REQUIREMENTS MANAGEMENT

PART ONE: REQUIREMENT MANAGEMENT PRACTICES (CH27)

TRUTH BEHIND STORY

Shari: “I finally finished implementing the multivendor catalogue query feature. That was a lot of work!”

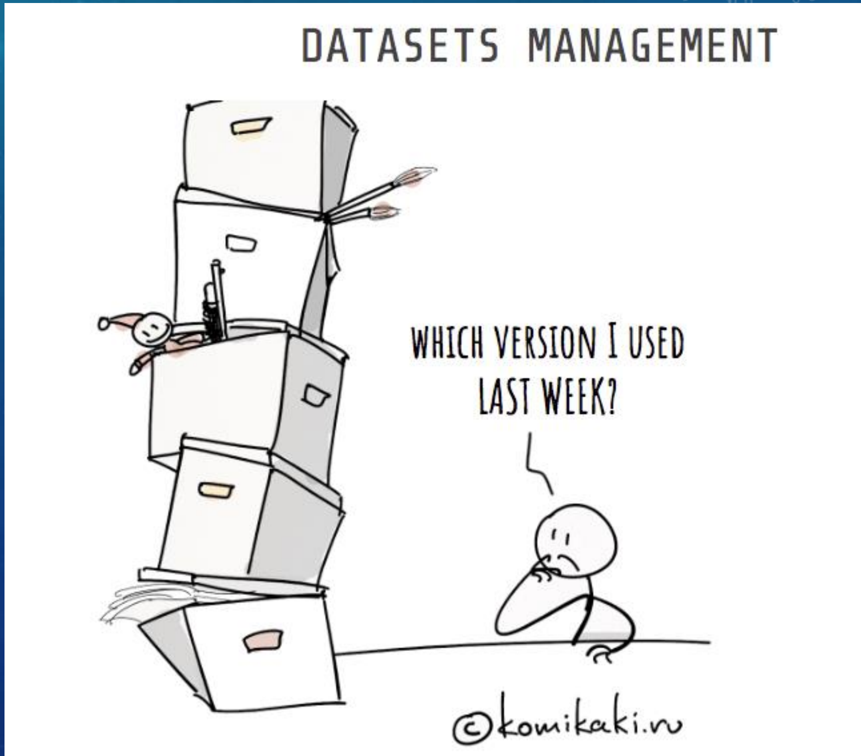
Dave: “Oh, the customers cancelled that feature two weeks ago.”

S: “Didn’t you get the revised SRS? What do you mean, it was cancelled? Those requirements are at the top of page 6 of my latest SRS.”

D: “Hmmm, they’re not in my copy. I’ve got version 1.5 of the SRS. What version are you looking at?”

S: “Mine says version 1.5 also. These documents should be identical, but obviously they’re not. So, is this feature still needed, or did I just waste 30 hours of my life?”

This story tells the importance of requirement management and effective communication.
Having the well-presented meaningful requirements is only the halfway of SE
Version control is one of the important aspects of requirement management which is an element of requirement engineering.
Change control is significantly serious in requirement engineering.

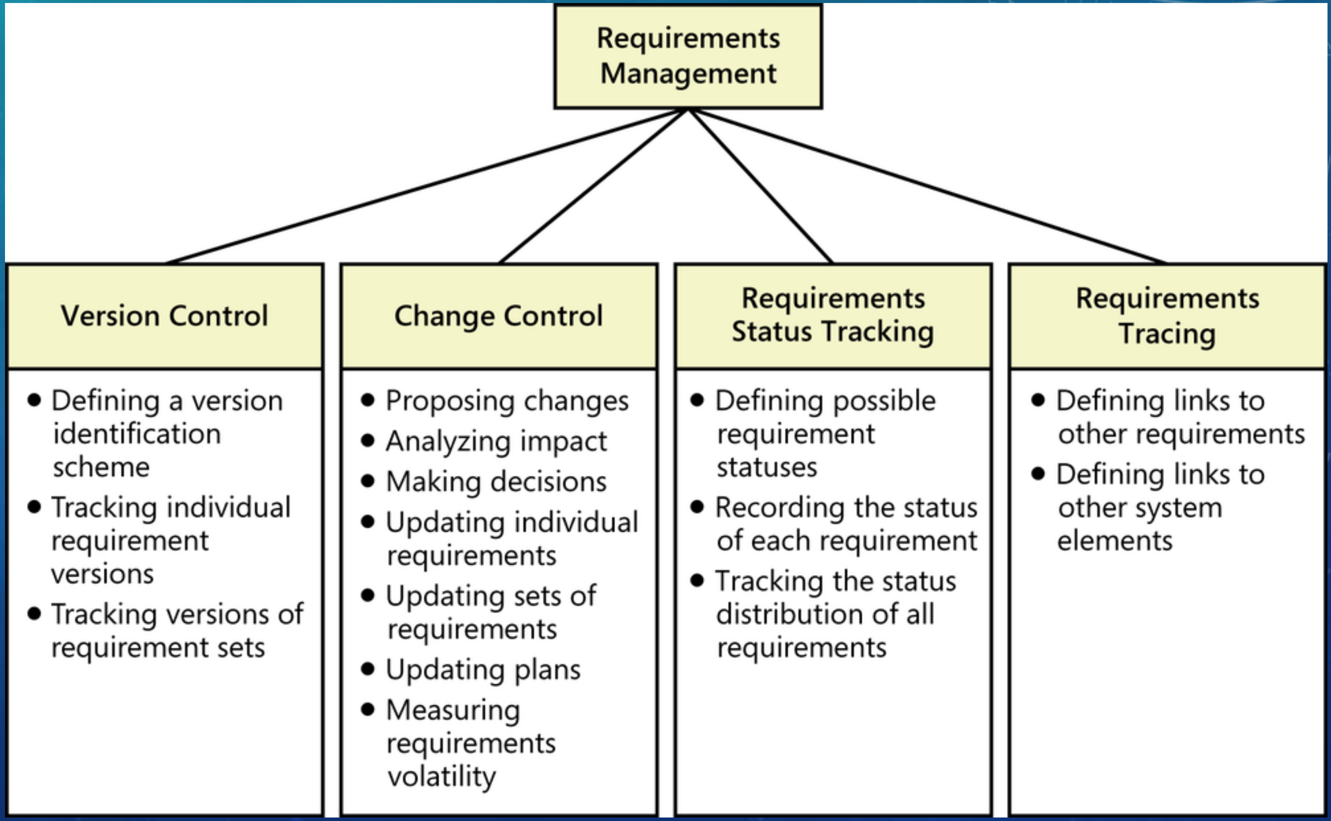


REQUIREMENT MANAGEMENT PROCESS

Aiming to maintain the integrity, accuracy, and currency of requirements agreements throughout the project

Four aspects:

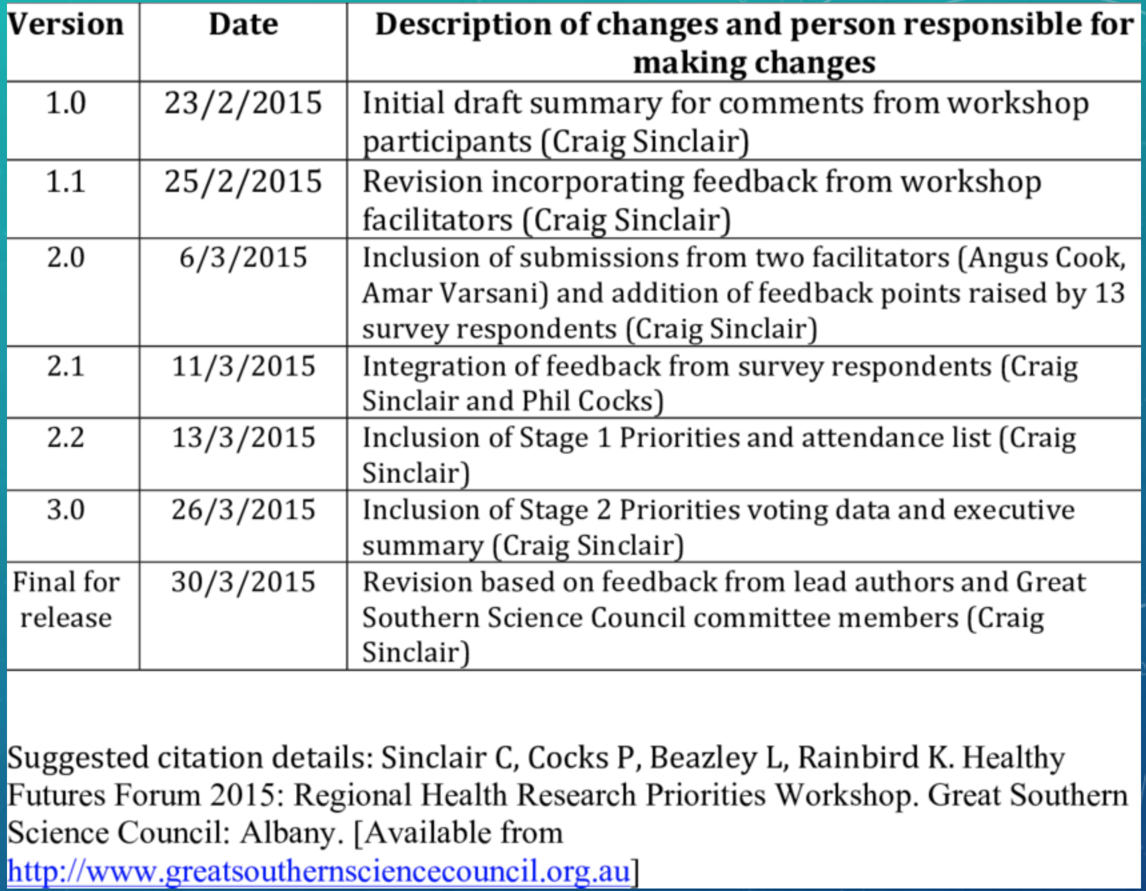
- Version control: to avoid different team members to use different requirements
- Change control: to handle changes on the requirements
- Requirements status tracking: to be aware of the current status of requirements (plus responsibilities to inform whomever to take actions)
- Requirements tracing: to monitor the impacts of a change to other elements within the same system or to other external systems.


Requirements Management
Version Control
• Defining a version identification scheme
• Tracking individual requirement versions
• Tracking versions of requirement sets
Change Control
• Proposing changes
• Analyzing impact
• Making decisions
• Updating individual requirements
• Updating sets of requirements
• Updating plans
• Measuring requirements volatility
Requirements Status Tracking
• Defining possible requirement statuses
• Recording the status of each requirement
• Tracking the status distribution of all requirements
Requirements Tracing
• Defining links to other requirements
• Defining links to other system elements

REQUIREMENT MANAGEMENT PROCESS

SRS Version control

- Version ID – e.g. v1.2, but my habit is Vddmmyy
- Accessibility – all team members use the same version
- Permission to updating – only designated persons can modify
- Version history – keep all previous versions
- Other attributes:
  - Date the requirement was created
  - Author who wrote the requirement
  - Priority/Status/Origin or source of the requirement (refer to previous lectures about priority and origin – from which stakeholder)
  - Rationale behind the requirement (what to achieve with the completement of the requirement)
  - Release number or iteration to which the requirement is allocated (mid-term report, deliverable number. etc.)
  - Stakeholders to contact with questions or to make decisions about changes (refer to source)
  - Validation method to be used or acceptance criteria (formal or informal peer review?)



Version	Date	Description of changes and person responsible for making changes
1.0	23/2/2015	Initial draft summary for comments from workshop participants (Craig Sinclair)
1.1	25/2/2015	Revision incorporating feedback from workshop facilitators (Craig Sinclair)
2.0	6/3/2015	Inclusion of submissions from two facilitators (Angus Cook, Amar Varsani) and addition of feedback points raised by 13 survey respondents (Craig Sinclair)
2.1	11/3/2015	Integration of feedback from survey respondents (Craig Sinclair and Phil Cocks)
2.2	13/3/2015	Inclusion of Stage 1 Priorities and attendance list (Craig Sinclair)
3.0	26/3/2015	Inclusion of Stage 2 Priorities voting data and executive summary (Craig Sinclair)
Final for release	30/3/2015	Revision based on feedback from lead authors and Great Southern Science Council committee members (Craig Sinclair)

Suggested citation details: Sinclair C, Cocks P, Beazley L, Rainbird K. Healthy Futures Forum 2015: Regional Health Research Priorities Workshop. Great Southern Science Council: Albany. [Available from http://www.greatsouthernsciencecouncil.org.au]

REQUIREMENT MANAGEMENT PROCESS

Change control

- Changes to be made only through the project's defined change control procedure (more in Part 2)
- New or changed requirements can be accommodate in various ways:
  - By deferring lower-priority requirements to later iterations or cutting them completely
  - By obtaining additional staff or outsourcing some of the work
  - By extending the delivery schedule or adding iterations to an agile project
  - By sacrificing quality to ship by the original date

Change control in Agile

- One of the manifestos of Agile software project management is to value Responding to **changes** over following a plan
- Together with other three
  - Value Individuals and interactions over processes and tools
  - Value Working software over comprehensive documentation
  - Value Customer collaboration over contract negotiation

REQUIREMENT MANAGEMENT PROCESS

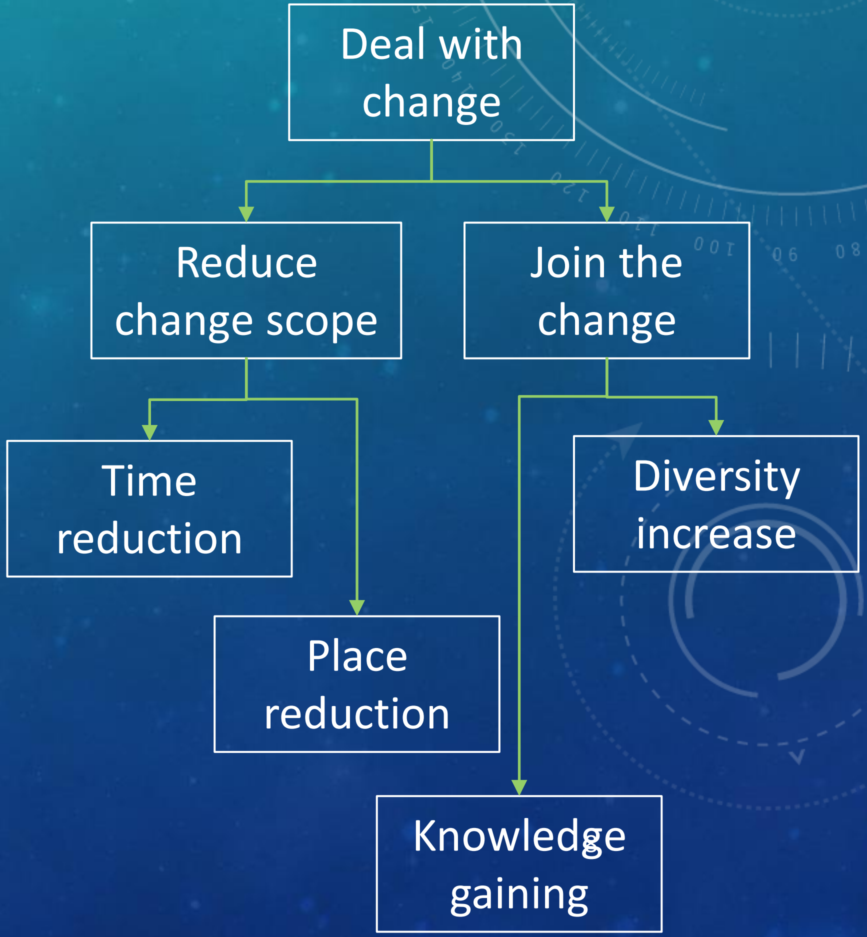
Agile explains the reasons for changes as the followings:

- It is hard to have a long-term plan (actually schedule) for all tasks with the starting and the ending dates (in the form, e.g. Gantt chart) because
  - Business process in customers' organisations can change.
  - Customers need time to clarify/confirm their requirements
  - Developers' understandings on tasks (e.g. whether they are still needed or some new tasks may appear, how complex they are, how long time the developers would need to complete them, etc.) change along the time.
- The solution is to set up short-term plans, such as those that are set up for 2 weeks, can be more realistic.
- Of course, teams will have to follow these short-term plans . By executing these short plans, developers will gain better understandings on the followed-up tasks and, hence, will be in better positions to develop the next bunch of short-term plans for the further development of the project they participate in.

REQUIREMENT MANAGEMENT PROCESS

Portkin's conceptual framework for change reduction

- Reduce change scope
    - Time reduction: reducing the time from concepts to reproductivity, i.e. dividing project into modules as small as possible so the completion of one module will require less time and the change in one module will require less time to reproduce the module.
    - Place reduction: dividing into isolated modules so that the module will be unlikely affected by changes introduced into others
- Join the change
    - Diversity increase: producing large number of offspring to increase diversity so the chance that one individual would be able to face the change increases
    - Knowledge gaining: intelligent mechanism being able to track the changes in the world and adapt the software itself.



REQUIREMENT MANAGEMENT PROCESS

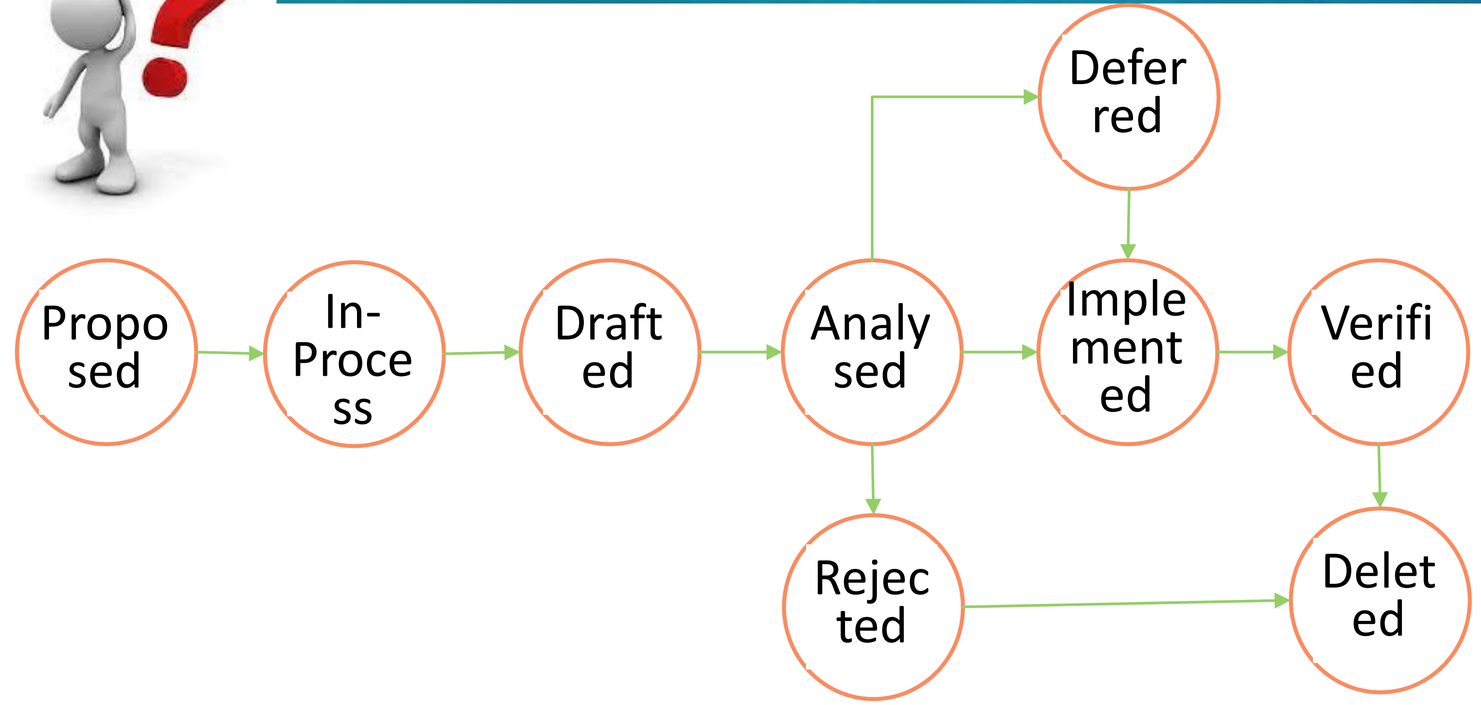
Status tracking

- A requirement can have 9 status
- Update a requirement's status only when specified transition conditions are satisfied

Status	Definition
Proposed	The requirement has been requested by an authorised source.
In Progress	A BA is actively working on handling the requirement. Drafted The initial version of the requirement has been written.
Drafted	The requirement has been analysed and specified.
Approved	The requirement has been validated and it has been allocated to a specific release. The stakeholders have agreed to incorporate the requirement, and a development team has committed to implement it.
Implemented	The code that implements the requirement has been designed, written, and unit tested. The requirement has been traced to the relevant design and code elements. The software that implements the requirement is now ready for testing, review.
Verified	The requirement has satisfied its acceptance criteria, meaning that the correct functioning of the implemented requirement has been confirmed. The requirement has been traced to pertinent tests. It is now considered complete.
Deferred	An approved requirement is now planned for implementation in a later release.
Deleted	An implemented and approved requirement has been removed from the baseline.
Rejected	The requirement was proposed but was never approved and is not planned for implementation in any upcoming release. Include an explanation of why and by whom the decision was made to reject it.

REQUIREMENT MANAGEMENT PROCESS

- Requirements have different issues when they are in different status
- Resolving the issues will change the status.
- Issues (see the table)



Issue type	Description
Requirement question	Something isn't understood or decided about a requirement.
Missing requirement	Developers uncovered a missed requirement during design or implementation.
Incorrect requirement	A requirement was wrong. It should be corrected or removed.
Implementati on Question	As developers implement requirements, they have questions about how something should work or about design alternatives.
Duplicate Requirement	Two or more equivalent requirements are discovered. Delete all but one of them.
Unneeded Requirement	A requirement simply isn't needed anymore.

REQUIREMENT MANAGEMENT PROCESS

Requirement tracing

(see part 3)

REQUIREMENT MANAGEMENT PROCESS

Monitoring requirements effort

- Monitoring requirement efforts to adjust project plans
- Efforts of BAs are in twofold:

Record the number of hours spent on requirements development activities such as the following:

- Planning requirements-related activities for the project
- Holding workshops and interviews, analysing documents, and performing other elicitation activities
- Writing requirements specifications, creating analysis models, and prioritizing requirements
- Creating and evaluating prototypes intended to assist with requirements development
- Reviewing requirements and performing other validation activities

Count the effort devoted to the following activities as requirements management effort:

- Configuring a requirements management tool for a project
- Submitting requirements changes and proposing new requirements
- Evaluating proposed changes, including performing impact analysis and making decisions
- Updating the requirements repository
- Communicating requirements changes to affected stakeholders
- Tracking and reporting requirements status
- Creating requirements trace information

PART TWO: MANAGING CHANGES (CH28)

CHANGES IN REQUIREMENT

Changes are inevitable

- The requirements for software systems typically grow between 1 percent and 3 percent per calendar month
- Changes need to be carefully managed because:
  - Cost -- developers sometimes do not, or cannot, produce realistic estimates of the cost (resources and time needed) of a proposed software change.
  - Back door issue -- developer may agree changes a user request to the users without having the changes officially approved by the right stakeholders, leading to the situation where only the developer and the user know the change but all others do not.
  - Scope creep -- the project becomes bigger and bigger because it continuously incorporates more functionalities without adjusting resources, schedules, or quality goals.
  - Impacts to others components -- the late changes can have a big impact on work already performed.

PRINCIPLES AND POLICIES

Principles

- Proposed requirements changes are thoughtfully evaluated before being committed to (cost and scope).
- Appropriate individuals make informed business decisions about requested changes (back door).
- Change activity is made visible to affected stakeholders (impact).
- Approved changes are communicated to all affected participants (back door and impact).
- The project incorporates requirements changes in a consistent and effective fashion (scope, back door).

Control policies

- All changes must follow the process.
- No design or implementation other than feasibility exploration will be performed on unapproved changes.
- The project's change control board (CCB) will decide which changes to implement.
- The contents of the change database must be visible to all project stakeholders (or it can be a spreadsheet).
- Impact analysis must be performed for every change.
- Every change must be traceable to an approved change request.
- The rationale behind every approval or rejection of a change request must be recorded.

PROCESS AND CHANGE CONTROL BOARD (CCB)

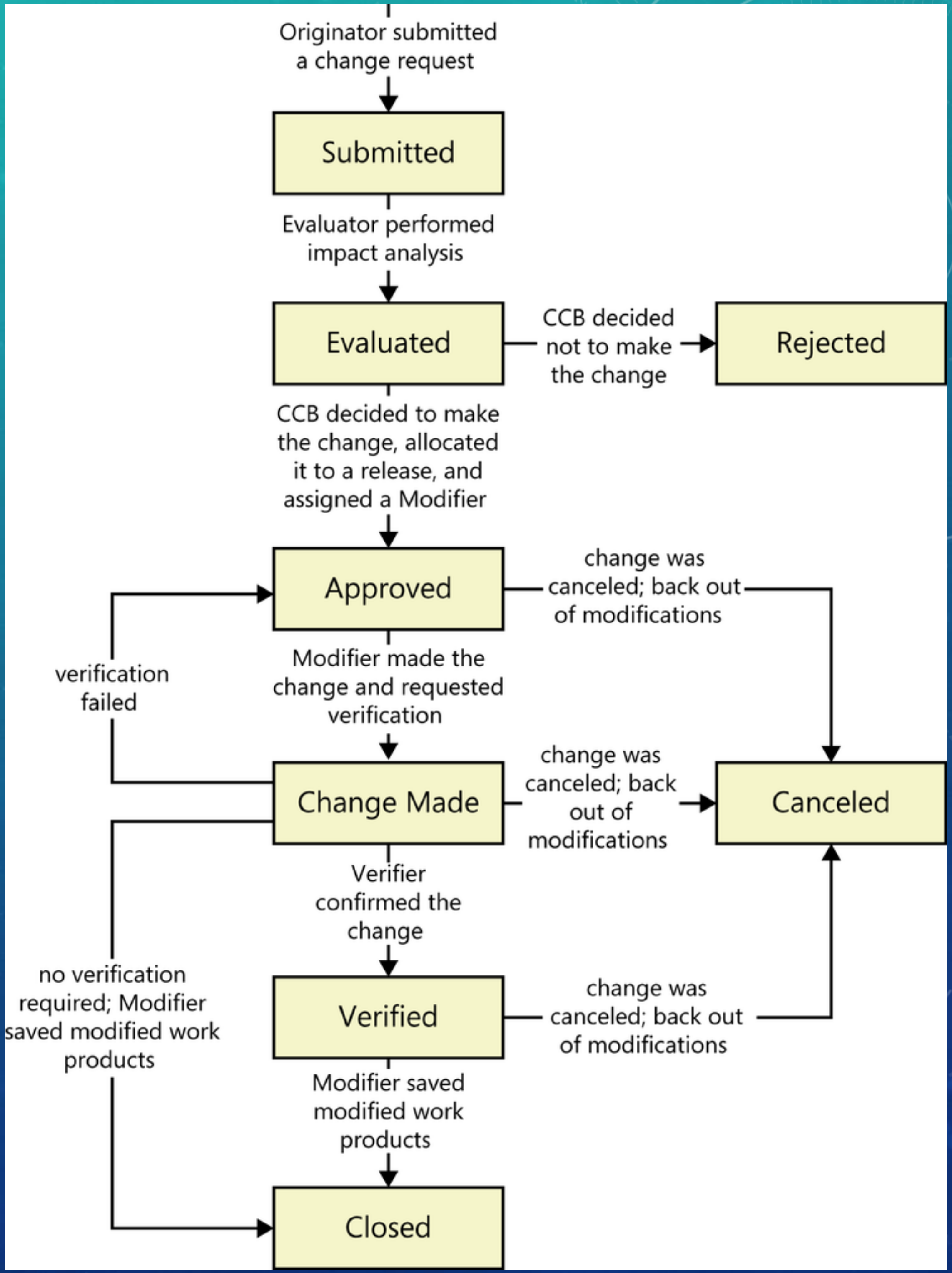
CCB

- Responsibility – decide changes based on the proposed change or a set of changes, and the rational
- Roles – who will take part and what are the responsibilities of the participants

Role	Description and responsibilities
Chair	Chairperson of the change control board; generally has final decision-making authority if the CCB does not reach agreement; identifies the Evaluator and the Modifier for each change request
Evaluator	Person whom the CCB Chair asks to analyse the impact of a proposed change (to cost, scope, impacts, resources)
Modifier	Person who is responsible for making changes in a work product in response to an approved change request
Originator	Person who submits a new change request (BA)
Request Receiver	Person who initially receives newly submitted change requests (CCB secretory)
Verifier	Person who determines whether the change was made correctly
CCB	The group that decides to approve or reject proposed changes fora specific project

PROCESS AND CHANGE CONTROL BOARD (CCB)

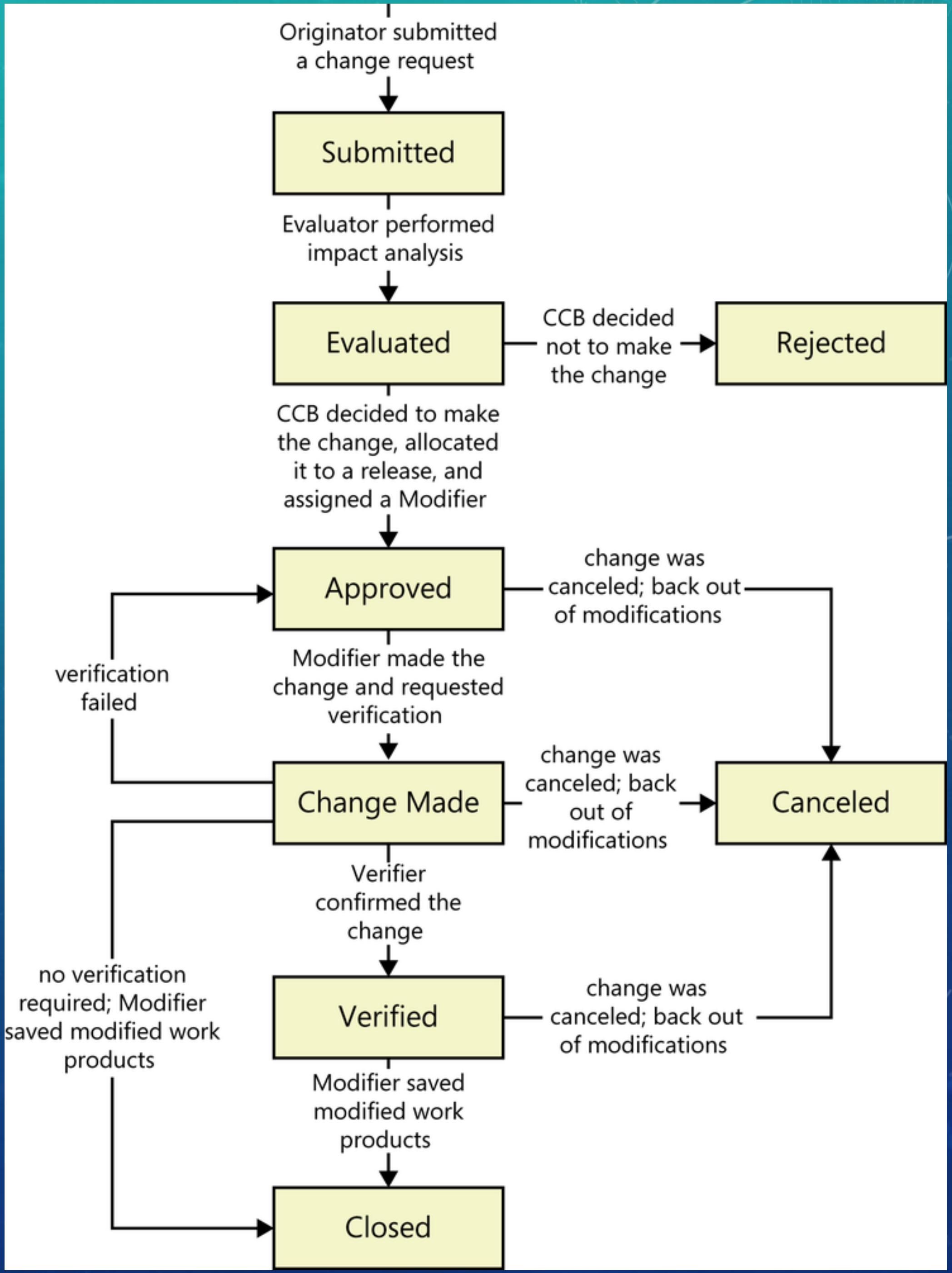
- CCB process (rectangles represent the status of changes)
  - Entry criteria -- the conditions that must be satisfied before the process execution can begin
  - The various tasks of CCB:
    - Evaluate change request: technical feasibility, cost, alignment with the project's business requirements and resource constraints, impact analysis, risk and hazard analysis
    - Make a decision on whether accept or reject, and if accept, priority, target implementation date, allocating the change to a specific iteration or release
    - Implement change
    - Verification – against a set of criteria



Originator submitted a change request
Submitted
Evaluator performed impact analysis
Evaluated
CCB decided not to make the change
Rejected
CCB decided to make the change, allocated it to a release, and assigned a Modifier
Approved
change was canceled; back out of modifications
verification failed
Modifier made the change and requested verification
Change Made
change was canceled; back out of modifications
Canceled
Verifier confirmed the change
no verification required; Modifier saved modified work products
Verified
change was canceled; back out of modifications
Modifier saved modified work products
Closed

PROCESS AND CHANGE CONTROL BOARD (CCB)

- CCB process (continue)
  - Exit criteria, the conditions that indicate when the process is successfully completed, such as:
    - The status of the request is Rejected, Closed, or Cancelled.
    - All modified work products are updated and stored in the correct locations.
    - The relevant stakeholders have been notified of the change details and the status of the change request.
  - Reporting --


Originator submitted a change request
Submitted
Evaluator performed impact analysis
Evaluated
CCB decided not to make the change
Rejected
CCB decided to make the change, allocated it to a release, and assigned a Modifier
Approved
change was canceled; back out of modifications
verification failed
Modifier made the change and requested verification
Change Made
change was canceled; back out of modifications
Canceled
Verifier confirmed the change
no verification required; Modifier saved modified work products
Verified
change was canceled; back out of modifications
Modifier saved modified work products
Closed

PROCESS AND CHANGE CONTROL BOARD (CCB)

Members from

- Project or program management (scope, resources)
- Business analysis or product management (cost)
- Development (feasibility)
- Testing or quality assurance (traceability)
- Marketing, the business for which the application is being built, or customer representatives (impact)
- Technical support or help desk (impact)

Decision making

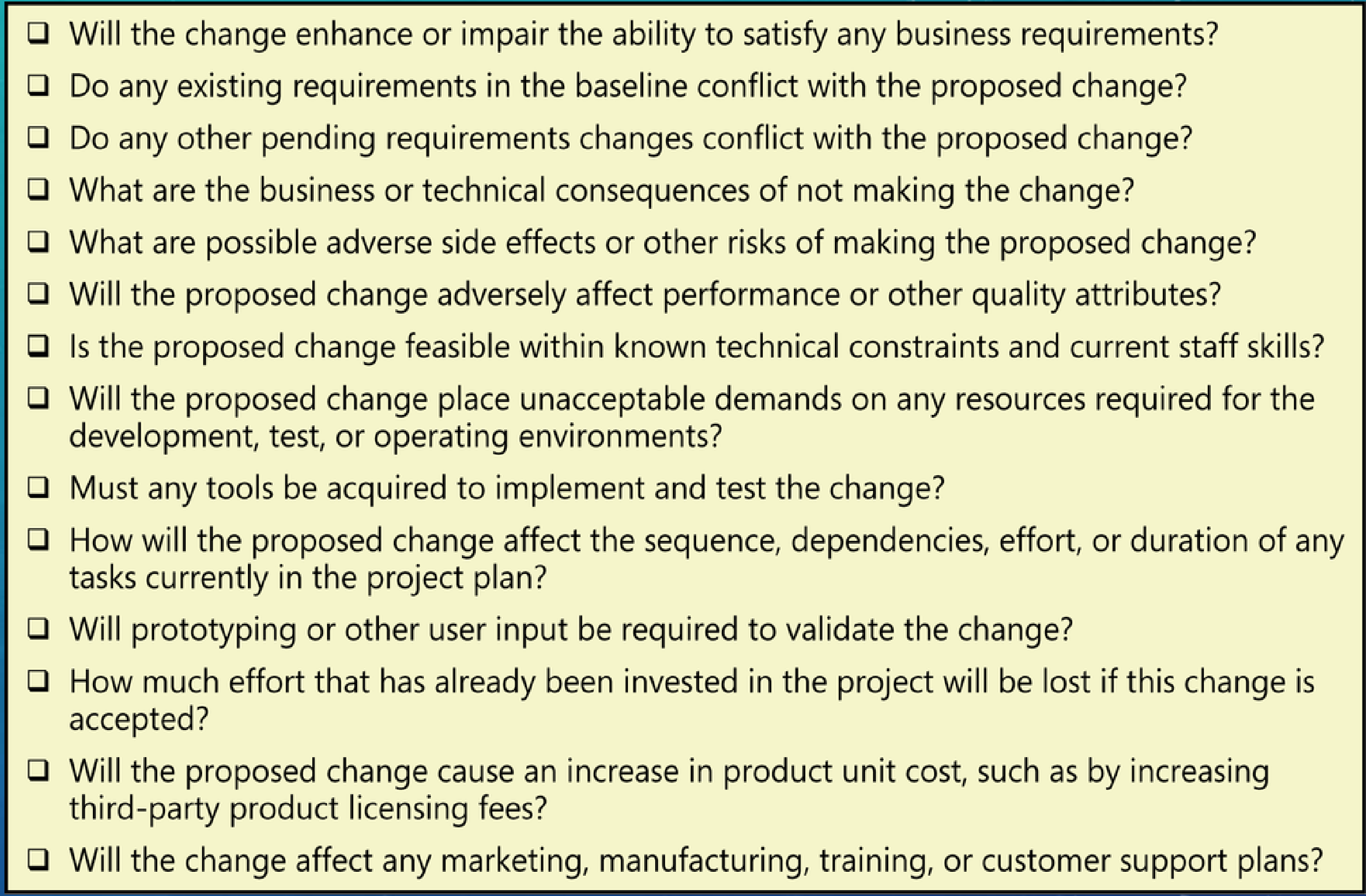
- The CCB membership
- The decision rules to be used
- Whether the CCB Chair can overrule the CCB's collective decision
- Whether CCB Chair or management must ratify the group's decision

CHANGE IMPACT ANALYSIS

Impact analysis involves three steps:

- Step 1: Understand the possible implications of making the change. A requirement change often produces a large side-effect, leading to modifications in other requirements, architectures, designs, code, and tests. Changes can lead to conflicts with other requirements or can compromise quality attributes, such as performance or security.

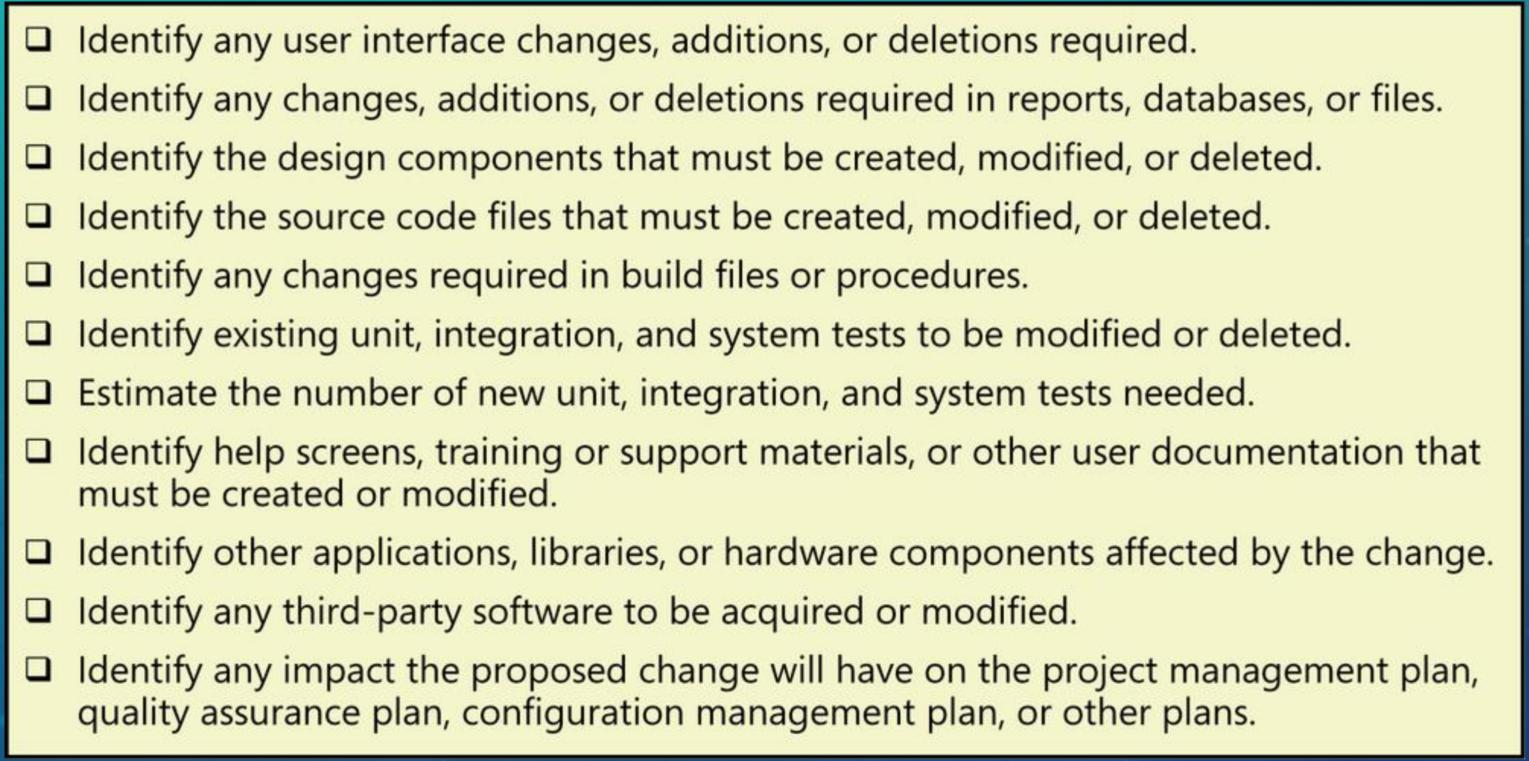
(checklist of questions to help the evaluator understand the implications of accepting a proposed change)



- ❑ Will the change enhance or impair the ability to satisfy any business requirements?
- ❑ Do any existing requirements in the baseline conflict with the proposed change?
- ❑ Do any other pending requirements changes conflict with the proposed change?
- ❑ What are the business or technical consequences of not making the change?
- ❑ What are possible adverse side effects or other risks of making the proposed change?
- ❑ Will the proposed change adversely affect performance or other quality attributes?
- ❑ Is the proposed change feasible within known technical constraints and current staff skills?
- ❑ Will the proposed change place unacceptable demands on any resources required for the development, test, or operating environments?
- ❑ Must any tools be acquired to implement and test the change?
- ❑ How will the proposed change affect the sequence, dependencies, effort, or duration of any tasks currently in the project plan?
- ❑ Will prototyping or other user input be required to validate the change?
- ❑ How much effort that has already been invested in the project will be lost if this change is accepted?
- ❑ Will the proposed change cause an increase in product unit cost, such as by increasing third-party product licensing fees?
- ❑ Will the change affect any marketing, manufacturing, training, or customer support plans?

CHANGE IMPACT ANALYSIS

- Step 2: Identify all the requirements, files, models, and documents that might have to be modified if the team incorporates the requested change.

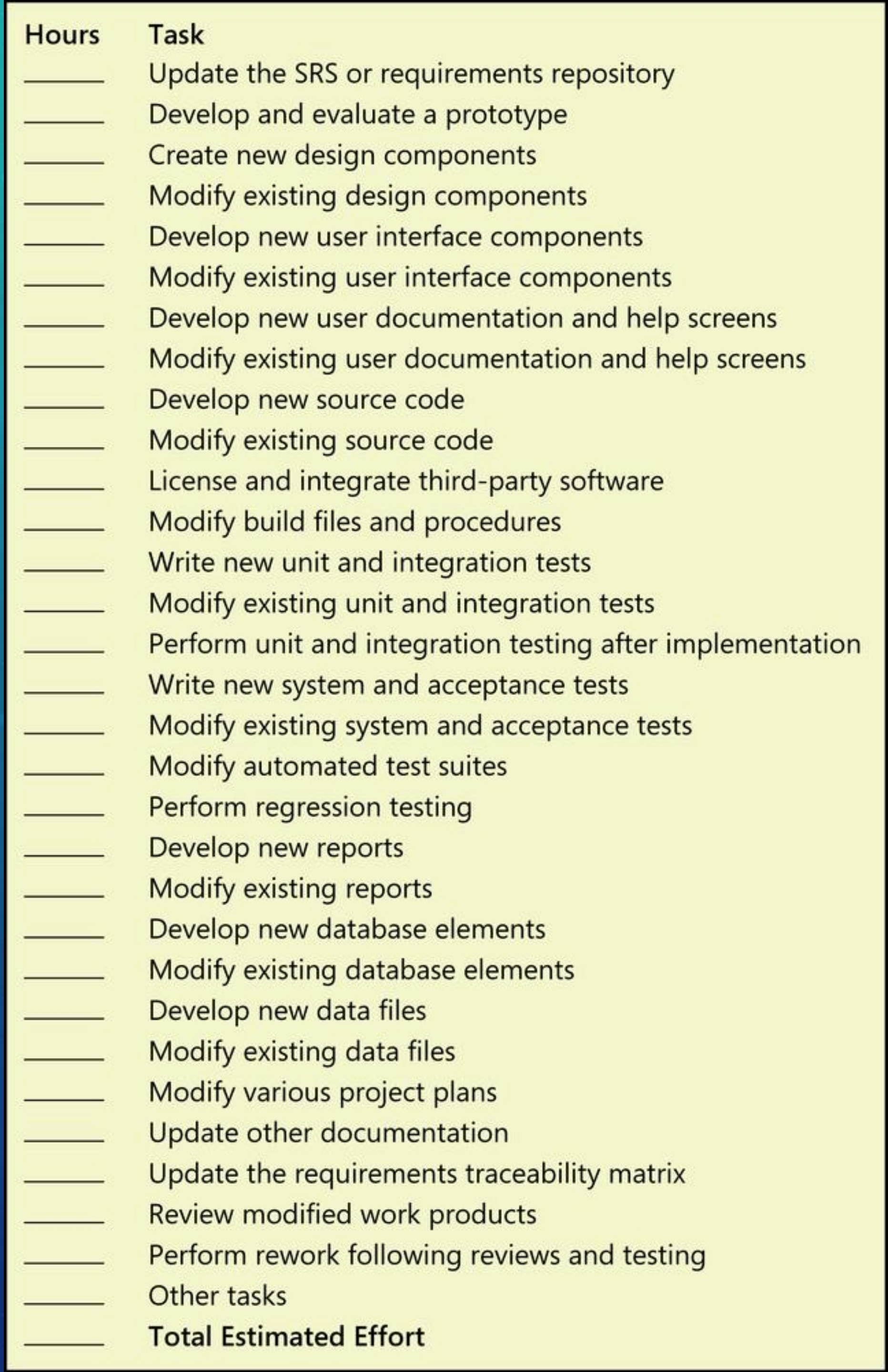


- ❑ Identify any user interface changes, additions, or deletions required.
- ❑ Identify any changes, additions, or deletions required in reports, databases, or files.
- ❑ Identify the design components that must be created, modified, or deleted.
- ❑ Identify the source code files that must be created, modified, or deleted.
- ❑ Identify any changes required in build files or procedures.
- ❑ Identify existing unit, integration, and system tests to be modified or deleted.
- ❑ Estimate the number of new unit, integration, and system tests needed.
- ❑ Identify help screens, training or support materials, or other user documentation that must be created or modified.
- ❑ Identify other applications, libraries, or hardware components affected by the change.
- ❑ Identify any third-party software to be acquired or modified.
- ❑ Identify any impact the proposed change will have on the project management plan, quality assurance plan, configuration management plan, or other plans.

(questions to help identify all software elements and other work products that the change might affect)

CHANGE IMPACT ANALYSIS

- Step 3: Identify the tasks required to implement the change, and estimate the effort needed to complete those tasks.



Hours	Task
______	Update the SRS or requirements repository
______	Develop and evaluate a prototype
______	Create new design components
______	Modify existing design components
______	Develop new user interface components
______	Modify existing user interface components
______	Develop new user documentation and help screens
______	Modify existing user documentation and help screens
______	Develop new source code
______	Modify existing source code
______	License and integrate third-party software
______	Modify build files and procedures
______	Write new unit and integration tests
______	Modify existing unit and integration tests
______	Perform unit and integration testing after implementation
______	Write new system and acceptance tests
______	Modify existing system and acceptance tests
______	Modify automated test suites
______	Perform regression testing
______	Develop new reports
______	Modify existing reports
______	Develop new database elements
______	Modify existing database elements
______	Develop new data files
______	Modify existing data files
______	Modify various project plans
______	Update other documentation
______	Update the requirements traceability matrix
______	Review modified work products
______	Perform rework following reviews and testing
______	Other tasks
______	**Total Estimated Effort**

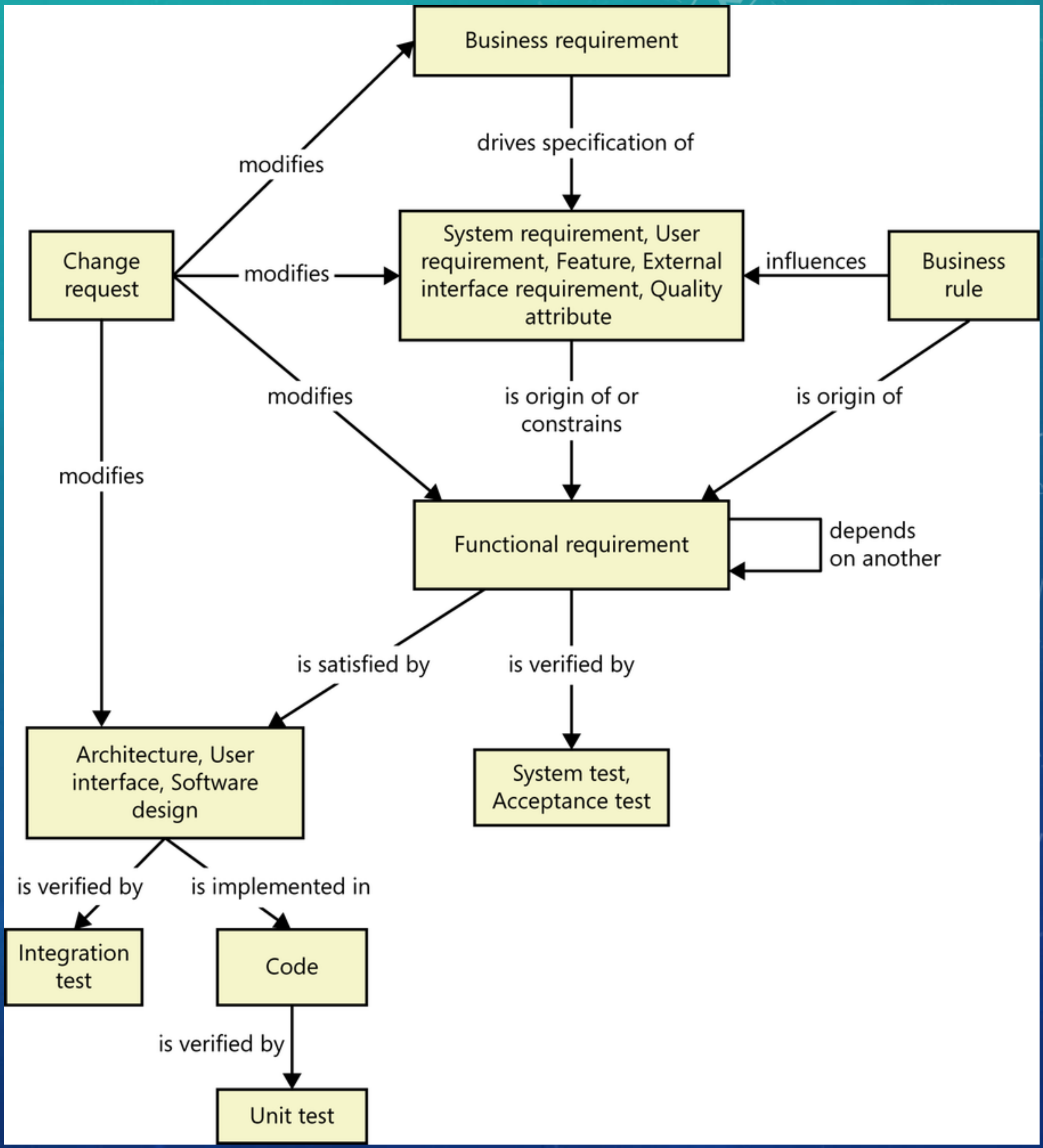
PART THREE: TRACE CHANGES (CH29)

BENEFITS

- Requirement changes seem simple but often have far-reaching impacts (e.g. scope creed can be not obvious)
- Tracing changes brings the following good things:
  - Finding missing requirements.
  - Finding unnecessary requirements via change impact analysis by comparing before and after the change is introduced
  - Certification and compliance -- trace information can demonstrate that all requirements were implemented.
  - Maintenance -- Reliable trace information tells changes are correctly completed.
  - Project tracking – trace information provides an accurate record of the implementation status of planned functionality.
  - Reengineering.
  - Reuse -- Trace information identifying packages of related requirements, designs, code, and tests
  - Testing -- links between tests, requirements, and code can tell the likely areas to examine for defects.

SHOW TRACEABILITY

- Diagram shows the traceability relationships between different types of requirements, and hence the impacts of a change in upstream to the components in downstream.
- For example, that a tester is confused about the testing results might mean some changes are introduced in functional requirement.
- Another example is that some changes may take place in functional requirement if architecture is no longer satisfying functional requirement.



SHOW TRACEABILITY

- The requirements traceability matrix shows how each functional requirement is linked backward to a specific use case and forward to one or more design, code, and test elements, for example

User requirement	Functional requirement	Design element	Code element	Test
UC-28	catalog.query.sort	Class catalog	CatalogSort()	search.7 search.8

UC-29	catalog.query.import	Class catalog	CatalogImport()
			CatalogValidate()
				search.12
				search.13
				search.14

REQUIREMENTS TRACING ACTIVITIES (GOOD PRACTICE)

- Explain the team and management about the concepts and importance of requirements tracing.
- Explain the selected link relationships from, e.g. the diagram from the previous slide.
- Explain traceability matrix, e.g. the table on the previous slide.
- Identify the parts of the software for which you want to maintain traceability information. Start with the critical core functions, the high-risk portions, or the portions that undergo the most maintenance and evolution over the product’s life.
- Identify the individuals who will supply each type of link information and the person (most likely a BA) who will coordinate the tracing activities and manage the data.
- Modify development procedures to remind developers to update the links after implementing a requirement or an approved change.
- Define the labelling conventions to give each system element a unique identifier so that they can be linked together.
- As development proceeds, let each participant provide the requested trace information as they complete small bodies of work.