7. Software Configuration Mgmt.

1. Key objective of the soft. process is to have change activity converge until the final product is stable enough to ship, the management of all changes is important.

2. Focus on code control.

3. SCM is a set of activities designed to manage change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made.

Need for Software Configuration Management: 1. Poor config. Mgmt. 2. Config. management helps to reduce these problems by coordinating the work products of the many different people who work on a common project. Problems • Simultaneous update – When two or more programmers work separately on the same program • Shared code – When a bug is fixed in code shared by several programmers • Common code – When common program functions are modified • Versions – Evolutionary release.

Basic Configuration Management Functions: Once an initial product level has stabilized, a first baseline is established. • With each successive set of enhancement, a new baseline is established in step with development. • Each baseline is retained in a permanent database, together with all the changes that produced it. The baseline is thus the official repository for the product, and it contains the most current. version. • Only tested code and approved changes are put in the baseline, which is fully protected. – It is the official source that all the programmers use to ensure their work is consistent with that of everyone else. **KEY TASKS:**

• **Configuration control:** The task of configuration control revolves around one official copy of the code. The simplest way to protect every system revision is to keep a separate official copy of each revision level. •While this can take a lot of storage, the most serious issue concerns code divergence. **Change Mgmt.: •Revisions• Versions • Deltas • Conditional code.** The **BASELINE** is the foundation for configuration management. • It provides the official standard on which subsequent work is based and to which only authorized changes are made. • After an initial baseline is established and frozen, every subsequent change is recorded as a delta until the next baseline is set. It is desirable to establish a baseline at an early point in every project. • Establishing a baseline too early, however, will impose unnecessary procedures and slow the programmers' work.

• If the programmers can work on individual modules with little interaction, a code baseline is not needed. As soon as integration begins, however, formal control is essential. The system library stores the development work products, including the source and object code for every baseline and change, the test cases, and the development tools. • It provides locks to prevent unauthorized changes, and it has the capability to build the various system configurations, test drivers, and test scenarios or buckets required by development.

QUALITY is a characteristic or attribute of something. Measurable, Things we can compare to known standards. ***Quality of design, Quality of conformance.*** The **GOAL of QA** is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals.

Quality Management encompasses…SQA Process, QC tasks, SE practices, Control of all SW work products, compliance, measurement, and reporting**. This SQA expertise** includes knowledge of statistical methods, quality control principles, the software process, and an ability to deal effectively with people in contentious situations. **QM GOALS:** Testing, each software task is satisfactorily completed, Deviations are exposed, auditable, quality work control, SQA plan and SD Plan. **GOALS of SQA: – To improve software quality** by appropriately. *monitoring both the software and the development process* that produces it – To ensure full *compliance with the established. standards and procedures* for the software and the software process – To ensure that *any inadequacies in the product, the process, or the standards are brought to mgmt's attention* so these inadequacies can be fixed. **SQA Functions:** – Quality Assurance (QA) practices – Software project planning evaluation – Requirements evaluation – Evaluation of the design process – Evaluation of coding practices – Evaluating the SW integration and test process – In-process evaluation of the mgmt. and project control process – Tailoring of Quality Assurance (QA) procedure. **SOFTWARE STANDARD** • A standard is a rule or basis for comparison that is used to assess the size, content, value, or quality of an object or activity. **Two Kinds** - Describes the nature of the object, Defines the way the work. **POLICY** A governing principle, typically used as the basis for regulations, procedures, or standards, and generally stated by the highest authority in the organization. **REGULATION**: A rule, law, instruction, typically established by some legislative or regulatory *body.*

SPECIFICATION: The precise and verifiable desc. of the charac. of a product. A process specification defines a method, procedure, or process to be used in performing a task and produced by technical experts. **GUIDLINES**: A suggested practice, method, or procedures. **Procedure**defined way to do work. **STANDARD:** A rule or basis for comparison that is used to assess size, content, or value, typically established by common practice or by a designated standards body. **CONVENTION**: A general agreement on practices, methods, or procedures. **METHOD**: A regular, orderly procedure or process for performing a task. **PRACTISE:** A usual procedure or process, typically a matter of habit or tacit agreement. **PROCESS:** A defined way to perform some. **Standards Development Process:** - Establish a standards strategy, - maintain this strategy, - individuals or small working groups to develop the top-priority standards, - define the areas of applicability, specify the introduction strategy, and propose an enforcement plan,- draft standards should be widely distributed and reviewed,- implemented in a limited test environment,-again review,- Implement and enforce,- evaluate. **MAINTAINING STANDARDS: -**kept current, modiefied and adjusted, if not maintained…less pertinent and less practical, if not corrected, responsibility one or group. **ENFORCING STANDARDS: -**basic rule of SQA org. , mix of reviews and tests, Exhaustive reviews are most appropriate when automated tools can be used to support the monitoring process or when the standard is so critical that no single deviation is acceptable.**SOFTWARE INSPECTION** provide a powerful way to improve the quality and prod. of the SW process. The soft inspection is a peer review or programmers.

Types of Reviews:

1.Management review
– Ensure progress
– Recommend corrective action
– Ensure proper alloc. of resources

2.Technical review
– Evaluate conformance to specifications and plans
– Ensure change integrity

3.Software inspection
– Detect and identify defects
– Verify resolution

4.Walkthrough
– Detect defects
– Examine alternatives
– Forum for learning

Inspection Objectives: find errors at the earliest possible point, appropriate parties technically agree on the work, work meets predefined criteria, formally complete a technical task, provide data on the product and the inspection process. **PRINCIPLES**:
– The inspection is a formal, structured process with a system of checklist and defined roles for the participants.
– Generic checklists and standards are developed for each inspection type and, where appropriate, they are tailored to specific project needs.
– The reviewers are prepared in advance and have identified their concerns and questions before the inspection starts. **Inspection Participants:** • The moderator (or inspection leader) • The producers – The person(s) responsible for doing the work being inspected
• The reviewers (or inspectors)
• The recorder (or scribe).

Software Testing: • Unit or module tests• Integration tests • External function tests • Regression tests • System tests • Acceptance tests • Installation tests. **AXIOMS:** good test case is one that has a high probability of detecting a prev undiscovered defect. Knowing when to stop, own code own test impossible, avoid non reproduc, necessary part of every test case is a desc of expected output.

Axioms: Write test cases for invalid as well as valid input conditions, inspect the result of each test, the number of detected defects in a piece of software increases, the probability of the existence of more undetected defects also
Increases, best programmers to testing. design of a system should be such that each module is integrated into the system only once, never alter the program to make testing easier. **TYPES:** UNIT + WHITE BOX = PATH TESTS**,** Integration Testing approaches—top down, bottom up. Black box functional. In **bottom-up testing** the modules are individually
tested, using specially developed drivers that provide the needed system functions. **Top-down testing** is essentially a prototyping.
philosophy. • The initial tests establish a basic system skeleton from the top and each new module adds capability. • **Functional (or black box) tests** are designed to
exercise the program to its external specifications. • The testers are typically not biased by knowledge
of the program's design and thus will likely provide tests that resemble the user's environment.
– The two most typical problems with black box testing are the need for explicitly stated requirements and the ability of such tests to cover only a small portion
of the possible test conditions. **progressive phase** introduces and tests new functions and regressive phase concerns the effects of
the newly introduced changes on all the previously integrated code.
The purpose of **SYSTEMS TESTS** is to find those case in which the system does not work as intended.
AGILE: Key Practices of Agile Process • Iterative development
• Risk-driven and client-driven Iterative Planning • Time-boxing • Evolutionary development • Adaptive planning• Incremental delivery • Evolutionary delivery.

The Agile Manifesto
• Individuals and interactions over processes and tools • Working software over comprehensive
Documentation • Customer collaboration over contract
Negotiation • Responding to change over following a plan.
Principles behind the Agile Manifesto: Satisfy Customer, **welcome** changing req., deliver working software frequently, Businesspeople and developers must work together daily, Build projects around motivated individuals, face-to-face conversation. Working software is the primary measure of progress, promote sustainable development, Continuous attention to technical excellence, simplicity, self org. teams, at regular intervals, the team reflects. **Agile Development Methods:** • Scrum • XP • Evo
• Crystal • Agile Modeling
• (Agile) Unified Process (Agile UP)
• Dynamic Solutions Delivery Model (DSDM) • Feature-Driven Development (FDD) • Lean Development.
SCRUM: Scrum's distinctive emphasis among the
methods is its strong promotion of
– self-directed teams
– daily team measurement
– avoidance of prescriptive process.
-- client-driven adaptive planning, demos. Scrum: 30 days (4w) XP: 1 – 3 weeks UP: 2 – 6 weeks Evo: 1 – 2 weeks.
INTRO: Scrum is an IID method that emphasizes a set of
project management values and practices, rather than those in requirements, implementation, and so on. As such, it is easily combined with or complementary to other methods. A key Scrum theme is its emphasis on empirical rather than defined process. **Lifecycle: Pregame (Planning-Staging)-> DEVELOPMENT->RELEASE.**

Work product - Requirements
Roles: Product owner, scrum team, scrum master, others.
PRACTICES: Pre-game
planning and staging**,** **sprint** planning, sprint, self-directed and organizing team, scrum meeting, don't add to iteration, scrum master firewall, decision in 1 hr., blocks gone in 1 day, teams of 7, common room, daily build, sprint review. **SCRUM MEETING:** The Scrum Meeting provides a daily forum to update tasks,
and surface and remove impediments. • The meeting is ideally held in a stand-up circle to encourage brevity. • On average, 15 or 20 minutes for 7–10 people.
– Longer meetings are common near the start of an iteration.
• Non-team members (chickens) are outside the circle. • It is held next to a whiteboard at which all the tasks and blocks are written when reported. In addition to the work products (illustrated before), Scrum allows any other work products of value to the project.
VALUES: Openness (product backlog, daily scrums, work trend and velocity), Respect(team responsibility, strength and weakness). Process Mixtures
Scrum + XP
The Scrum practice of a demo to external stakeholders at the end of each iteration enhances XP's feedback and communication.
goals. – The Scrum Backlog and progress tracking approaches are minor variations of XP practices,
and so simple that they are well within the XP spirit of "do the simplest thing that could possibly.
work." Scrum's 30-day timeboxed iteration length is not completely consistent with XP, which prefers shorter – even one-week iterations.

Extreme Programming (XP) is a well-known agile method.
• **XP emphasizes:**
– Collaboration
– Quick and early software creation
– Skillful development practices
• XP is founded **on four values:**
– Communication, – Simplicity
– Feedback, – Courage.
12 core practices:
1. Planning game
2. Small, frequent releases
3. System metaphors
4. Simple design
5. Testing
6. Frequent factoring
7. Pair programming
8. Team code ownership
9. Continuous integration
10. Sustainable pace
11. Whole team together
12. Coding standards

XP is low on the ceremony. scale; it has only a small set of a predefined, informal work products, such as paper index cards for summarizing feature requests, called story cards.
For average projects, the recommended length of a
timeboxed iteration is between one and three weeks – somewhat shorter than for UP or Scrum.
• XP, created by Kent Beck [Beck 00], is an IID method that stresses customer satisfaction through:
– rapid creation of high-value software – skillful and sustainable software development techniques
– flexible response to change
• It is aimed at relatively small team projects, usually with delivery dates under one year. • Iterations are short – usually one to three weeks.
As the word 'programming' suggests, it provides explicit methods for programmers, so they can more confidently respond to changing requirements, even late in the project, and still produce quality code.

These include test-driven development, refactoring, pair programming, and continuous integration, among others.
XP is very communication and team oriented.
XP Lifecycle
• Exploration Phase
• Planning Phase
• Iterations to Release Phase
• Productionizing Phase
• Maintenance Phase
Project velocity: a measure showing how much move fast, and to which direction the project is going; not simply measure how much code is written, but measure the value from the customer's
view; how much user stories are completed during an iteration and estimation; the ratio between estimated time and real calendar time. **Metaphor**: a conceptual framework; a logical architecture
of the system; a shared story or description of how the
system works.; a description of how you intend to build
your system. The metaphor is defined during an
architectural spike early in the project (during the 1st
iteration or pre-iteration).
Architectural spike: intends to identify areas of high risk,
to get started with estimating them correctly.