REQUIREMENTS MODELING – SCENARIO-BASED METHODS (LESSON 06)

PHÂN TÍCH YÊU CẦU PHẦN MỀM (SOFTWARE REQUIREMENTS)

CONTENTS

THAM KHẢO:

https://www.uml-diagrams.org/index-examples.html

https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_tutorial.pdf

STUDENT BOOK 2021

LƯU HÀNH NỘI BỘ
(TÀI LIỆU TRÍCH LỌC)

PHÂN TÍCH
YÊU CẦU PHẦN MỀM
(SOFTWARE REQUIREMENTS)

AUGUST 12

COLLEGE OF INFORMATION TECHNOLOGY
& COMMUNICATION
Authored by: Collection Team

REQUIREMENTS ANALYSIS

- Requirements analysis results in the specification of software's operational characteristics, indicates software's interface with other system elements, and establishes constraints that software must meet.
- Requirements analysis allows you (regardless of whether you're called a software engineer, an analyst, or a modeler) to elaborate on basic requirements established during the inception, elicitation, and negotiation tasks that are part of requirements engineering (Chapter 2).

REQUIREMENTS ANALYSIS

- The requirements modeling action results in one or more of the following types of models:
  - Scenario-based models of requirements from the point of view of various system “actors.”
  - Class-oriented models that represent object-oriented classes (attributes and operations) and the manner in which classes collaborate to achieve system requirements.
  - Behavioral and patterns-based models that depict how the software be-haves as a consequence of external “events.”
  - Data models that depict the information domain for the problem.
  - Flow-oriented models that represent the functional elements of the system and how they transform data as they move through the system.

OVERALL OBJECTIVES AND PHILOSOPHY

- Throughout analysis modeling (or requirement modeling), your primary focus is on "what", not "how".
- What user interaction occurs in a particular circumstance,
  - what objects does the system manipulate,
  - what functions must the system perform,
  - what behaviors does the system exhibit,
  - what interfaces are defined, and
  - what constraints apply?

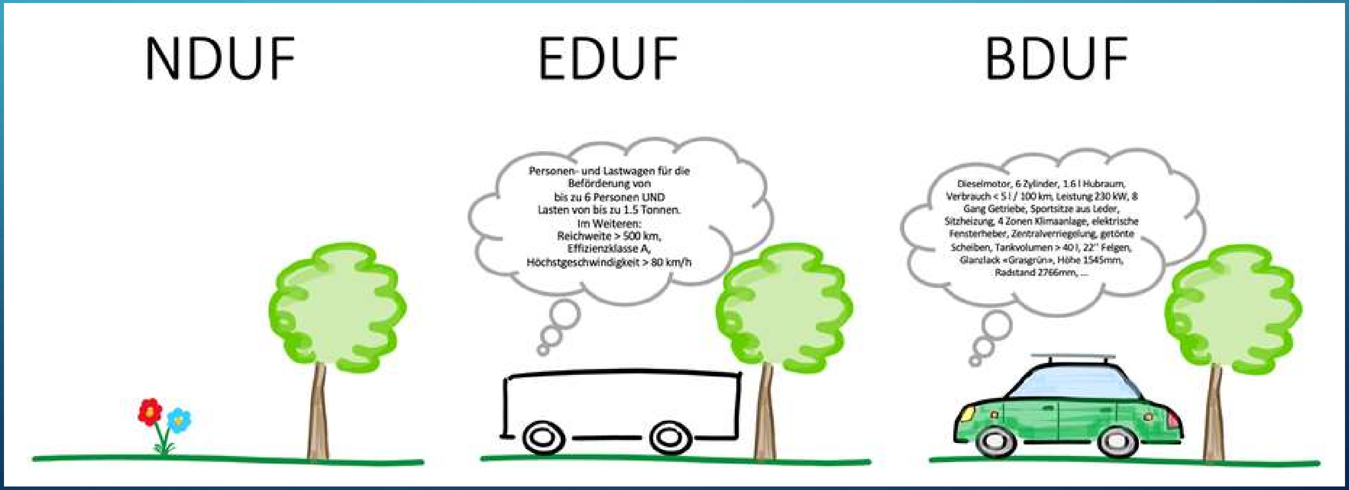
REQUIREMENTS ANALYSIS

- Complete specification of requirements may not be possible at this stage:
  - Customer may be unsure of precisely what is required for certain aspects of the system.
  - Developer may be unsure that a specific approach will properly accomplish function and performance.
- These realities mitigate in favor of an iterative approach to requirements analysis and modeling.
- Analyst should model what is known and use that model as the basis for design of the software increment.

DSDM: ENOUGH DESIGN UP FRONT (EDUF)

Traditional approaches use time contingency to reduce risk. Traditional projects use BDUF (Big Design Up Front).

Some Agile projects use NDUF (No Design Up Front). DSDM uses the EDUF (Enough Design Up Front) approach. It is because requirements are sometimes unknowable up front or are likely to change during the project.


NDUF
EDUF
Personen- und Lastwagen für die Beförderung von bis zu 6 Personen UND Lasten von bis zu 1.5 Tonnen. Im Weiteren: Reichweite > 500 km, Effizienzklasse A, Höchstgeschwindigkeit > 80 km/h
BDUF
Dieselmotor, 6 Zylinder, 1.6 l Hubraum, Verbrauch < 5 l / 100 km, Leistung 230 kW, 8 Gang Getriebe, Sportsitze aus Leder, Sitzheizung, 4 Zonen Klimaanlage, elektrische Fensterheber, Zentralverriegelung, getönte Scheiben, Tankvolumen > 40 l, 22'' Felgen, Glanzlack «Grasgrün», Höhe 1545mm, Radstand 2766mm, …

NDUF (NO DESIGN UP FRONT)

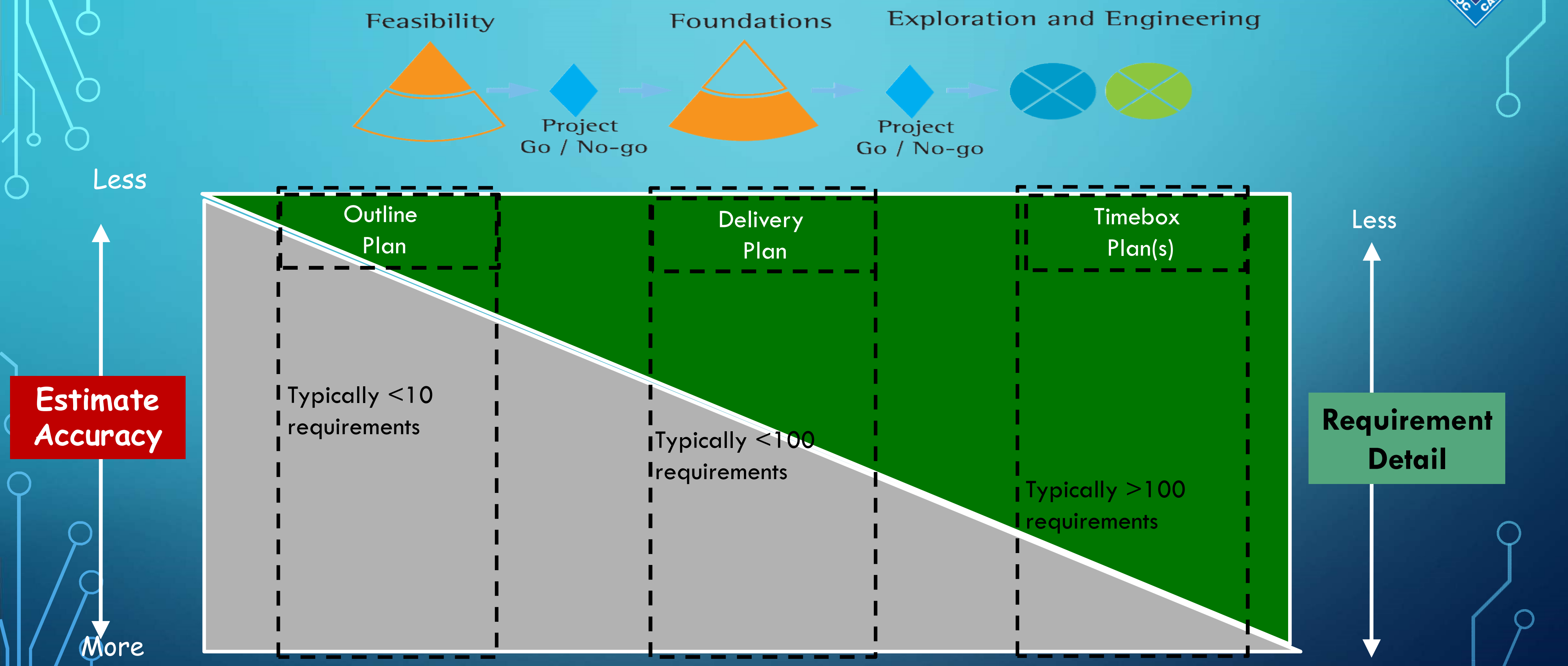
- You can start with the development practically **immediately**, without investing any time in a design. Often a product vision and a rough product backlog serve as the starting point. "Sprint zero" on the first productive iteration resp. «Sprint 1» is being prepared.
- During the development, new inputs and ideas are continuously recorded and implemented. Such a free approach requires a great deal of experience. Most of them run the risk of developing unstable architectures and investing resources in developments that later turn out to be redundant or unusable.

"Big design up front is **dumb**, but doing no design up front is even dumber." (BDUF is stupid, but NDUF is **even more stupid.**)

(**Dave Thomas**, one of the authors of the "Agile Manifesto")

https://www.agileagreement.com/2020/07/21/enough-design-upfront-vor-big-design-upfront/

Requirements in the DSDM Lifecycle



Source: Image from pixabay.com © 2016

PRIMARY OBJECTIVES OF REQUIREMENTS MODEL

- The requirements model must achieve three primary objectives: (1) to describe what the customer requires, (2) to establish a basis for the creation of a software design, and (3) to define a set of requirements that can be validated once the software is built.

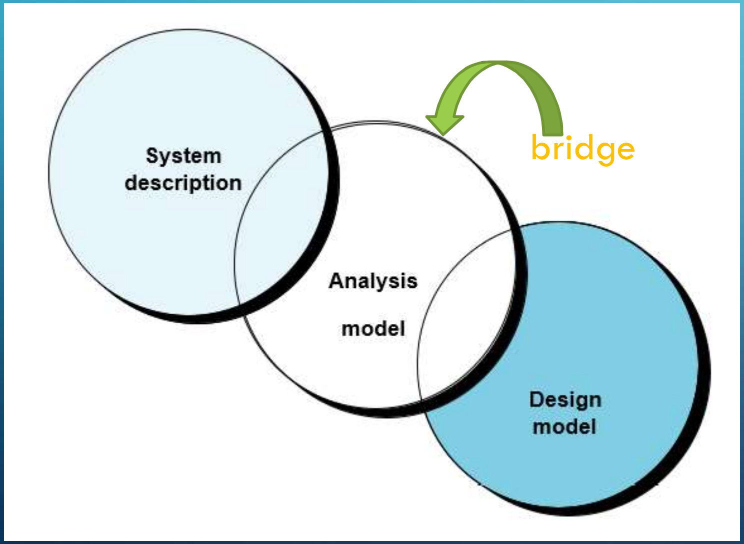


Figure 6.1. The requirements model as a bridge between the system description and the design model.

ANALYSIS RULES OF THUMB

- Arlow and Neustadt [Arl02] suggest a **number of worthwhile rules of thumb** that should be followed when creating the analysis model:
  - The model should focus on requirements that are visible within the problem or business domain.
  - Each element of the requirements model should add to an overall understanding of software requirements and provide insight into the information domain, function, and behavior of the system.
  - Delay consideration of infrastructure and other nonfunctional models until design.
  - Minimize coupling throughout the system. It is important to represent relationships between classes and functions.
  - Be certain that the requirements model provides value to all stakeholders.
  - Keep the model as simple as it can be.

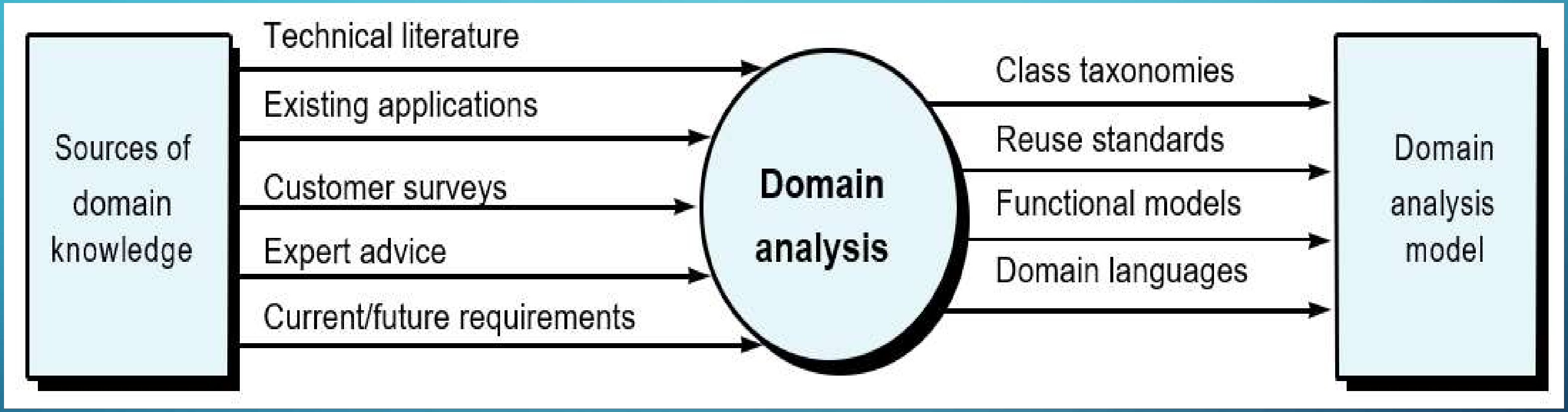
REQUIREMENTS GATHERING TECHNIQUES

ARTICLE
11 Requirements Gathering Techniques for Agile Product Teams
jama software

- Interviews
- Questionnaires or Surveys
- User Observation
- Document Analysis
- Interface analysis
- Workshops
- Brainstorming
- Role-play
- Use Cases and Scenarios
- Focus Groups
- Prototyping

https://www.jamasoftware.com/blog/11-requirements-gathering-techniques-for-agile-product-teams/

DOMAIN ANALYSIS



DOMAIN ANALYSIS

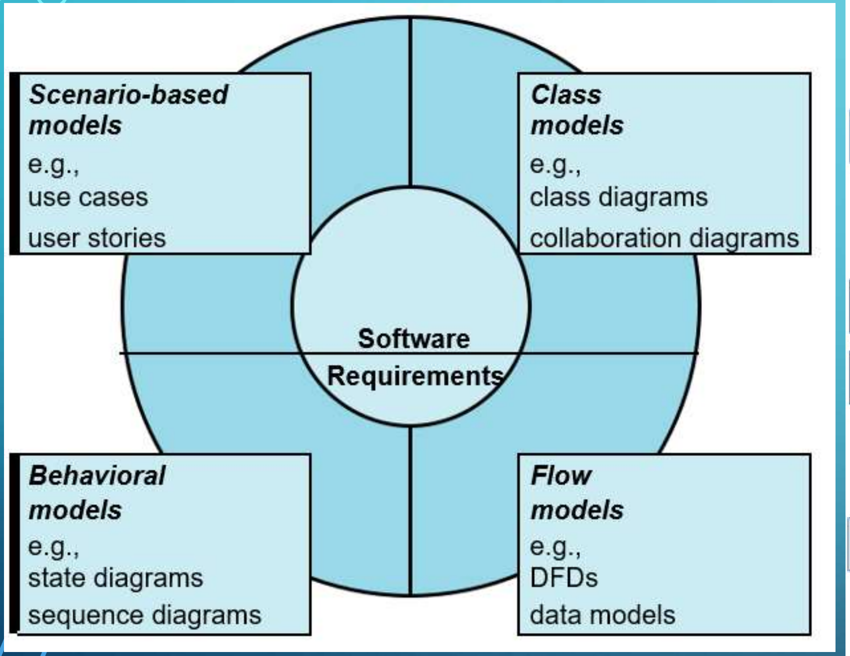
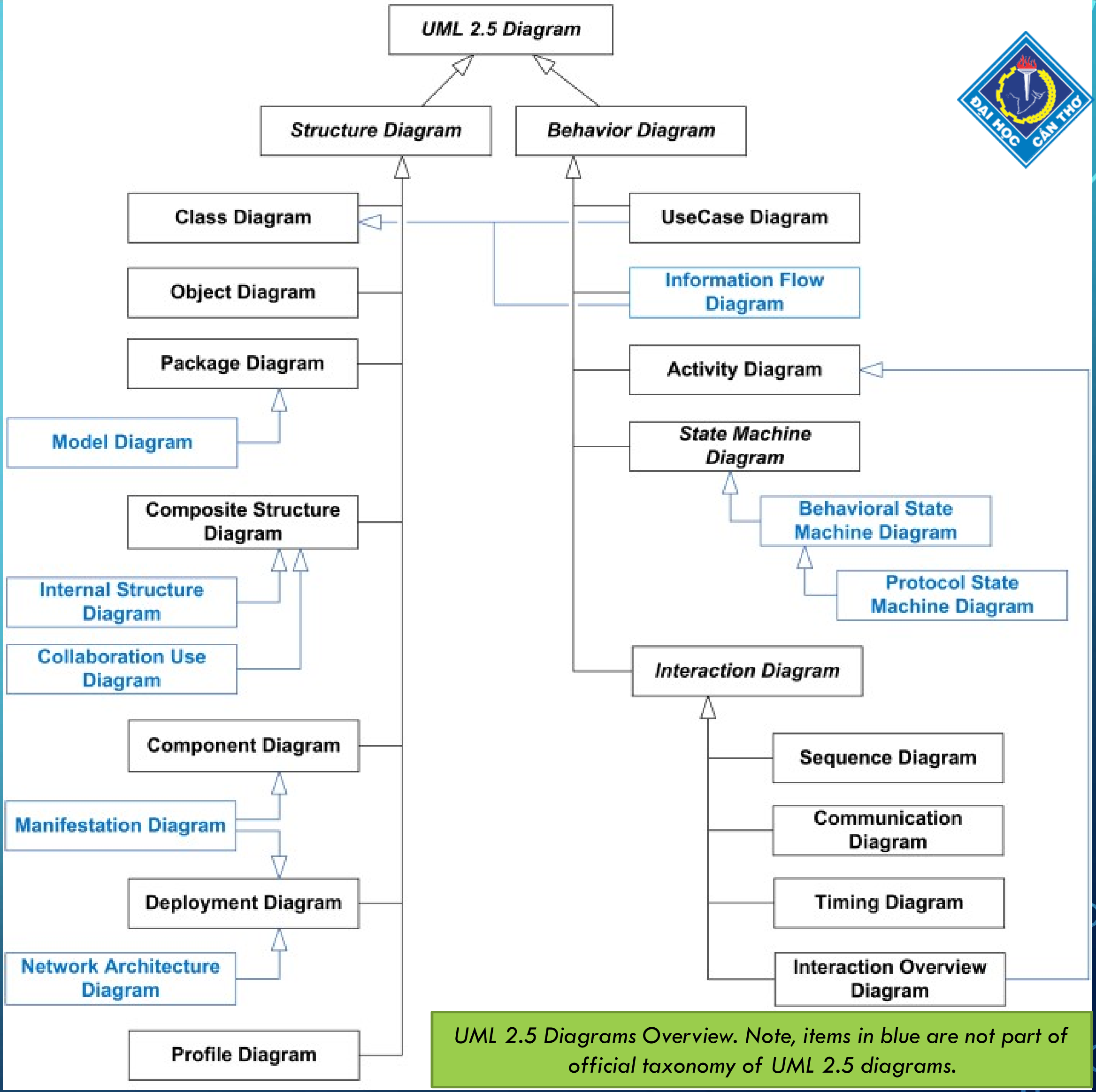


Figure 6.3. Elements of the analysis model.



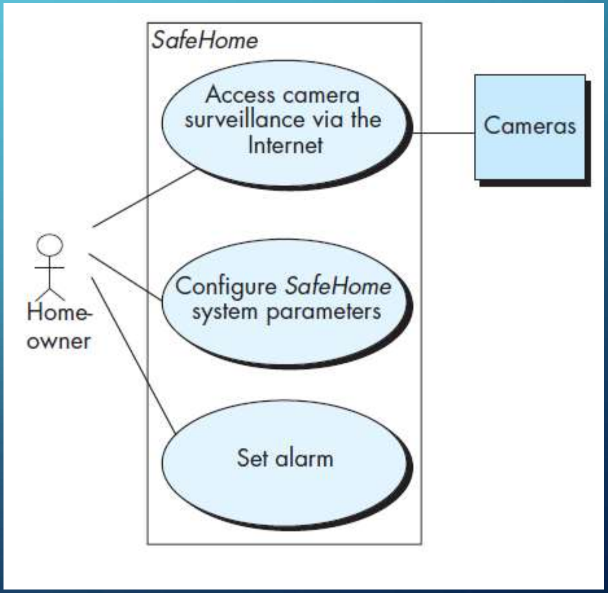
UML 2.5 Diagrams Overview. Note, items in blue are not part of official taxonomy of UML 2.5 diagrams.

SCENARIO-BASED MODELING

- Although the success of a computer-based system or product is measured in many ways, user satisfaction resides at the top of the list.
- If you understand how end users (and other actors) want to interact with a system, your software team will be better able to properly characterize requirements and build meaningful analysis and design models.
- Hence, requirements modeling with UML begins with the creation of scenarios in the form of use cases, activity diagrams, and swimlane diagrams.

CREATING THE PRELIMINARY USE CASE

- The SafeHome home surveillance function (subsystem) identifies the following functions (an abbreviated list) that are performed by the homeowner actor:
  - Select camera to view.
  - Request thumbnails from all cameras.
  - Display camera views in a PC window.
  - Control pan and zoom for a specific camera.
  - Selectively record camera output.
  - Replay camera output.
  - Access camera surveillance via the Internet.


SafeHome
Access camera surveillance via the Internet
Cameras
Configure *SafeHome* system parameters
Home-owner
Set alarm

SAFEHOME PROJECT

- Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)
- Actor: Homeowner
- If I'm at a remote location, I can use any PC with appropriate browser software to log on to the SafeHome Products website. I enter my user ID and two levels of passwords and once I'm validated, I have access to all functionality for my installed SafeHome system. To access a specific camera view, I select "surveillance" from the major function buttons displayed. I then select "pick a camera" and the floor plan of the house is displayed. I then select the camera that I'm interested in. Alternatively, I can look at thumbnail snapshots from all cameras simultaneously by selecting "all cameras" as my viewing choice. Once I choose a camera, select "view" and a one-frame-per-second view appears in a viewing window that is identified by the camera ID. If I want to switch cameras, I select "pick a camera" and the original viewing window disappears and the floor plan of the house is displayed again. I then select the camera that I'm interested in. A new viewing window appears.

SAFEHOME PROJECT

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

Actor: homeowner

1. The homeowner logs onto the SafeHome Products website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

REFINING A PRELIMINARY USE CASE

- Each step in the primary scenario is evaluated by asking the following questions [Sch98a]:
  - Can the actor take some other action at this point?
  - Is it possible that the actor will encounter some error condition at this point? If so, what might it be?
  - Is it possible that the actor will encounter some other behavior at this point (e.g., behavior that is invoked by some event outside the actor's control)? If so, what might it be?

WRITING A FORMAL USE CASE - 1

PROJECT: SAFEHOME
Use Case Template for Surveillance
Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)
Iteration: 2, last modifi cation: January 14 by V. Raman.
Primary actor: Homeowner.
Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.
Preconditions: System must be fully confi gured; appropriate user ID and passwords must be obtained.
Trigger: The homeowner decides to take a look inside the house while away.
Scenario:
1. The homeowner logs onto the SafeHome Productswebsite.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identifi ed by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

WRITING A FORMAL USE CASE - 2

Exceptions:
1. ID or passwords are incorrect or not recognized see use case Validate ID and passwords.
2. Surveillance function not confi gured for this system—system displays appropriate error message; see use case Configure surveillance function.
3. Homeowner selects "View thumbnail snapshots for all camera"—see use case View thumbnail snapshots for all cameras.
4. A floor plan is not available or has not been confi gured—display appropriate error message and see use case Confi gure fl oor plan.
5. An alarm condition is encountered—see use case alarm condition encountered.

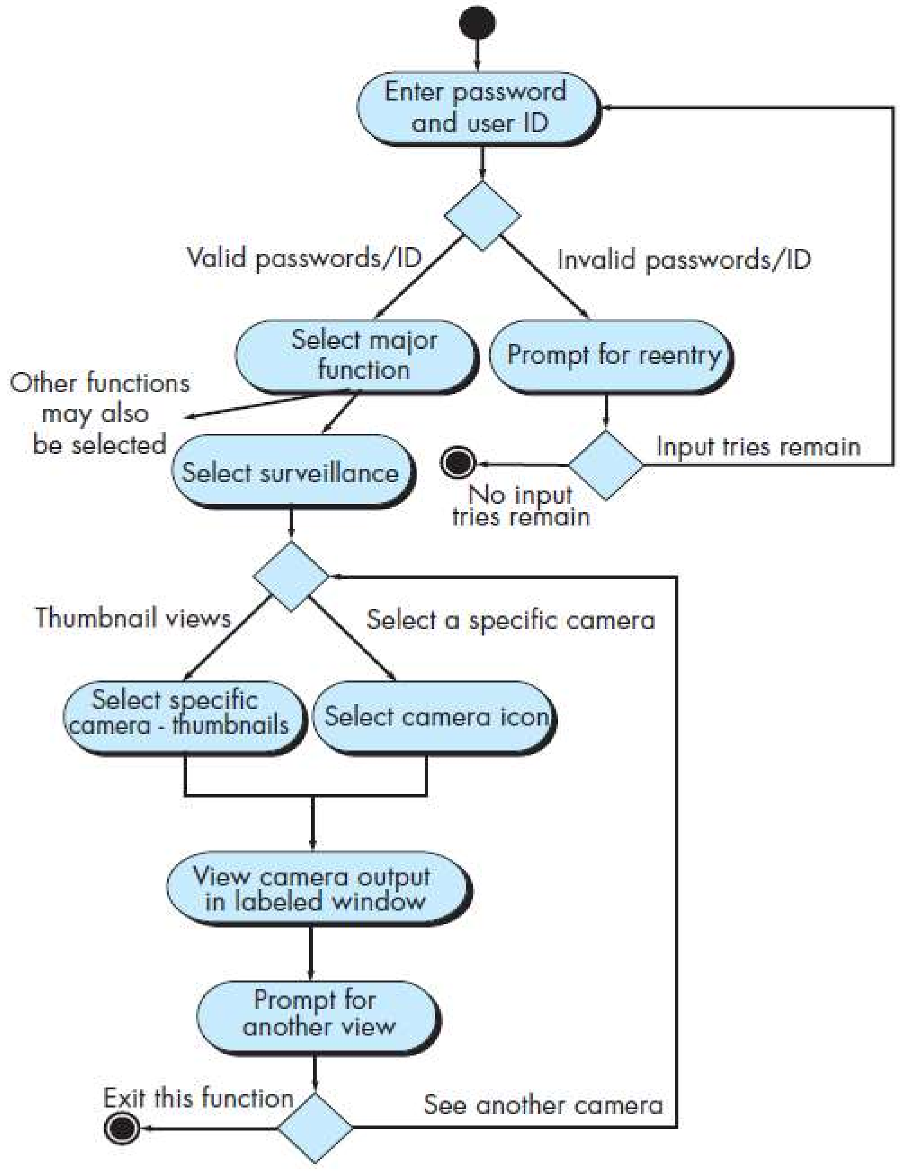
Priority: Moderate priority, to be implemented after basic functions.
When available: Third increment.
Frequency of use: Infrequent.
Channel to actor: Via PC-based browser and Internet connection.
Secondary actors: System administrator, cameras.
Channels to secondary actors:
1. System administrator: PC-based system.
2. Cameras: wireless connectivity.

Open issues:
1. What mechanisms protect unauthorized use of this capability by employees of SafeHome Products?
2. Is security suffi cient? Hacking into this feature would represent a major invasion of privacy.
3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
4. Will we develop a capability to provide video at a higher frames-per-second rate when highbandwidth connections are available?

DEVELOPING AN ACTIVITY DIAGRAM

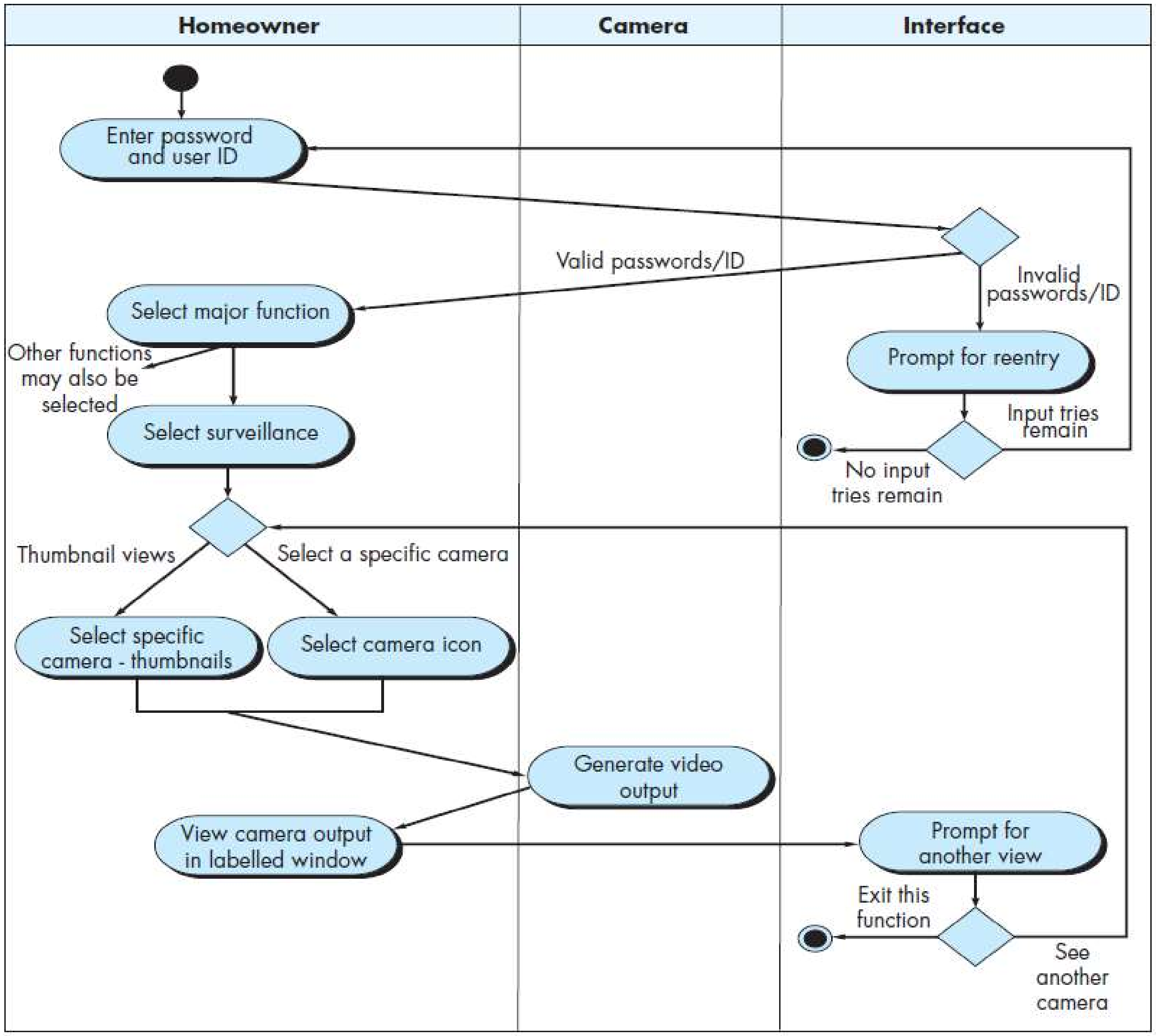
- The UML activity diagram supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario.
- Similar to the flowchart, an activity diagram uses rounded rectangles to imply a specific system function, arrows to represent flow through the system, decision diamonds to depict a branching decision (each arrow emanating from the diamond is labeled), and solid horizontal lines to indicate that parallel activities are occurring.

Figure 6.5. Activity diagram for Access camera surveillance via the Internet—display camera views function.



SWIMLANE DIAGRAMS

- UML swimlane diagram:
  - is a useful variation of the activity diagram.
  - allows you to represent the flow of activities described by the use case and
  - at the same time indicate which actor (if there are multiple actors involved in a specific use case) or analysis class has responsibility for the action described by an activity rectangle.
- Responsibilities are represented as parallel segments that divide the diagram vertically, like the lanes in a swimming pool.


Homeowner
Camera
Interface
Enter password and user ID
Valid passwords/ID
Invalid passwords/ID
Select major function
Other functions may also be selected
Prompt for reentry
Input tries remain
No input tries remain
Select surveillance
Thumbnail views
Select a specific camera
Select specific camera - thumbnails
Select camera icon
Generate video output
View camera output in labelled window
Prompt for another view
Exit this function
See another camera

SUMMARY

- The objective of requirements modeling is to
  - create a variety of representations that describe what the customer requires,
  - establish a basis for the creation of a software design, and
  - define a set of requirements that can be validated once the software is built.
- The requirements model bridges the gap between
  - a system-level description that describes overall system and business functionality and
  - a software design that describes the software’s application architecture, user inter-face, and component-level structure.

Q & A

1. Is it possible to begin coding immediately after a requirements model has been created? Explain your answer and then argue the counterpoint.
2. An analysis rule of thumb is that the model "should focus on requirements that are visible within the problem or business domain." What types of requirements are not visible in these domains? Provide a few examples.
3. What is the purpose of domain analysis? How is it related to the concept of requirements patterns?
4. Is it possible to develop an effective analysis model without developing all four elements shown in Figure 9.3? Explain.

(PROBLEMS AND POINTS TO PONDER)