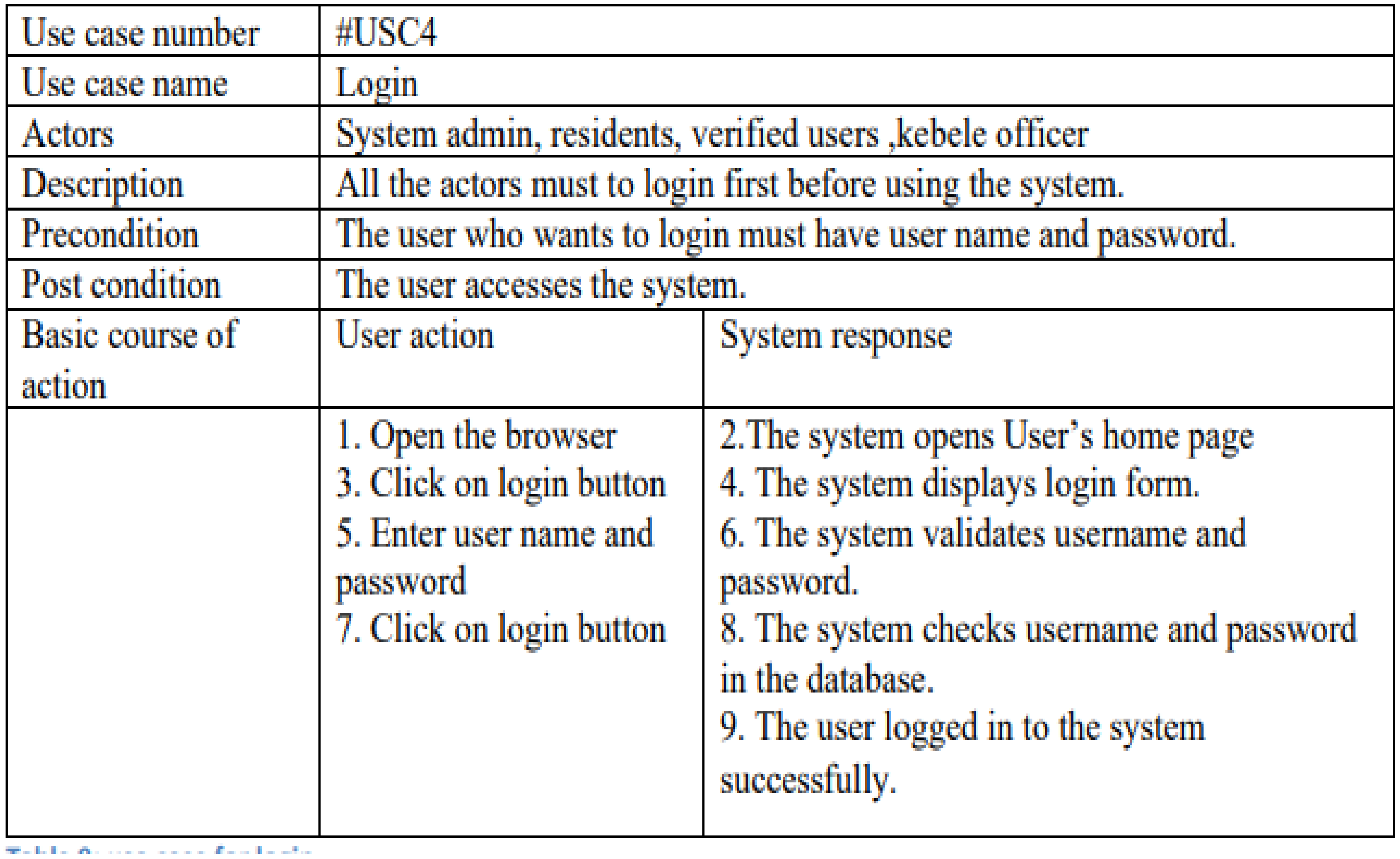
- ***During requirements elicitation***, *developers access many different sources of information, including* ***client-supplied documents*** *about the application domain,* ***manuals*** *and* ***technical documentation*** *of legacy systems that the future system will replace, and most important, the users and clients themselves.*
- *Developers interact the most with users and clients during requirements elicitation.*
- *We* ***focus on two methods for eliciting information***, ***making decisions with users and clients,*** *and managing dependencies among requirements and other artifacts:*
  - ***Joint Application Design*** *(JAD) focuses on building consensus among developers, users, and clients by jointly developing the requirements specification.*
  - ***Traceability*** *focuses on recording, structuring, linking, grouping, and maintaining dependencies among requirements and between requirements and other work products*

Cntd…



National ID system for citizen Use case diagram

Cntd…



Use case number	#USC4	
Use case name	Login	
Actors	System admin, residents, verified users ,kebele officer	
Description	All the actors must to login first before using the system.	
Precondition	The user who wants to login must have user name and password.	
Post condition	The user accesses the system.	
Basic course of action	User action	System response
	1. Open the browser 3. Click on login button 5. Enter user name and password 7. Click on login button	2.The system opens User's home page 4. The system displays login form. 6. The system validates username and password. 8. The system checks username and password in the database. 9. The user logged in to the system successfully.

use case table for login

Requirements Elicitation Concepts

- *In this section, we describe the main requirements elicitation concepts used.*
- *In particular, we describe*
    - ✓ *Functional Requirements*
    - ✓ *Nonfunctional Requirements*

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.*

Cntd…

*Nonfunctional Requirements describe **aspects of the system** that are not directly related to the functional behavior of the system.*

- *It include a broad variety of requirements that apply to many different aspects of the system, from usability to performance.*
- ***The following are 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.*

Cntd…

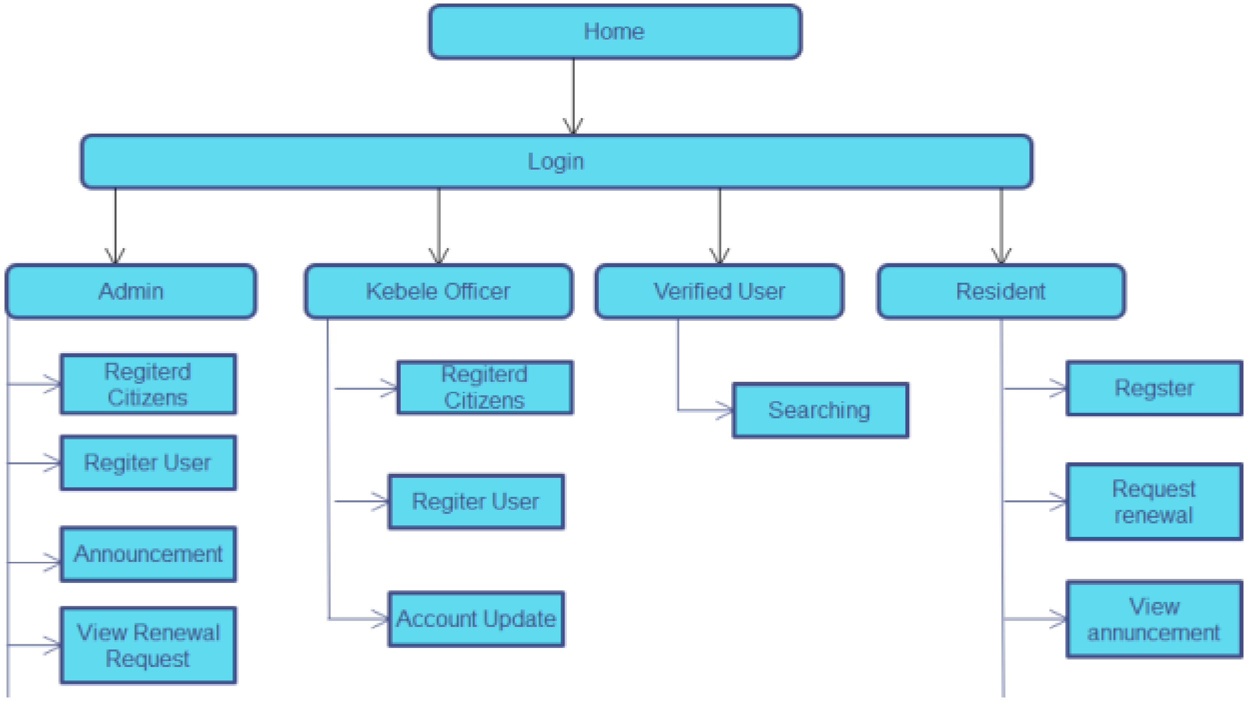
- ***Reliability** is the ability of a system or component to perform its required functions under stated conditions for a specified period of time. Reliability requirements include,*
- *for example, an acceptable mean time to failure and the ability to detect specified faults or to withstand specified security attacks.*
- ***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),*

Cntd…

- *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, a**daptability** (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).*

User Interface Prototyping

- *User interface (UI) prototyping is an iterative analysis technique in which users are actively involved in the mocking -up of the UI for a system.*
- *UI prototypes have several purposes: As an analysis artifact that enables you to explore the problem space with your stakeholders*



Domain modeling with class responsibility collaborator (CRC) cards

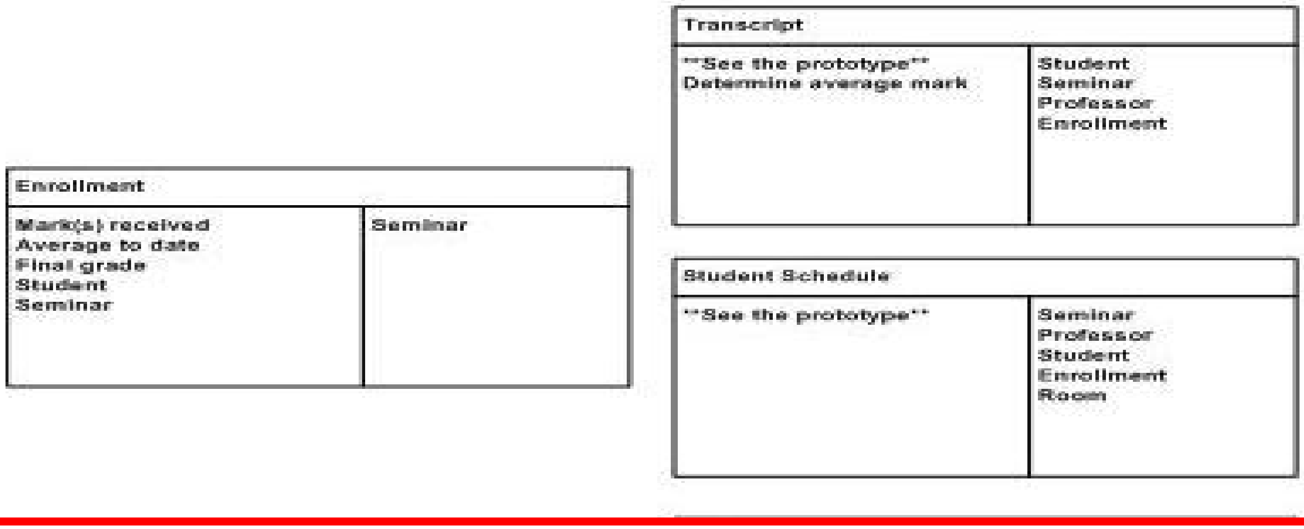
- **Domain Modeling** is a way to describe and model real world entities and the relationships between them, which collectively describe the problem domain space.
- Derived from an understanding of system-level requirements, identifying domain entities and their relationships provides an effective basis for understanding and helps practitioners design systems for maintainability, testability, and incremental development. Because there is often a gap between understanding the problem domain and the interpretation of requirements.
- **Class-responsibility-collaborator cards** (CRC cards) are not a part of the UML specification, but they are a useful tool for organizing classes during analysis and design.
- A CRC card is a physical card representing a single class.

Cntd…

- Each card lists the class's name, attributes and methods (its responsibilities), and class associations (collaborations).
- The collection of these CRC cards is the CRC model.
- Using CRC cards is a straightforward addition to object-oriented analysis and design:
  - Identify the classes.
  - List responsibilities.
  - List collaborators.
- CRC cards can be used during analysis and design while classes are being discovered in order to keep track of them.

Cntd…

- ***A ``collaborator''*** *for a class (``class A'') is another class (``class B'') that receives a message from class A, sent because the class A needs one of the class B's services to be performed, in order for one of class A's responsibilities to be fulfilled. you should list class B as a collaborator of class A, by listing class B in the ``collaborator'' column on class A's index card, to the right of the listing for the responsibility that required the message to be sent.*



Enrollment	
Mark(s) received Average to date Final grade Student Seminar	Seminar

Transcript	
See the prototype Determine average mark	Student Seminar Professor Enrollment

Student Schedule	
See the prototype	Seminar Professor Student Enrollment Room

Developing a supplementary Specification

- ***The Supplementary Specification*** *provides an overview of the entire document.*
- *It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this Supplementary Specification.*
- *It also captures the system requirements* ***that are not readily captured*** *in the use cases of the use-case model. Such requirements include:*
  - *Legal and regulatory requirements, including application standards.*
  - *Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.*
  - *Other requirements such as operating systems and environments, compatibility requirements, and design constraints.*

Cntd…

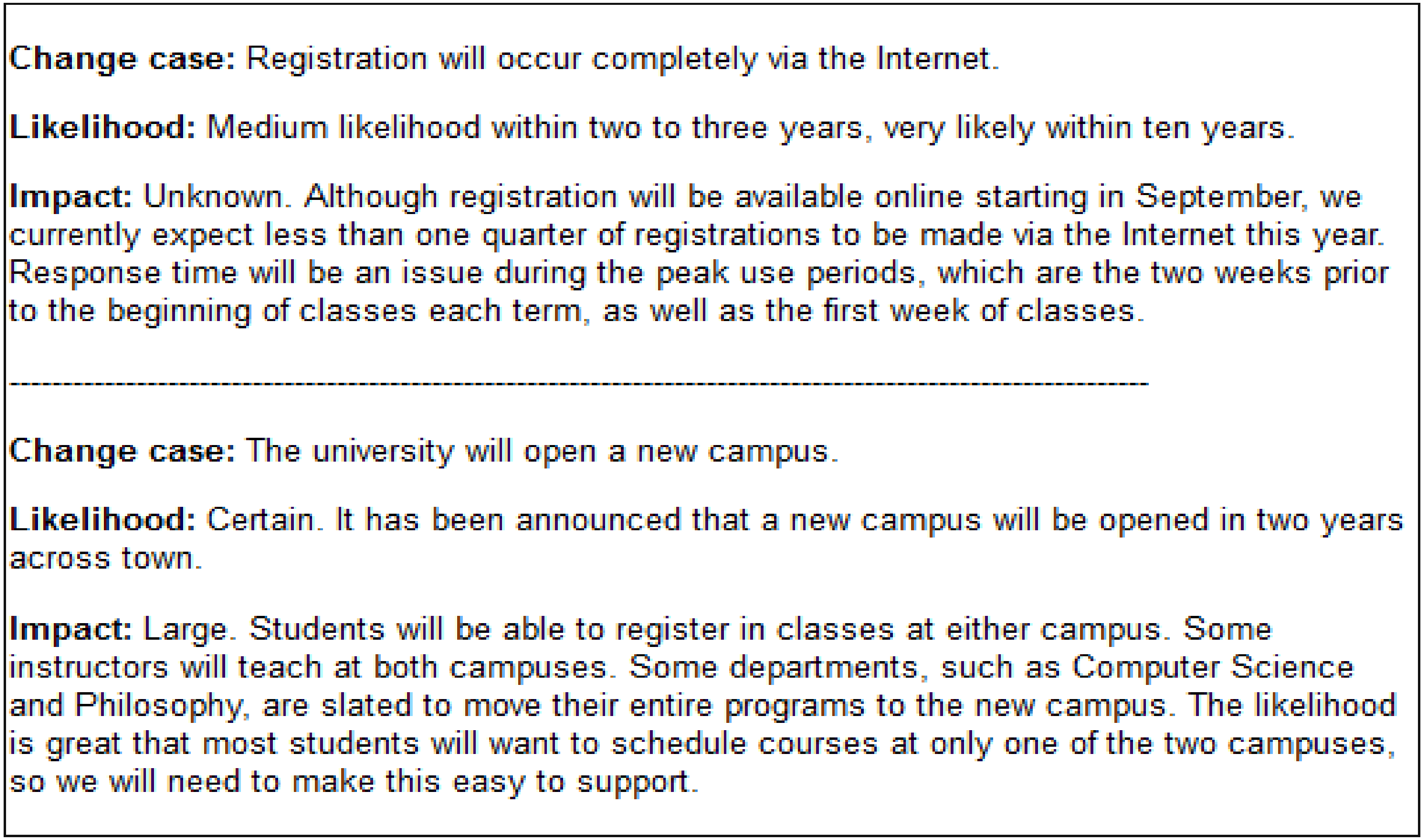
Purpose

- *The purpose of the Supplementary Specification is to capture the system requirements that are not readily captured in the use cases of the use-case model.*
- *It includes requirements such as legal and regulatory requirements, application standards, quality attributes of the system to be built (**usability**, **reliability, performance,** and **supportability requirements**) and other requirements such as operating systems and environment, compatibility requirements and design constraint.*

Identifying Change Cases

- ***Change cases*** *are used to describe new potential requirements for a system or modifications to existing requirements.*
- *Change cases are modeled in a simple manner.*
- *You describe the potential change to your existing requirements, indicate the **likeliness of that change occurring,** and indicate the **potential impact of that change**.*
- *Figure 1 presents two change cases, one potential change that is motivated by technical innovation-in this case the use of the Internet-and a second by a change in your business environment. Notice how both change cases are short and to the point, making them easy-to-understand.*
- *The name of a change case should describe the potential change itself.*

Cntd…



Change case: Registration will occur completely via the Internet.

Likelihood: Medium likelihood within two to three years, very likely within ten years.

Impact: Unknown. Although registration will be available online starting in September, we currently expect less than one quarter of registrations to be made via the Internet this year. Response time will be an issue during the peak use periods, which are the two weeks prior to the beginning of classes each term, as well as the first week of classes.

Change case: The university will open a new campus.

Likelihood: Certain. It has been announced that a new campus will be opened in two years across town.

Impact: Large. Students will be able to register in classes at either campus. Some instructors will teach at both campuses. Some departments, such as Computer Science and Philosophy, are slated to move their entire programs to the new campus. The likelihood is great that most students will want to schedule courses at only one of the two campuses, so we will need to make this easy to support.

figure: change case

Cntd…

- Change cases can be identified throughout the course of your overall development effort.
- Change cases are often the result of brainstorming with your stakeholders. Good questions to consider include:
  - ✓ How can the business change?
  - ✓ What is the long-term vision for our organization?
  - ✓ What technology can change?
  - ✓ What legislation can change?
  - ✓ What is your competition doing?
  - ✓ What systems will we need to interact with?
  - ✓ Who else might use the system and how?

End of Chapter Six

Any question???