Unit V

SOFTWARE RISK: RISK IDENTIFICATION, RISK ASSESSMENT, RISK PROJECTION, RISK REFINEMENT, RMMM PLAN.

SOFTWARE CONFIGURATION MANAGEMENT: SCM, SCM REPOSITORIES, SCM PROCESS, CHANGE CONTROL AND VERSION CONTROL.

6TH EDITION
Chapter 25: Risk Management
Chapter 27: Change Management

7TH EDITION
Chapter 28: Risk Management
Chapter 22: Software Configuration Management

Concepts

- Risks:
  - Are those events that *may* occur and whose occurrences if it does take place have a harmful or negative effect on the project.
- Risk analysis and management:
  - Are series of steps that help a software team to understand and manage uncertainty.
  - Everyone involved in the software process – managers, engineers and stakeholders participate in risk analysis
- Why is it needed?
  - Understanding risks and taking preventive measures to avoid or manage are good PM practice.

Reactive Risk Management

- project team reacts to risks when they occur
- mitigation—plan for additional resources in anticipation of fire fighting
- fix on failure—resource are found and applied when the risk strikes
- crisis management—when managing risk fails, then project is in jeopardy

Proactive Risk Management

- Intelligent strategy
- Begins before technical work is initiated
- Potential risks are identified, probability and impact are assessed and ranked by their importance
- primary objective is to avoid risk
- If not avoidable, develop a contingency plan to control the effects.

Software Risks Categories

- Product Specific Risks:
  - Project, Technical, Business
- Generic Risks:
  - Known risks: that can be uncovered after careful evaluation of project plan, business and technical environment (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment)
  - Predictable risks: extrapolated from past project experience (e.g., staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced).
  - Unpredictable risks: can and do occur and extremely difficult to identify in advance

Software Risks Categories

- Project Risks:
  - Threaten project plan
  - If it occurs then Project schedule will slip and cost will increase
  - Identify budgetary, schedule, resource, stakeholders and requirements problem
  - Risk factors include Project complexity, size and degree of structural uncertainty
- Technical Risks:
  - Threaten quality and timeliness
  - If it occurs then Implementation may become difficult or impossible
  - Identify Design, implementation, interface, verification and maintenance problems
  - Risk factors include Specification ambiguity, technical uncertainty, technical obsolescence, leading edge technology

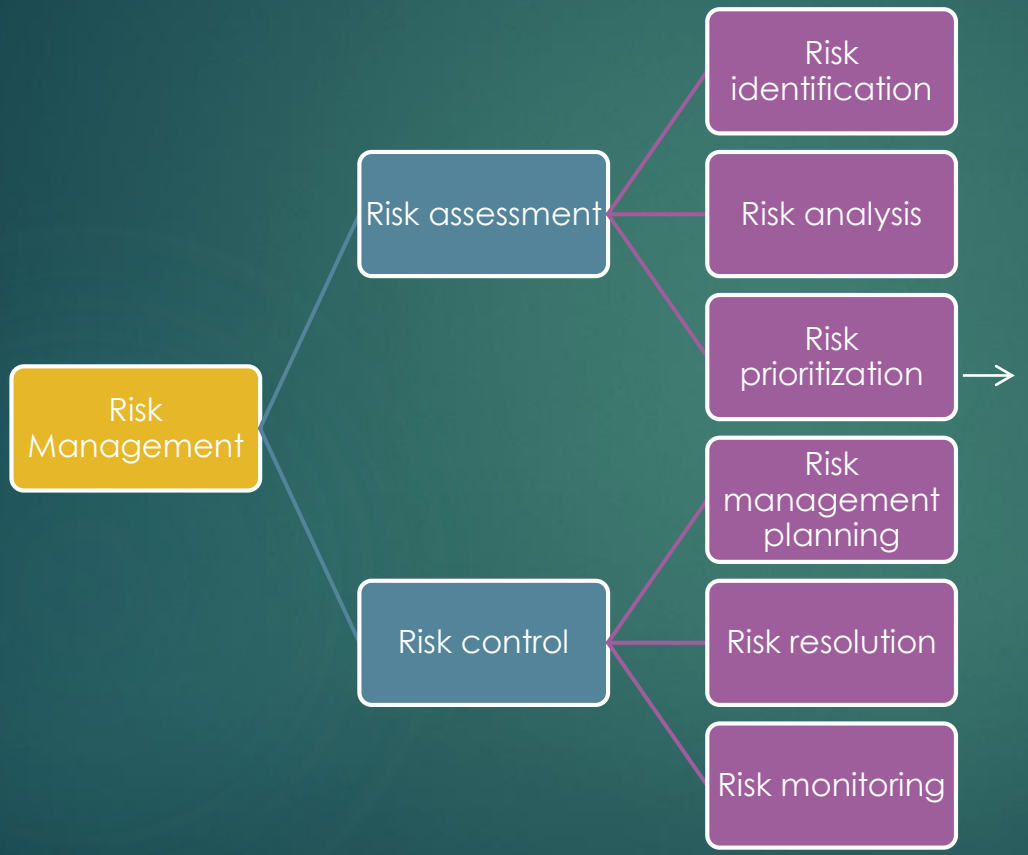
Software Risks Categories

- Business Risks:
  - Threaten the viability of software to be build
  - Jeopardize the project or product
  - Market risks: building an excellent product that nobody wants
  - Strategic risks: product that no longer fits into overall business strategy
  - Sales risk: product that sales force doesn't understand how to sell
  - Management risk: losing support of senior management due to loss in focus or change in people
  - Budget risk: losing budgetary or personnel commitment

Seven Principles of RM

- **Maintain a global perspective**
  - view software risks within the context of system and the business problem
- **Take a forward-looking view**
  - think about the risks that may arise in the future; establish contingency plans
- **Encourage open communication**
  - if someone states a potential risk, don't discount it.
- **Integrate**
  - a consideration of risk must be integrated into the software process
- **Emphasize a continuous process**
  - the team must be vigilant throughout the software process, modifying identified risks as more information is known and adding new ones as better insight is achieved.
- **Develop a shared product vision**
  - if all stakeholders share the same vision of the software, it likely that better risk identification and assessment will occur.
- **Encourage teamwork**
  - the talents, skills and knowledge of all stakeholder should be pooled

Steps in Risk management



- Probability of occurrence
- Consequences

Risk Exposure = Prob(R) * Loss(R)

Risk Identification

- Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.).
- By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and controlling them when necessary.
- There are two distinct types of risks for each of the categories mentioned earlier:
  - *Generic risks* are a potential threat to every software project.
  - *Product-specific risks* can be identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the software that is to be built.
  - To identify product-specific risks, the project plan and the software statement of scope are examined, and an answer to the following question is developed: "What special characteristics of this product may threaten our project plan?"

Risk Identification (checklist)

- *One method of identifying risk is to create a checklist that help focus on some subset of known and predictable risks*
  - *Product size*—risks associated with the overall size of the software to be built or modified.
  - *Business impact*—risks associated with constraints imposed by management or the marketplace.
  - *Customer characteristics*—risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
  - *Process definition*—risks associated with the degree to which the software process has been defined and is followed by the development organization.
  - *Development environment*—risks associated with the availability and quality of the tools to be used to build the product.
  - *Technology to be built*—risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.
  - *Staff size and experience*—risks associated with the overall technical and project experience of the software engineers who will do the work.

Risk Identification

- The risk item checklist can be organized in different ways.
- Questions relevant to each of the topics can be answered for each software project.
- The answers to these questions allow you to estimate the impact of risk.
- A different risk item checklist format simply lists characteristics that are relevant to each generic subcategory.
- Finally, a set of “risk components and drivers” are listed along with their probability of occurrence.

When should the risk plan be made

- Assess the project risk based on the following questions!
- Questions are based on past experience
- Questions in the order of their relative importance

Assessing Project Risk

1. Have top software and customer managers formally committed to support the project?
2. Are end-users enthusiastically committed to the project and the system/product to be built?
3. Are requirements fully understood by the software engineering team and their customers?
4. Have customers been involved fully in the definition of requirements?
5. Do end-users have realistic expectations?
6. Is project scope stable?
7. Does the software engineering team have the right mix of skills?
8. Are project requirements stable?
9. Does the project team have experience with the technology to be implemented?
10. Is the number of people on the project team adequate to do the job?
11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

If any one of these questions is answered negatively, mitigation, monitoring, and management steps should be instituted without fail. The degree to which the project is at risk is directly proportional to the number of negative responses to these questions.

Risk Components and Drivers

the project manager identify the risk drivers that affect software risk components—performance, cost, support, and schedule.

- *performance risk*—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- *cost risk*—the degree of uncertainty that the project budget will be maintained.
- *support risk*—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- *schedule risk*—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

Impact on assessment

The impact category is chosen based on the characterization that best fits the description in the table.

		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet requirements would result in mission failure		Failure results in increased costs and schedule delays	
	2	Significant degradation in non achievement of tech performanct	Non responsive or unsupportable staff	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet requirements would degrade system performance to a point where mission is questionable		Failure results in operational delays and/or increased costs with expected value of 100 K to 500 K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and / or schedule impact with expected value of less than $1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget underrun	Early achievable IOC

1. **Potential consequence of Undetected software errors or faults**
2. **Potential consequence If desired outcome is not achieved**

Risk Projection

- *Risk projection*, also called *risk estimation,* attempts to rate each risk in two ways
  - the likelihood or probability that the risk is real
  - the consequences of the problems associated with the risk, should it occur.
- There are four risk projection steps:
  - establish a scale that reflects the perceived likelihood of a risk
  - delineate the consequences of the risk
  - estimate the impact of the risk on the project and the product,
  - note the overall accuracy of the risk projection so that there will be no misunderstandings.
- The intent of these steps is to consider risks in a manner that leads to prioritization.
- No software team has the resources to address every possible risk with the same degree of rigor. By prioritizing risks, you can allocate resources where they will have the most impact.

Building the Risk Table

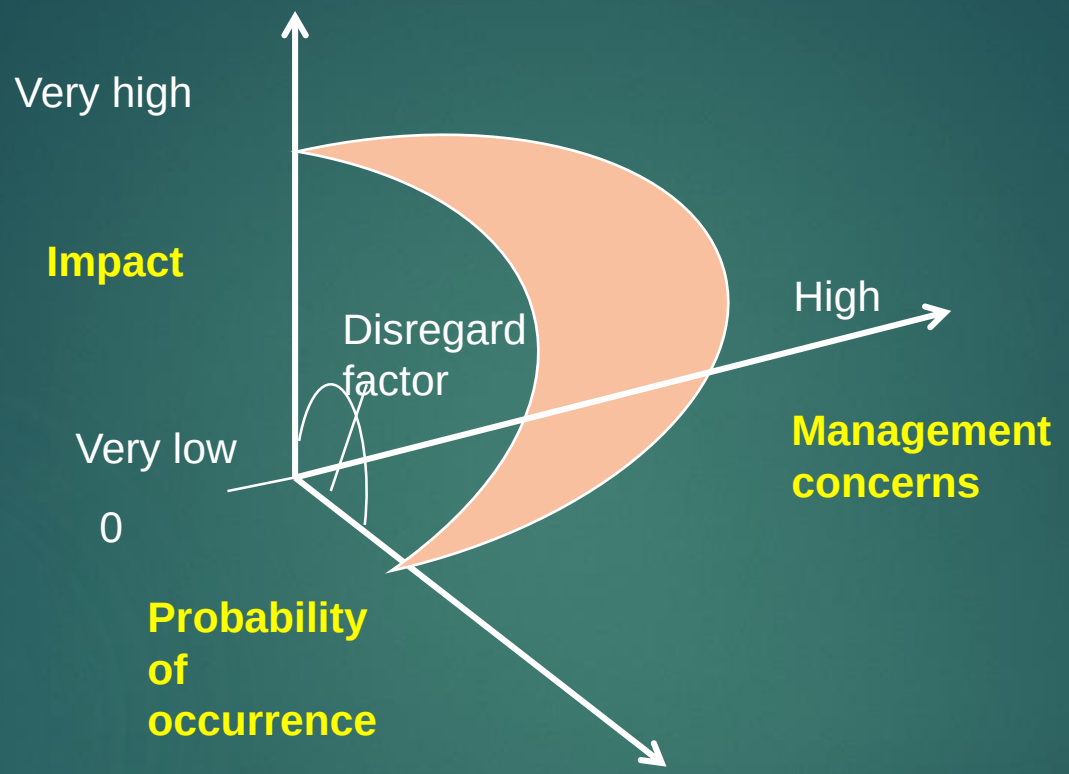
- List all risks (refer list item checklists)
- Categorize each risk
- Estimate the probability of occurrence
- Estimate the impact on the project on a scale of 1 to 5, where
  - 1 = low impact on project success
  - 5 = catastrophic impact on project success
- sort the table by probability and impact

Building a Risk Table

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover will be high	ST	60%	2	
Σ				
Σ				
Σ				

Impact values:
1—catastrophic
2—critical
3—marginal
4—negligible

Cutoff line



- **High impact – moderate to high probability**
- **Low impact – high probability**

Assessing Risk Impact

apply the following steps to determine the overall consequences of a risk:

(1) determine the average probability of occurrence value for each risk component;

(2) determine the impact for each component based on the criteria shown, and

(3) complete the risk table and analyze the results

The overall *risk exposure,* RE, is determined using the following relationship [HAL98]:

$$RE = P \times C$$

where

P is the probability of occurrence for a risk, and

C is the cost to the project should the risk occur.

Risk Exposure Example

- **Risk identification.** Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.
- **Risk probability.** 80% (likely).
- **Risk impact.** 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.
- **Risk exposure.** RE = 0.80 x 25,200 ~ $20,200.

Building a Risk Table

Risks	Category	Probability	Impact
Computer Crash	TI	70%	1
Late Delivery	BU	30%	1
Technology will not Meet Expectations	TE	25%	1
End Users Resist System	BU	20%	1
Changes in Requirements	PS	20%	2
Lack of Development Experience	TI	20%	2
Lack of Database Stability	TI	40%	2
Poor Quality Documentation	BU	35%	2
Deviation from Software Engineering Standards	PI	10%	3
Poor Comments in Code	TI	20%	4

Risk Exposure Example

- Consider a software project with 77 percent of risk probability in which 15 components were developed from the scratch. Each component have on an average 500 LOC and each LOC have an average cost of $10. Then the risk exposure can be calculated as

  - Cost = 15*500*10 = $75000
  - RE = 0.77 * 75000 = $57750

Risk Refinement

- Syntax for CTC format (Condition - transition - consequence) for risk reuse
- Given that <condition> then there is concern that (possibly) <consequence>
- E.g.
  - Given that all reusable software must conform to specific design standards and that some do no conform, then there is concern that (possibly) only 70% of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30% of components.

Risk Refinement

- Given that all reusable software must conform to specific design standards and that some do no conform, then there is concern that (possibly) only 70% of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30% of components.

- This is refined in following manner:
  - Subcondition1. certain reusable components were developed by a third party with no knowledge of internal design standards
  - Subcondition2. the design standard for component interfaces has not been solidified and may not conform to certain existing reusable components
  - Subcondition3. certain reusable components have been implemented in a language that is not supported on the target environment.

Risk Mitigation, Monitoring, and Management

- mitigation—how can we avoid the risk?
- monitoring—what factors can we track that will enable us to determine if the risk is becoming more or less likely?
- management—what contingency plans do we have if the risk becomes a reality?

Risk Information Sheet			
Risk ID: P02-4-32	Date: 5/9/04	Prob: 80%	Impact: high

Description:
Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

Refinement / context:

Subcondition1. certain reusable components were developed by a third party with no knowledge of internal design standards

Subcondition2. the design standard for component interfaces has not been solidified and may not conform to certain existing reusable components

Subcondition3. certain reusable components have been implemented in a language that is not supported on the target environment

Mitigation/monitoring:

1. Contact third party to determine conformance with design standards
2. Press for interface standards completion; consider component structure when deciding on interface protocol
3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired

Management / contingency plan / trigger:
RE computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly
Trigger: Mitigation steps unproductive as of 1/7/14

Current Status:
12/5/14: Mitigation steps initiated

Originator: D. Gagne	Assigned: B. Laster

Example

- Create a RMMM plan for high staff turnover project risk.

Risk Information Sheet

Risk ID: P02-4-32	Date: 7/2/18	Prob: 70%	Impact: critical

Description:
It is observed that generally there is high staff turnover. The rate of employees leaving the job is quite high.

Refinement / context:
Subcondition1. considerable time spent on knowledge transfer
Subcondition2. critical impact on project cost leading to increase in budget
Subcondition3. critical impact on project schedule causing delay in the project delivery to considerable extent

Mitigation/monitoring:
1. Meet with current staff to determine causes for turnover
2. Mitigate those causes that are under our control before the project starts
3. Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave
4. Monitor general attitude of team members based on project pressures
5. Interpersonal relationships among team members
6. Availability of jobs within company and outside it.

Management / contingency plan / trigger:
1. Organize project teams so that information about each development activity is widely dispersed.
2. Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner
3. Assign a backup staff member for every critical technologist
4. Allocate budget as a contingency measure well before the project initiates
5. Meet with customer to adjust the schedule of delivery of increments

Current Status:
7/2/18: Mitigation steps initiated

Originator: Kiran Bhowmick	Assigned:

Risk Due to Product Size

Attributes that affect risk:

- estimated size of the product in LOC or FP?
- estimated size of product in number of programs, files, transactions?
- percentage deviation in size of product from average for previous products?
- size of database created or used by the product?
- number of users of the product?
- number of projected changes to the requirements for the product? before delivery? after delivery?
- amount of reused software?

Risk Due to Business Impact

Attributes that affect risk:

- affect of this product on company revenue?
- visibility of this product by senior management?
- reasonableness of delivery deadline?
- number of customers who will use this product
- interoperability constraints
- sophistication of end users?
- amount and quality of product documentation that must be produced and delivered to the customer?
- governmental constraints
- costs associated with late delivery?
- costs associated with a defective product?

Risks Due to the Customer

Questions that must be answered:

- Have you worked with the customer in the past?
- Does the customer have a solid idea of requirements?
- Has the customer agreed to spend time with you?
- Is the customer willing to participate in reviews?
- Is the customer technically sophisticated?
- Is the customer willing to let your people do their job—that is, will the customer resist looking over your shoulder during technically detailed work?
- Does the customer understand the software engineering process?

Risks Due to Process Maturity

Questions that must be answered:

- Have you established a common process framework?
- Is it followed by project teams?
- Do you have management support for software engineering
- Do you have a proactive approach to SQA?
- Do you conduct formal technical reviews?
- Are CASE tools used for analysis, design and testing?
- Are the tools integrated with one another?
- Have document formats been established?

Technology Risks

Questions that must be answered:

- **Is the technology new to your organization?**
- **Are new algorithms, I/O technology required?**
- **Is new or unproven hardware involved?**
- **Does the application interface with new software?**
- **Is a specialized user interface required?**
- **Is the application radically different?**
- **Are you using new software engineering methods?**
- **Are you using unconventional software development methods, such as formal methods, AI-based approaches, artificial neural networks?**
- **Are there significant performance constraints?**
- **Is there doubt the functionality requested is "do-able?"**

Staff/People Risks

Questions that must be answered:

- **Are the best people available?**
- **Does staff have the right skills?**
- **Are enough people available?**
- **Are staff committed for entire duration?**
- **Will some people work part time?**
- **Do staff have the right expectations?**
- **Have staff received necessary training?**
- **Will turnover among staff be low?**

Recording Risk Information

Project: Embedded software for XYZ system
Risk type: schedule risk
Priority (1 low ... 5 critical): 4
Risk factor: Project completion will depend on tests which require hardware component under development. Hardware component delivery may be delayed
Probability: 60 %
Impact: Project completion will be delayed for each day that hardware is unavailable for use in software testing
Monitoring approach:
Scheduled milestone reviews with hardware group
Contingency plan:
Modification of testing strategy to accommodate delay using software simulation
Estimated resources: 6 additional person months beginning 7-1-96

Risks	Category	Probability	Impact
Computer Crash	TI	70%	1
Late Delivery	BU	30%	1
Technology will not Meet Expectations	TE	25%	1
End Users Resist System	BU	20%	1
Changes in Requirements	PS	20%	2
Lack of Development Experience	TI	20%	2
Lack of Database Stability	TI	40%	2
Poor Quality Documentation	BU	35%	2
Deviation from Software Engineering Standards	PI	10%	3
Poor Comments in Code	TI	20%	4

Risks	Category	Probability	Impact
Equipment failure	TI	70%	1
Late delivery	BU	30%	1
Technology will not meet expectations	TE	25%	1
End users resist system	BU	20%	1
Changes in requirements	PS	20%	2
Deviation from software engineering standards	PI	10%	3
Less reuse than planned	PS	60%	3
Poor comments in code	TI	20%	4

Software Engineering: A Practitioner's Approach, 6/e

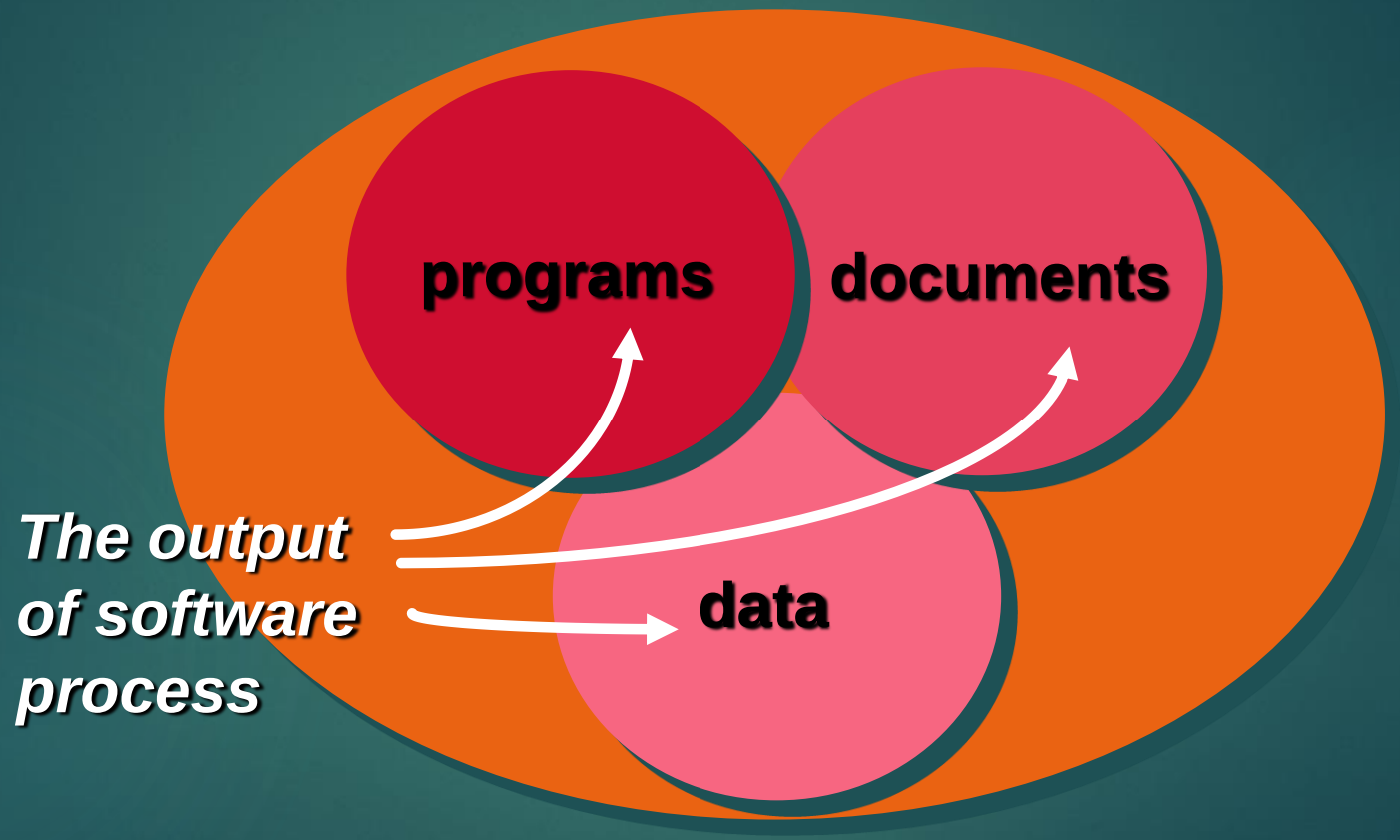
Chapter 27
Change Management

The "First Law"

No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle.

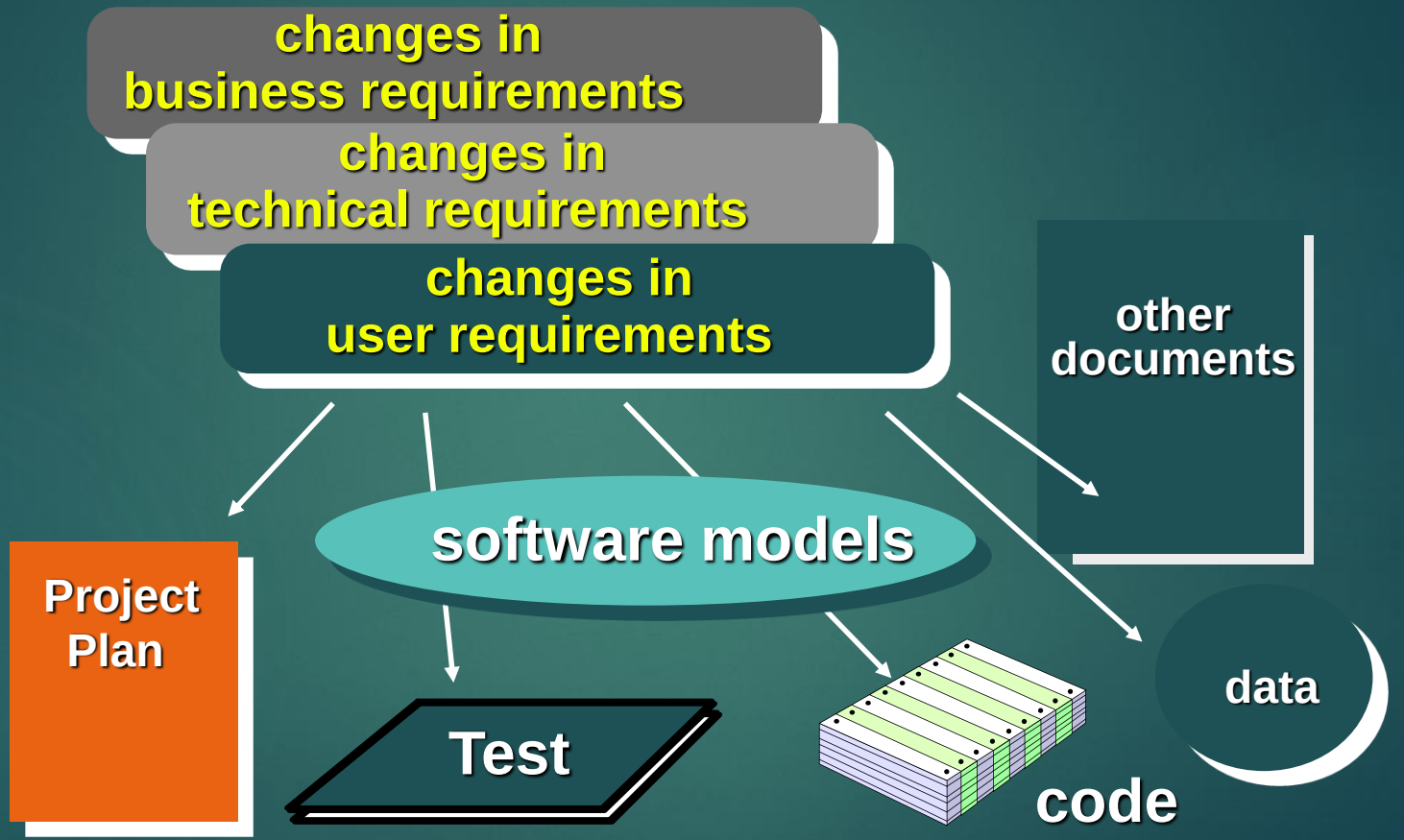
Bersoff, et al, 1980

The Software Configuration



Software configuration is the collection of all the items that comprise of all information produced as part of the software process

What Are These Changes?



What is SCM?

- It is a set of activities that have been developed to manage change throughout the life cycle of computer software.
- SCM can be viewed as a software quality assurance activity that is applied throughout the software process.

SCM Scenario

- Project Manager – in-charge of software group
- Configuration Manager – in-charge of CM procedure and policies
- Software Engineers – for developing and maintaining the software product
- Customer – uses the product and, indicates bugs and requests changes

SCM Elements

- *Component elements*—a set of tools coupled within a file management system (e.g., a database) that enables access to and management of each software configuration item.
- *Process elements*—a collection of procedures and tasks that define an effective approach to change management (and related activities) for all constituencies involved in the management, engineering and use of computer software.
- *Construction elements*—a set of tools that automate the construction of software by ensuring that the proper set of validated components (i.e., the correct version) have been assembled.
- *Human elements*—to implement effective SCM, the software team uses a set of tools and process features (encompassing other CM elements)

Baselines

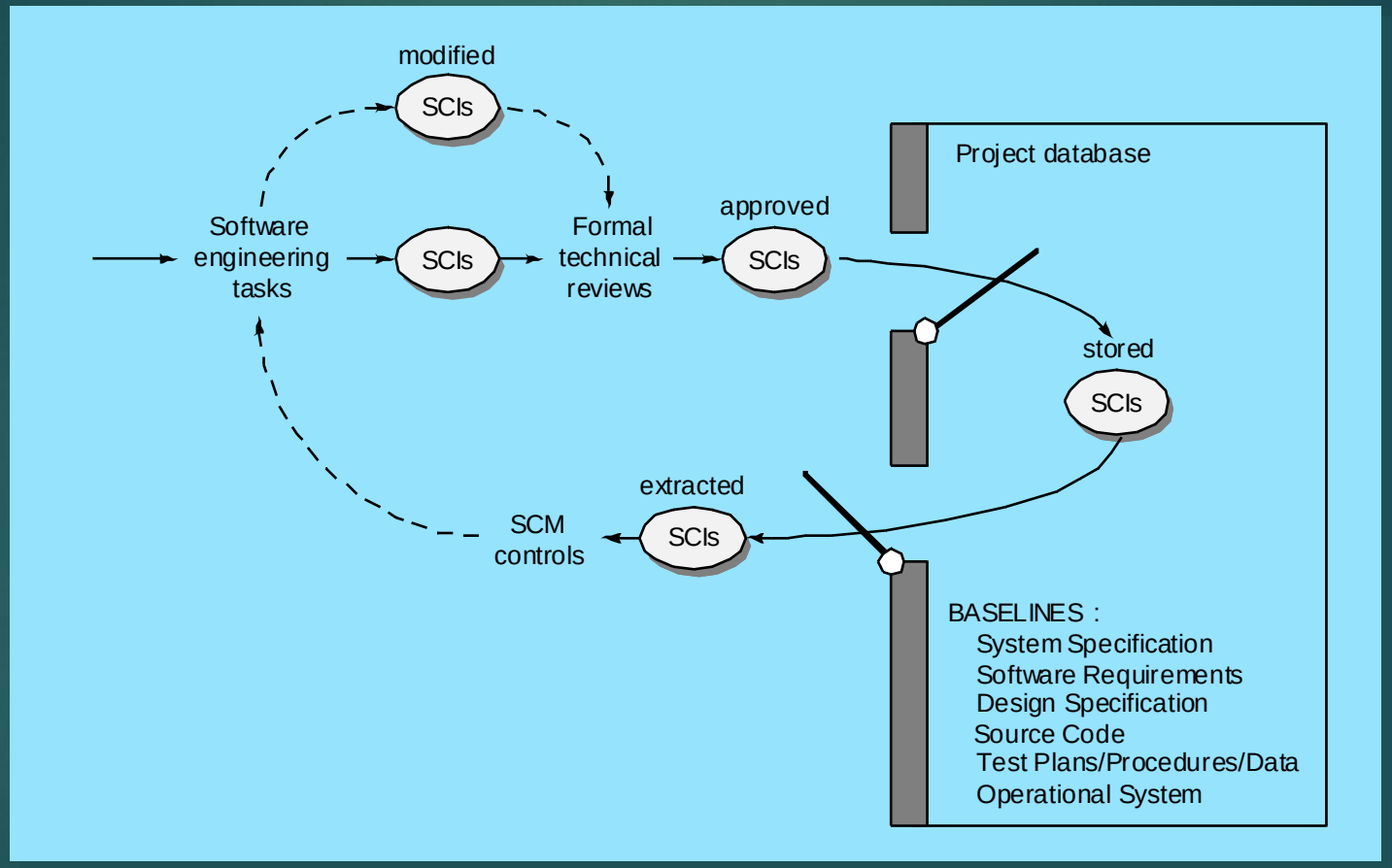
- A baseline is a Software configuration management concept that helps us to control change without seriously impeding justifiable change.

- The IEEE (IEEE Std. No. 610.12-1990) defines a baseline as:

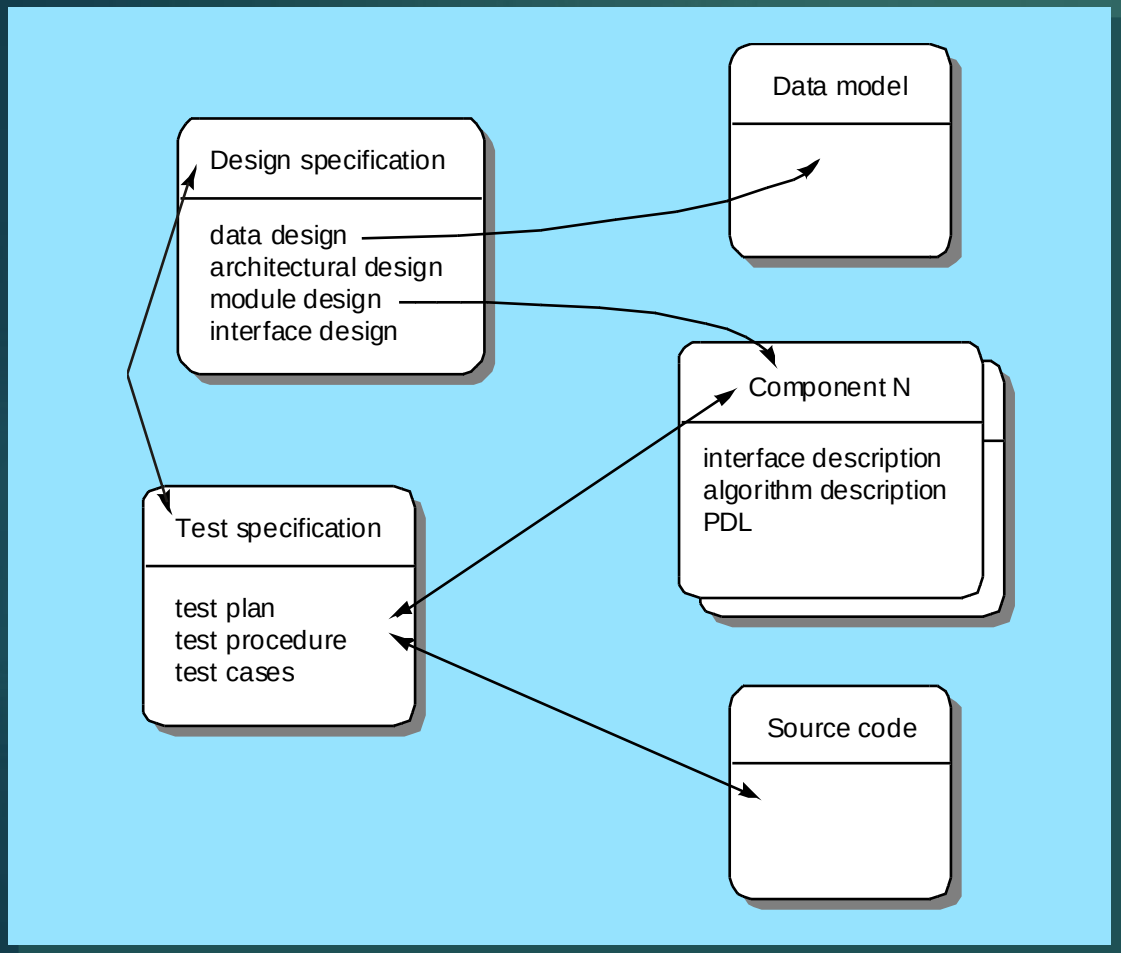
A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

- a baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and the approval of these SCIs is obtained through a formal technical review

Baselines


modified
SCIs
Software engineering tasks
SCIs
Formal technical reviews
approved
SCIs
Project database
stored
SCIs
extracted
SCIs
SCM controls
BASELINES :
System Specification
Software Requirements
Design Specification
Source Code
Test Plans/Procedures/Data
Operational System

Software Configuration Objects



SCIs are organized to form configuration objects that may be cataloged in the project database with a single name.

A *configuration object* has a name, attributes, and is "connected" to other objects by relationships

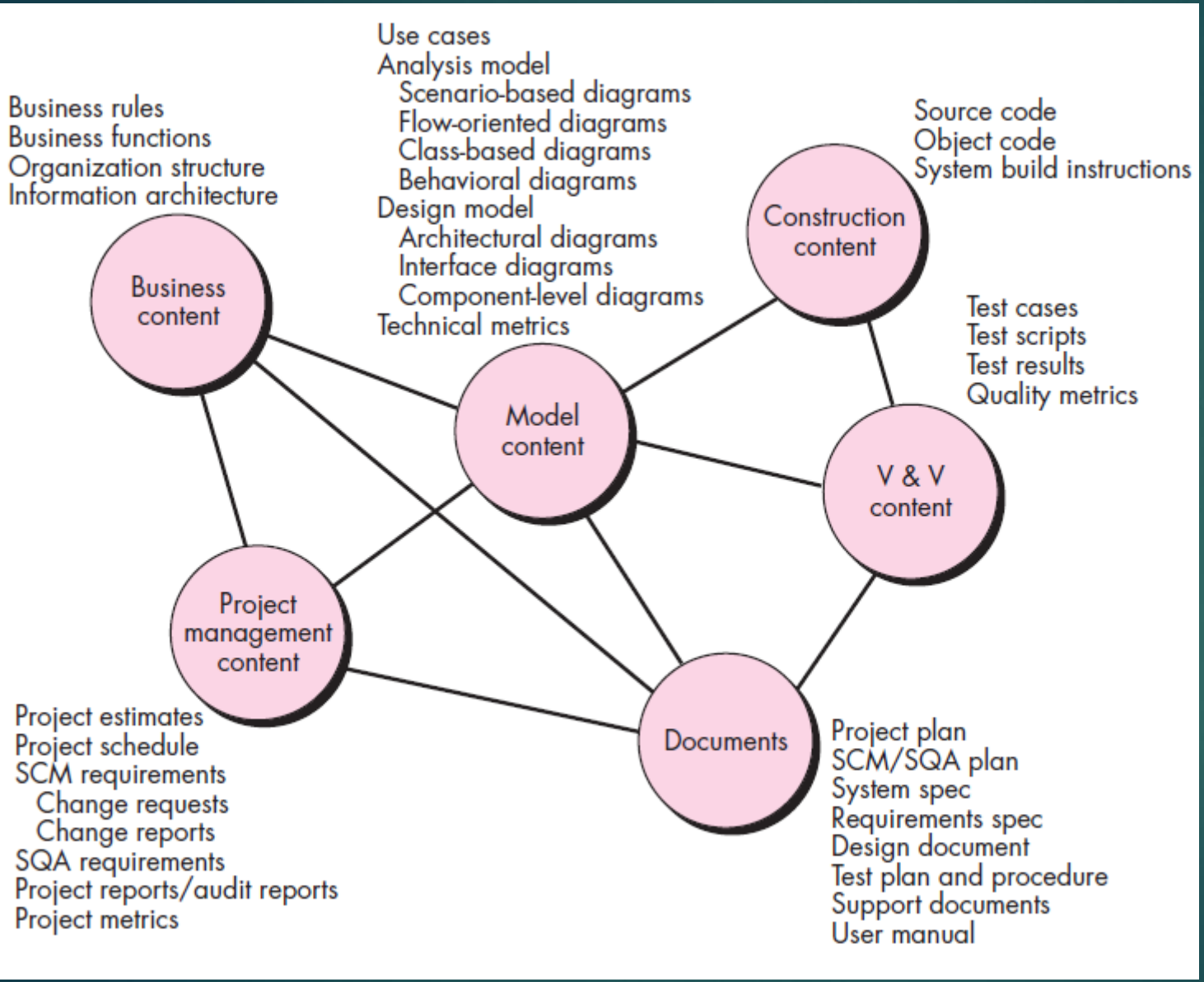
SCM Repository

- The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner
- The repository performs or precipitates the following functions
  - Data integrity – validate entries in repository, ensure consistency among related objects and automatically perform cascading modifications.
  - Information sharing – information sharing among multiple tools, multiple developers, multiple users and locking/unlocking objects to avoid inadvertent changes
  - Tool integration – provision for accessing objects and controlling access to the data through software engineering tools
  - Data integration – provides database function that allow various SCM tasks to perform on SCIs
  - Methodology enforcement – defined ER model amongst objects such that the relationships and objects define steps to build the contents of repository
  - Document standardization – standard approach for creation of Software Engineering documents

General Features and Content

- Two perspectives: What is stored in the repository and What services it provides
- Services provided by repository:
  - Sophisticated Database Management System services
  - Software engineering specific services
    - Integrate and Support Process management functions
    - Support specific rules that govern the SCM function and data maintained in the repository
    - Provide interface to other software engineering tools
    - Accommodate storage of sophisticated data objects like graphics, audio, video, text etc

Content of Repository



Repository Features

- **Versioning**
- **Dependency tracking and change management**
- **Requirements tracing**
- **Configuration management**
- **Audit trails**

- **Versioning.**
  - saves all of these versions to enable effective management of product releases and to permit developers to go back to previous versions
  - Control wide variety of object types like graphics, bit maps, complex documents, screens, reports
  - Track versions at different granularity levels like data definition or modules or component level
- **Dependency tracking and change management.**
  - The repository manages a wide variety of relationships among the data elements stored in it.
  - Relationships can be associations, dependencies or mandatory
  - Crucial to the integrity of stored products and generation of work products
  - E.g. if UML class design is changed, repository can detect if related classes, interfaces and code components also require change

Repository Features

- **Versioning**
- **Dependency tracking and change management**
- **Requirements tracing**
- **Configuration management**
- **Audit trails.**

- **Requirements tracing.**
  - Provides the ability to track all the design and construction components and deliverables that result from a specific requirement specification (forward tracing)
  - Given a work product which requirements are generated (backward tracing)

- **Configuration management.**
  - Keeps track of a series of configurations representing specific project milestones or production releases.

- **Audit trails.**
  - establishes additional information about when, why, and by whom changes are made.
  - Information about source of changes are entered as specific attributes of specific objects in repository

The SCM Process

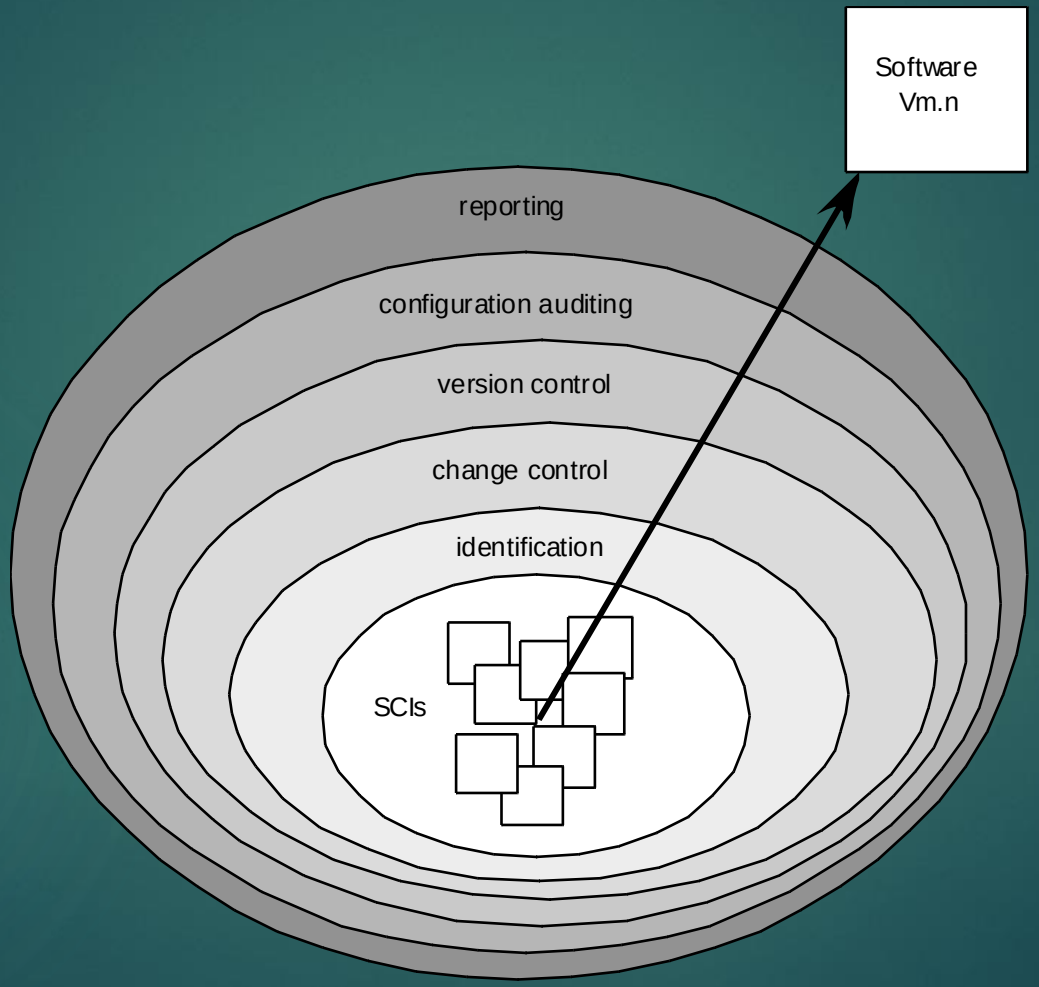
Four objectives:

- To identify all items that collectively define the software configuration
- To manage changes to one or more of these items
- To facilitate the construction of different versions of an application
- To ensure that software quality is maintained as the configuration evolves over time

Addresses the following questions …

- How does a software team identify the discrete elements of a software configuration?
- How does an organization manage the many existing versions of a program (and its documentation) in a manner that will enable change to be accommodated efficiently?
- How does an organization control changes before and after software is released to a customer?
- Who has responsibility for approving and ranking changes?
- How can we ensure that changes have been made properly?
- What mechanism is used to appraise others of changes that are made?

The SCM Process



Identification of objects in the software configuration

- Two types of objects:
  - Basic objects
    - unit of information created by s/w engg during analysis, design, code or test
    - E.g. section of SRS, part of design model, source code of component, suite of test cases
  - Aggregate objects
    - Collection of basic objects and aggregate of objects

Object

Name
Description
List of resources
Realization

Name - unique character string to identify object
Description - List of data items that identify the SCI type (model element, program, data), project identifier and change and/ or version information

Identification of objects in the software configuration

- Notation considering relationship amongst objects
  - Hierarchy of SCIs

Class diagram <part of> analysis model;

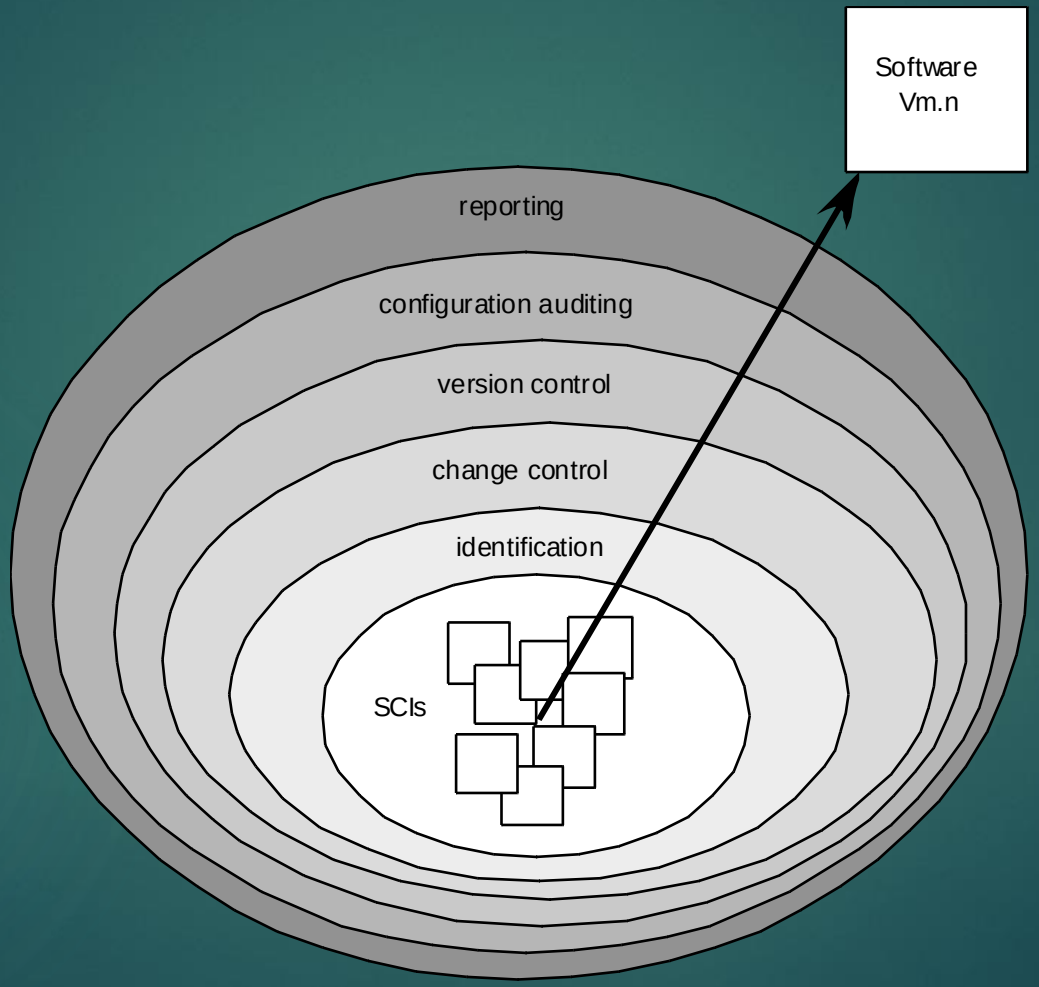
Analysis model <part of> requirements specification;

  - Cross structural relationships

Data model <interrelated> data flow model;

Data model <interrelated> test case class m;

The SCM Process



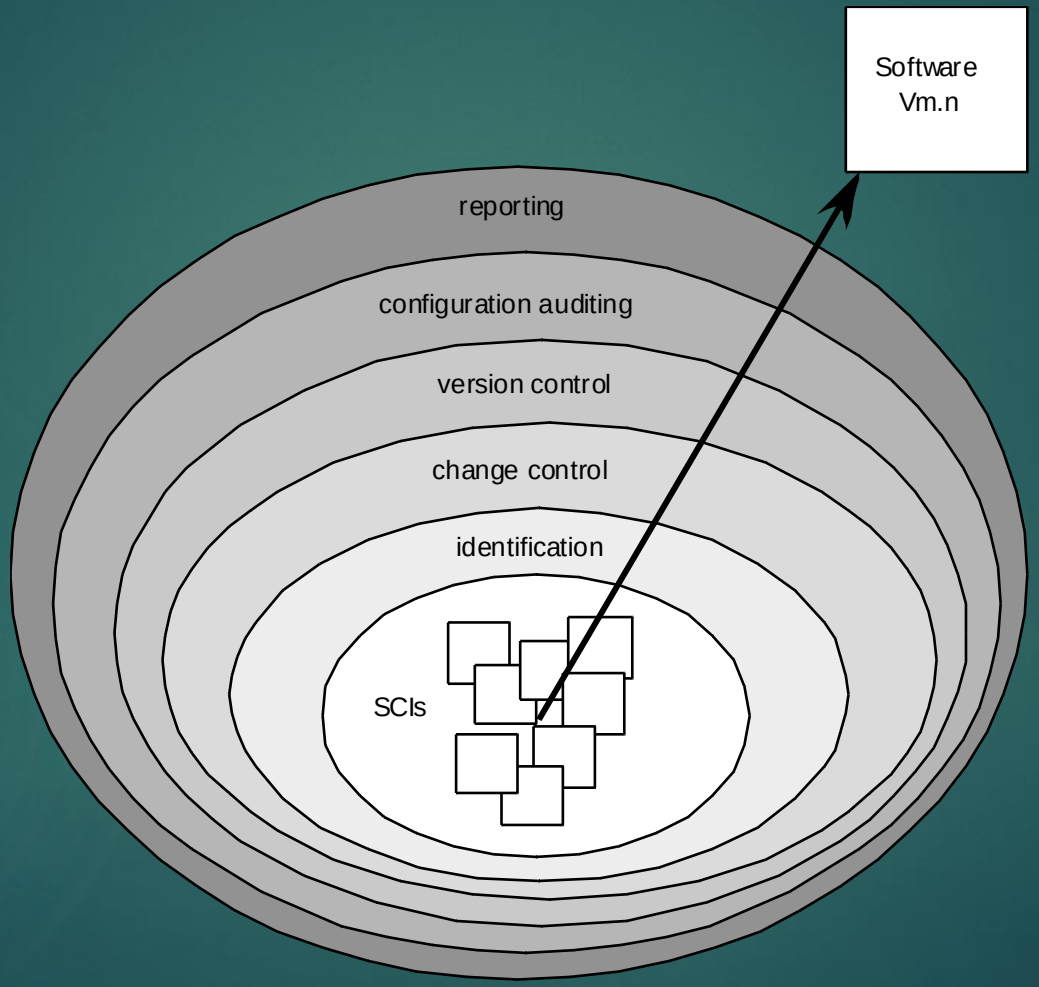
Version Control

- Version control combines procedures and tools to manage different versions of configuration objects that are created during the software process
- A version control system implements or is directly integrated with four major capabilities:
  - a *project database (repository)* that stores all relevant configuration objects
  - a *version management* capability that stores all versions of a configuration object (or enables any version to be constructed using differences from past versions);
  - a *make facility* that enables the software engineer to collect all relevant configuration objects and construct a specific version of the software.
  - an *issues tracking* (also called *bug tracking*) capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.

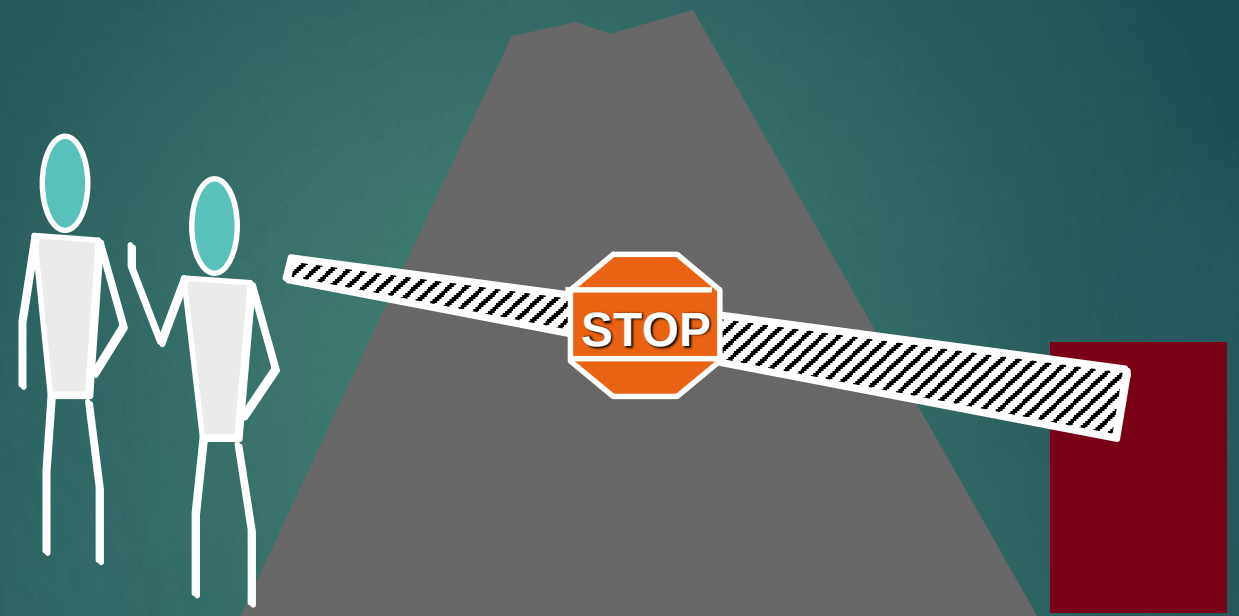
Version Control

- Change set – collection of all changes that are required to create a specific version of the software
- Captures all changes to all files in the configuration along with the reason for changes and details of who made the changes and when.
- A number of named change sets can be identified for an application or system. Software engineer can create a version by specifying the change sets name that must be applied to the baseline configuration

The SCM Process

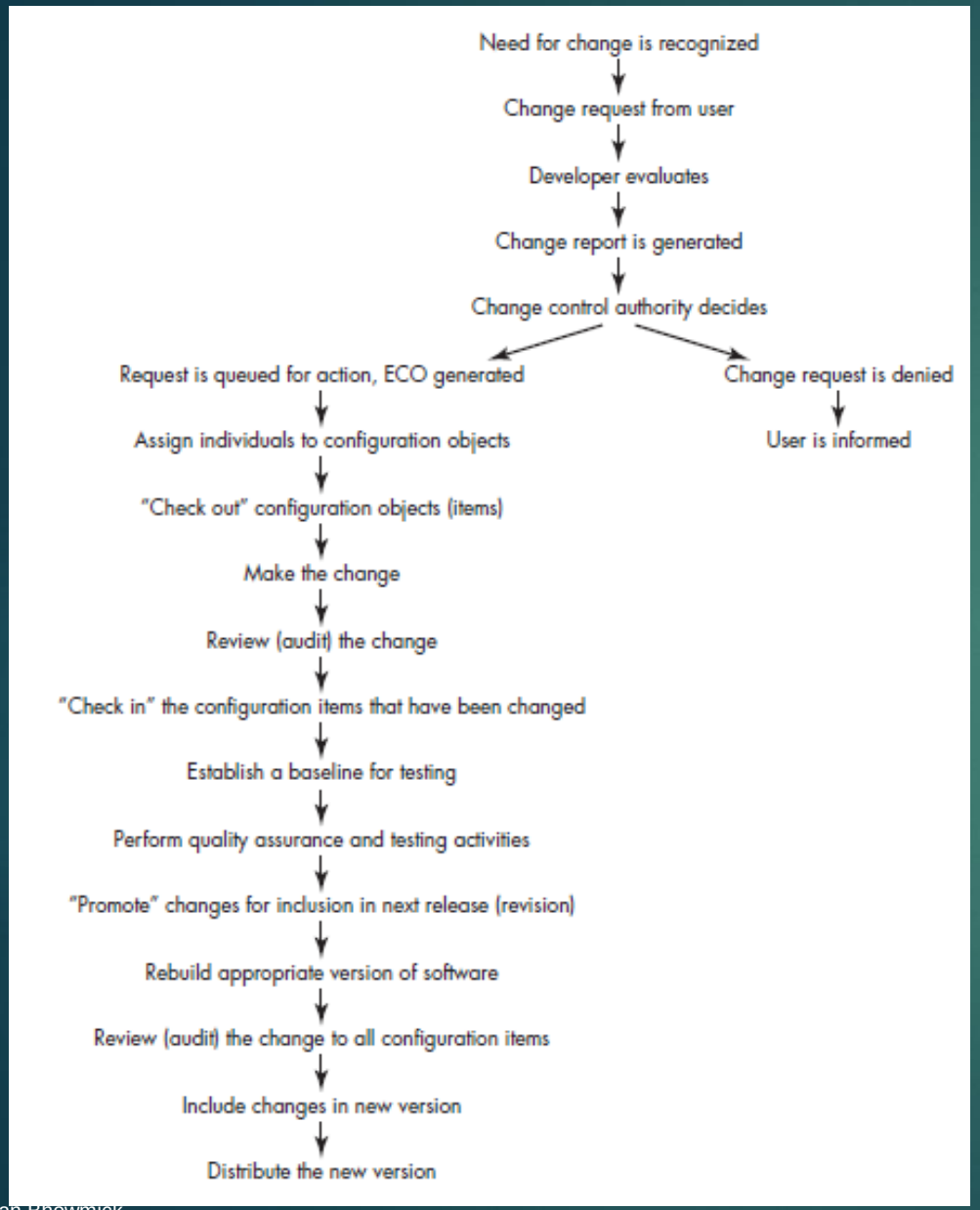


Change Control

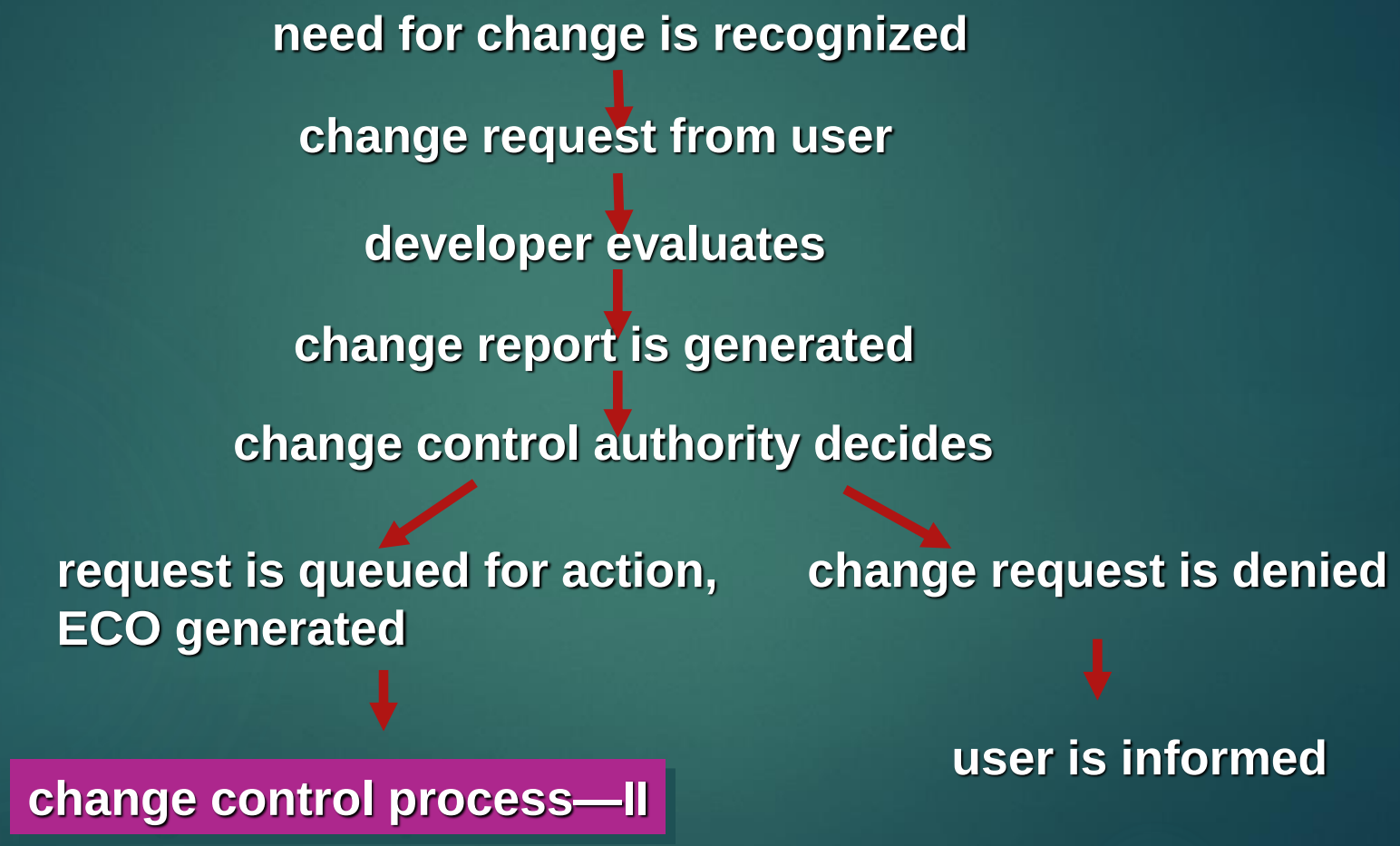


Reality of change control:
Too much change control, and we create problems.
Too little change control, and we create other problems

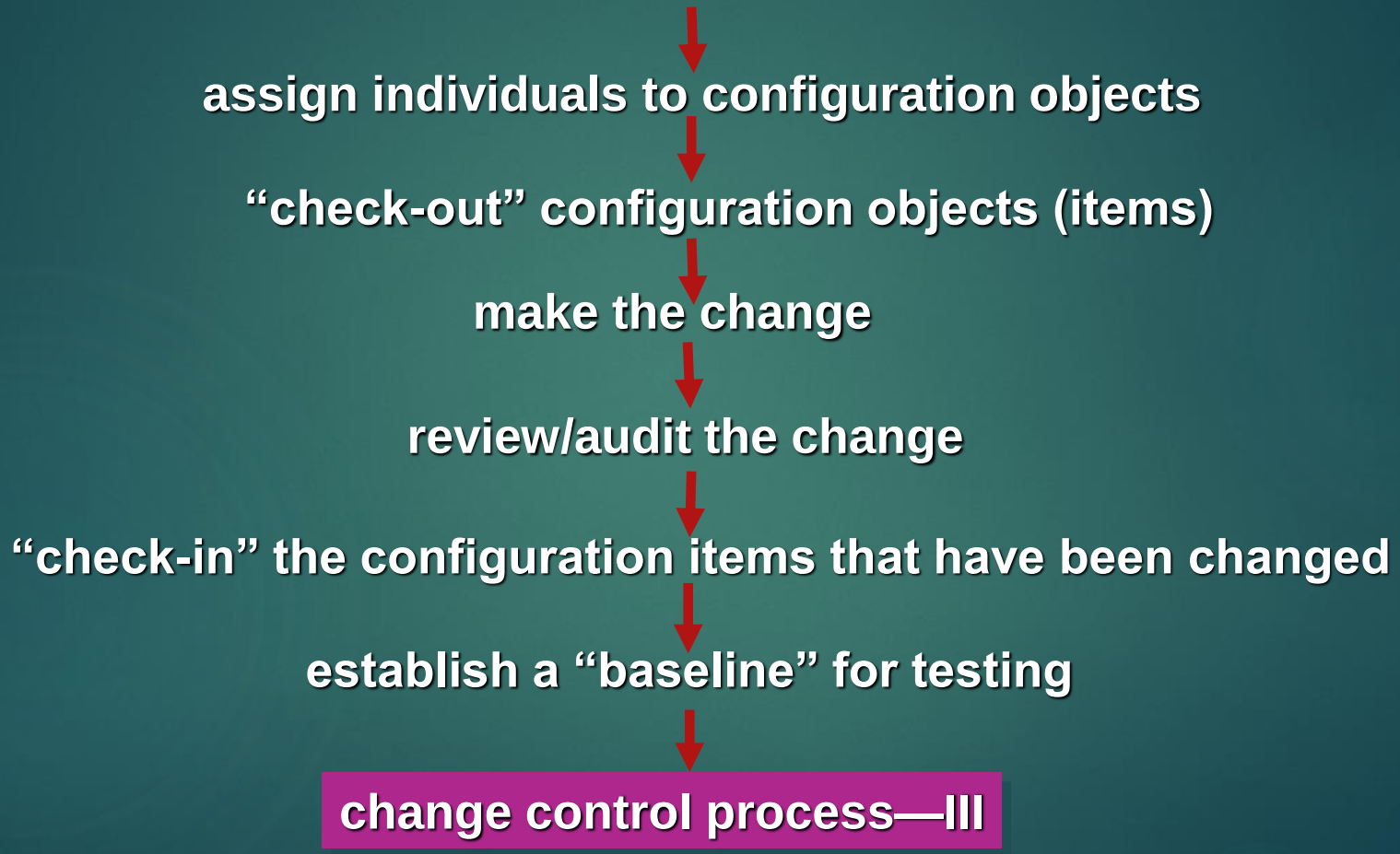
Change Control Process



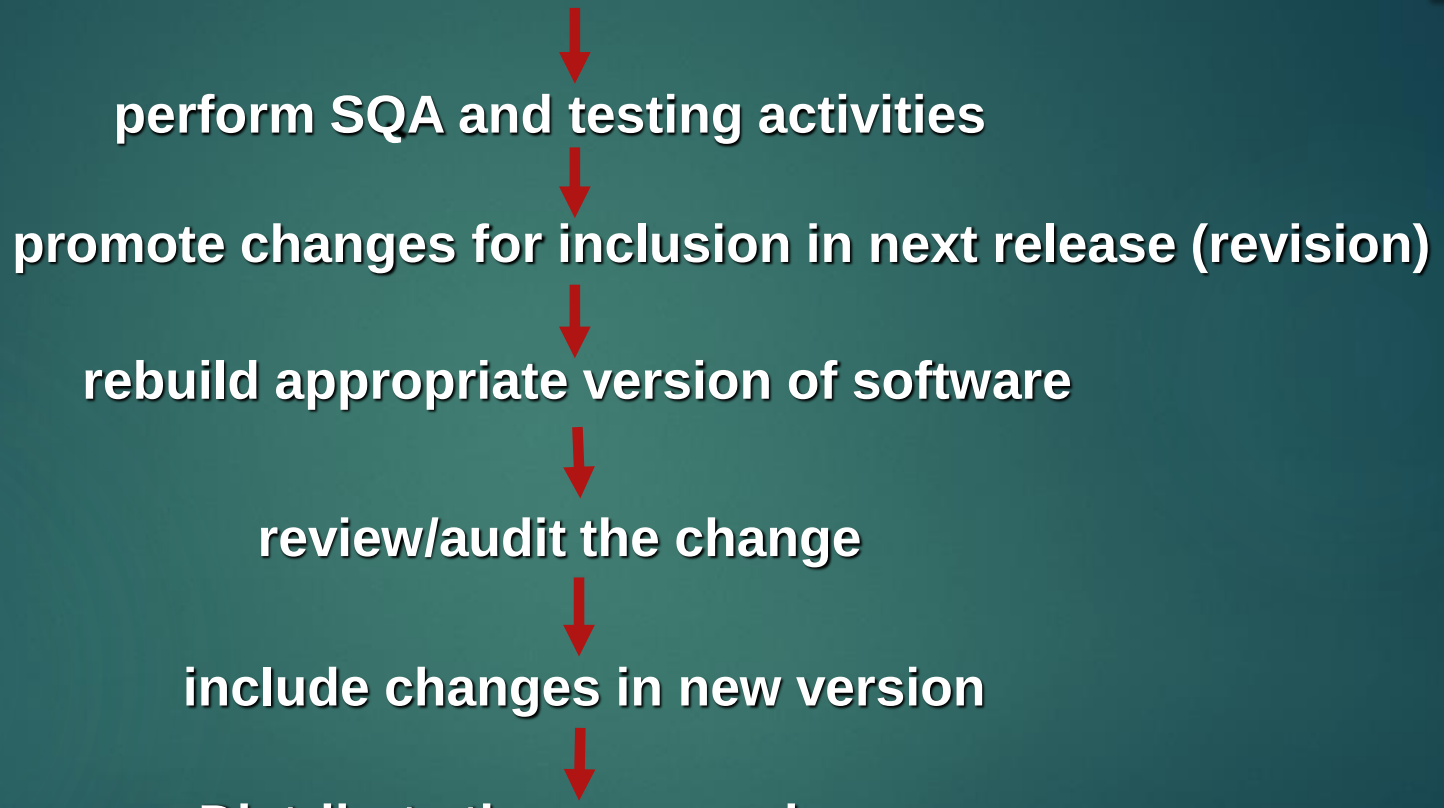
Change Control Process—I



Change Control Process-II

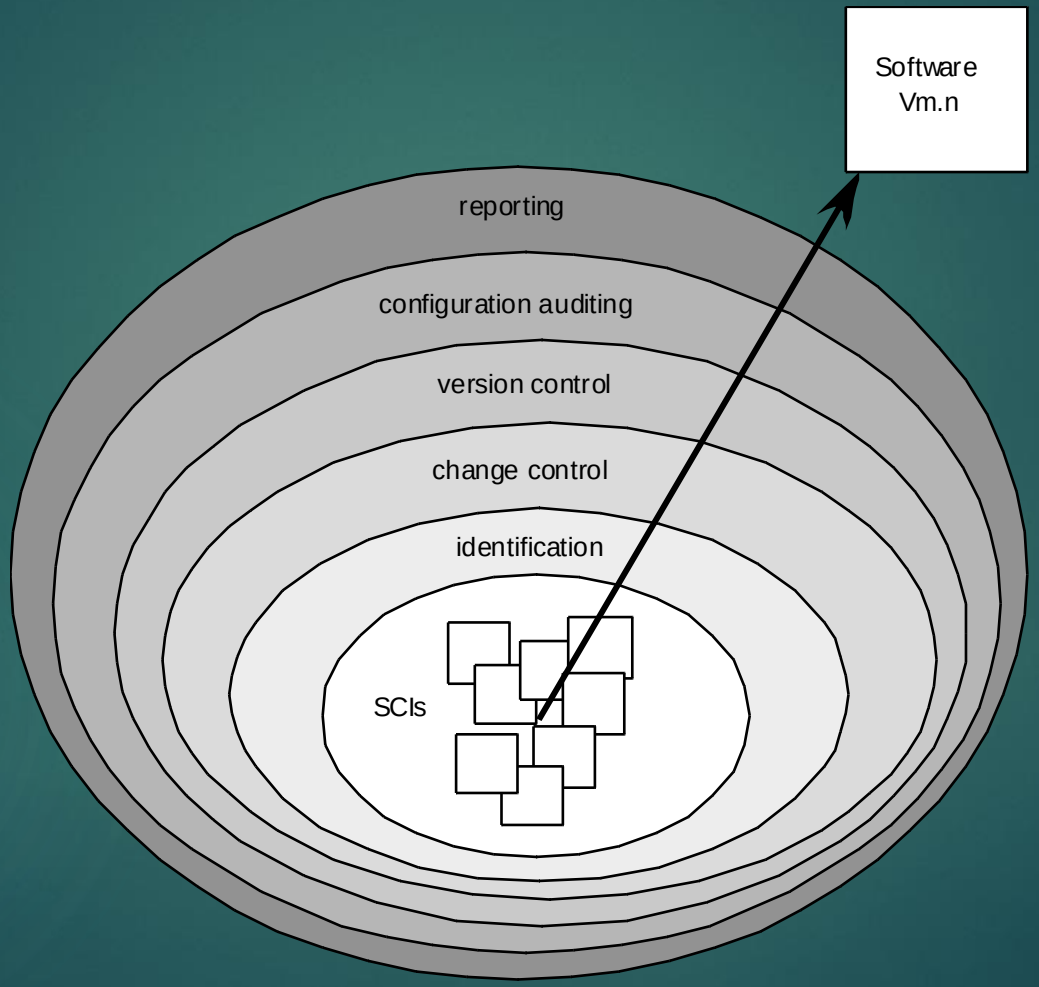


Change Control Process-III

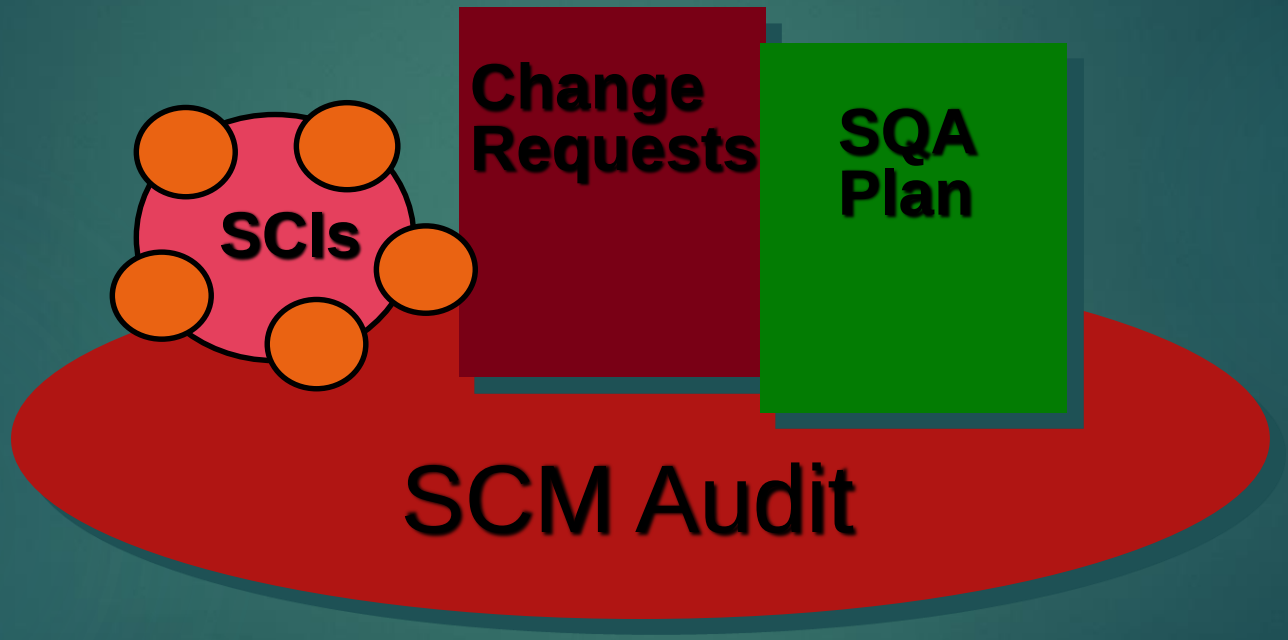


- Version control mechanisms integrated within change control process implements two important elements
- Access control – governs which software engineers have the authority to access
- Synchronization control – helps to ensure that parallel changes don't overwrite one another

The SCM Process



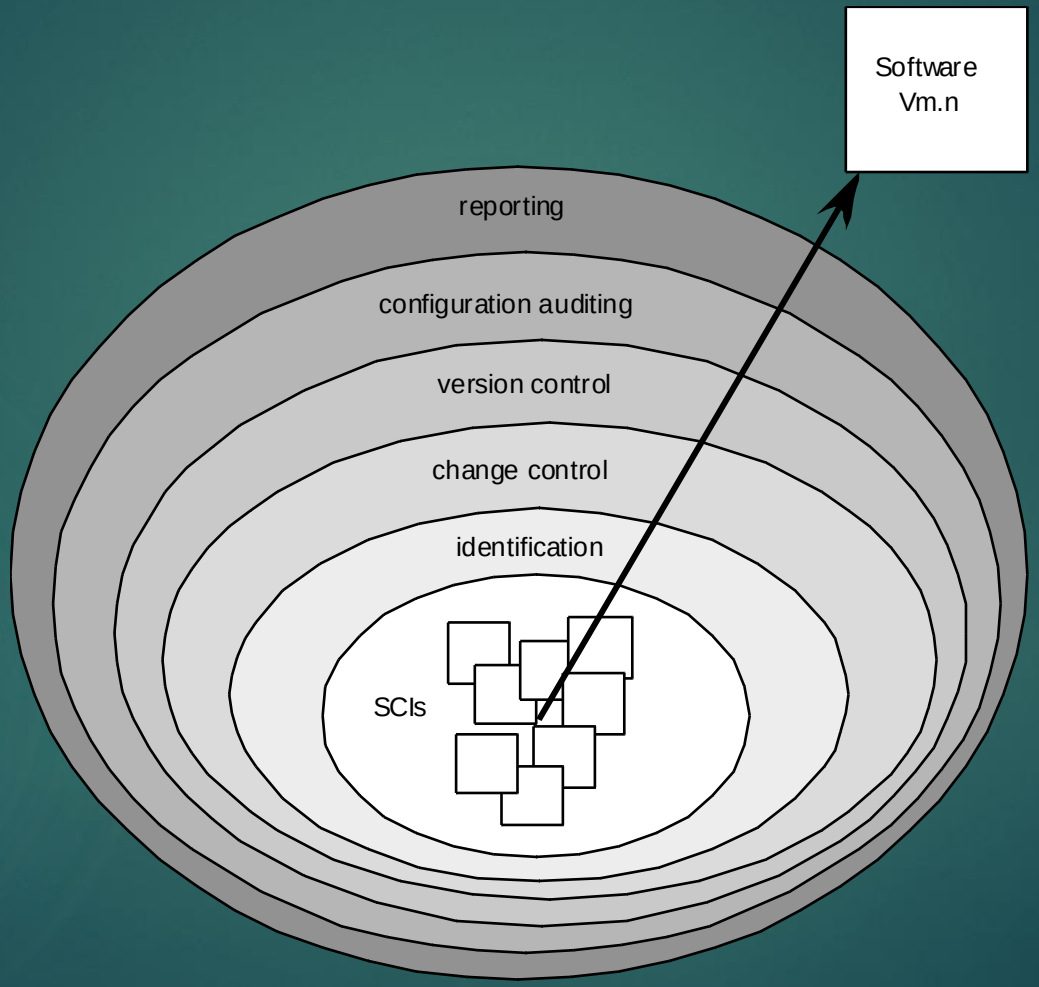
Auditing



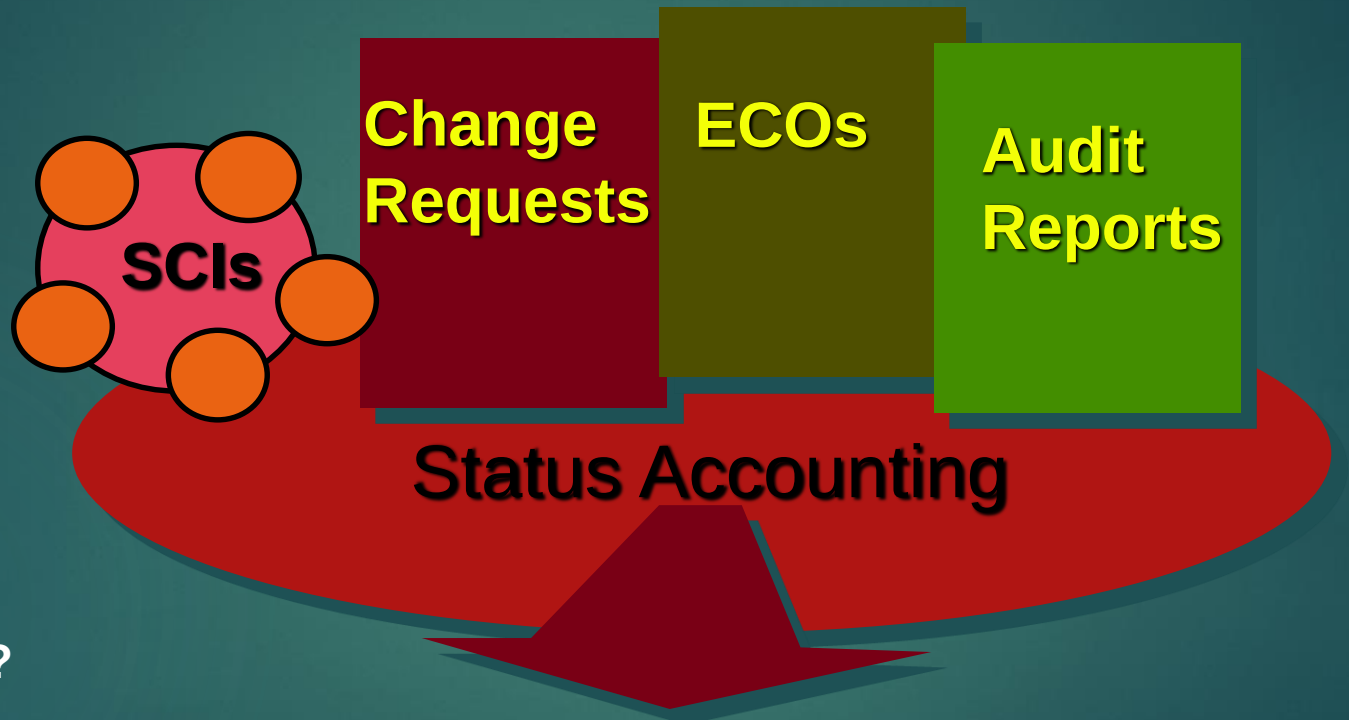
Configuration Audit

- To ensure change is properly implemented
  - Formal technical review – focusses on technical correctness of configuration object that has been modified
  - Software configuration audit –
    - Has the change been specified in ECO? Any additional modification?
    - Has FTR been conducted to assess the technical correctness
    - Has software process been followed and standards applied?
    - Has change been highlighted in SCI? Change data and change author specified?
    - Have SCM procedure followed for noting the change, recording it and reporting it been followed
    - Have all related SCIs been properly updated?

The SCM Process



Configuration Status reporting / Status Accounting



What happened?
Who did it?
When did it happen?
What else will be effected?