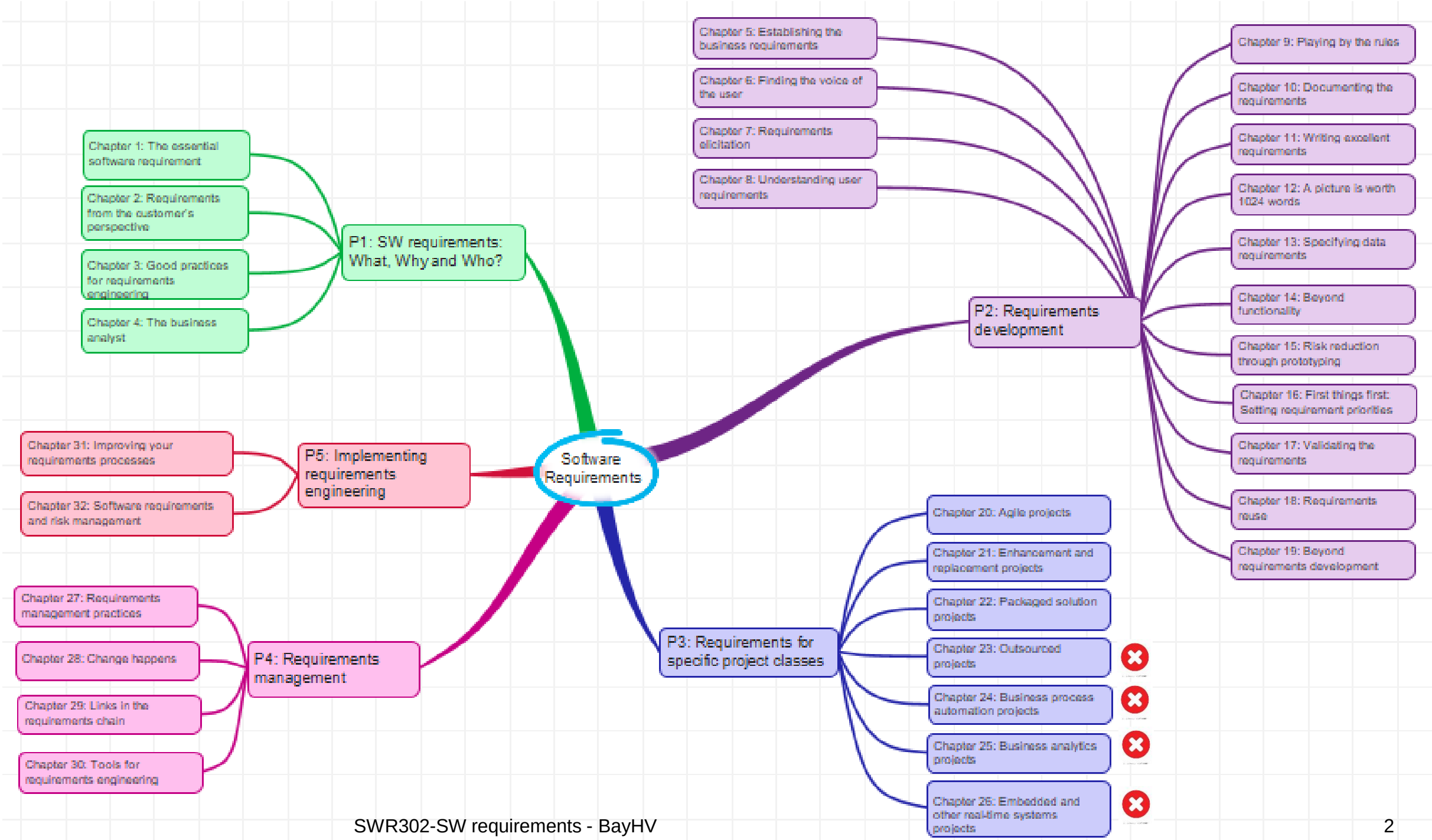
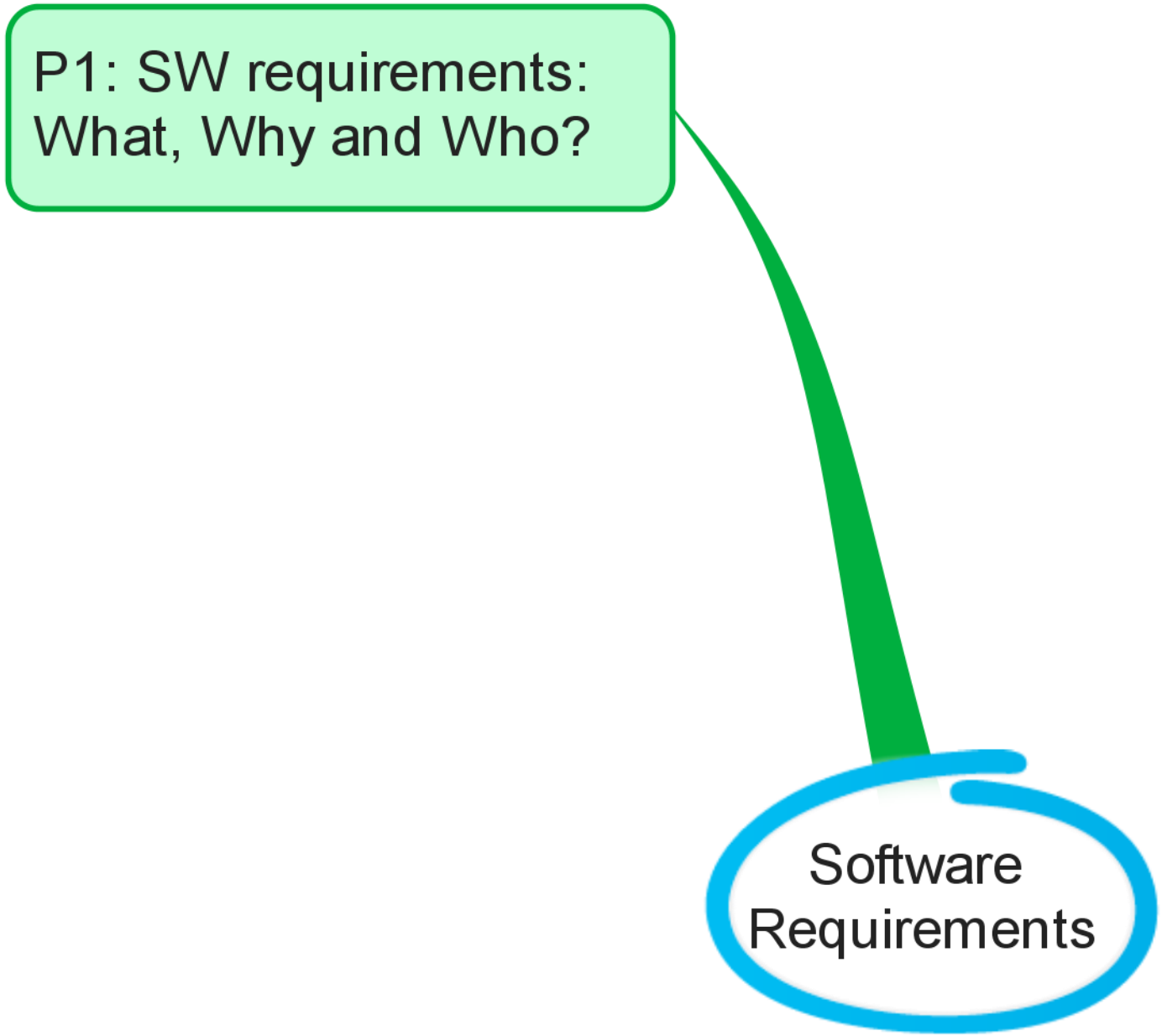

Software
Requirements


Software Requirements
P1: SW requirements: What, Why and Who?
Chapter 1: The essential software requirement
Chapter 2: Requirements from the customer's perspective
Chapter 3: Good practices for requirements engineering
Chapter 4: The business analyst
P2: Requirements development
Chapter 5: Establishing the business requirements
Chapter 6: Finding the voice of the user
Chapter 7: Requirements elicitation
Chapter 8: Understanding user requirements
Chapter 9: Playing by the rules
Chapter 10: Documenting the requirements
Chapter 11: Writing excellent requirements
Chapter 12: A picture is worth 1024 words
Chapter 13: Specifying data requirements
Chapter 14: Beyond functionality
Chapter 15: Risk reduction through prototyping
Chapter 16: First things first: Setting requirement priorities
Chapter 17: Validating the requirements
Chapter 18: Requirements reuse
Chapter 19: Beyond requirements development
P3: Requirements for specific project classes
Chapter 20: Agile projects
Chapter 21: Enhancement and replacement projects
Chapter 22: Packaged solution projects
Chapter 23: Outsourced projects
Chapter 24: Business process automation projects
Chapter 25: Business analytics projects
Chapter 26: Embedded and other real-time systems projects
P4: Requirements management
Chapter 27: Requirements management practices
Chapter 28: Change happens
Chapter 29: Links in the requirements chain
Chapter 30: Tools for requirements engineering
P5: Implementing requirements engineering
Chapter 31: Improving your requirements processes
Chapter 32: Software requirements and risk management


P1: SW requirements:
What, Why and Who?
Software
Requirements

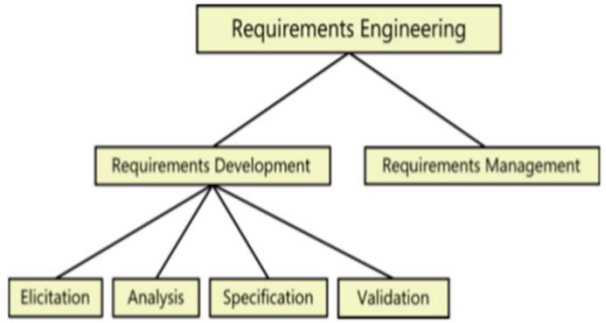
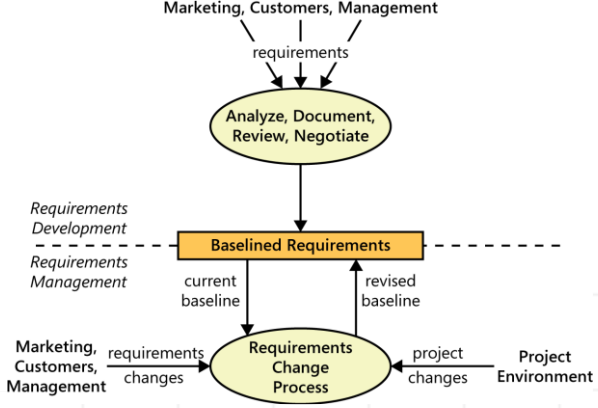


FIGURE 1-4 Subdisciplines of software requirements engineering.

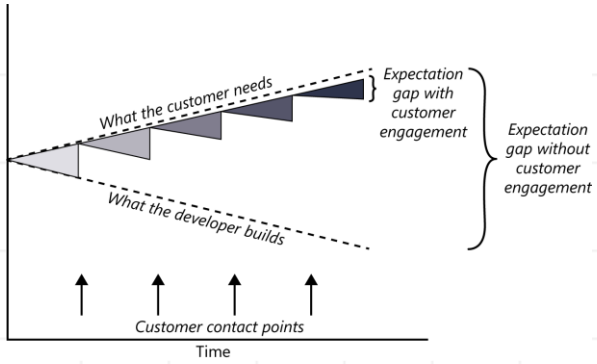


"Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system".

The errors introduced during requirements activities account for 40 to 50% of all defects found in software product.

Some types of requirements information

- Business requirement
- Business rule
- Constraint
- Functional requirement
- Nonfunctional requirement
- System requirement
- User requirement

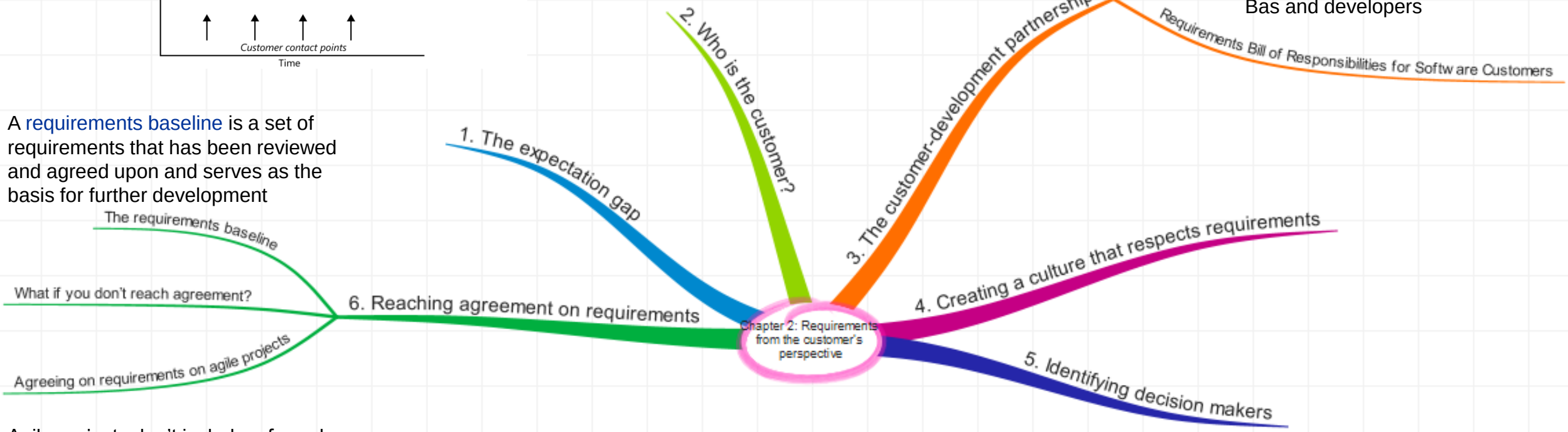


A *stakeholder* is a person, group, or organization that is actively involved in a project, is affected by its process or outcome, or can influence its process or outcome. Stakeholders can be internal or external to the project team and to the developing organization.

10 rights of customers to BAs and developers

10 responsibilities of customers to Bas and developers

A requirements baseline is a set of requirements that has been reviewed and agreed upon and serves as the basis for further development



Agile projects don't include a formal sign-off action. They generally maintain requirements in the form of user stories in a product backlog.

The decision-making group needs to identify its *decision leader* and to select a *decision rule*, which describes how they will arrive at their decisions.

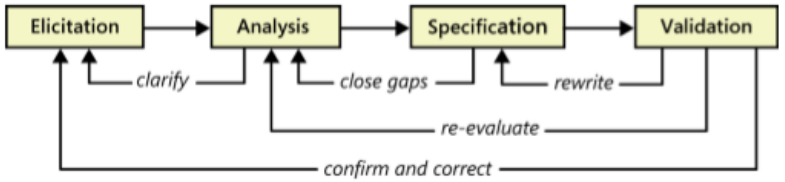
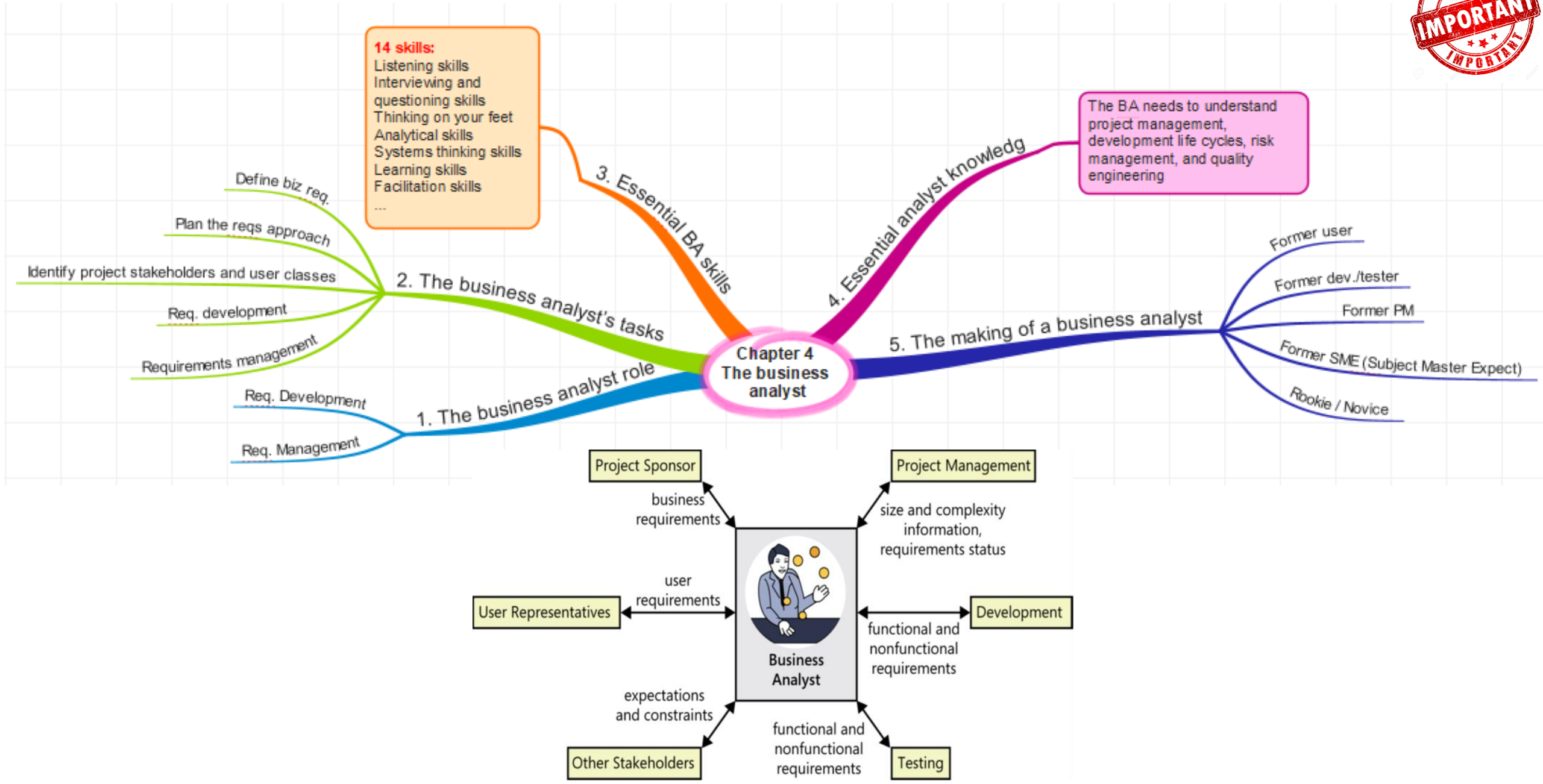
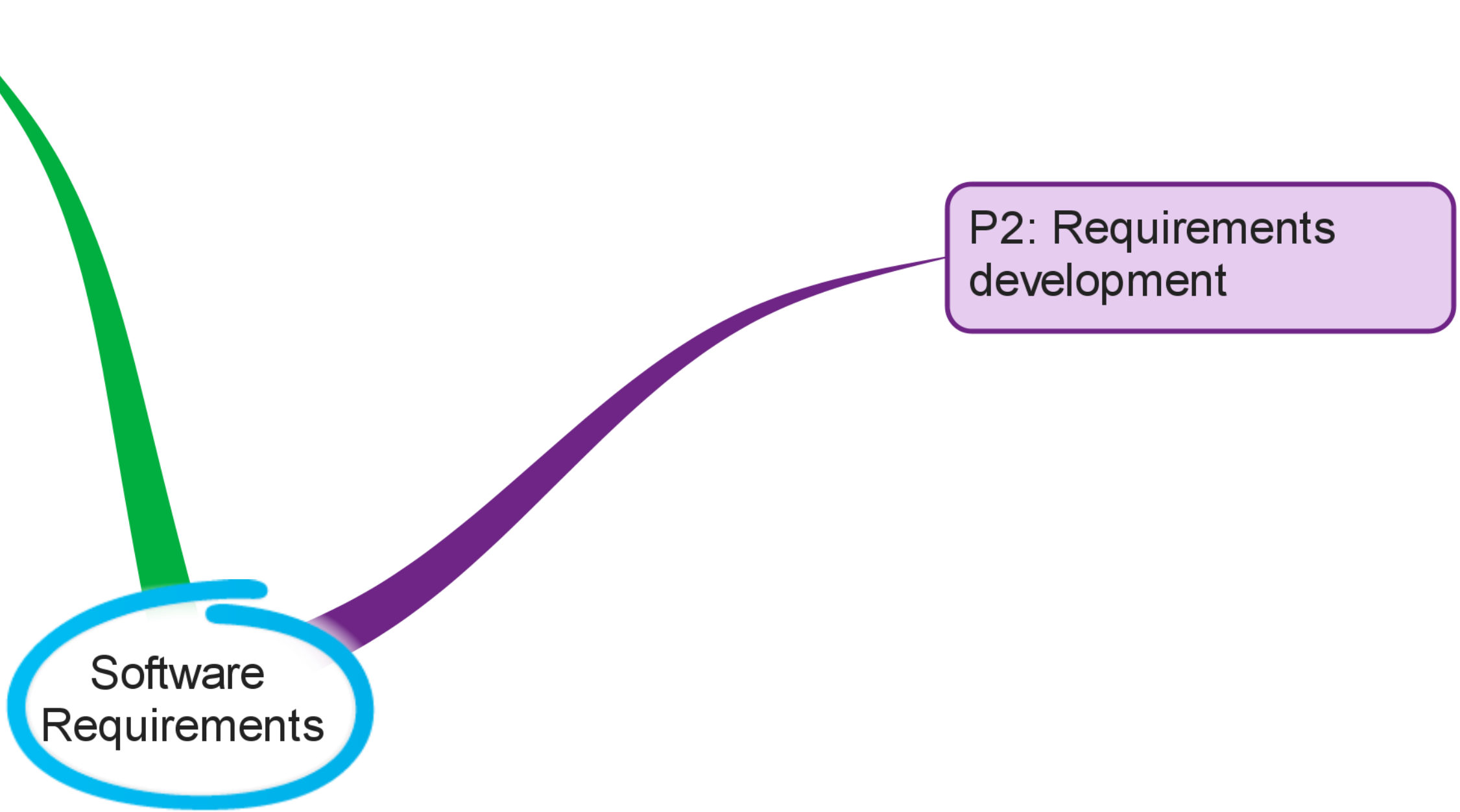


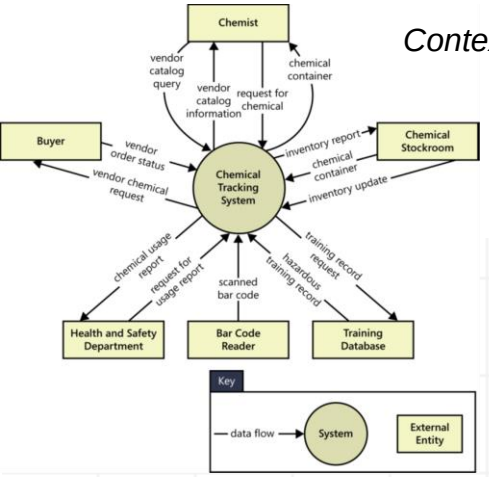
FIGURE 3-1 Requirements development is an iterative process.

There are more than 50 practices, grouped into 7 categories.

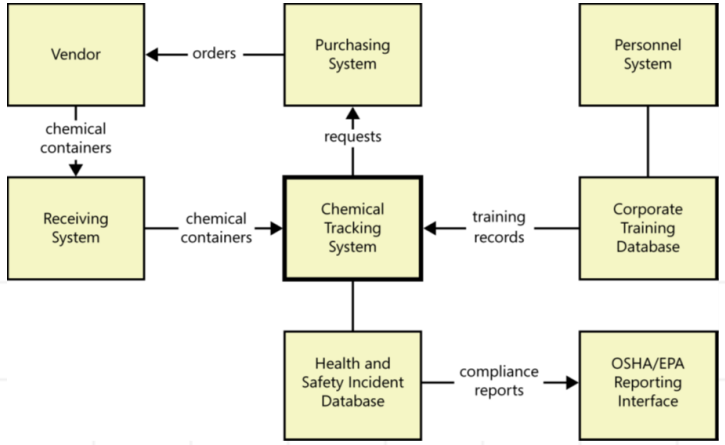
Software development life cycle:
Traditional or Agile project approach?


Chapter 4
The business analyst
1. The business analyst role
Req. Development
Req. Management
2. The business analyst's tasks
Define biz req.
Plan the reqs approach
Identify project stakeholders and user classes
Req. development
Requirements management
3. Essential BA skills
14 skills:
Listening skills
Interviewing and questioning skills
Thinking on your feet
Analytical skills
Systems thinking skills
Learning skills
Facilitation skills
...
4. Essential analyst knowledg
The BA needs to understand project management, development life cycles, risk management, and quality engineering
5. The making of a business analyst
Former user
Former dev./tester
Former PM
Former SME (Subject Master Expect)
Rookie / Novice
Project Sponsor
business requirements
Project Management
size and complexity information, requirements status
user requirements
User Representatives
Development
functional and nonfunctional requirements
Business Analyst
expectations and constraints
Other Stakeholders
functional and nonfunctional requirements
Testing

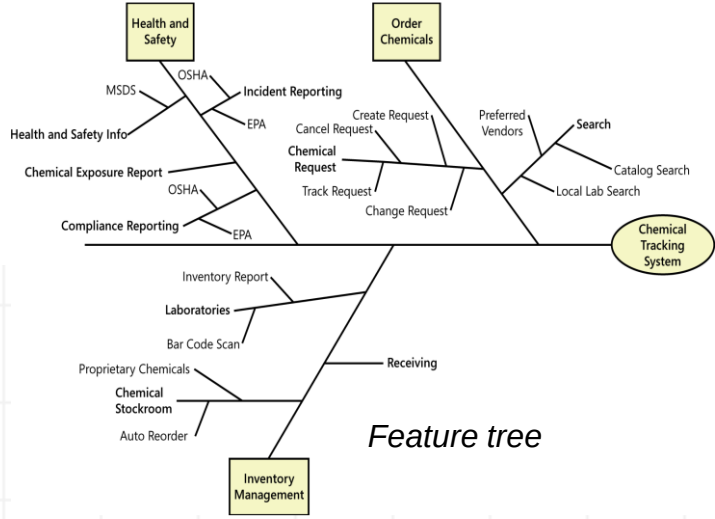

P2: Requirements development
Software Requirements



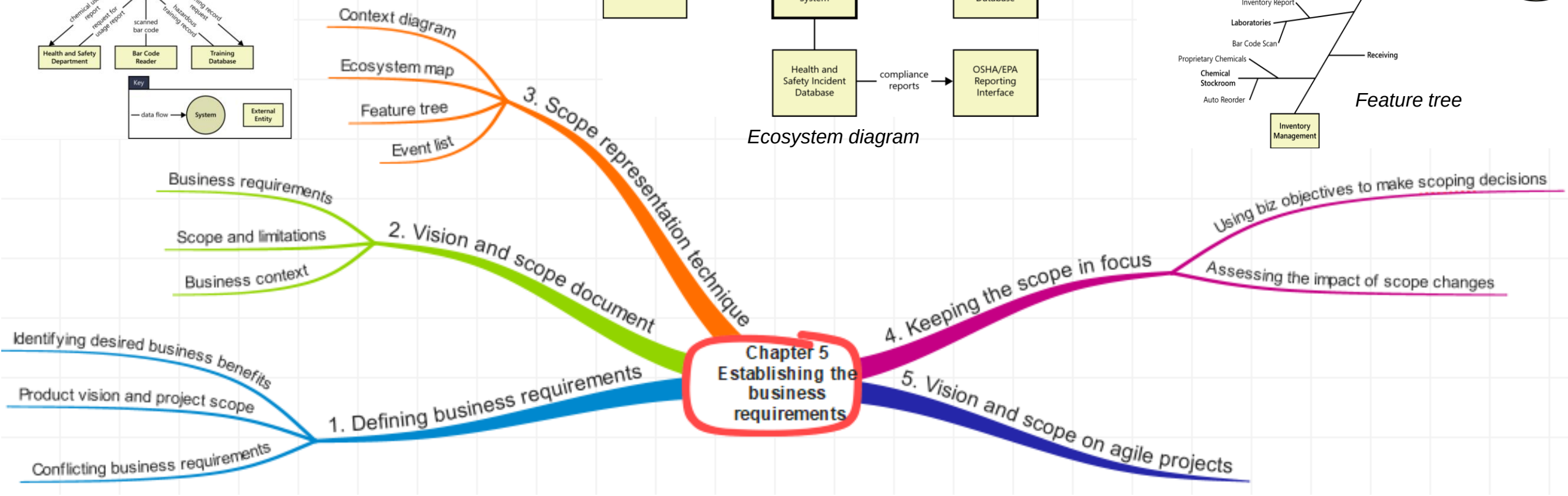
Context diagram



Ecosystem diagram



Feature tree



"**Biz requirements**" refers to a set of information that describes a need that leads to one or more projects to deliver a solution and the desired ultimate biz outcomes.
Biz opportunities, biz objectives, success metrics, and a vision statement make up the biz requirements.

Product vision succinctly (súc tích) describes the ultimate product that will achieve the biz objectives. The vision describes what the product is about and what it ultimately could become.

Project scope identifies what portion of the ultimately product vision the current project or development iteration will address. The statement of scope draws the boundary between what's in and what's out for the project.

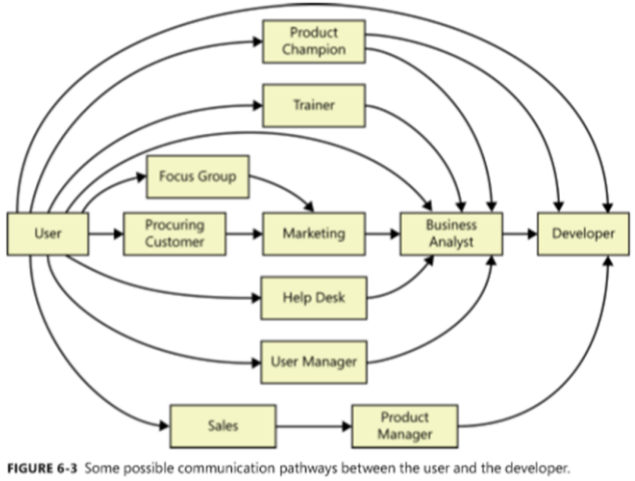


FIGURE 6-3 Some possible communication pathways between the user and the developer.

Each of our projects included a few key members of our user community to provide the requirements. We called these people ***product champions***.

If you encounter resistance, point out that insufficient user involvement is a leading cause of SW project failure.

A **persona** is a description of a hypothetical (có tính giả thuyết), generic person who serves as a stand-in (người đóng thế vai) for a group of users having similar characteristics and needs.

In agile projects, a single representative of stakeholders called ***product owner*** in the team to serve as the voice of the customer.

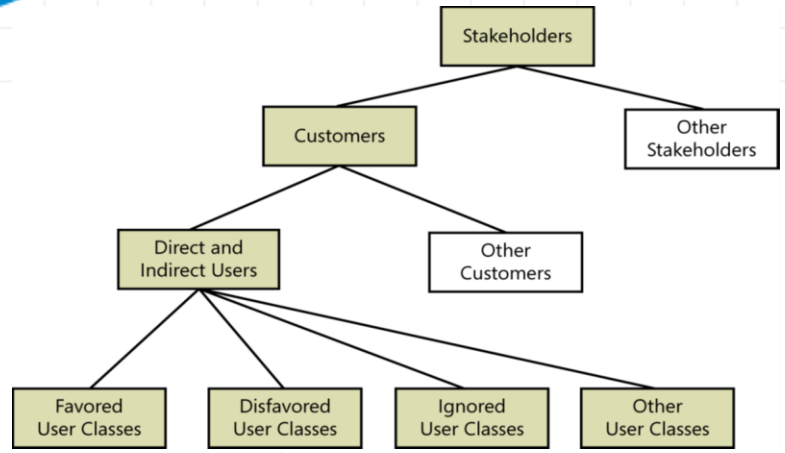
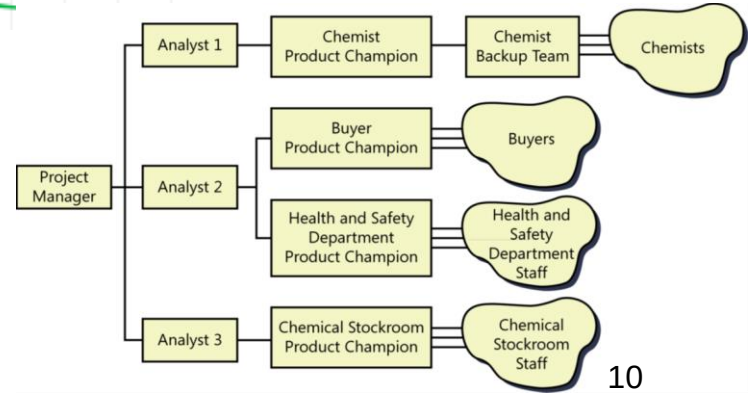


FIGURE 6-1 A hierarchy of stakeholders, customers, users, and user classes.



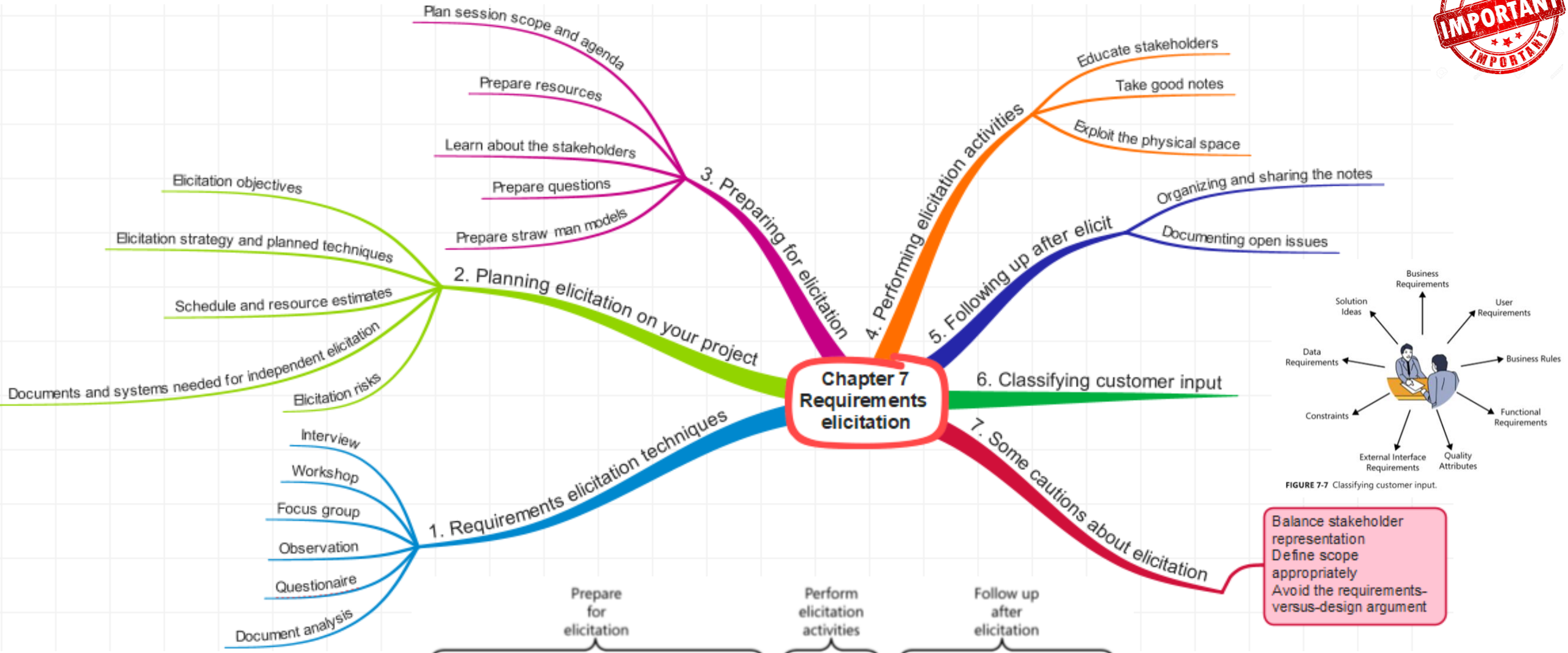


FIGURE 7-7 Classifying customer input.

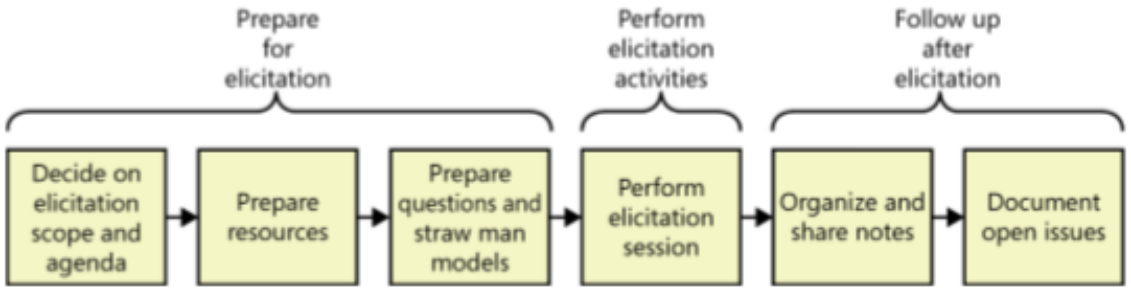
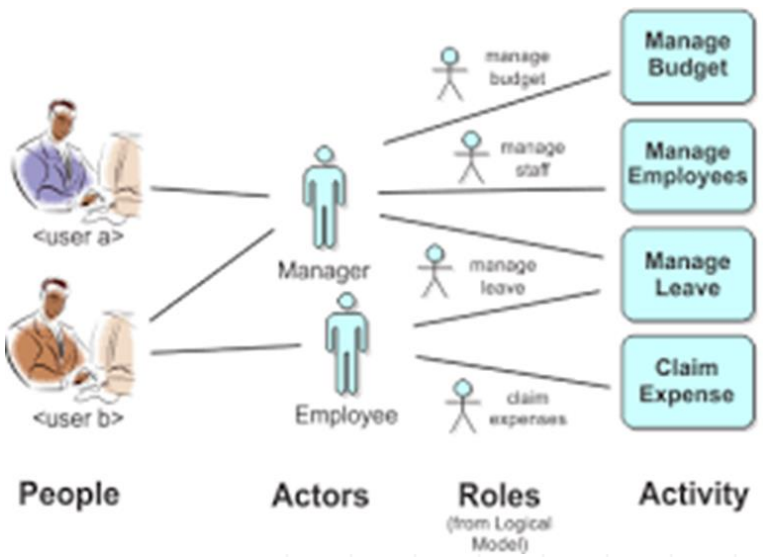
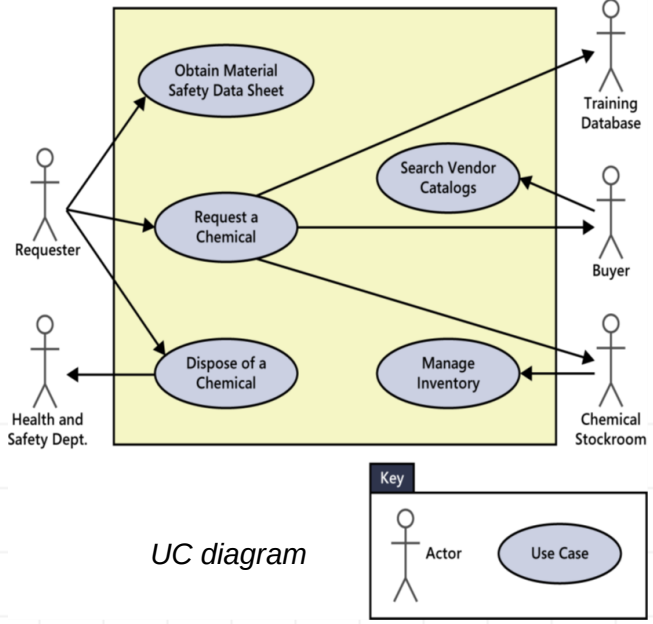


FIGURE 7-2 Activities for a single requirements elicitation session.





UC diagram

ID and Name:	UC-4 Request a Chemical
Created By:	Lori — Date Created: 8/22/13
Primary Actor:	Requester — Secondary Actors: Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.
Trigger:	Requester indicates that he wants to request a chemical.
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.
Normal Flow:	**4.0 Request a Chemical from the Chemical Stockroom** 1. Requester specifies the desired chemical. 2. System lists containers of the desired chemical that are in the chemical stockroom, if any. 3. System gives Requester the option to View Container History for any container. 4. Requester selects a specific container or asks to place a vendor order (see 4.1). 5. Requester enters other information to complete the request. 6. System stores the request and notifies the Chemical Stockroom.
Alternative Flows:	**4.1 Request a Chemical from a Vendor** 1. Requester searches vendor catalogs for the chemical (see 4.1.E1). 2. System displays a list of vendors for the chemical with available container sizes, grades, and prices. 3. Requester selects a vendor, container size, grade, and number of containers. 4. Requester enters other information to complete the request. 5. System stores the request and notifies the Buyer.
Exceptions:	**4.1.E1 Chemical Is Not Commercially Available** 1. System displays message: No vendors for that chemical. 2. System asks Requester if he wants to request another chemical (3a) or to exit (4a). 3a. Requester asks to request another chemical. 3b. System starts normal flow over. 4a. Requester asks to exit. 4b. System terminates use case.
Priority:	High
Frequency of Use:	Approximately 5 times per week by each chemist, 200 times per week by chemical stockroom staff
Business Rules:	BR-28, BR-31
Other Information:	The system must be able to import a chemical structure in the standard encoded form from any of the supported chemical drawing packages.
Assumptions:	Imported chemical structures are assumed to be valid.

A use case describes a sequence of interactions between a system and an external actor that results in the actor being able to achieve some outcome of value.

As used on agile project, a ***user story*** is a "short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system..

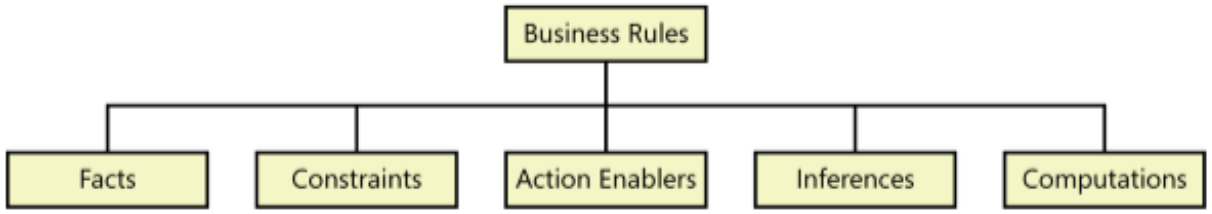
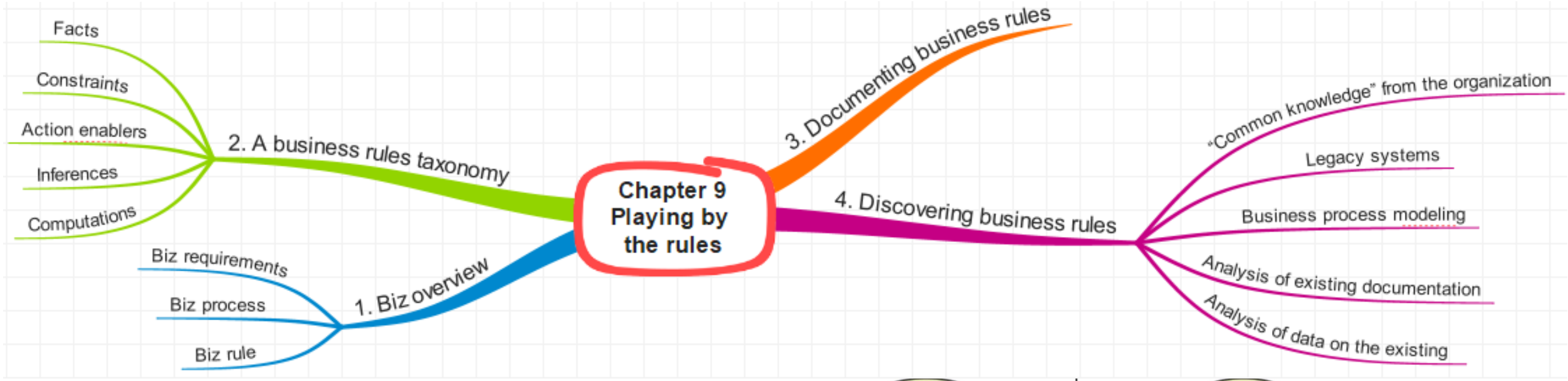


FIGURE 9-1 A simple business rule taxonomy.



Biz requirement states desirable outcome or a high-level objective of the organization that builds or procures a SW solution.

Biz process describes a series of activities that transform inputs into outputs to achieve a specific result.

Biz rules influence biz processes by establishing vocabulary, imposing restriction, triggering actions, and governing (điều chỉnh) how computations are carried out.

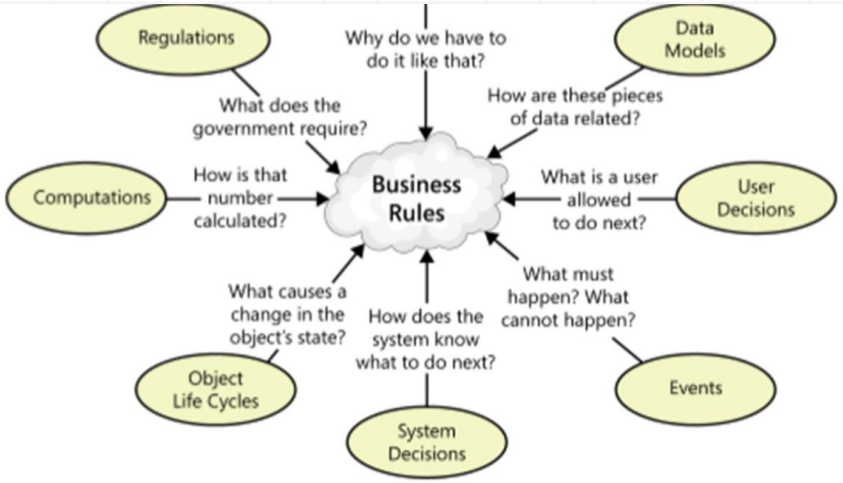
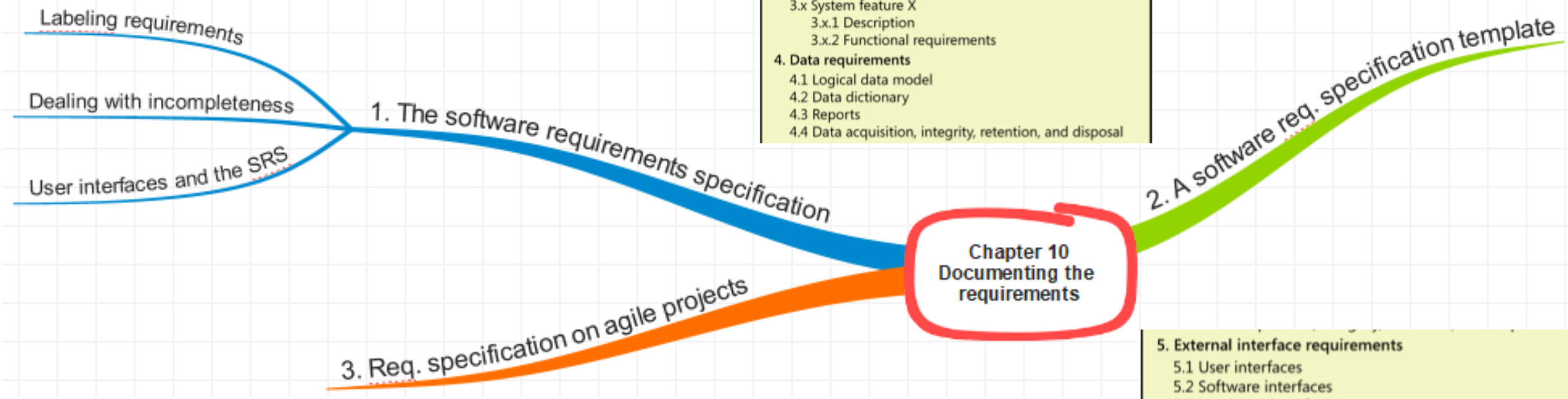


FIGURE 9-3 Discovering business rules by asking questions from different perspectives.

The **SRS** states the functions and capabilities that a SW system must provide, its characters, and the constraints that it must respect.

The **SRS** is the basis for subsequent project planning, design, and coding, as well as the foundation for system testing and user documentation..



1. Introduction
   - 1.1 Purpose
   - 1.2 Document conventions
   - 1.3 Project scope
   - 1.4 References
2. Overall description
   - 2.1 Product perspective
   - 2.2 User classes and characteristics
   - 2.3 Operating environment
   - 2.4 Design and implementation constraints
   - 2.5 Assumptions and dependencies
3. System features
   - 3.x System feature X
     - 3.x.1 Description
     - 3.x.2 Functional requirements
4. Data requirements
   - 4.1 Logical data model
   - 4.2 Data dictionary
   - 4.3 Reports
   - 4.4 Data acquisition, integrity, retention, and disposal

5. External interface requirements
   - 5.1 User interfaces
   - 5.2 Software interfaces
   - 5.3 Hardware interfaces
   - 5.4 Communications interfaces
6. Quality attributes
   - 6.1 Usability
   - 6.2 Performance
   - 6.3 Security
   - 6.4 Safety
   - 6.x [others]
7. Internationalization and localization requirements
8. Other requirements

Appendix A: Glossary

Appendix B: Analysis models

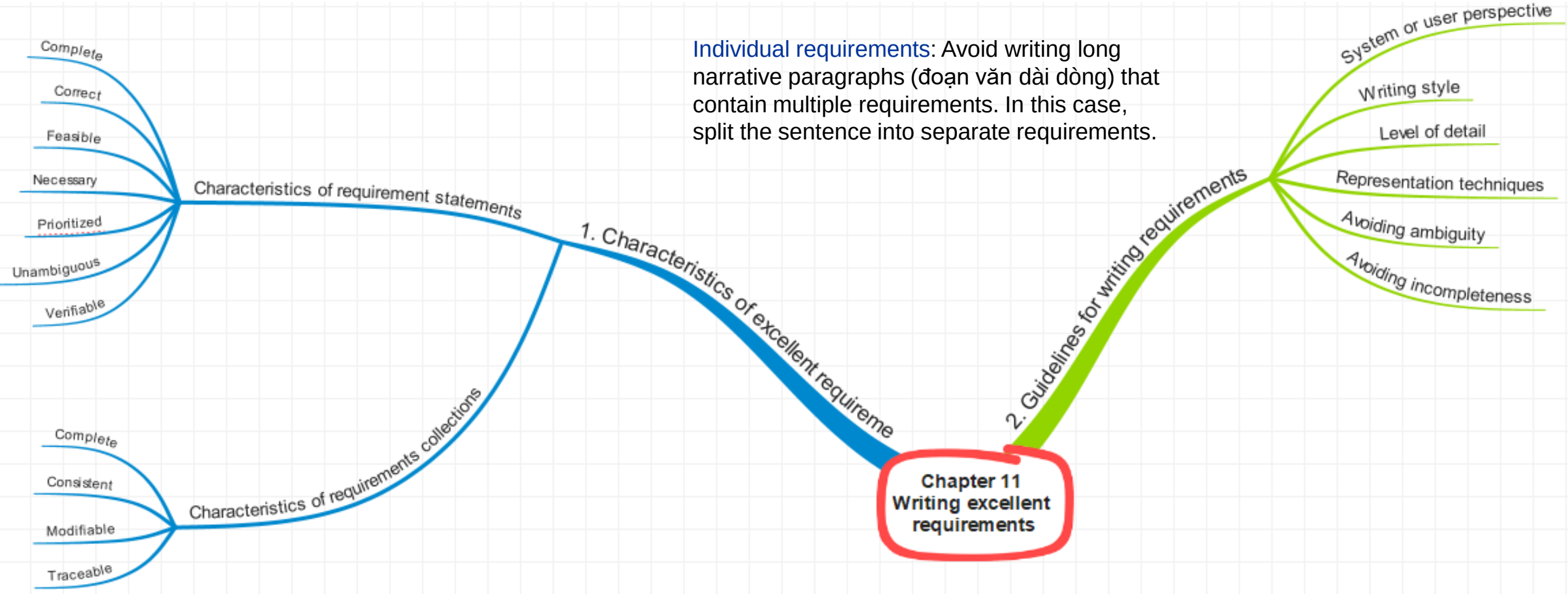
Dealing with incompleteness: Sometimes you know that you lack a piece of information about specific requirement. Use the notation TBD (to be determined) to flag these knowledge gaps. Plan to resolve all TBDs before implementing a set of requirements.

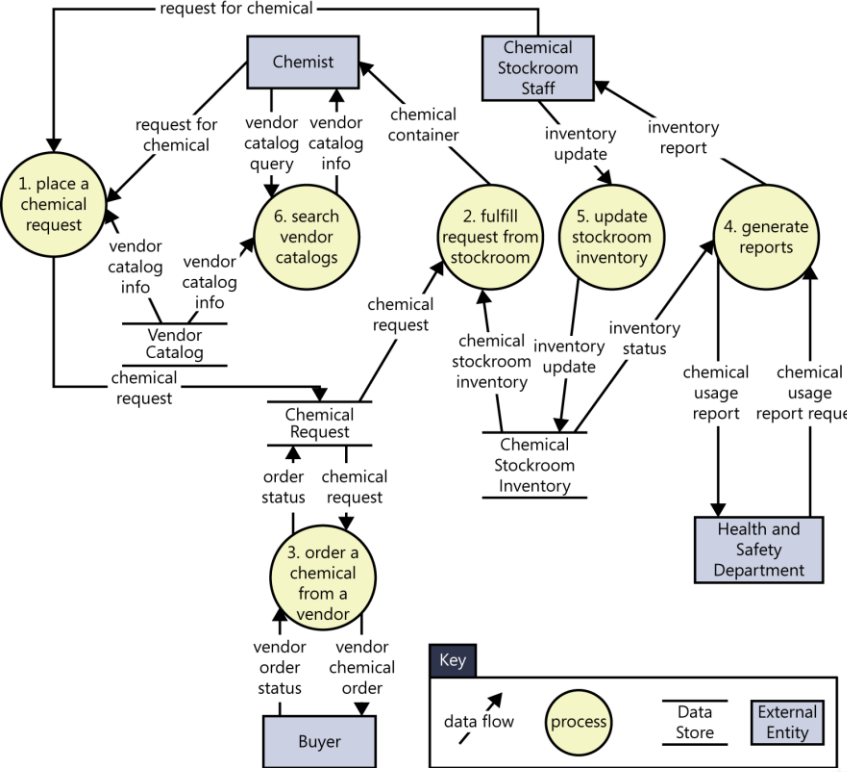
Clarity and conciseness: Write requirements in complete sentences using proper grammar, spelling, and punctuation. Keep sentences and paragraphs sort and direct, avoiding jargon. Define specialized terms in a glossary.

The keyword "shall": In general, the BAs often use "shall" in writing requirements.

Active voice: Write in the active voice to make it clear what entity is taking the action described.

Individual requirements: Avoid writing long narrative paragraphs (đoạn văn dài dòng) that contain multiple requirements. In this case, split the sentence into separate requirements.





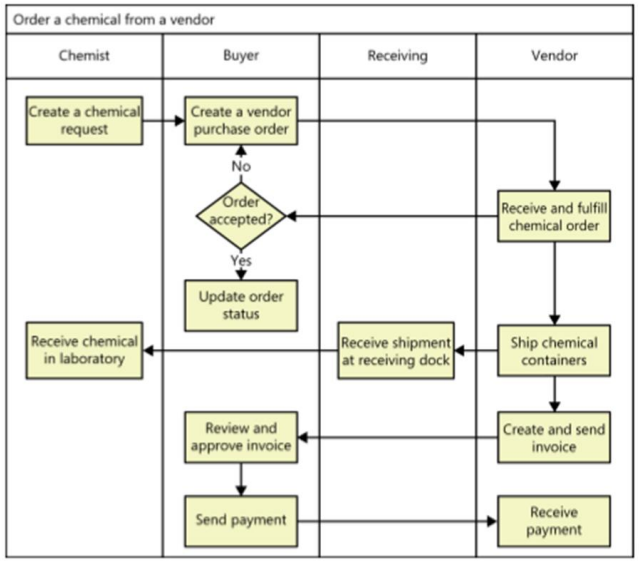


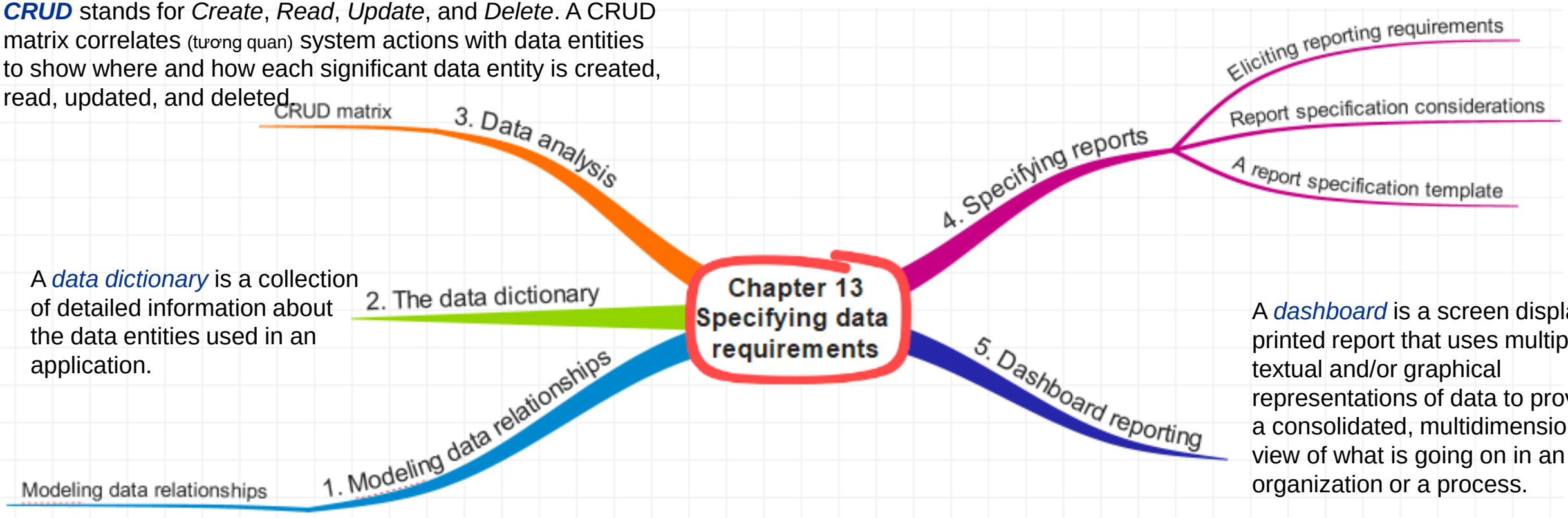
FIGURE 12-2 Partial swimlane diagram for a process in the Chemical Tracking System.

Diagrams communicate certain types of information more efficiently than text can.

Visual requirements models can help you identify missing, extraneous (ko liên quan), and inconsistent requirements.

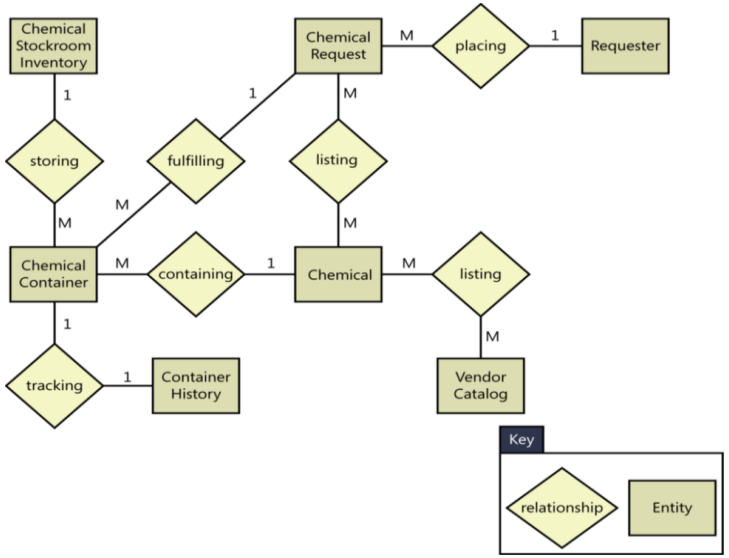
FIGURE 12-3 A partial state-transition diagram for a chemical request in the Chemical Tracking System.

CRUD stands for *Create*, *Read*, *Update*, and *Delete*. A CRUD matrix correlates (tương quan) system actions with data entities to show where and how each significant data entity is created, read, updated, and deleted.



A *data dictionary* is a collection of detailed information about the data entities used in an application.

A *dashboard* is a screen display or printed report that uses multiple textual and/or graphical representations of data to provide a consolidated, multidimensional view of what is going on in an organization or a process.

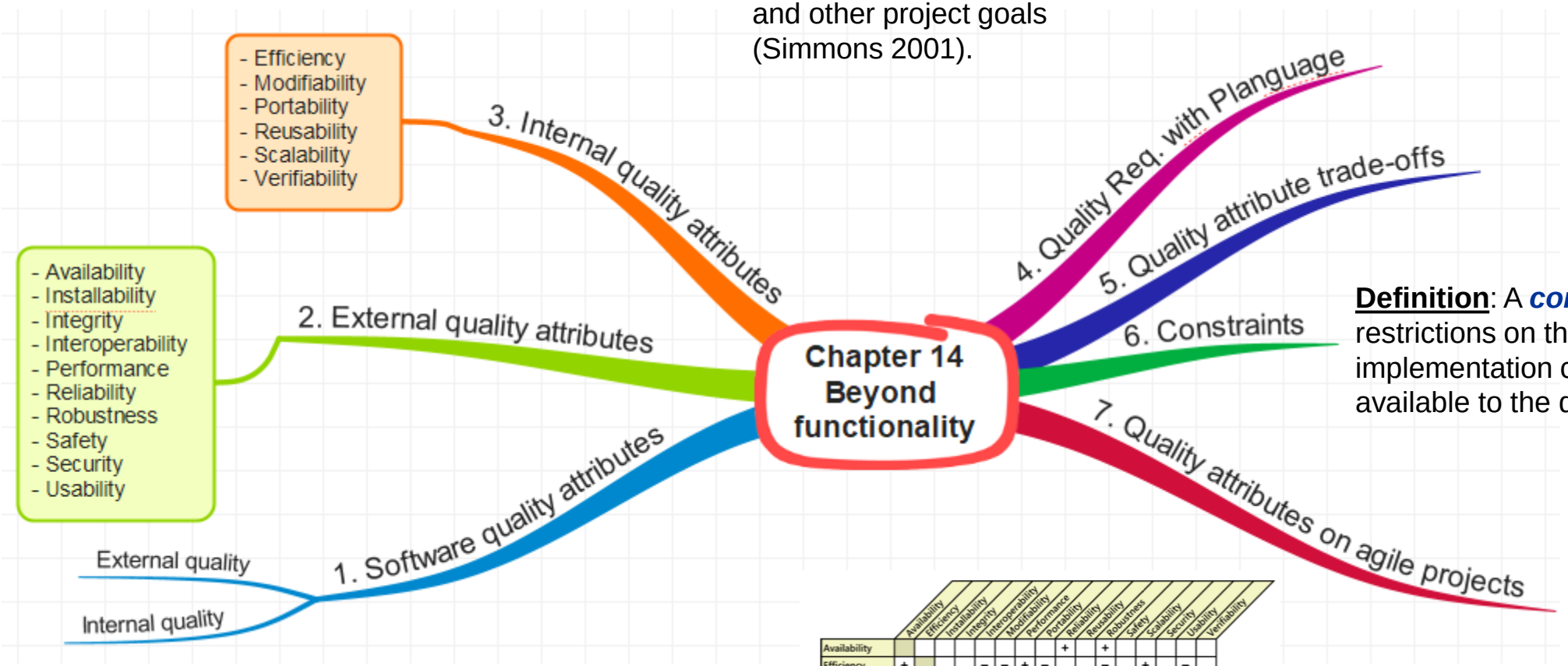


Use Case \ Entity	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R
Change Order	U, D		R	R
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R	R	
Edit Requesters			C, U	

FIGURE 13-5 Sample CRUD matrix for the Chemical Tracking System.

Definition: ***Planguage***, a language with a rich set of keywords that permits precise statements of quality attributes and other project goals (Simmons 2001).

Purpose: address the problem of ambiguous and incomplete nonfunctional requirements.



Definition: A ***constraint*** places restrictions on the design or implementation choices available to the developer.

External quality factors are primarily important to users, whereas internal qualities are more significant to development and maintenance staff.

Internal quality attributes indirectly contribute to customer satisfaction by making the product easier to enhance, correct, test, and migrate to new platforms.

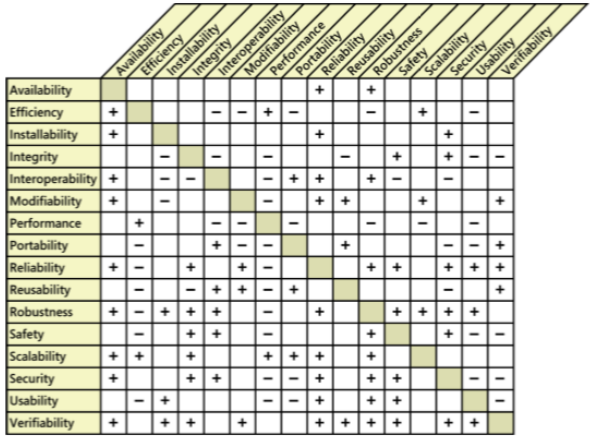


FIGURE 14-2 Positive and negative relationships among selected quality attributes.

In agile projects, the develop requirements and deliver functionality in small increments need to specify significant quality attributes and constraints early in the project.

A mock-up is also called a *horizontal prototype*. Such a prototype focuses on a portion of the user interface; it doesn't dive into (đi sâu vào) all the architectural layers or into detailed functionality.

A proof of concept, also known as a *vertical prototype*, implements a slice of application functionality from the user interface through all the technical services layers..

Build a *throwaway prototype* to answer questions, resolve uncertainties, and improve requirements quality, it is used just only once time.

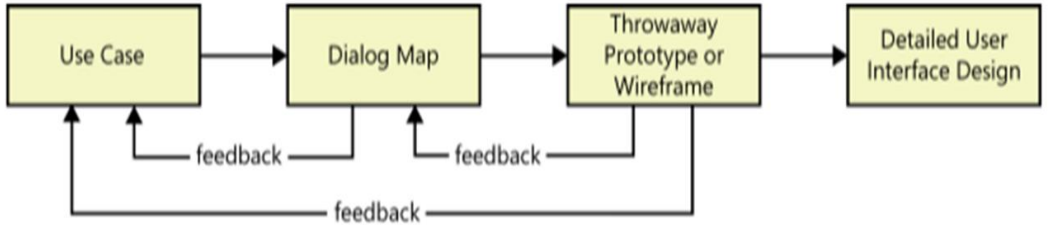
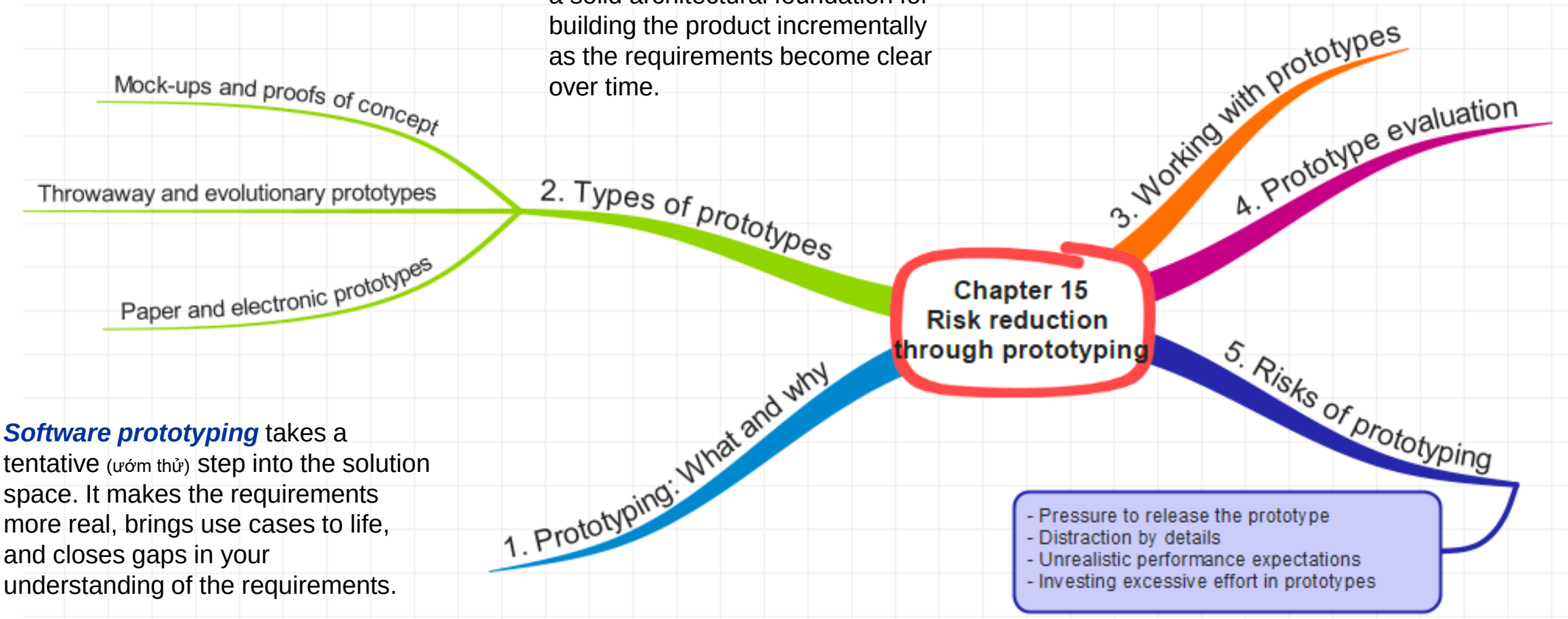


FIGURE 15-2 Activity sequence from use cases to user interface design using a throwaway prototype.

An *evolutionary prototype* provides a solid architectural foundation for building the product incrementally as the requirements become clear over time.

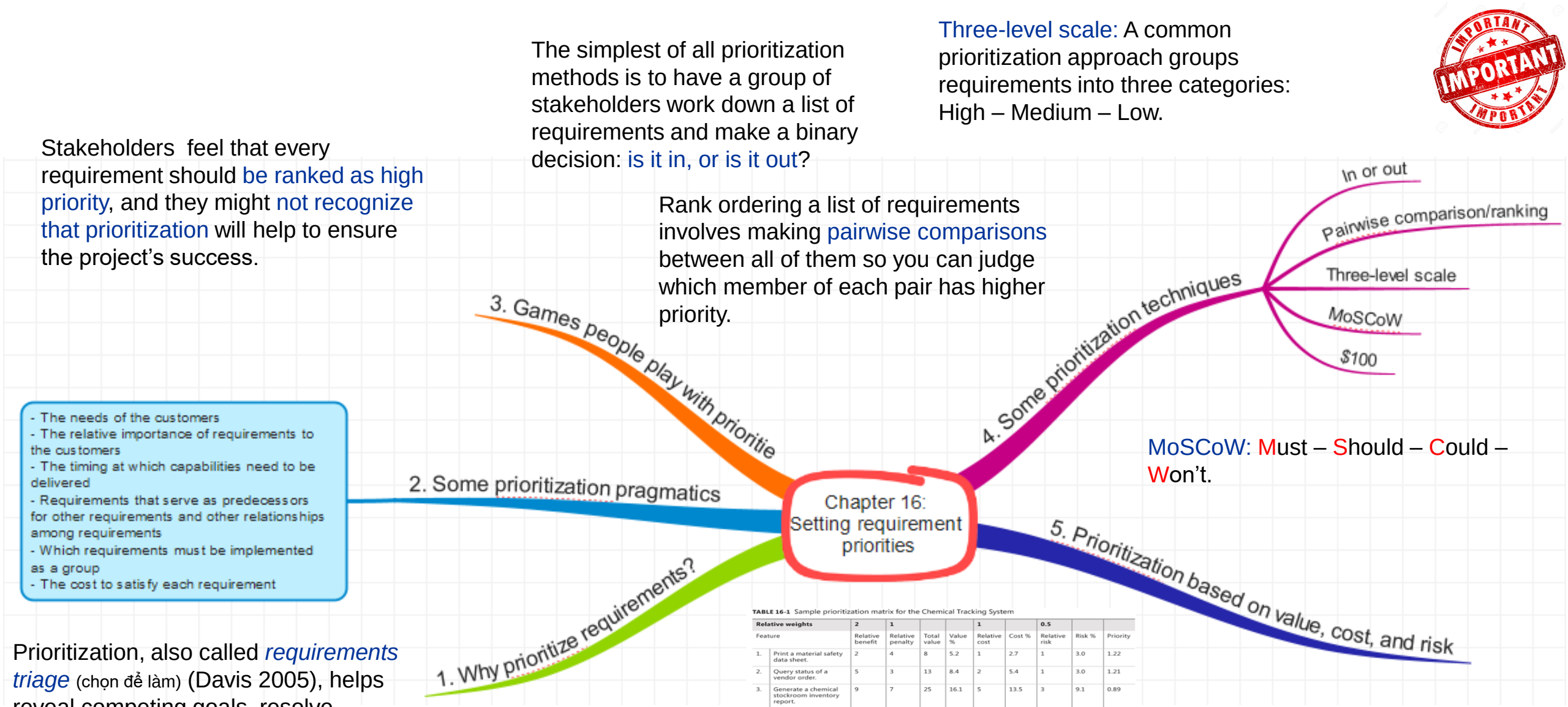


Software prototyping takes a tentative (ướm thử) step into the solution space. It makes the requirements more real, brings use cases to life, and closes gaps in your understanding of the requirements.

Stakeholders feel that every requirement should be ranked as high priority, and they might not recognize that prioritization will help to ensure the project's success.

The simplest of all prioritization methods is to have a group of stakeholders work down a list of requirements and make a binary decision: is it in, or is it out?

Three-level scale: A common prioritization approach groups requirements into three categories: High – Medium – Low.



MoSCoW: Must – Should – Could – Won't.

Prioritization, also called *requirements triage* (chọn để làm) (Davis 2005), helps reveal competing goals, resolve conflicts, plan for staged or incremental deliveries, control scope creep, and make the necessary trade-off decisions (QĐ đánh đổi)

TABLE 16-1 Sample prioritization matrix for the Chemical Tracking System

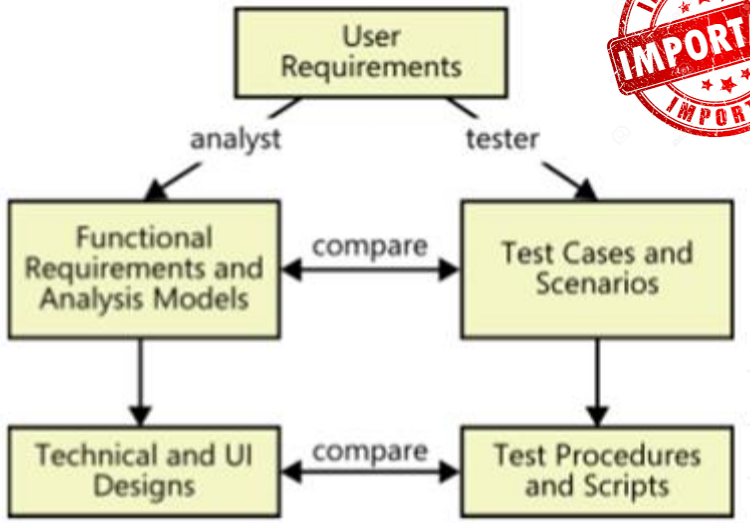
	Relative weights	2	1			1		0.5		
	Feature	Relative benefit	Relative penalty	Total value	Value %	Relative cost	Cost %	Relative risk	Risk %	Priority
1.	Print a material safety data sheet.	2	4	8	5.2	1	2.7	1	3.0	1.22
2.	Query status of a vendor order.	5	3	13	8.4	2	5.4	1	3.0	1.21
3.	Generate a chemical stockroom inventory report.	9	7	25	16.1	5	13.5	3	9.1	0.89
4.	See history of a specific chemical container.	5	5	15	9.7	3	8.1	2	6.1	0.87
5.	Search vendor catalogs for a specific chemical.	9	8	26	16.8	3	8.1	8	24.2	0.83
6.	Maintain a list of hazardous chemicals.	3	9	15	9.7	3	8.1	4	12.1	0.68
7.	Change a pending chemical request.	4	3	11	7.1	3	8.1	2	6.1	0.64
8.	Generate a laboratory inventory report.	6	2	14	9.0	4	10.8	3	9.1	0.59
9.	Check training database for hazardous chemical training record.	3	4	10	6.5	4	10.8	2	6.1	0.47
10.	Import chemical structures from structure drawing tools.	7	4	18	11.6	9	24.3	7	21.2	0.33
	Totals	53	49	155	100.0	37	100.0	33	100.0	

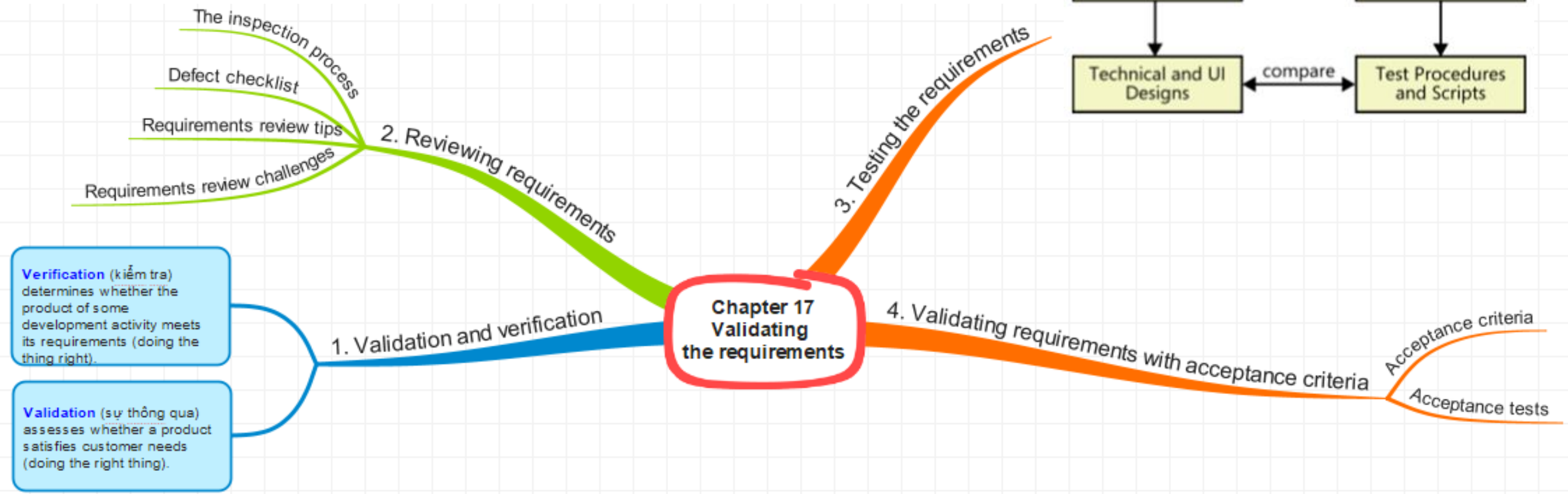
The inspection process:

- Who are participants? Max: 7 people
- Inspection roles: Author, Moderator, Reader, Recorder.
- Entry criteria: used to preparing for an inspection.
- Inspection stages: Planning → Preparation → Inspection Meeting → Rework → Follow up

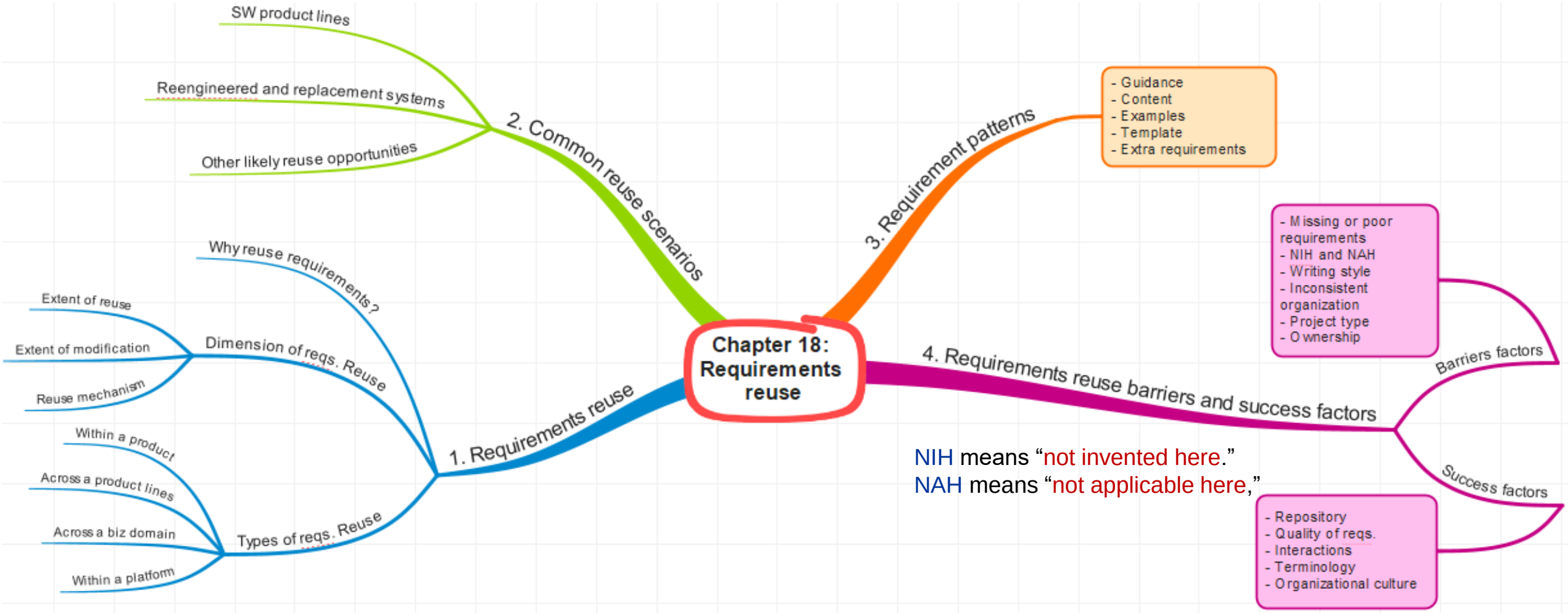
Requirements review challenges:

- Large requirements documents
- Large inspection teams
- Geographically separated reviewers
- Unprepared reviewers





One study found that it took an average of 30 minutes to fix a defect discovered during the requirements phase. In contrast, 5 to 17 hours were needed to correct a defect identified during system testing



NIH means "not invented here."
NAH means "not applicable here,"

Why? Faster delivery, lower development costs, consistency both within and across applications, higher team productivity, fewer defects, and reduced rework

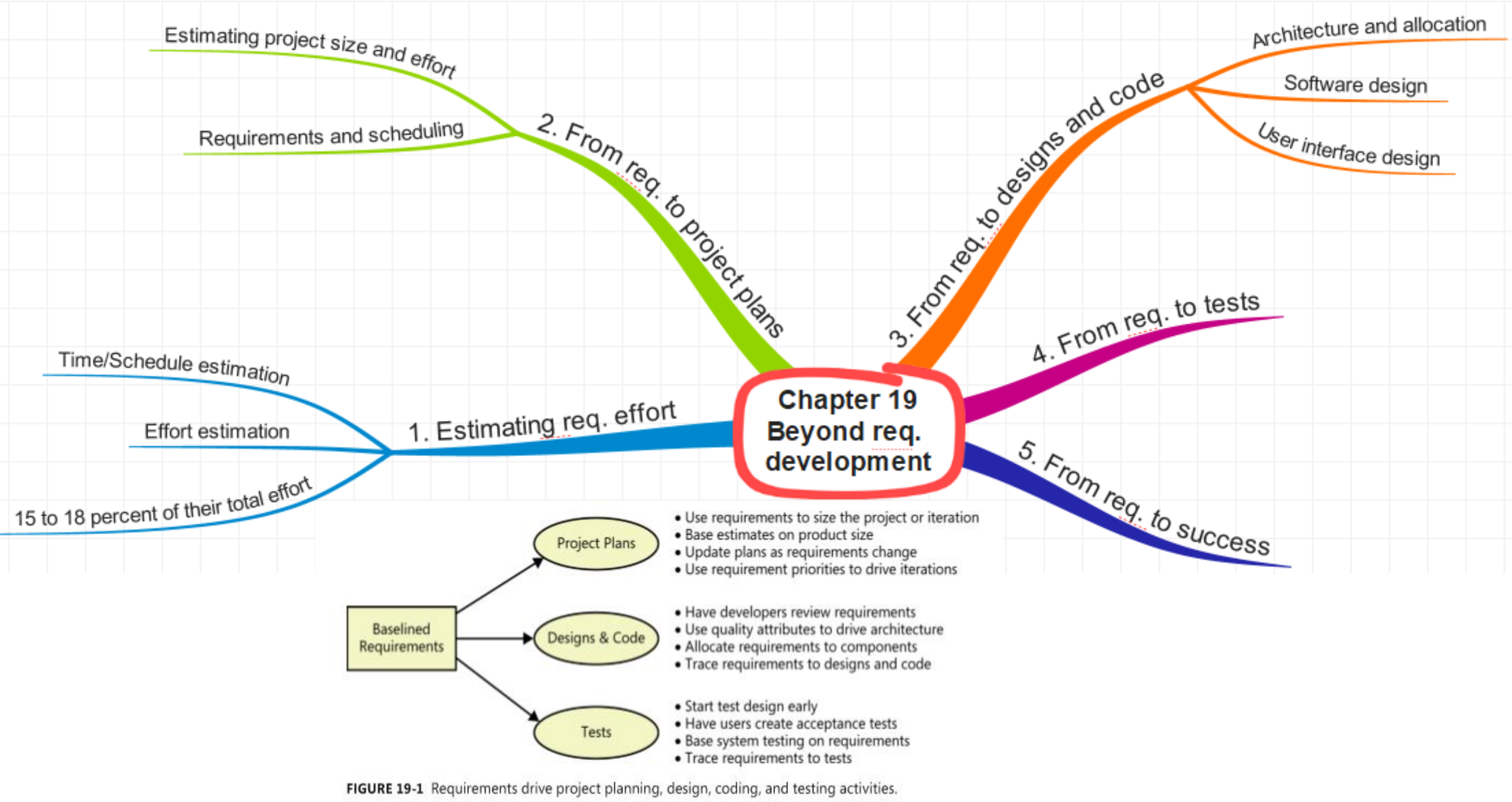
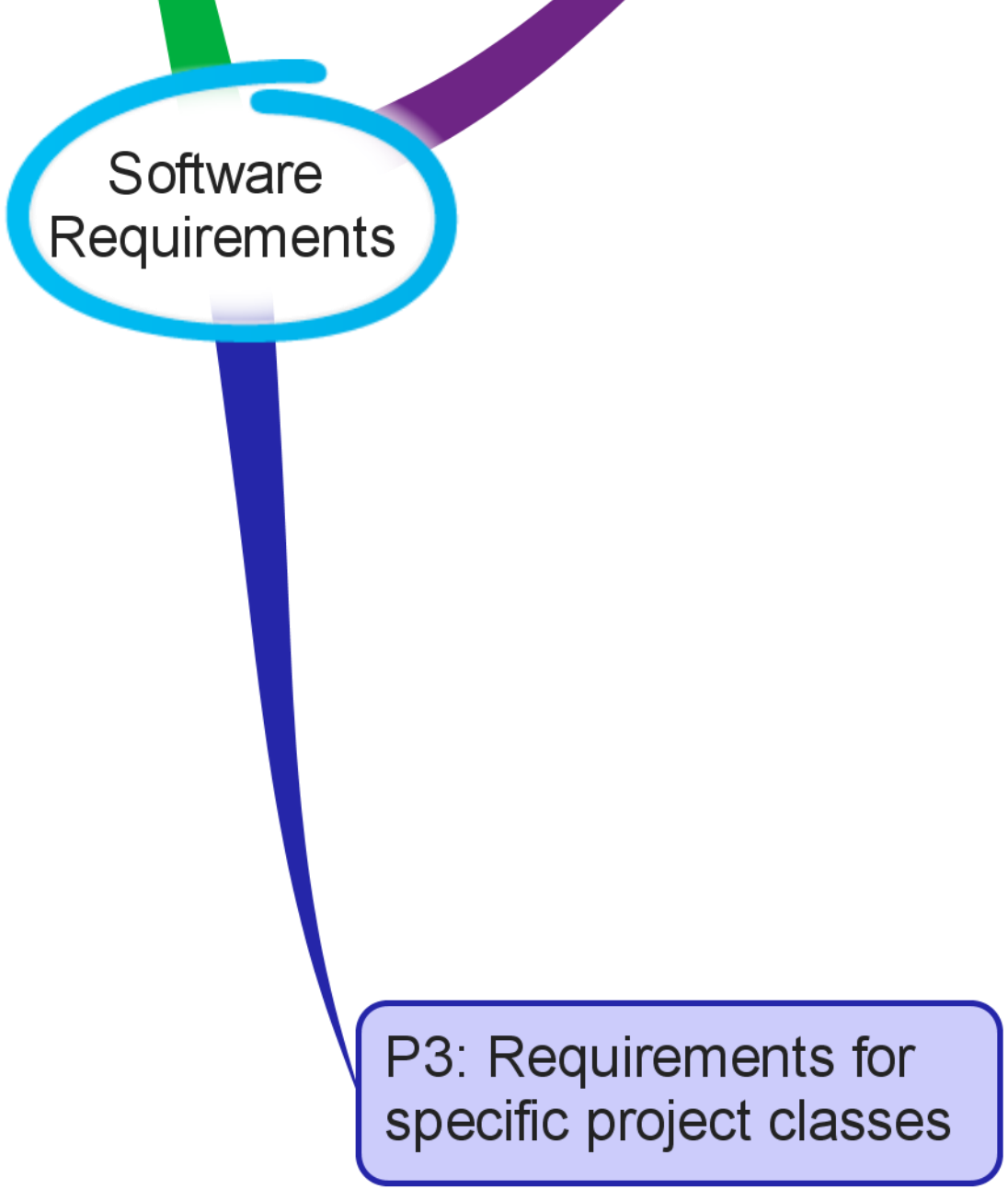


FIGURE 19-1 Requirements drive project planning, design, coding, and testing activities.


Software Requirements
P3: Requirements for specific project classes

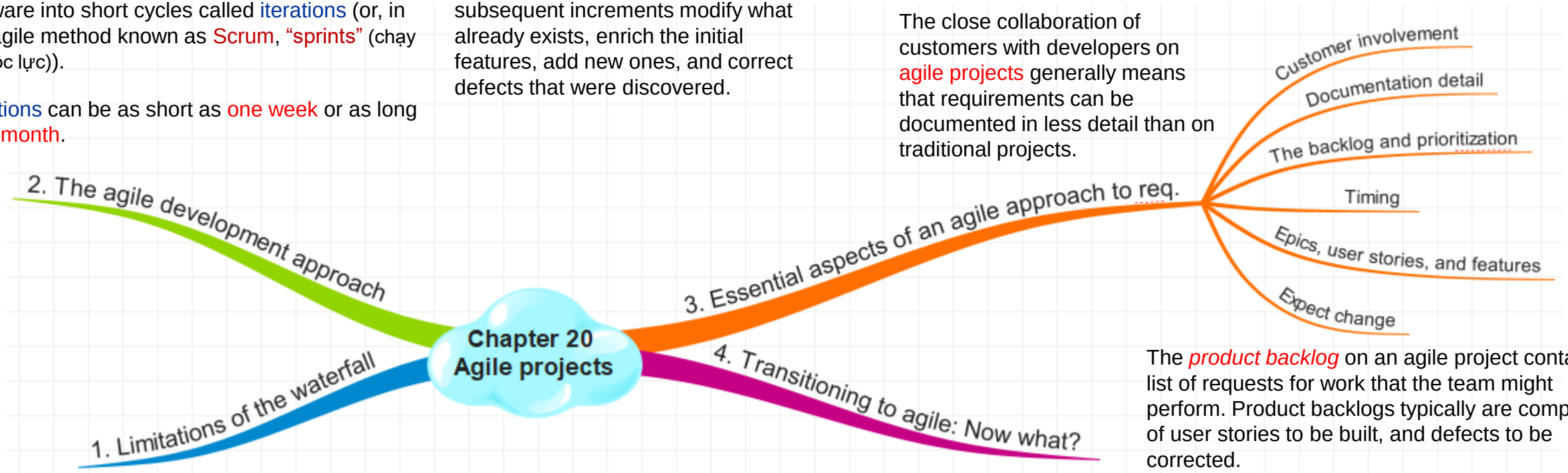
Agile methods focus on iterative and incremental development, breaking the development of software into short cycles called iterations (or, in the agile method known as Scrum, "sprints" (chạy hết tốc lực)).

Iterations can be as short as one week or as long as a month.

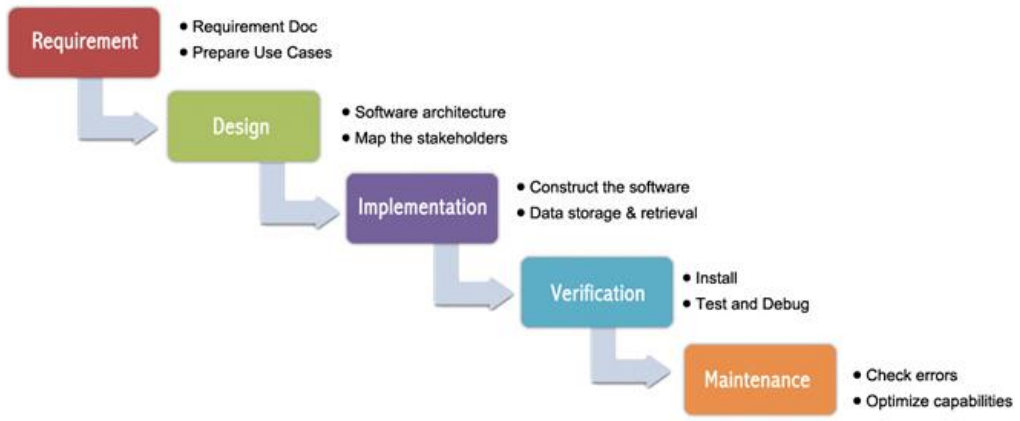
Benefits: Minimum the defects at the final product, because of subsequent increments modify what already exists, enrich the initial features, add new ones, and correct defects that were discovered.

On agile projects, customers (or a product owner) are engaged continuously throughout the project.

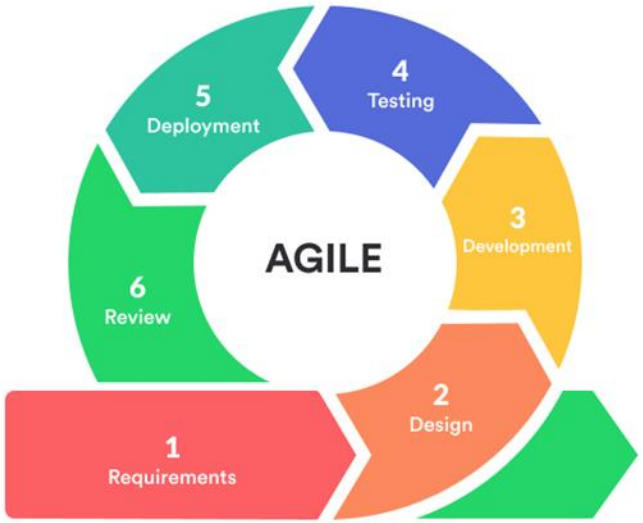
The close collaboration of customers with developers on agile projects generally means that requirements can be documented in less detail than on traditional projects.

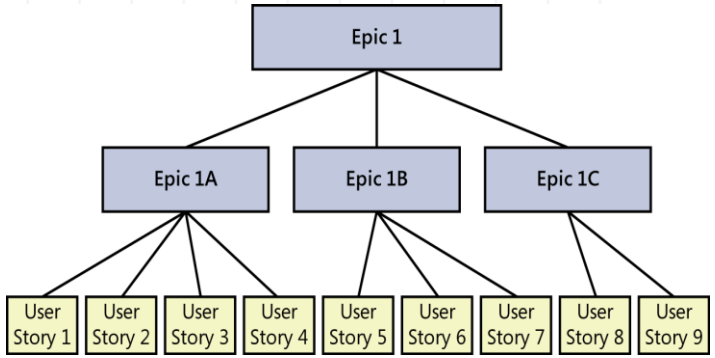


The *product backlog* on an agile project contains a list of requests for work that the team might perform. Product backlogs typically are composed of user stories to be built, and defects to be corrected.



Waterfall model





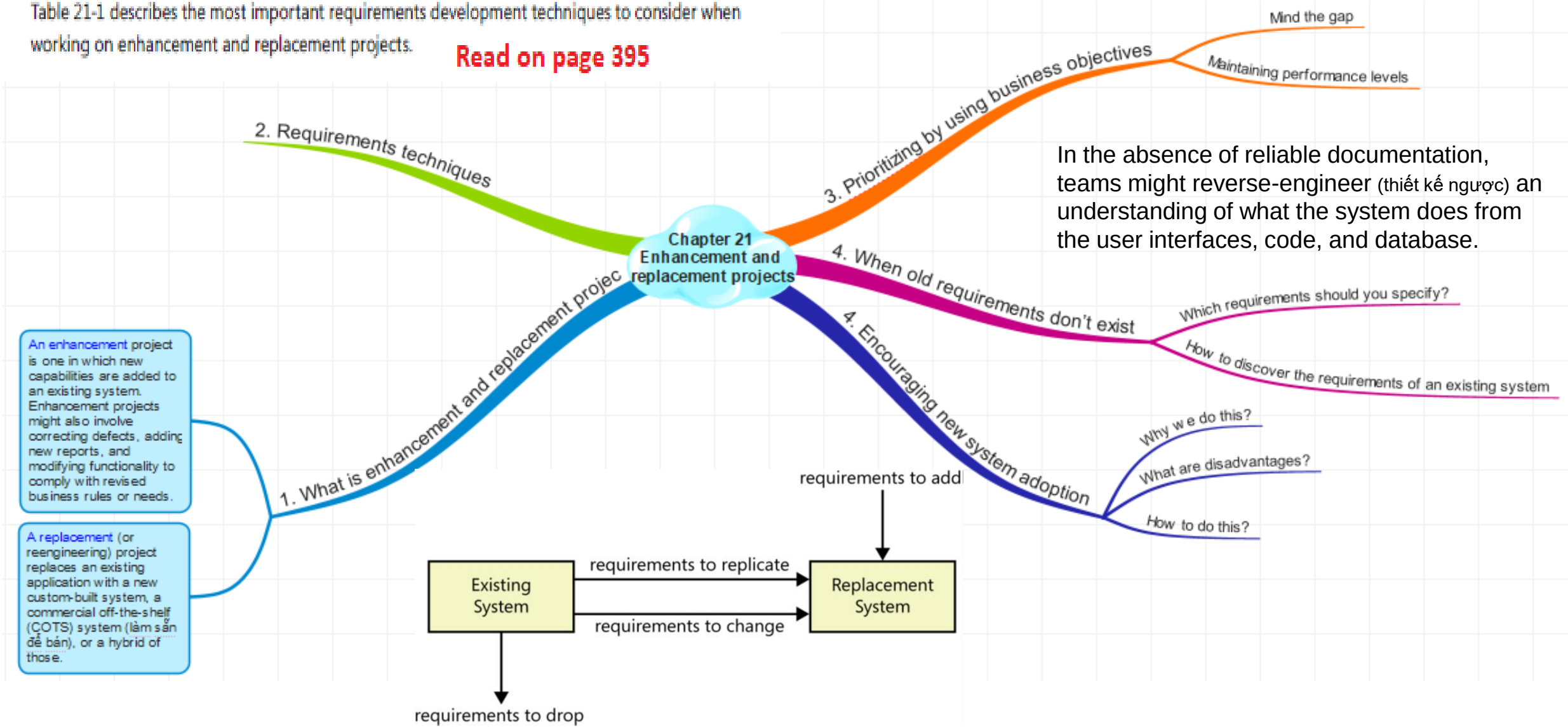
It encourages customer collaboration to create or change user stories and prioritize each change request against everything else that's already in the backlog.

Requirements techniques when there is an existing system

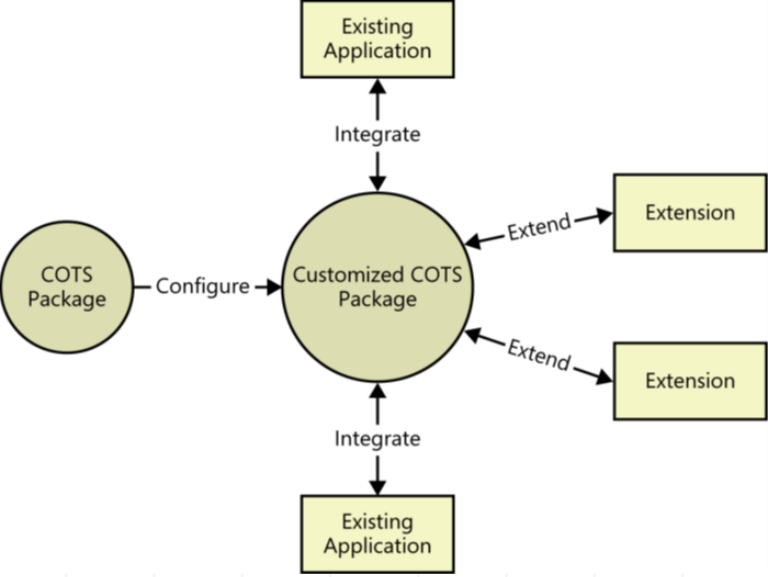
A ***gap analysis*** is a comparison of functionality between an existing system and a desired new system.

Table 21-1 describes the most important requirements development techniques to consider when working on enhancement and replacement projects.

Read on page 395



In the absence of reliable documentation, teams might reverse-engineer (thiết kế ngược) an understanding of what the system does from the user interfaces, code, and database.



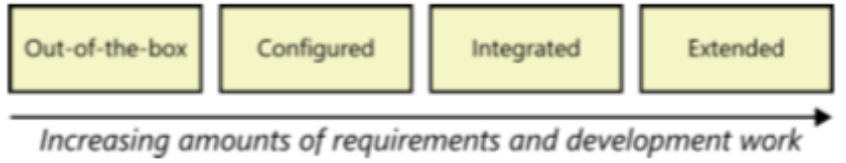
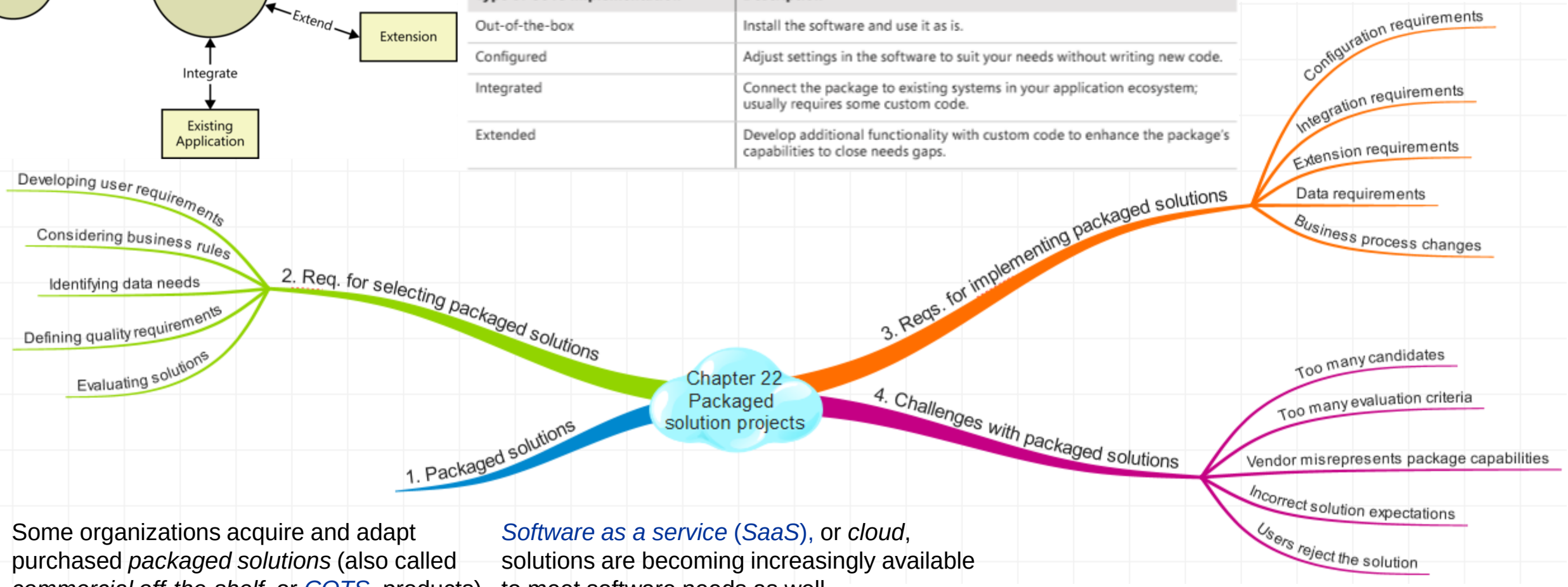


FIGURE 22-3 A spectrum of implementation effort for packaged solutions.

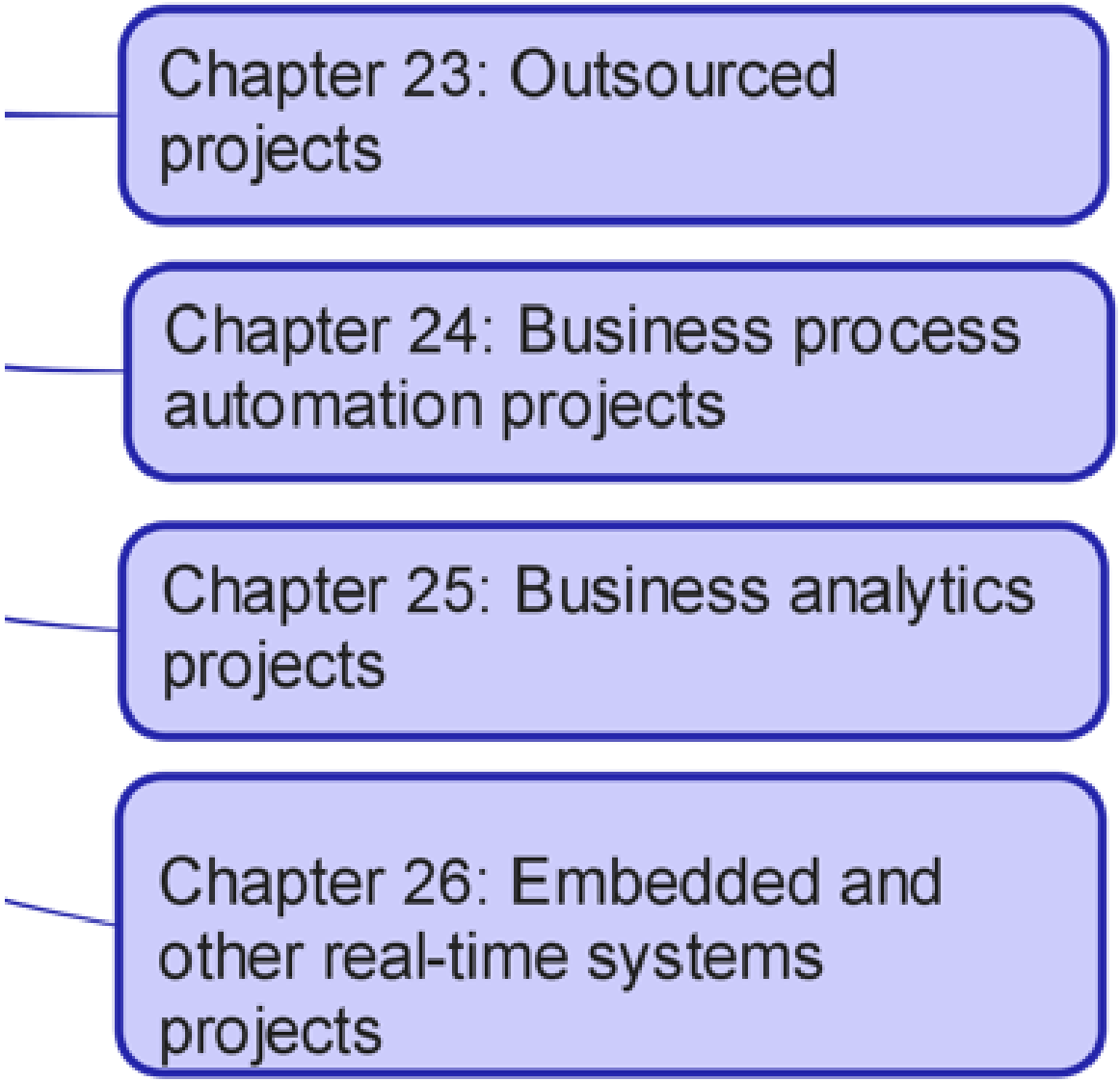
TABLE 22-1 COTS package implementation approaches

Type of COTS implementation	Description
Out-of-the-box	Install the software and use it as is.
Configured	Adjust settings in the software to suit your needs without writing new code.
Integrated	Connect the package to existing systems in your application ecosystem; usually requires some custom code.
Extended	Develop additional functionality with custom code to enhance the package's capabilities to close needs gaps.

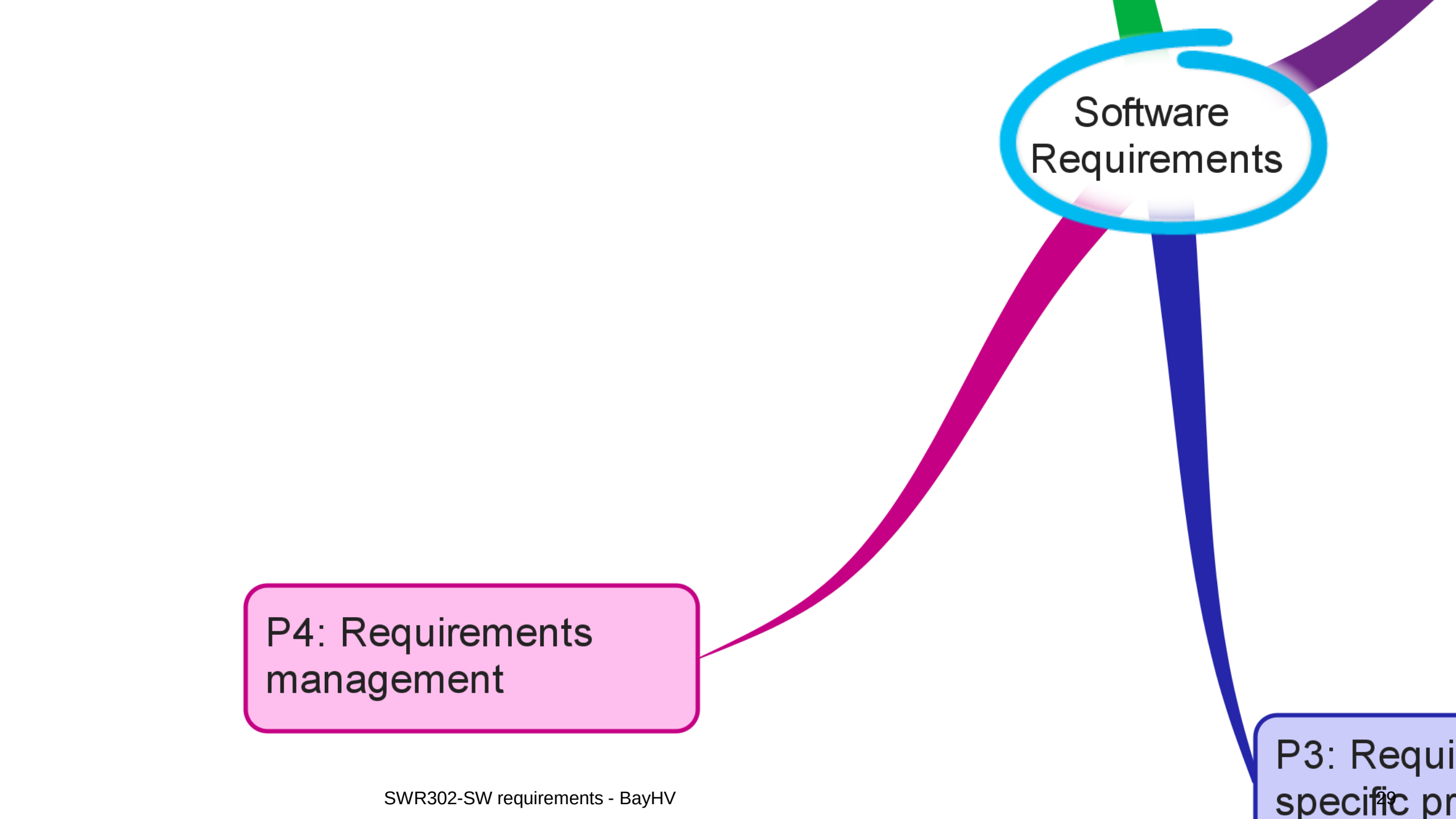


Some organizations acquire and adapt purchased *packaged solutions* (also called *commercial off-the-shelf*, or *COTS*, products) to meet their software needs, instead of building new systems from scratch (từ đầu).

Software as a service (*SaaS*), or *cloud*, solutions are becoming increasingly available to meet software needs as well.


Chapter 23: Outsourced projects
Chapter 24: Business process automation projects
Chapter 25: Business analytics projects
Chapter 26: Embedded and other real-time systems projects

Self study


Software Requirements
P4: Requirements management
P3: Requi
specific pr

Version control—uniquely identifying different versions of an item—applies at the level of both individual requirements and requirements sets, most commonly represented in the form of documents.

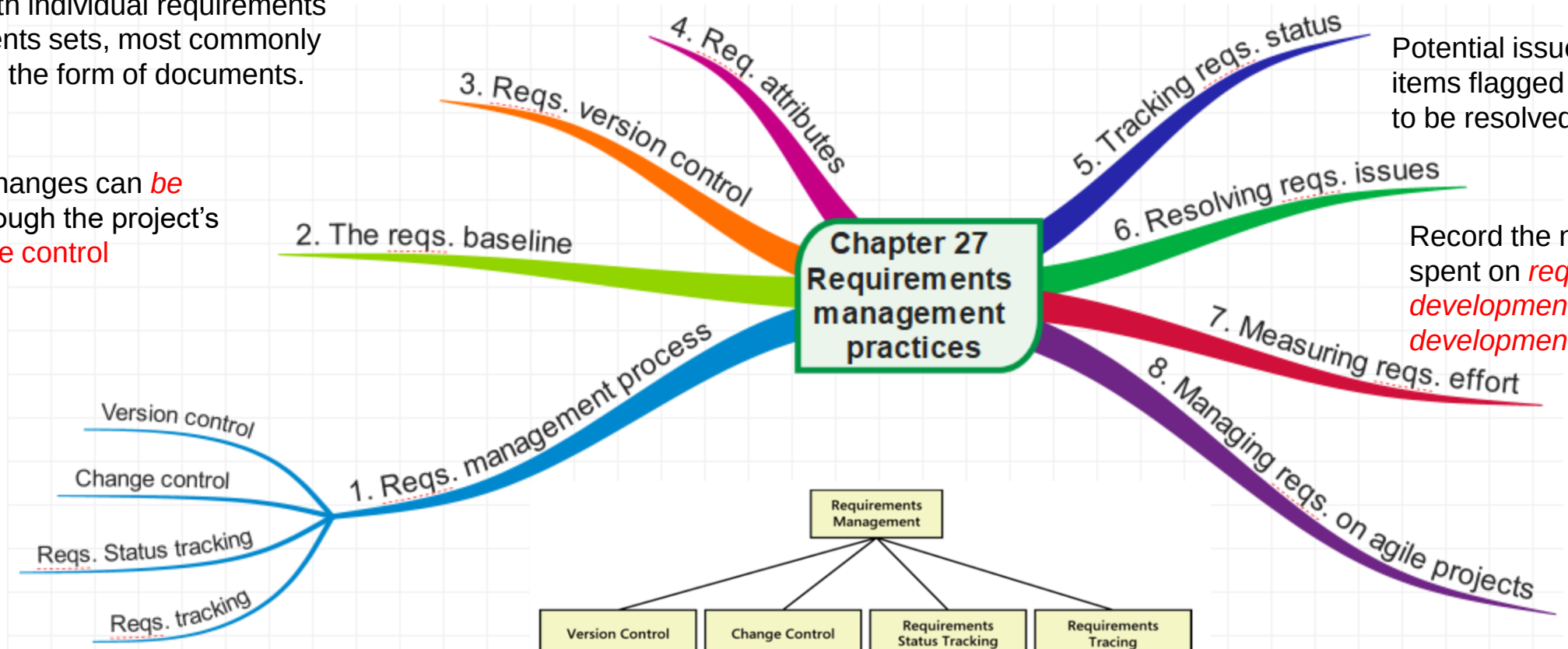
Think of each requirement as an object with properties that distinguish it from other requirements.

Tracking status means comparing where you really are at a particular time against (so với) the expectation of what "complete" means for this development cycle.

Potential issues include items flagged as TBD need to be resolved.

Subsequent changes can *be made only* through the project's defined change control process.

Record the number of hours spent on *requirements development and development activities*.



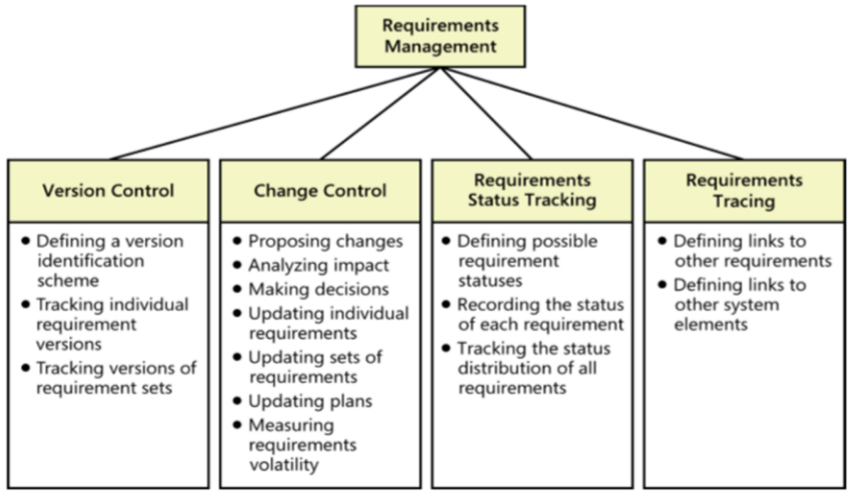
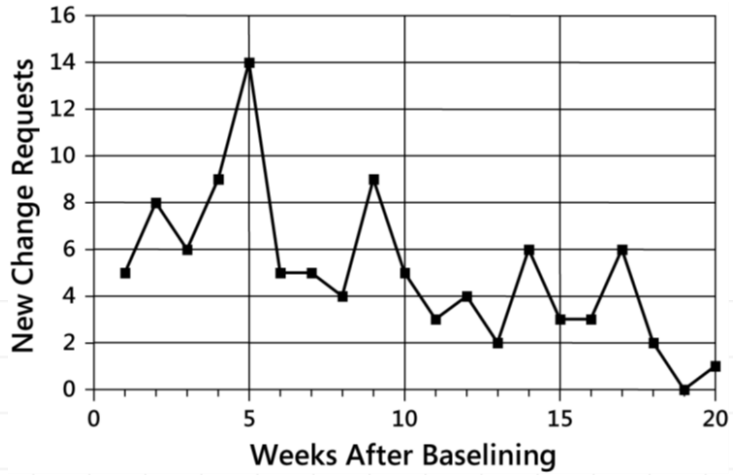


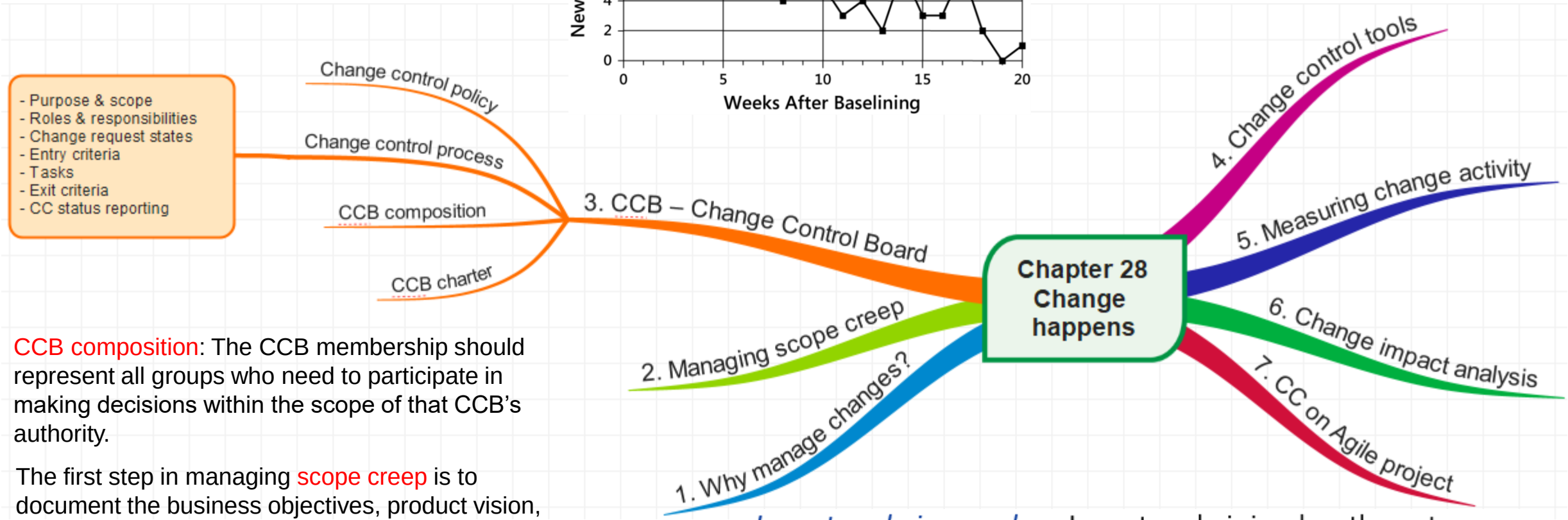
FIGURE 27-1 Major requirements management activities.

A requirements baseline is a set of requirements that stakeholders have agreed to, often defining the contents of a specific planned release or development iteration.

Agile projects accommodate (điều tiết) change by building the product through a series of development iterations and managing a dynamic product backlog of work remaining to be done.



Measuring change activity is a way to assess the stability of the requirements. It also reveals opportunities for process improvements that might lead to fewer changes in the future.



CCB composition: The CCB membership should represent all groups who need to participate in making decisions within the scope of that CCB's authority.

The first step in managing scope creep is to document the business objectives, product vision, project scope, and limitations of the new system.

Why we manage changes? Because the uncontrolled change is a common source of project chaos, schedule slips, quality problems, and hard feelings.

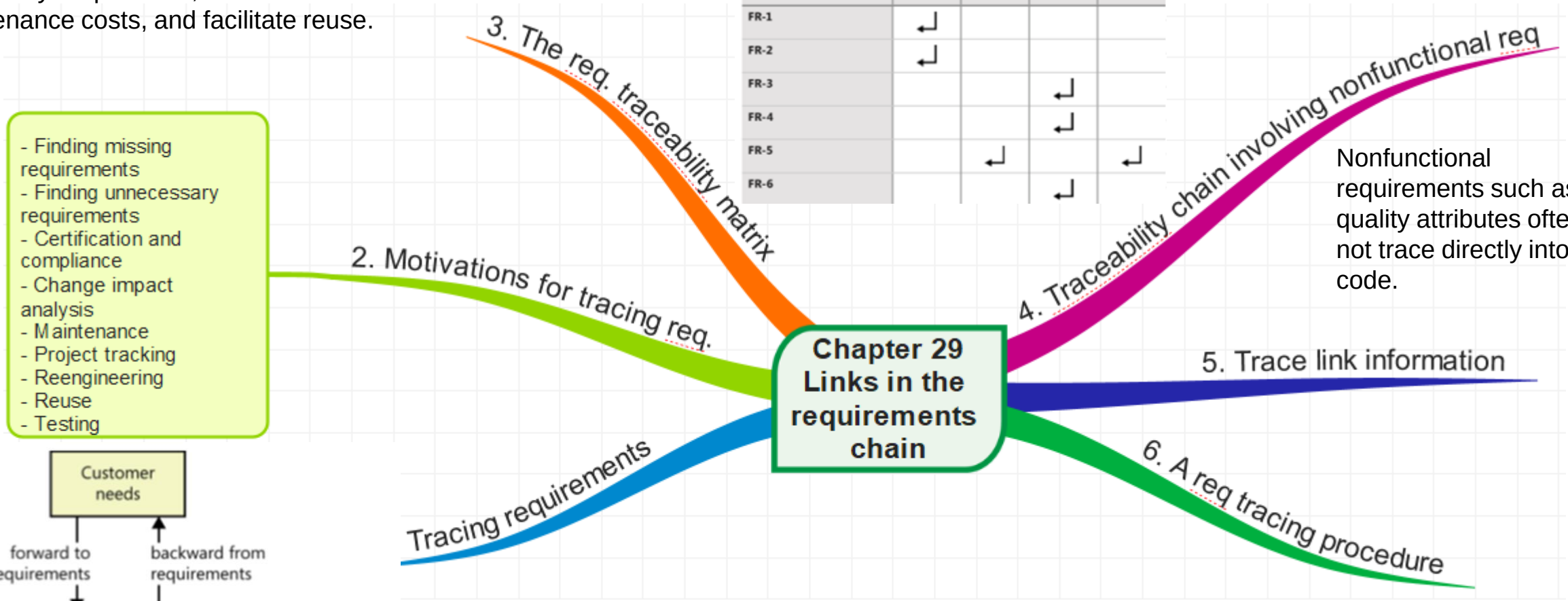
Impact analysis procedure: Impact analysis involves three steps:

- Understand the possible implications of making the change. Changes can lead to conflicts with other requirements or can compromise (làm tổn thương) quality attributes, such as performance or security.
- Identify all the requirements, files, models, and documents that might have to be modified if the team incorporates the requested change.
- Identify the tasks required to implement the change, and estimate the effort needed to complete those tasks.

At an organization level, implementing requirements tracing can improve the quality of your products, reduce maintenance costs, and facilitate reuse.

TABLE 29-2 Requirements traceability matrix showing links between use cases and functional requirements

	Use case			
Functional requirement	**UC-1**	**UC-2**	**UC-3**	**UC-4**
FR-1	↵			
FR-2	↵			
FR-3			↵	
FR-4			↵	
FR-5		↵		↵
FR-6			↵	



Nonfunctional requirements such as quality attributes often do not trace directly into code.

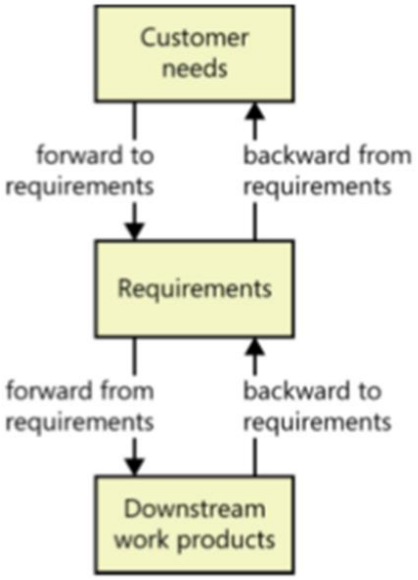


FIGURE 29-1 Four types of requirements tracing.

Tracing procedure: 9 steps. Read more details on pages from 499 to 500.

Link information: Determine the roles and individuals who should supply each type of trace information for your project.

- Manage versions and changes
- Store requirements attributes
- Facilitate impact analysis
- Identify missing and extraneous reqs.
- Track reqs status
- Control access
- Communicate with stakeholders
- Reuse reqs
- Track issue status
- Generate tailored subsets

Benefits of using an RM tool

RM tool capabilities

2. Req. management tools

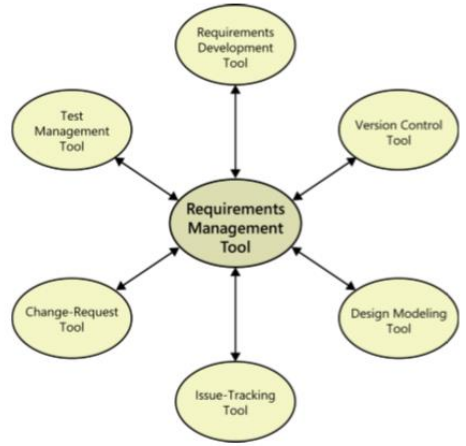


FIGURE 30-2 RM tools integrate with other kinds of software tools.

Selecting a tool

Setting up the tool and processes

Facilitating user adoption

3. Selecting a implementing a reqs tool

Chapter 30 Tools for requirements engineering

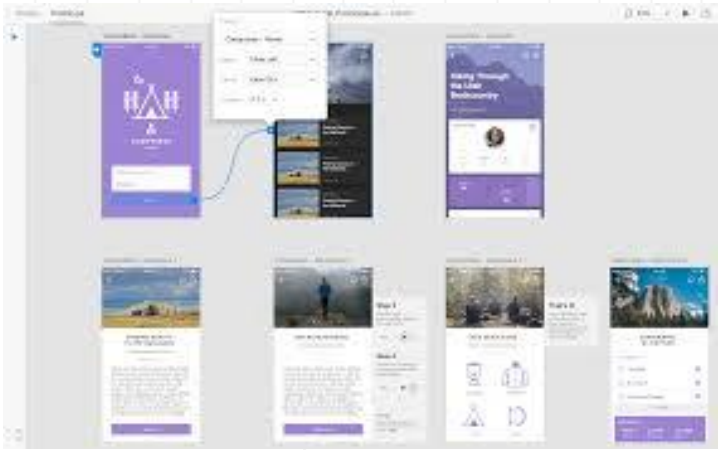
Elicitation tools

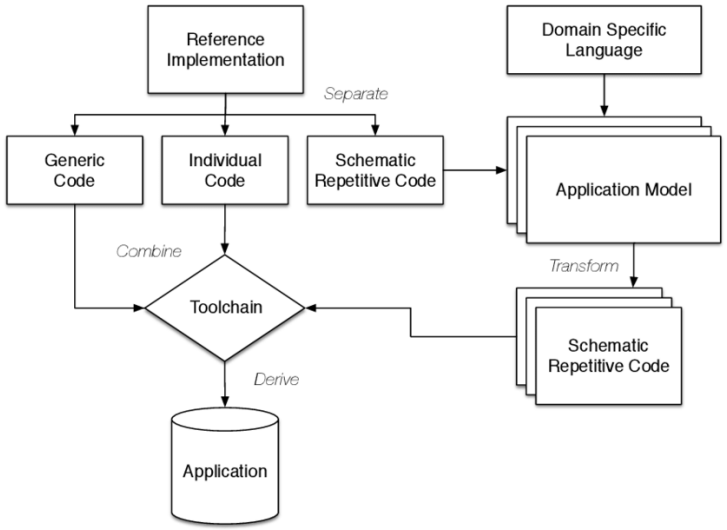
Prototyping tools

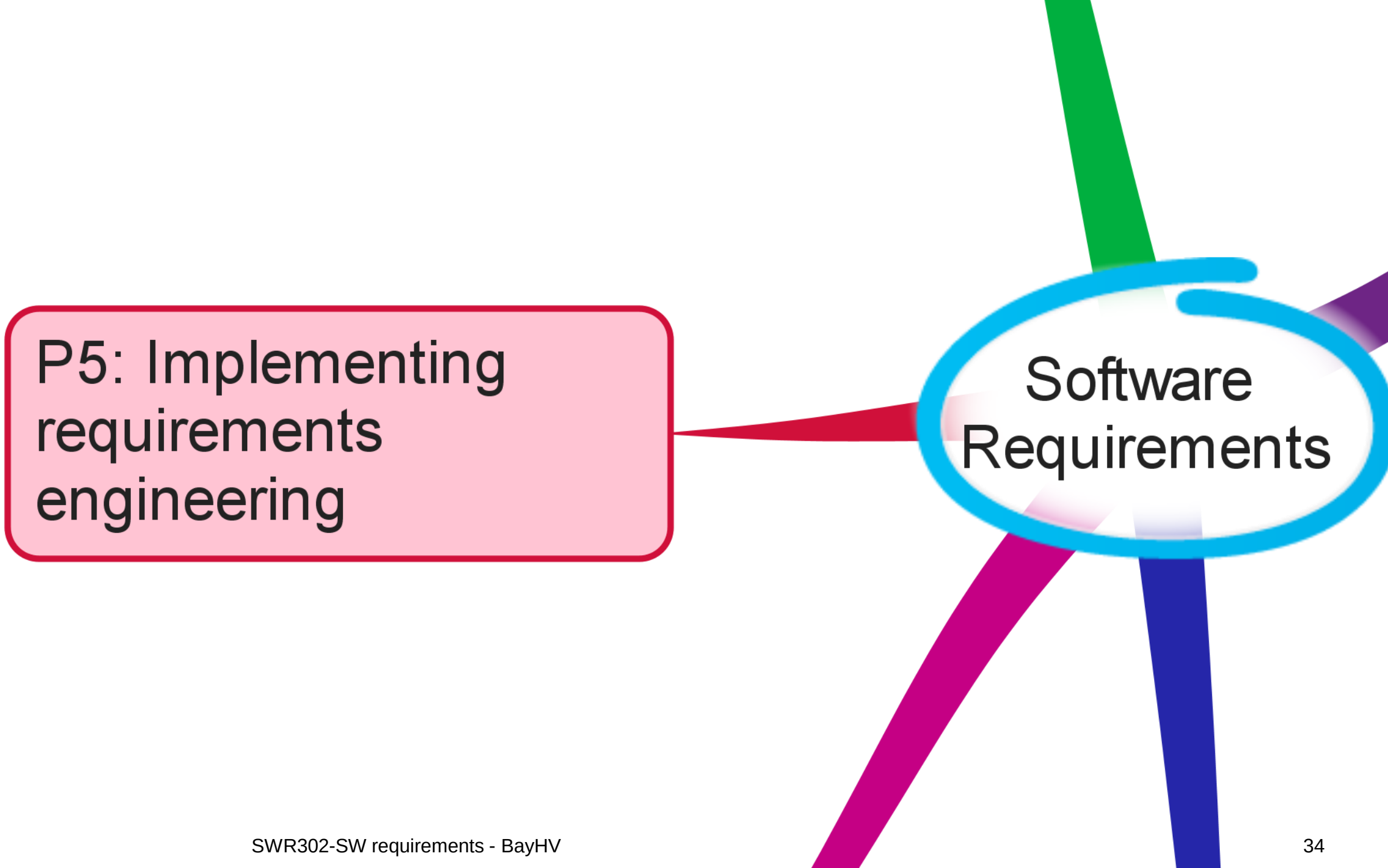
Modeling tools

1. Req. development tools

Audio & Video pens






P5: Implementing requirements engineering
Software Requirements

Gaining commitment to change: People don't like to be forced out of their comfort zone, so there are some resistance from them to the process changes you propose. Understand the origins of the resistance so you can both respect it and defuse it (xoa dịu nó).

Gaining commitment to change:

- Process improvement should be evolutionary and continuous
- People and organizations change only when they have an incentive (khích lệ) to do so.
- Process changes should be goal-oriented
- Treat your improvement activities as mini-projects

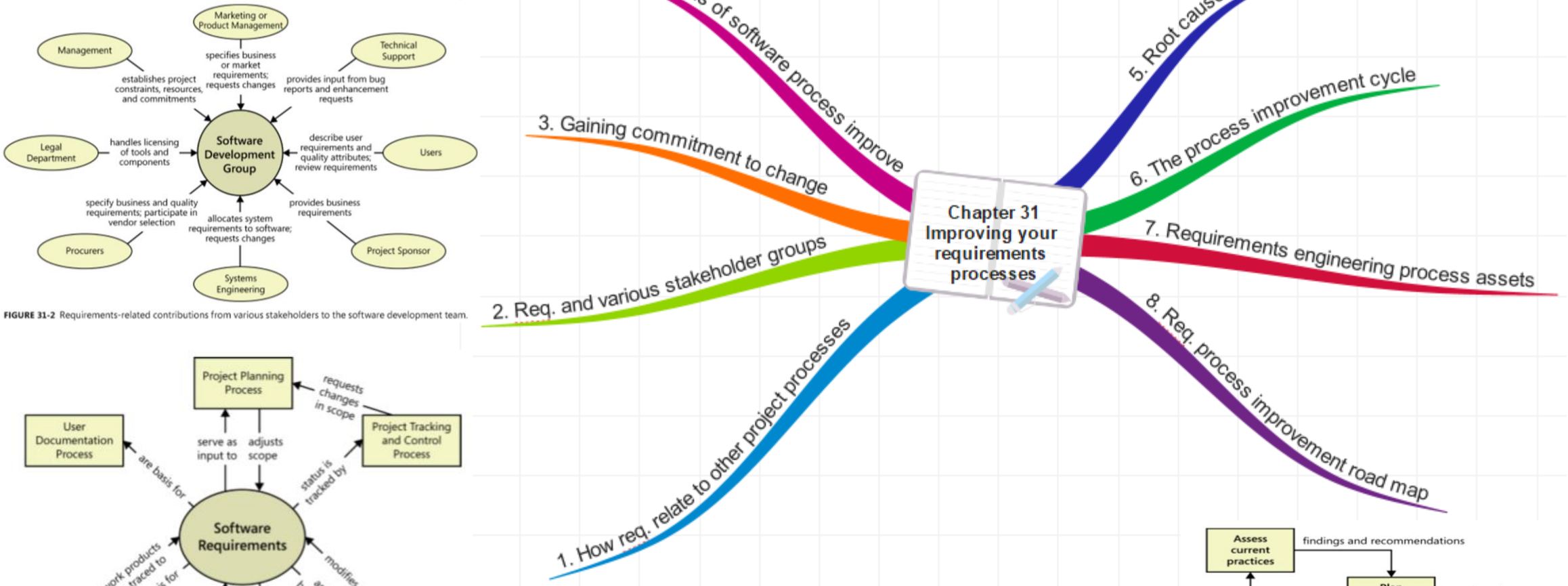


FIGURE 31-2 Requirements-related contributions from various stakeholders to the software development team.

FIGURE 31-1 Relationship of requirements to other project processes.

The ultimate objective of process improvement is to *reduce the cost of creating* and *maintaining software*, thereby *increasing the value delivered by projects*.

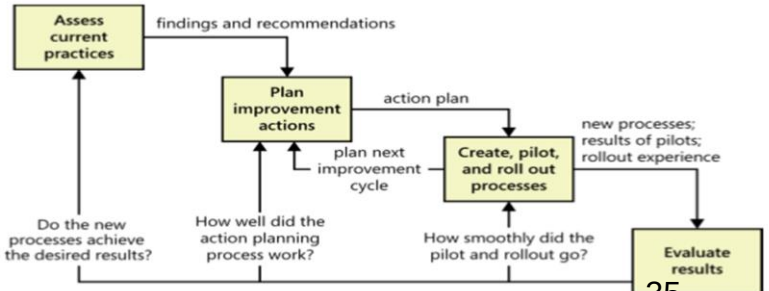
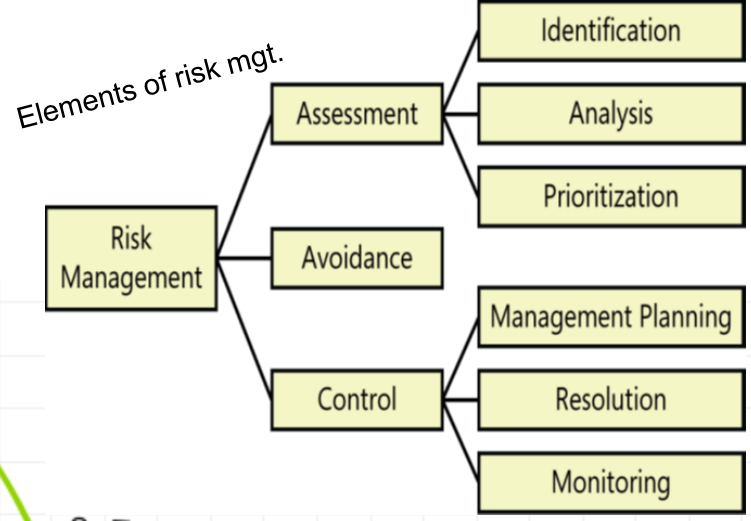


FIGURE 31-5 The software process improvement cycle.



ID: 1 **Submitter:** Yuhong Li

Date Opened: 8/22/13 **Date Closed:** (open)

Risk Statement:
If we have insufficient user involvement in requirements elicitation, then we might need to perform extensive user interface rework after beta testing.

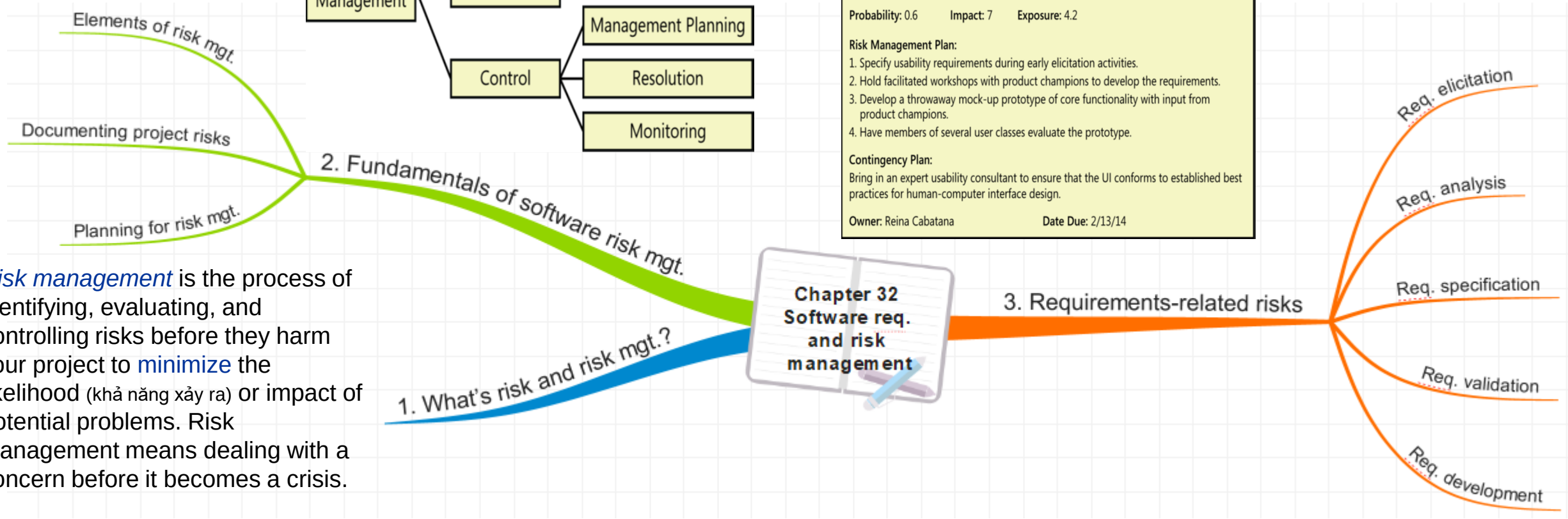
Scope of Impact:
Could affect entire system, including any customizations done on integrated COTS components.

Probability: 0.6 **Impact:** 7 **Exposure:** 4.2

Risk Management Plan:
1. Specify usability requirements during early elicitation activities.
2. Hold facilitated workshops with product champions to develop the requirements.
3. Develop a throwaway mock-up prototype of core functionality with input from product champions.
4. Have members of several user classes evaluate the prototype.

Contingency Plan:
Bring in an expert usability consultant to ensure that the UI conforms to established best practices for human-computer interface design.

Owner: Reina Cabatana **Date Due:** 2/13/14



Risk management is the process of identifying, evaluating, and controlling risks before they harm your project to minimize the likelihood (khả năng xảy ra) or impact of potential problems. Risk management means dealing with a concern before it becomes a crisis.

A *risk* is a condition that could cause some loss or otherwise threaten the success of a project.
This condition causes actually problems. These potential problems might have an adverse (bất lợi) impact on the project's cost, schedule, or technical success; the product's quality; or the team's effectiveness.

In every phase of requirements development and management, there are many risks you need to identify early and manage them to minimize the impact on your project.

THE END
THANK YOU!