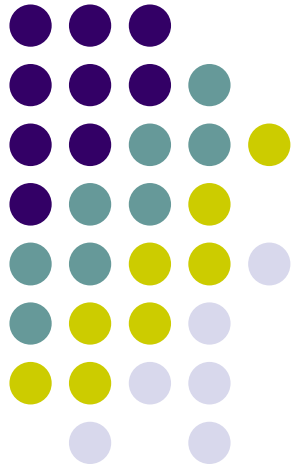
INTRODUCTION TO SOFTWARE ENGINEERING

Part I – Overview of Software Engineering



Overview of Software Engineering

- What is software engineering?
- Software process models
- Software project management
  - Managing people
  - Quality management
  - Software configuration management
  - …

Topics

- Selecting staff
- Motivating people
- Managing groups

Selecting staff

- People are an organisation's most important assets.
- An important project management task is team selection.
- Information on selection comes from:
  - Information provided by the candidates.
  - Information gained by interviewing and talking with candidates.
  - Recommendations and comments from other people who know or who have worked with the candidates.

Staff selection factors

Application domain experience	For a project to develop a successful system, the developers must understand the application domain. It is essential that some members of a development team have some domain experience.
Platform experience	This may be significant if low-level programming is involved. Otherwise, not usually a critical attribute.
Programming language experience	This is normally only significant for short duration projects where there is not enough time to learn a new language. While learning a language itself is not difficult, it takes several months to become proficient in using the associated libraries and components.
Problem solving ability	This is very important for software engineers who constantly have to solve technical problems. However, it is almost impossible to judge without knowing the work of the potential team member.

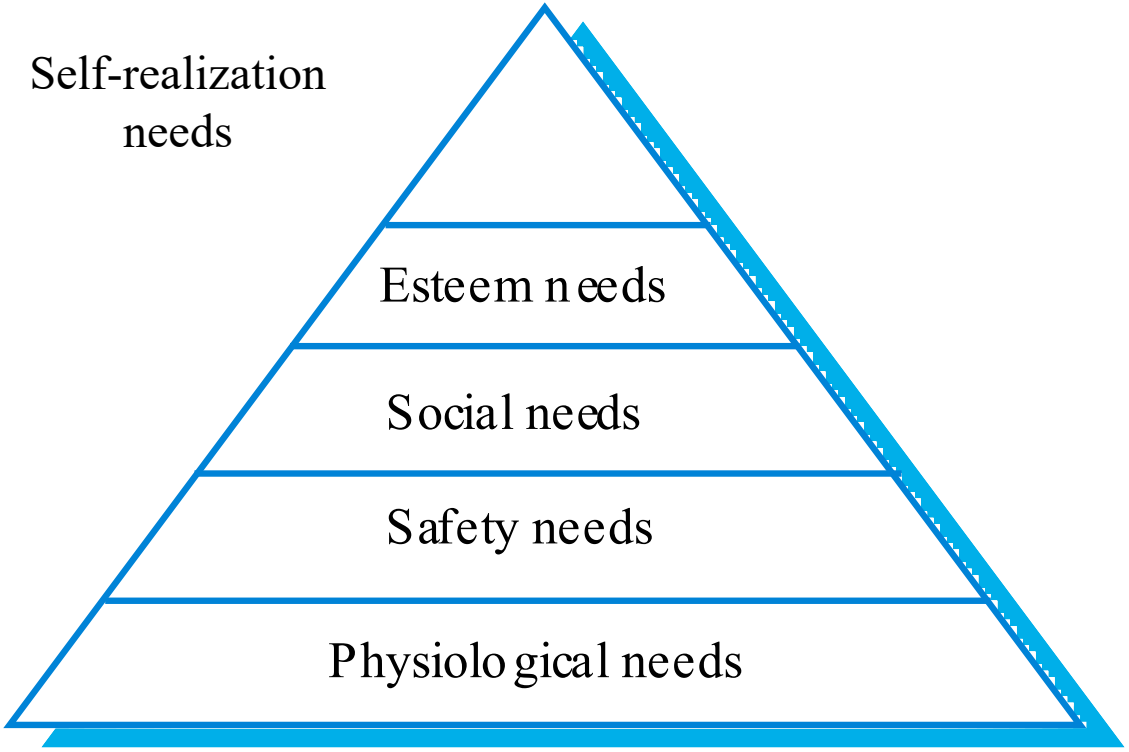
Staff selection factors

Educational background	This may provide an indicator of the basic fundamentals that the candidate should know and of their ability to learn. This factor becomes increasingly irrelevant as engineers gain experience across a range of projects.
Communication ability	This is important because of the need for project staff to communicate orally and in writing with other engineers, managers and customers.
Adaptability	Adaptability may be judged by looking at the different types of experience that candidates have had. This is an important attribute as it indicates an ability to learn.
Attitude	Project staff should have a positive attitude to their work and should be willing to learn new skills. This is an important attribute but often very difficult to assess.
Personality	This is an important attribute but difficult to assess. Candidates must be reasonably compatible with other team members. No particular type of personality is more or less suited to software engineering.

Motivating people

- An important role of a manager is to motivate the people working on a project.
- Motivation is a complex issue but it appears that there are different types of motivation based on:
  - Basic needs (e.g. food, sleep, etc.);
  - Personal needs (e.g. respect, self-esteem);
  - Social needs (e.g. to be accepted as part of a group).

Human needs hierarchy



Refer: Ian Sommerville, Software Engineering 8th edition.

Need satisfaction

- Social
  - Provide communal facilities;
  - Allow informal communications.
- Esteem
  - Recognition of achievements;
  - Appropriate rewards.
- Self-realization
  - Training - people want to learn more;
  - Responsibility.

Personality types

- The needs hierarchy is almost certainly an over-simplification of motivation in practice.
- Motivation should also take into account different personality types:
  - Task-oriented;
  - Self-oriented;
  - Interaction-oriented.

Personality types

- Task-oriented
  - The motivation for doing the work is the work itself;
- Self-oriented
  - The work is a means to an end which is the achievement of individual goals - e.g. to get rich, to play tennis, to travel etc.;
- Interaction-oriented
  - The principal motivation is the presence and actions of co-workers. People go to work because they like to go to work.

Managing groups

- Most software engineering is a group activity
  - The development schedule for most non-trivial software projects is such that they cannot be completed by one person working alone.
- Group interaction is a key determinant of group performance.
- Flexibility in group composition is limited
  - Managers must do the best they can with available people.

Factors influencing group working

- Group composition
- Group cohesiveness
- Group communications
- Group organization

Group composition

- Group composed of members who share the same motivation can be problematic
  - Task-oriented - everyone wants to do their own thing;
  - Self-oriented - everyone wants to be the boss;
  - Interaction-oriented - too much chatting, not enough work.
- An effective group has a balance of all types.
- This can be difficult to achieve software engineers are often task-oriented.
- Interaction-oriented people are very important as they can detect and defuse tensions that arise.

Group leadership

- Leadership depends on respect not titular status.
- There may be both a technical and an administrative leader.
- Democratic leadership is more effective that autocratic leadership.

Group cohesiveness

- In a cohesive group, members consider the group to be more important than any individual in it.
- The advantages of a cohesive group are:
  - Group quality standards can be developed;
  - Group members work closely together so inhibitions caused by ignorance are reduced;
  - Team members learn from each other and get to know each other's work;
  - Egoless programming where members strive to improve each other's programs can be practised.

Developing cohesiveness

- Cohesiveness is influenced by factors such as the organisational culture and the personalities in the group.
- Cohesiveness can be encouraged through
  - Social events;
  - Developing a group identity and territory;
  - Explicit team-building activities.
- Openness with information is a simple way of ensuring all group members feel part of the group.

Group loyalties

- Group members tend to be loyal to cohesive groups.
- 'Groupthink' is preservation of group irrespective of technical or organizational considerations.
- Management should act positively to avoid groupthink by forcing external involvement with each group.

Group communications

- Good communications are essential for effective group working.
- Information must be exchanged on the status of work, design decisions and changes to previous decisions.
- Good communications also strengthens group cohesion as it promotes understanding.

Group communications

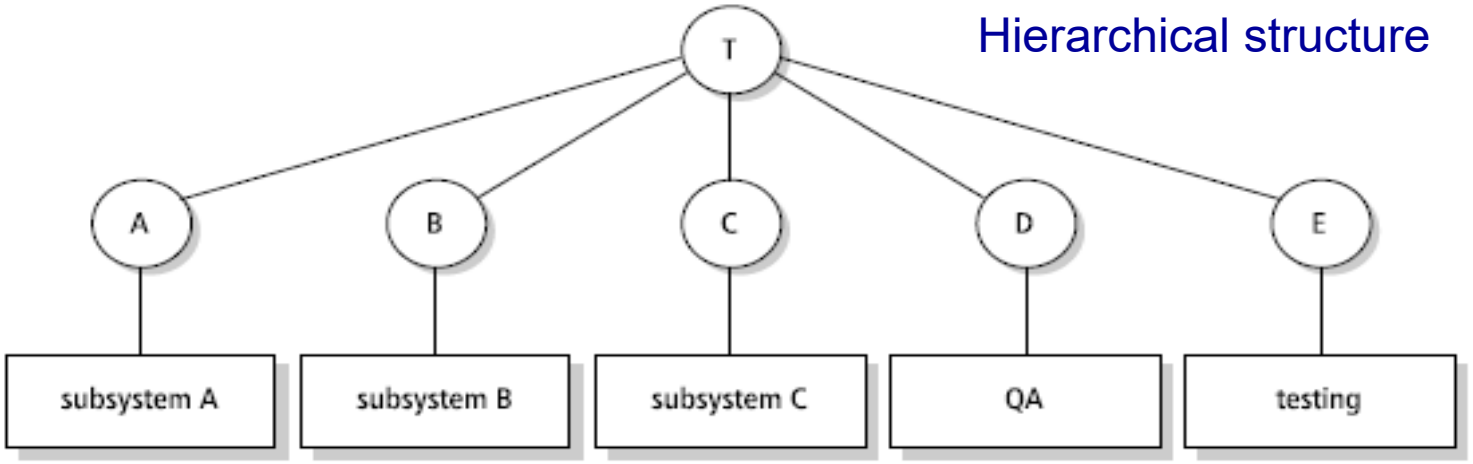
- Group size
  - The larger the group, the harder it is for people to communicate with other group members.
- Group structure
  - Communication is better in informally structured groups than in hierarchically structured groups.
- Group composition
  - Communication is better when there are different personality types in a group and when groups are mixed rather than single sex.
- The physical work environment
  - Good workplace organisation can help encourage communications.

Group organisation

- Small software engineering groups are usually organised informally without a rigid structure.
- For large projects, there may be a hierarchical structure where different groups are responsible for different sub-projects.

Group organisation

Hierarchical structure



Matrix structure

	real-time programming	graphics	databases	QA	testing
project C	X			X	X
project B	X		X	X	X
project A		X	X	X	X

Extreme programming groups; Chief programmer teams; SWAT (Skilled With Advanced Tools); …

Overview of Software Engineering

- What is software engineering?
- Software process models
- Software project management
  - Managing people
  - Quality management
  - Software configuration management
  - …

Topics

- Quality assurance and standards
- Quality planning
- Quality control

Software quality management

- Concerned with ensuring that the required level of quality is achieved in a software product.
- Involves defining appropriate quality standards and procedures and ensuring that these are followed.
- Should aim to develop a ‘quality culture’ where quality is seen as everyone’s responsibility.

Refer: Ian Sommerville, Software Engineering 8th edition.

What is quality?

- Quality, simplistically, means that a product should meet its specification.
- This is problematical for software systems
  - There is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.);
  - Some quality requirements are difficult to specify in an unambiguous way;
  - Software specifications are usually incomplete and often inconsistent.

the quality compromise

Refer: Ian Sommerville, Software Engineering 8th edition.

Scope of quality management

- Quality management is particularly important for large, complex systems. The quality documentation is a record of progress and supports continuity of development as the development team changes.
- For smaller systems, quality management needs less documentation and should focus on establishing a quality culture.

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality management activities

- Quality assurance
  - Establish organisational procedures and standards for quality.
- Quality planning
  - Select applicable procedures and standards for a particular project and modify these as required.
- Quality control
  - Ensure that procedures and standards are followed by the software development team.
- Quality management should be separate from project management to ensure independence.

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality assurance and standards

- Standards are the key to effective quality management.
- They may be international, national, organizational or project standards.
- Product standards define characteristics that all components should exhibit e.g. a common programming style.
- Process standards define how the software process should be enacted.

Importance of standards

- Encapsulation of best practice- avoids repetition of past mistakes.
- They are a framework for quality assurance processes - they involve checking compliance to standards.
- They provide continuity - new staff can understand the organisation by understanding the standards that are used.

Refer: Ian Sommerville, Software Engineering 8th edition.

Product and process standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of documents to CM
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

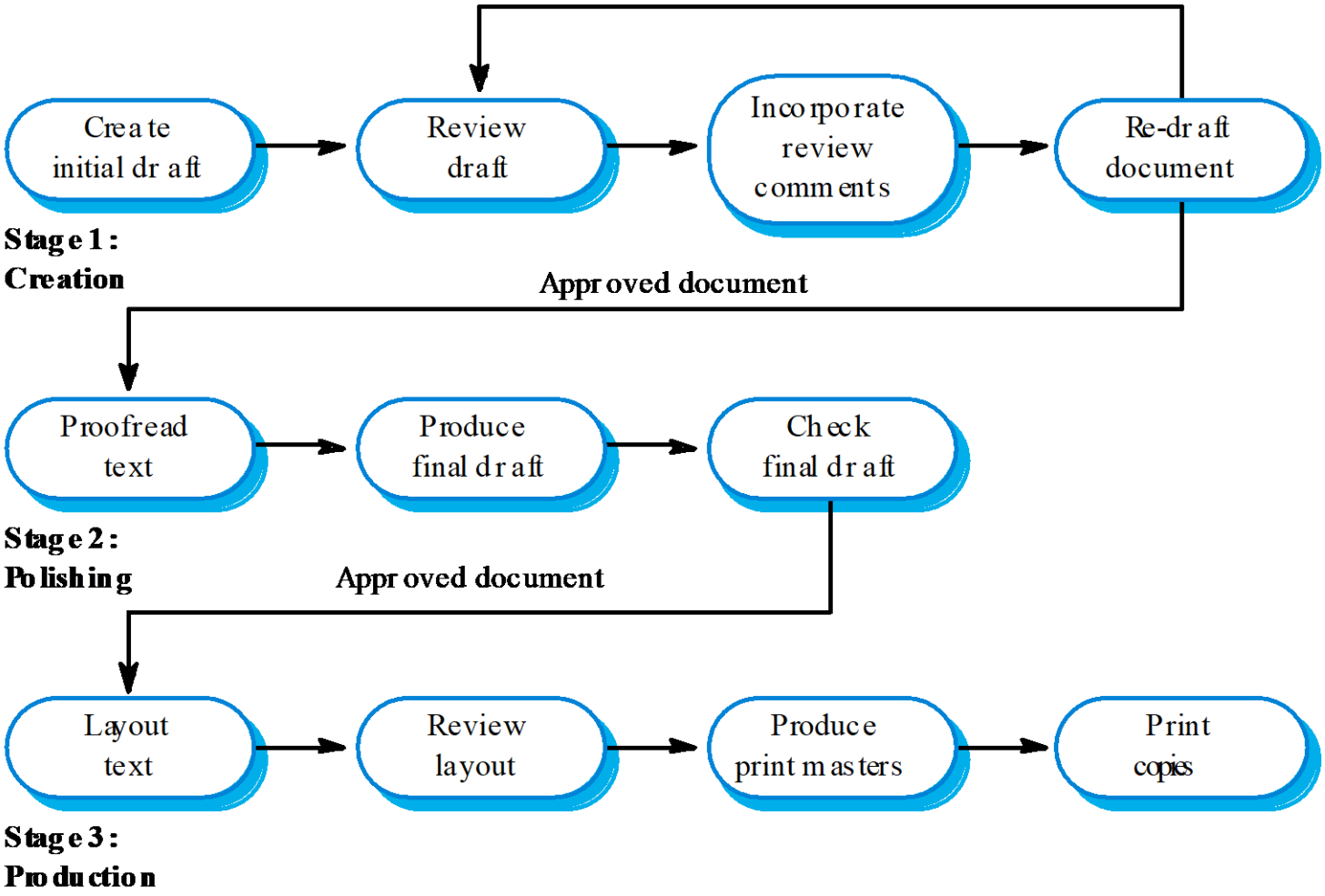
Refer: Ian Sommerville, Software Engineering 8th edition.

Document

- Particularly important - documents are the tangible manifestation of the software.
- Documentation process standards
  - Concerned with how documents should be developed, validated and maintained.
- Document standards
  - Concerned with document contents, structure, and appearance.
- Document interchange standards
  - Concerned with the compatibility of electronic documents.

Refer: Ian Sommerville, Software Engineering 8th edition.

Documentation process



Refer: Ian Sommerville, Software Engineering 8th edition.

Document standards

- Document identification standards
  - How documents are uniquely identified.
- Document structure standards
  - Standard structure for project documents.
- Document presentation standards
  - Define fonts and styles, use of logos, etc.
- Document update standards
  - Define how changes from previous versions are reflected in a document.

Refer: Ian Sommerville, Software Engineering 8th edition.

Document interchange standards

- Interchange standards allow electronic documents to be exchanged, mailed, etc.
- Documents are produced using different systems and on different computers. Even when standard tools are used, standards are needed to define conventions for their use e.g. use of style sheets and macros.
- Need for archiving. The lifetime of word processing systems may be much less than the lifetime of the software being documented. An archiving standard may be defined to ensure that the document can be accessed in future.

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality planning

- A quality plan sets out the desired product qualities and how these are assessed and defines the most significant quality attributes.
- The quality plan should define the quality assessment process.
- It should set out which organisational standards should be applied and, where necessary, define new standards to be used.

Refer: Ian Sommerville, Software Engineering 8th edition.

Software quality attributes

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Refer: Ian Sommerville, Software Engineering 8th edition.

Software quality attributes

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality plans

- Quality plan structure
  - Product introduction;
  - Product plans;
  - Process descriptions;
  - Quality goals;
  - Risks and risk management.
- Quality plans should be short, succinct documents
  - If they are too long, no-one will read them.

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality control

- This involves checking the software development process to ensure that procedures and standards are being followed.
- There are two approaches to quality control
  - Quality reviews;
  - Automated software assessment and software measurement.

Quality reviews

- This is the principal method of validating the quality of a process or of a product.
- A group examines part or all of a process or system and its documentation to find potential problems.
- There are different types of review with different objectives
  - Inspections for defect removal (product);
  - Reviews for progress assessment (product and process);
  - Quality reviews (product and standards).

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality reviews

- A group of people carefully examine part or all of a software system and its associated documentation.
- Code, designs, specifications, test plans, standards, etc. can all be reviewed.
- Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.

Refer: Ian Sommerville, Software Engineering 8th edition.

Quality reviews

- The objective is the discovery of system defects and inconsistencies.
- Any documents produced in the process may be reviewed.
- Review teams should be relatively small and reviews should be fairly short.
- Records should always be maintained of quality reviews.

Software measurement and metrics

- Software measurement is concerned with deriving a numeric value for an attribute of a software product or process.
- This allows for objective comparisons between techniques and processes.

Refer: Ian Sommerville, Software Engineering 8th edition.

Overview of Software Engineering

- What is software engineering?
- Software process models
- Software project management
  - Managing people
  - Quality management
  - Software configuration management
  - …

Topics

- Configuration management (CM) planning
- Change management
- Version and release management
- System building
- CASE tools for configuration management

Refer: Ian Sommerville, Software Engineering 8th edition.

Configuration management

- New versions of software systems are created as they change:
  - For different machines/OS;
  - Offering different functionality;
  - Tailored for particular user requirements.
- Configuration management is concerned with managing evolving software systems:
  - System change is a team activity;
  - CM aims to control the costs and effort involved in making changes to a system.

Refer: Ian Sommerville, Software Engineering 8th edition.

Configuration management

- Involves the development and application of procedures and standards to manage an evolving software product.
- CM may be seen as part of a more general quality management process.
- When released to CM, software systems are sometimes called *baselines* as they are a starting point for further development.

Configuration management planning

- All products of the software process may have to be managed:
  - Specifications;
  - Designs;
  - Programs;
  - Test data;
  - User manuals.
- Thousands of separate documents may be generated for a large, complex software system.

Refer: Ian Sommerville, Software Engineering 8th edition.

The CM plan

- Defines the types of documents to be managed and a document naming scheme.
- Defines who takes responsibility for the CM procedures and creation of baselines.
- Defines policies for change control and version management.
- Defines the CM records which must be maintained.

Refer: Ian Sommerville, Software Engineering 8th edition.

The CM plan

- Describes the tools which should be used to assist the CM process and any limitations on their use.
- Defines the process of tool use.
- Defines the CM database used to record configuration information.
- May include information such as the CM of external software, process auditing, etc.

Refer: Ian Sommerville, Software Engineering 8th edition.

Change management

- Software systems are subject to continual change requests:
  - From users;
  - From developers;
  - From market forces.
- Change management is concerned with keeping track of these changes and ensuring that they are implemented in the most cost-effective way.

Refer: Ian Sommerville, Software Engineering 8th edition.

The change management process

```
Request change by completing a change request form
Analyze change request
if change is valid then
    Assess how change might be implemented
    Assess change cost
    Record change request in database
    Submit request to change control board
    if change is accepted then
        repeat
            make changes to software
            record changes and link to associated change request
            submit changed software for quality approval
        until software quality is adequate
        create new system version
    else
        reject change request
else
    reject change request
```

Refer: Ian Sommerville, Software Engineering 8th edition.

Version and release management

- Invent an identification scheme for system versions.
- Plan when a new system version is to be produced.
- Ensure that version management procedures and tools are properly applied.
- Plan and distribute new system releases.

Versions/variants/releases

- Version An instance of a system which is functionally distinct in some way from other system instances.
- Variant An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.
- Release An instance of a system which is distributed to users outside of the development team.

Refer: Ian Sommerville, Software Engineering 8th edition.

Version identification

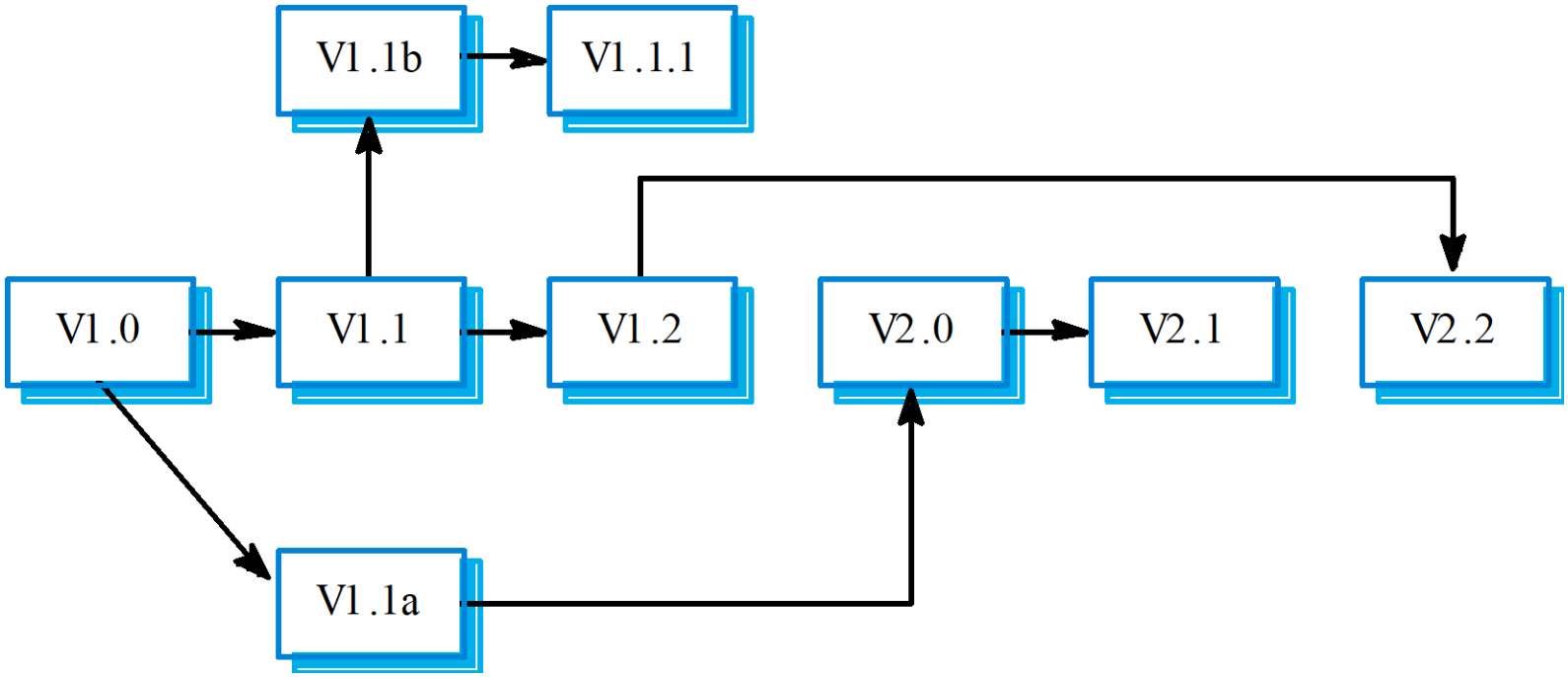
- Procedures for version identification should define an unambiguous way of identifying component versions.
- There are three basic techniques for component identification
  - Version numbering;
  - Attribute-based identification;
  - Change-oriented identification.

Refer: Ian Sommerville, Software Engineering 8th edition.

Version numbering

- Simple naming scheme uses a linear derivation
  - V1, V1.1, V1.2, V2.1, V2.2 etc.
- The actual derivation structure is a tree or a network rather than a sequence.
- Names are not meaningful.
- A hierarchical naming scheme leads to fewer errors in version identification.

Version derivation structure



Refer: Ian Sommerville, Software Engineering 8th edition.

Attribute-based identification

- Attributes can be associated with a version with the combination of attributes identifying that version
  - Examples of attributes are Date, Creator, Programming Language, Customer, Status etc.
- This is more flexible than an explicit naming scheme for version retrieval; However, it can cause problems with uniqueness - the set of attributes have to be chosen so that all versions can be uniquely identified.
- In practice, a version also needs an associated name for easy reference.

Refer: Ian Sommerville, Software Engineering 8th edition.

Change-oriented identification

- Integrates versions and the changes made to create these versions.
- Used for systems rather than components.
- Each proposed change has a change set that describes changes made to implement that change.
- Change sets are applied in sequence so that, in principle, a version of the system that incorporates an arbitrary set of changes may be created.

Refer: Ian Sommerville, Software Engineering 8th edition.

Release management

- Releases must incorporate changes forced on the system by errors discovered by users and by hardware changes.
- They must also incorporate new system functionality.
- Release planning is concerned with when to issue a system version as a release.

System releases

- Not just a set of executable programs.
- May also include:
  - Configuration files defining how the release is configured for a particular installation;
  - Data files needed for system operation;
  - An installation program or shell script to install the system on target hardware;
  - Electronic and paper documentation;
  - Packaging and associated publicity.

Q&A