CASE WESTERN RESERVE UNIVERSITY

Agile Practices and Software Inspections

CSDS 393/493: Software Engineering

Spring 2025

Agenda

- Agile practices
  - Scrum
  - Scaling agile
- Software Inspections

Team Projects

- Quiz 1: February 6
  - On Requirement (see module Week 2)

- Discuss project proposals with TAs
- Project milestones
  - **Demo 1**: Week of Feb 11
    - Setup and installations
      - IDEs/Frameworks such as Android Studio, DB server, etc.)
      - GitHub (initial commits, .gitignore, branching, pull request)
      - Trello (creating and assigning tasks)
    - Implementation of at least one simple feature (e.g., user login)
    - Teamwork

Scrum

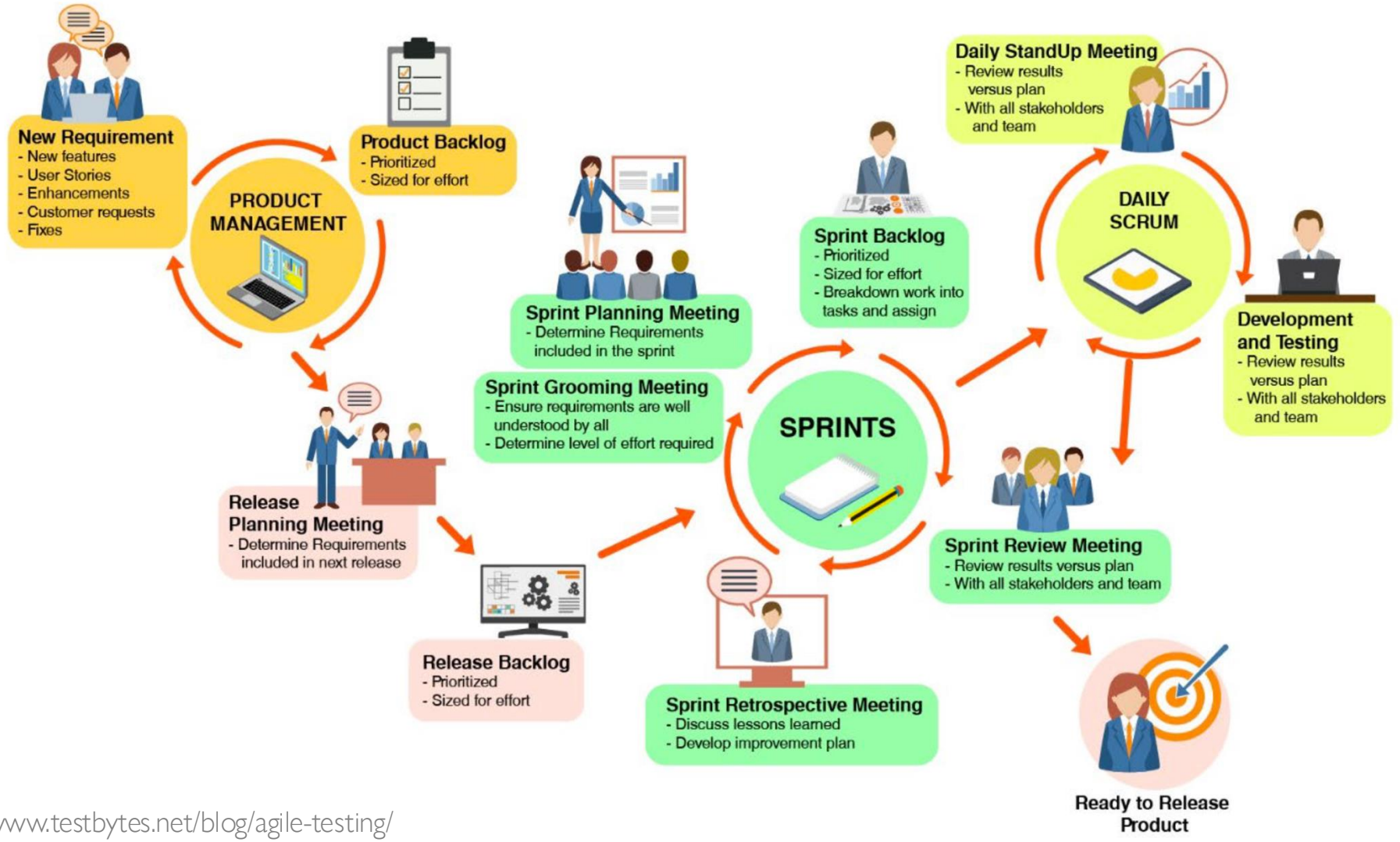
- Based on iterative and incremental development
- Somewhat more conventional than XP
- Projects split into iterations called sprints

- A sprint typically takes 1-4 weeks

Scrum Roles

- **Product owner**
  - Maximizes product value
  - Manages product backlog
  - Decides when to ship
- **Development team**
  - Creates and delivers product increment
  - "Self-organizing and cross-functional"
- **Scrum Master**
  - Ensures Scrum process is followed
  - Coaches team
  - Removes impediments
  - May take on project management tasks

Scrum Lifecycle



https://www.testbytes.net/blog/agile-testing/

Scrum Release Planning

- Comprehensive product backlog
- Definition of the delivery date and functionality
- Mapping of backlog items to product packets
- Appropriate risk controls
- Selection and validation of development tools and infrastructure
- Estimation of release cost, including development, material, marketing, training, and rollout
- Verification of management approval and funding

The Sprint

- Limited to one month
- Consists of Sprint Planning, Daily Scrums, Development, Sprint Review, and Sprint Retrospective
- Has definition of what is to be built, a design, and a flexible implementation plan

Sprint Planning

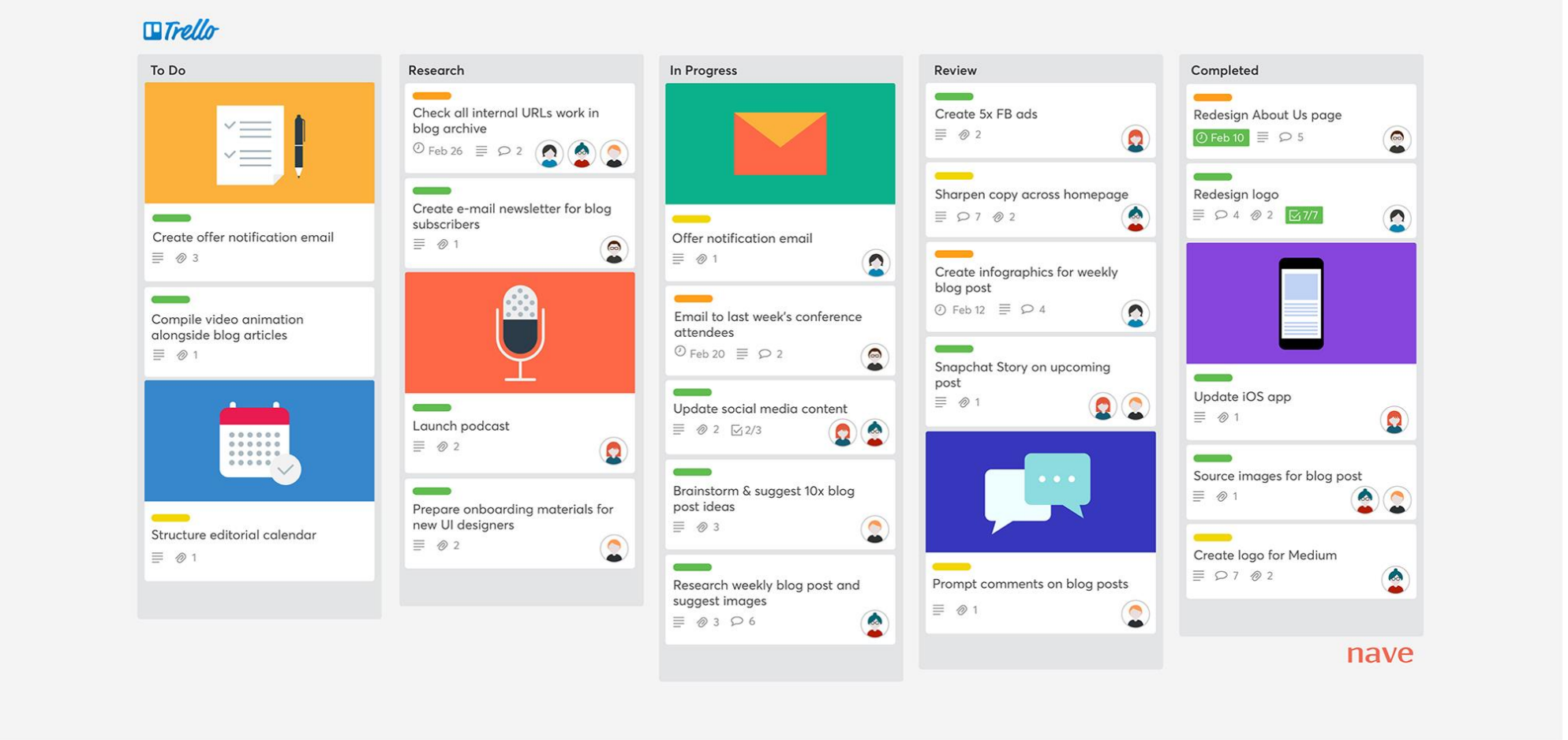
- Done by entire Scrum team
- At most 8 hours for one sprint
- **Input**: Product Backlog
- Answers two questions:
  - What can be delivered in this increment?
  - How will the work be achieved?
    - Involves design, work scheduling
- Produces Sprint Goal and Sprint Backlog

Hobby Site Product Backlog Items chosen for this Sprint

Item ID	Bug ID	Summary	Rank	Category	Accomplished?
1		Search other hobby sites: bring back results in a list	1	Search	X
2		Search hobby content & bring back results in a list	3	Search	X
3		Show direct display of hobby content in results	14	Search	
4		Use variety of methods for relevancy	4	Search	X
5		Keep track of queries for buckets	37	Search	X
6		Include user-generated content (message boards) in index	19	Search	X
7		Include Ads on search	16	Search	X
8		Include sites to index	2	Search	X
9		Search infrastructure/ops/machines		Search	X
10		"Did you mean?" support for misspellings, etc.	18	Search	X
11		Media for Message boards	23	Boards	
12		View/post to boards	6	Boards	X
13		View list of forums		Boards	X
14		See a list of threads on this board	7	Boards	X
15		Show different views of threads	40	Boards	X
16		Show messages/all messages in thread		Boards	X
17		Sort list of threads	43	Boards	X
18		Sort postings by rating	42	Boards	X
19		Signed in users can rate another user's posted message	41	Boards	X
20		Signed in users can start a thread/message	8	Boards	X
21		Signed in users can preview messages before submitting	22	Boards	X
22		Find all posts by another user	45	Boards	
23		Show user info for each message	24	Boards	

https://www.scrumalliance.org/community/articles/2005/september/how-scrum-works

“Kanban”-Style Backlog



https://getnave.com/blog/trello-kanban-boards/

Design in Scrum

- Software design is incorporated into the regular sprint cycles:
  - High-level architecture and key design decisions are made during initial sprint planning
  - Detailed design work is done within sprints as needed
  - Design evolves and is refined sprint-by-sprint

Daily Scrum

- 15-minute event for Dev Team
  - Synchronize activities
  - Create plan for next 24 hours
- Dev Team members explain:
  1. What I did yesterday
  2. What I will do today
  3. Impediments encountered
- Dev Team members may have detailed discussions afterward

Sprint Review

- **4-hour meeting at end of sprint**
- Reviews latest increment and adapts Product Backlog
- Attended by Scrum Team and key stakeholders
- Dev Team demonstrates work done and answers questions
- Review of timeline, budget, marketplace for next release

Sprint Retrospective

- Opportunity for Scrum Team to inspect itself and plan process improvements
- Occurs after Sprint Review and prior to next Sprint Planning
- Considers people, relationships, process, and tools

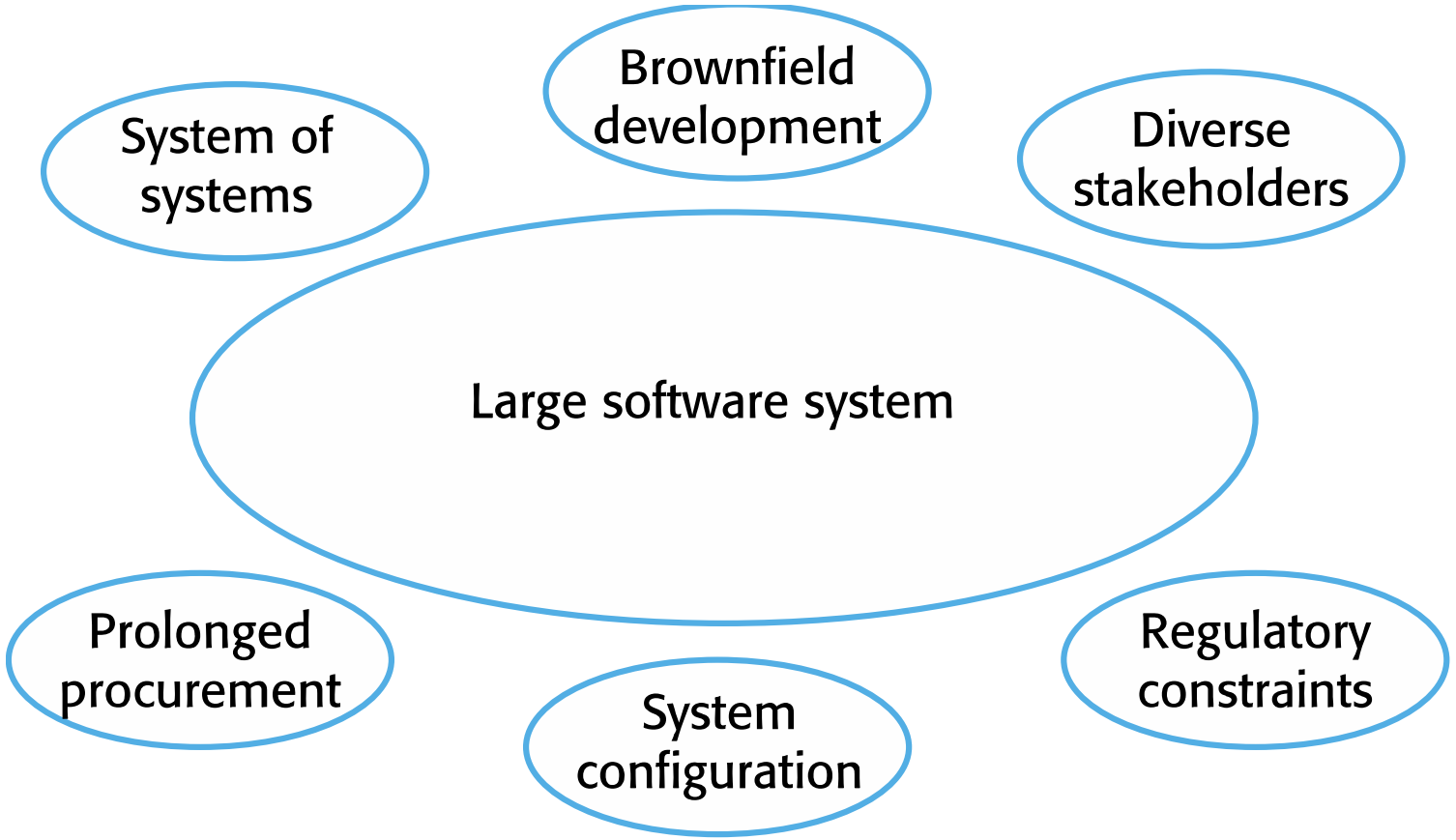
Sprint Retrospective

- Key discussion points
  - What went well in the Sprint
  - What problems were encountered and how they were (or weren't) solved
  - What could be improved
  - What the team will commit to improve in the next Sprint

Scrum Closure Phase

- Occurs when management feels time, competition, requirements, cost, and quality call for a new release
- Prepares the developed product for general release
- Tasks include: integration, system test, user documentation, preparation of training and marketing material

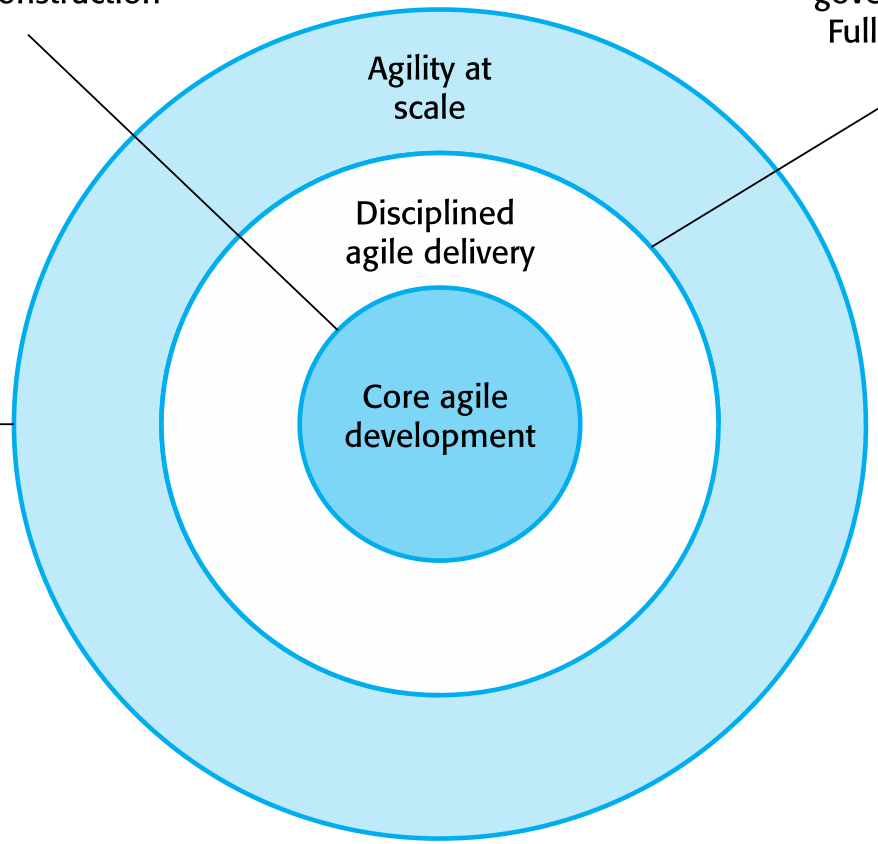
Factors in Large Systems



IBM's Agility at Scale Model

Core agile development
Value-driven life-cycle
Self-organizing teams
Focus on construction

Disciplined agile delivery
Risk+value driven lifecycle
Self-organizing with appropriate governance framework
Full delivery life-cycle



Agility at scale
Disciplined agile delivery where scaling factors apply:
Large team size
Geographic distribution
Regulatory compliance
Domain complexity
Organization distribution
Technical complexity
Organizational complexity
Enterprise discipline

Multi-Team Scrum

- Role replication
- Product architects
- Release alignment
- Scrum of Scrums

Further Reading

- Abrahamsson, Pekka, et al. "New directions on agile methods: a comparative analysis." 25th International Conference on Software Engineering, 2003. Proceedings.. IEEE, 2003.
- Ambler, S. W. 2010. "Scaling Agile: A Executive Guide"

Software Inspections

Inspections

- Involve developers carefully reviewing documents or code to identify:
  - **Errors** of omission or commission
  - **Ambiguity** and lack of clarity
  - **Violations** of standards
  - Other issues

- They typically involve a team of developers
- Studies have shown inspections are very effective
- They complement other forms of validation like testing

Summary of Inspection Results

Reference	Environment	Result
Fagan [40][41]	Aetna Life Casuality	38 defects from 46 detected
	IBM Respond, United Kingdom	93% of all defects were detected by inspections
	Standard Bank of South Africa	Over 50% of all defects detected by inspection
Weller [132]	Bull HN Information Systems	70% of all defect detected by inspection
Grady and van Slack [51]	Hewlett-Packard	60%-70% of all defects detected by inspection
Shirey [122]		60%-70% of all defects detected by inspection
Barnard and Price [4]	AT&T Bell Laboratories	30%-75% of all defects detected by inspection
McGibbon [92]	Cardiac Pacemakers Inc.	70% to 90% of all defects detected by inspection
Collofello and Woodfield [26]	Large real time software project	Defect detection effectiveness is 54% for design inspection, 64% for code inspec-tion, and 38% for testing
Kitchenham et al. [71]	ICL	57.7% of all defects found by code in-spection
Franz and Shih [43]	Hewlett Packard	19% of all defects found by inspection
A. Gately [46]	Raytheon Systems Company	The average number of defects found by inspection is 18.2.
Conradi et al. [27]	Ericsson	The average number of defects found by inspection is 3.41.

Laitenberger, Oliver. "A survey of software inspection technologies." *Handbook of Software Engineering and Knowledge Engineering*. 2002.

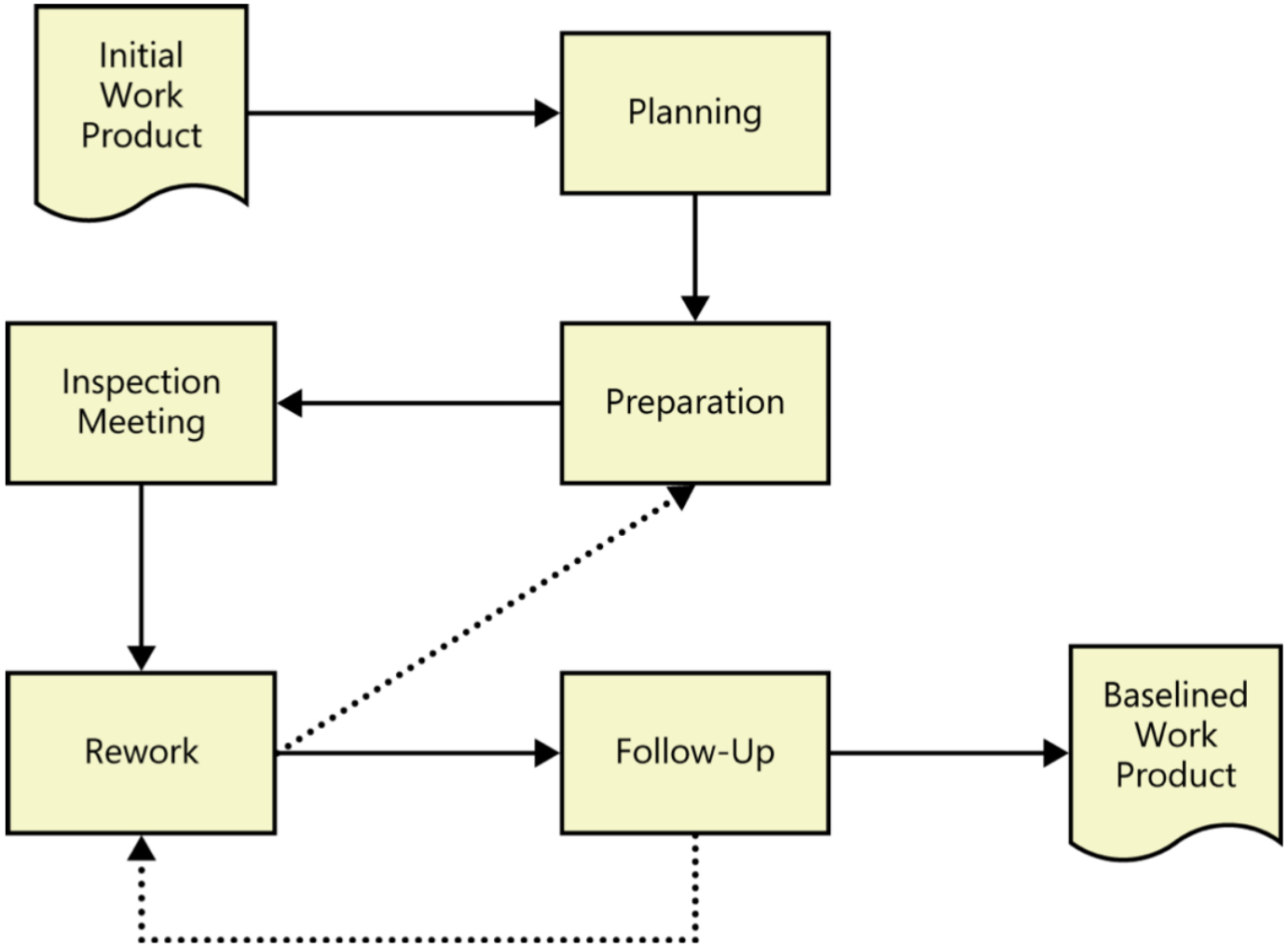
Traditional Inspection Roles

- Author – developer responsible for work product
- Inspectors – inspect work product
- Scribe or recorder – records issues
- Moderator – directs preparation and inspection meeting; reports results to manager
- Manager – schedules inspection, assigns moderator and team, manages follow-up

Inspection Meeting Rules

- Manager is not present
- Inspectors take turns presenting issues
- Inspectors are tactful
- Producer does not defend work
- Problems are not solved at the meeting

Inspection Process



Outline of Inspection Meeting

- Planning
  - Team is assembled
  - Materials are distributed
- Overview (Optional)
  - Author provides background information on artifact
- Preparation
  - Team members individually review material and note potential defects
- Inspection Meeting
  - Reader walks through the artifact
  - Inspectors raise issues
  - Recorder logs all defects identified
  - Moderator keeps meeting on track
- Rework
  - Author addresses issues
- Follow-up
  - Moderator verifies changes
  - Determines re-inspection

Checklists

- Useful for reminding inspectors of important issues
- May inhibit them from discovering other issues, however
- Perhaps best consulted after first reading of work product

Requirements Review Checklist

Completeness

- ❑ Do the requirements address all known customer or system needs?
- ❑ Is any needed information missing? If so, is it identified as TBD?
- ❑ Have algorithms intrinsic to the functional requirements been defined?
- ❑ Are all external hardware, software, and communication interfaces defined?
- ❑ Is the expected behavior documented for all anticipated error conditions?
- ❑ Do the requirements provide an adequate basis for design and test?
- ❑ Is the implementation priority of each requirement included?
- ❑ Is each requirement in scope for the project, release, or iteration?

Correctness

- ❑ Do any requirements conflict with or duplicate other requirements?
- ❑ Is each requirement written in clear, concise, unambiguous, grammatically correct language?
- ❑ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ❑ Are any specified error messages clear and meaningful?
- ❑ Are all requirements actually requirements, not solutions or constraints?
- ❑ Are the requirements technically feasible and implementable within known constraints?

Quality Attributes

- ❑ Are all usability, performance, security, and safety objectives properly specified?
- ❑ Are other quality attributes documented and quantified, with the acceptable trade-offs specified?
- ❑ Are the time-critical functions identified and timing criteria specified for them?
- ❑ Have internationalization and localization issues been adequately addressed?
- ❑ Are all of the quality requirements measurable?

Organization and Traceability

- ❑ Are the requirements organized in a logical and accessible way?
- ❑ Are all cross-references to other requirements and documents correct?
- ❑ Are all requirements written at a consistent and appropriate level of detail?
- ❑ Is each requirement uniquely and correctly labeled?
- ❑ Is each functional requirement traced back to its origin (e.g., system requirement, business rule)?

Other Issues

- ❑ Are any use cases or process flows missing?
- ❑ Are any alternative flows, exceptions, or other information missing from use cases?
- ❑ Are all of the business rules identified?
- ❑ Are there any missing visual models that would provide clarity or completeness?
- ❑ Are all necessary report specifications present and complete?

Generic Checklist for Code Reviews

Structure

- ❑ Does the code completely and correctly implement the design?
- ❑ Does the code conform to any pertinent coding standards?
- ❑ Is the code well-structured, consistent in style, and consistently formatted?
- ❑ Are there any uncalled or unneeded procedures or any unreachable code?
- ❑ Are there any leftover stubs or test routines in the code?
- ❑ Can any code be replaced by calls to external reusable components or library functions?
- ❑ Are there any blocks of repeated code that could be condensed into a single procedure?
- ❑ Is storage use efficient?
- ❑ Are symbolics used rather than "magic number" constants or string constants?
- ❑ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

- ❑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
- ❑ Are all comments consistent with the code?

Variables

- ❑ Are all variables properly defined with meaningful, consistent, and clear names?
- ❑ Do all assigned variables have proper type consistency or casting?
- ❑ Are there any redundant or unused variables?

Arithmetic Operations

- ❑ Does the code avoid comparing floating-point numbers for equality?
- ❑ Does the code systematically prevent rounding errors?
- ❑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- ❑ Are divisors tested for zero or noise?

Loops and Branches

- ❑ Are all loops, branches, and logic constructs complete, correct, and properly nested?
- ❑ Are the most common cases tested first in IF- -ELSEIF chains?
- ❑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- ❑ Does every case statement have a default?
- ❑ Are loop termination conditions obvious and invariably achievable?
- ❑ Are indexes or subscripts properly initialized, just prior to the loop?
- ❑ Can any statements that are enclosed within loops be placed outside the loops?
- ❑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

- ❑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- ❑ Are imported data and input arguments tested for validity and completeness?
- ❑ Are all output variables assigned?
- ❑ Are the correct data operated on in each statement?
- ❑ Is every memory allocation deallocated?
- ❑ Are timeouts or error traps used for external device accesses?
- ❑ Are files checked for existence before attempting to access them?
- ❑ Are all files and devices are left in the correct state upon program termination?

Example: Inspection Comments

Reviewer's Name	Sophie (senior QA engineer)
Author's Name:	Dean (junior QA engineer)
Title:	Contract Certification – automated test script #TP-491-A
Review Date:	8/12/03
No. of Review hours:	2

Location	Comments
Global	Script does not adequately copy databases in when the data changes
Case 14	The test plan logs in as "Administrator", this script logs in as "Admin"
Case 52, 53	What exactly is printed? It's not clear, you should be looking for specific data.
Case 61	The test plan tests all of the preferences, but the script only tests the first five.
Global	Script does not adequately copy databases in when the data changes

https://www.stellman-greene.com/about/applied-software-project-management/applied-software-project-management-review-practices/

Inspection Problems and Remedies

Work product not available on time	Moderator informs manager, who reschedules
Inspectors spend insufficient time reading document	Record reading time Moderator reports poor preparation to manager
Some inspectors dominate meeting	Inspectors take turns reporting issues
Scribe can't keep up	Moderator pauses
Scribe doesn't accurately record issues	Scribe reads back issues Participants review issue
Producer defends product	Moderator intervenes

Inspection Follow-Up

Task	Responsible
Confirm that the author addressed every item on the Issue Log. Determine whether the author made appropriate decisions as to which defects not to correct.	Verifier
Examine the modified work product to judge whether the rework has been performed correctly. Report any findings to the author.	Verifier
Report the number of major and minor defects found and corrected and the actual rework effort to the moderator.	Author
Check whether the **exit criteria** for the inspection and for the peer review process have been satisfied. If so, the inspection is complete.	Moderator
Check the baselined work product into the project's configuration management.	Author
Deliver Inspection Summary Report and counts of defects found and defects corrected to peer review coordinator.	Author

Modern Code Review

Modern code review practices have evolved to become more lightweight:

- Tool-based process
- Small, incremental changes
- Asynchronous
- Automated checks: Static analysis tools, linters, and automated tests are used to catch basic issues before human review.
- Selective reviewing
- Metrics and goals: Organizations track metrics like review turnaround time and defect detection
- Checklist-driven
- Emphasis on knowledge sharing
- Constructive feedback
- Continuous process

Modern Code Review at Google

- Original impetus for code review at Google:
  - To force developers to write code that other developers could understand

- Additional benefits became clear:
  - Checking consistency of style and design
  - Ensuring adequate tests
  - Improving security through oversight of commits

Current Expectations

- Key themes Google developers expect from code review:
  - Education
  - Maintaining norms
    - e.g., in formatting or API usage
  - Gatekeeping
    - Establishing boundaries around code, design choices, or another artifact
  - Accident prevention
    - e.g., bugs, vulnerabilities
    - Tracking history of code changes

Ownership

- Google codebase is arranged in a tree structure
- Each directory is explicitly owned by a set of developers
- Any developer can propose a change to any part, but it must be reviewed and approved by an owner before it is committed

Code Review Flow at Google

- Creating: Author start modifying, adding, or deleting some code.
- Previewing: Author use Critique tool to view diff of change and view results of automatic code analyzers. When ready, they assign to one or more reviewers.
- Commenting: Reviewers view diff and draft comments.
- Addressing feedback: Author updated change or replies to comments, then updates new snapshot.
- Approving: Once all comments are addressed, reviewers approve change. Developer can then commit.
  - Usually only one reviewer is required.

Suggesting Reviewers

- Critique provides a tool that analyzes the change and suggests possible reviewers.
- It prioritizes reviewers that have recently edited or reviewed the included files.
- New team members are explicitly added as reviewers.

Code Analysis Results

- Critique tool shows code analysis results as comments along with human ones
- To vet changes before they're committed, pre-commit checks are triggered automatically
  - e.g., automated style checking, testing
  - Failure requires explicit override by developer
  - Can be configured by teams

Further Reading

- 10 Tips for Effective Code Review, Atlassian Summit 2016: https://www.youtube.com/watch?v=fatTnX8_ZRk
- Sadowski, Caitlin, et al. "Modern code review: a case study at google." Proceedings of the 40th international conference on software engineering: Software engineering in practice. 2018.