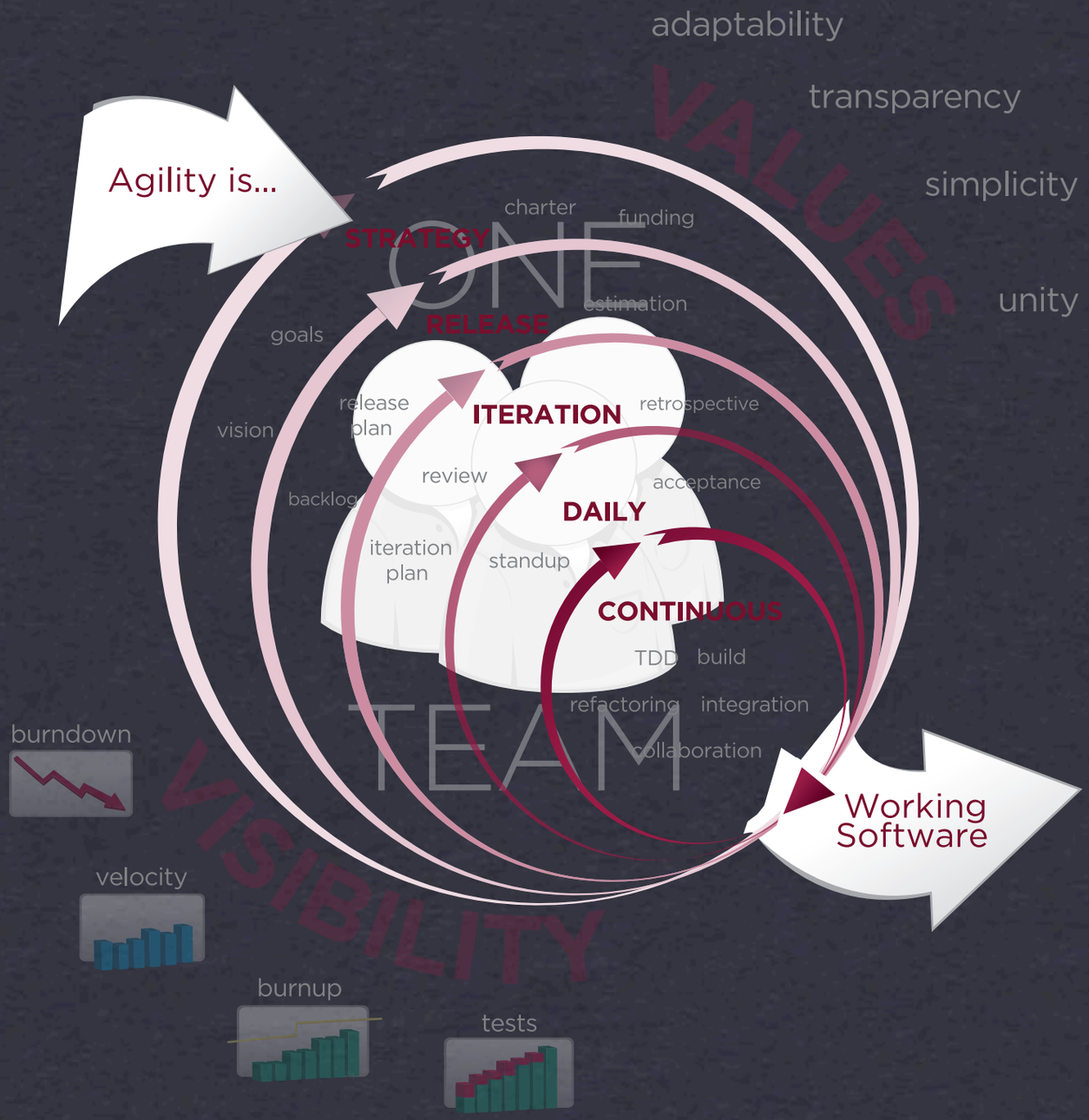


Material and some slide content from:
- Krzysztof Czarnecki
- Ian Sommerville

Process & Version Control

Reid Holmes

LIGHTWEIGHT VS. HEAVYWEIGHT PROCESSES

Heavyweight
e.g., V-Process

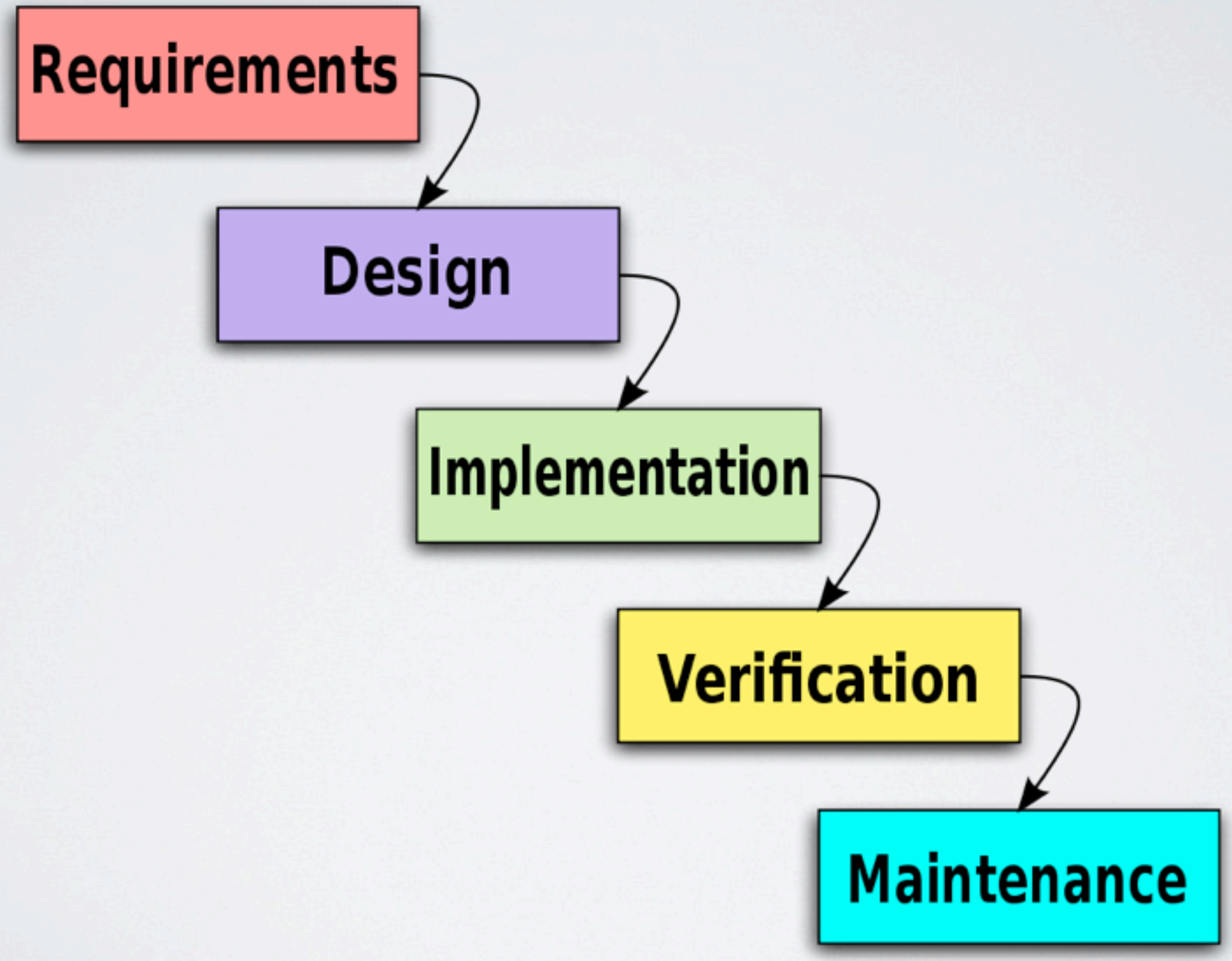
Customizable Framework
e.g., Rational Unified Process (RUP)

Agile (Lightweight)
e.g., eXtreme Programming (XP)

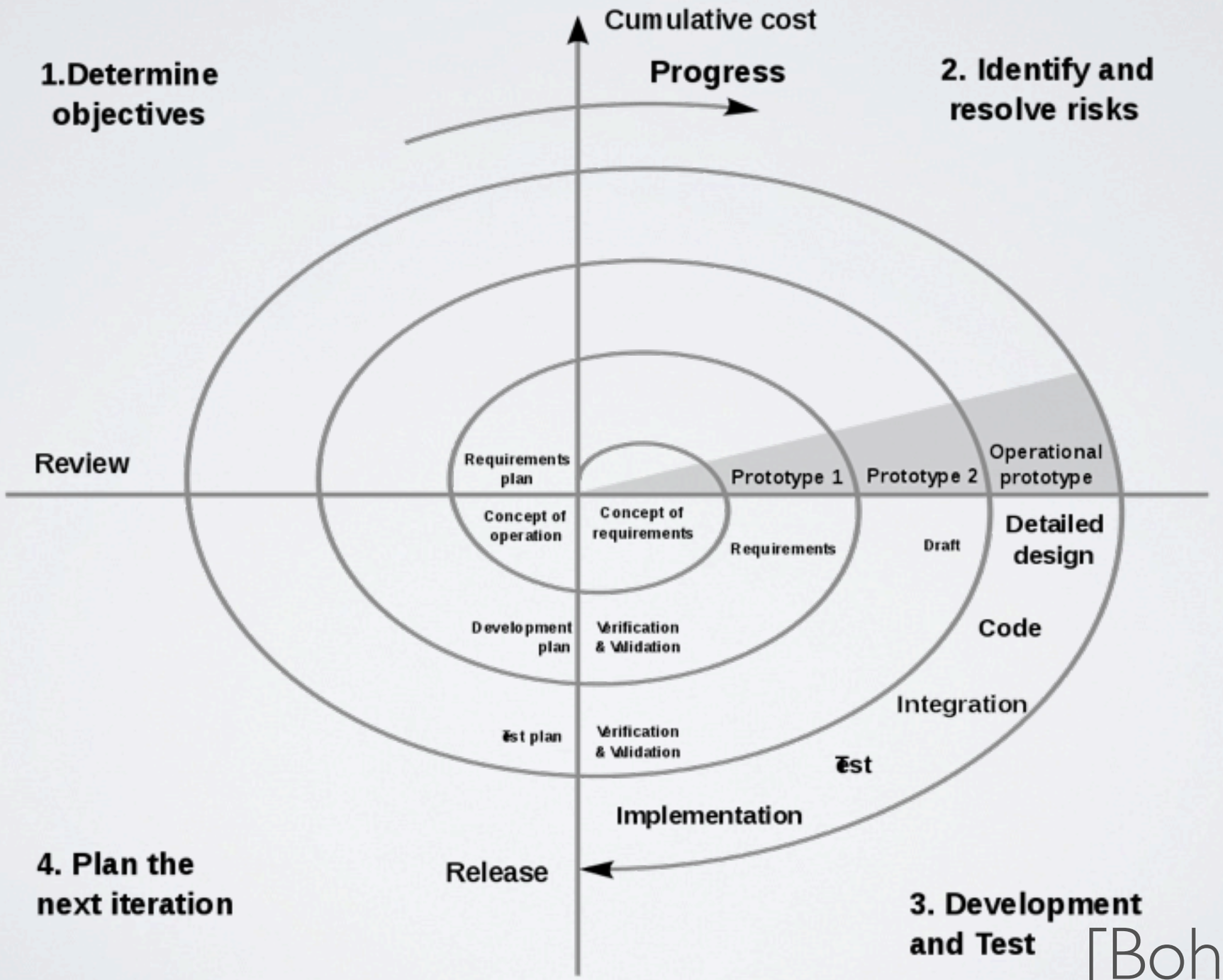
Document driven
Elaborate workflow definitions
Many different roles
Many checkpoints
High management overhead
Highly bureaucratic

Focus on working code
rather than documentation
Focus on direct communication
(between developers and
between developers and the customer)
Low management overhead

WATERFALL


Requirements
Design
Implementation
Verification
Maintenance

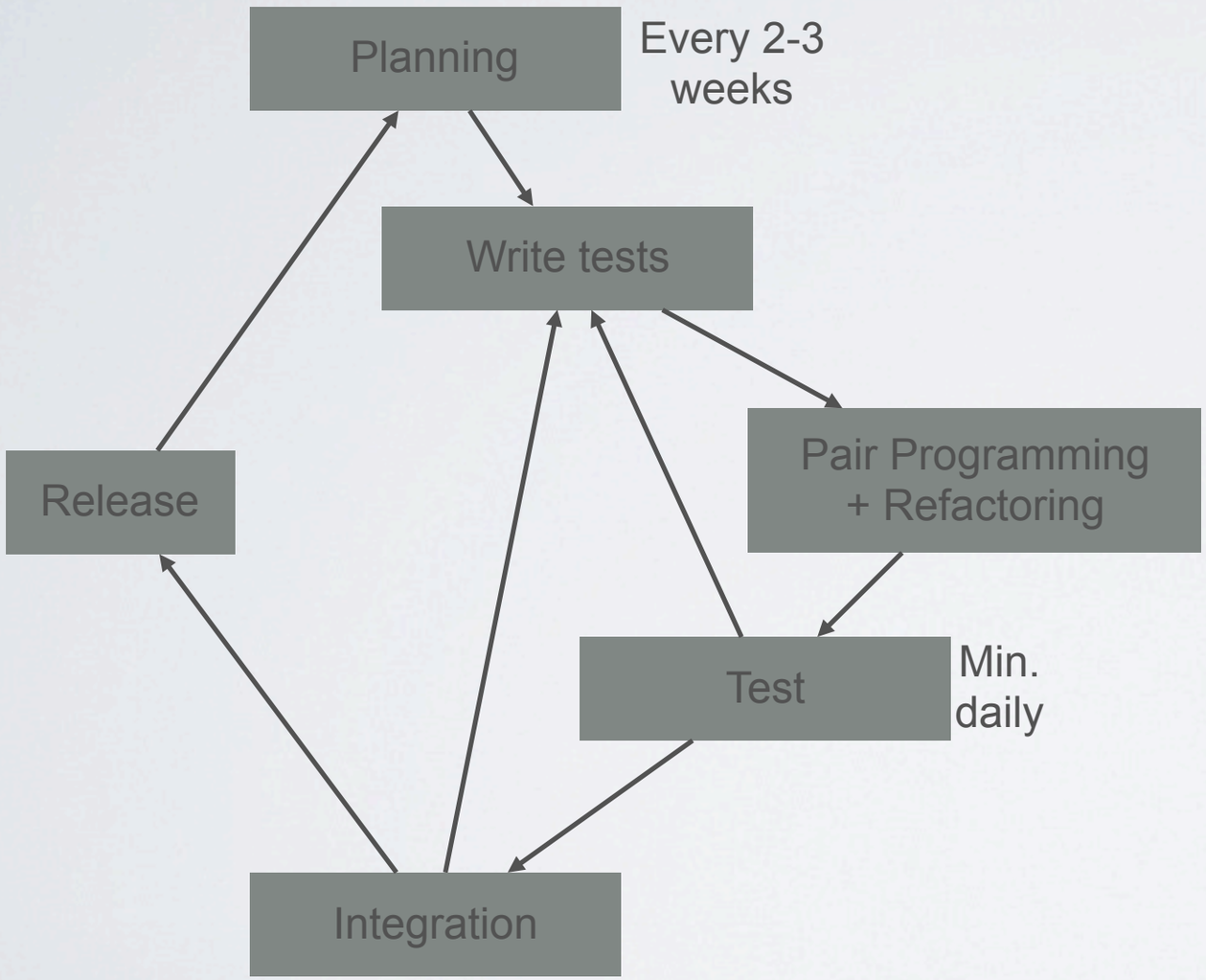
SPIRAL



[Bohem 1986]

AGILEMANIFESTO.ORG

EXTREME PROGRAMMING (XP)



Characteristics

- Evolutionary development
- Collection of 12 "Best Practices"
- Focus on working code that implements customer needs (rather than documents)
- Testing is a crucial element of the process

XP VALUES

- Five principle values:
  - Communication:
  - Simplicity: do the simplest thing that could work
  - Feedback: from the code (tests), customer (colocation), and team (planning)
  - Courage: be willing to iterate and throw away what doesn't work
  - Respect: think of your team

XP PRACTICES (I)

- The planning game
  - Stakeholder meeting to plan the next iteration
  - Business people decide on business value of features
  - Developers on the technical risk of features and predicted effort per feature
- Small releases
  - Start with the smallest useful feature set; release early and often, adding a few features each time

XP PRACTICES (II)

- Simple Design
  - Always use the simplest possible design that gets the job done (runs the tests and states intentions of the programmer)
  - No speculative generality
- Testing
  - Test-first: write test, then implement it
  - Programmers write unit tests and customers write acceptance tests
- Refactoring
  - Refactoring is done continuously; the code is always kept clean

XP PRACTICES (III)

- Pair programming
  - All production code written by two programmers
  - One programmer is thinking about implementing the current method, the other is thinking strategically about the whole system
  - Pairs are put together dynamically
  - http://www.cs.utah.edu/~lwilliam/Papers/ieeeSoftware.PDF
- Collective code ownership
  - Any programmer that sees an opportunity to add value to any portion of the code is required to do so at any time
- Continuous integration
  - Use of version and configuration management (e.g., CVS)
  - All changes are integrated into the code-base at least daily
  - The tests have to run 100% before and after the integration

XP PRACTICES (IV)

- 40-h week
  - Programmers go home on time
  - Overtime is a symptom of a serious problem
  - No errors by tired developers; better motivated developers
- On-site customer
  - Development team has continuous access to a real life customer/user
- Coding standards
  - Everyone codes to the same standards
  - Ideally, you should not be able to tell by looking at it who has written a specific piece of code

XP ADVANTAGES

- Integrated, simple concept
- Low management overhead (no complicated procedures to follow, no documentation to maintain, direct communication, pair programming)
- Continuous risk management (early feedback from the customer)
- Continuous effort estimation
- Emphasis on testing; tests help in evolution and maintenance

XP DISADVANTAGES

- Appropriate for small teams (up to 10 developers) only (does not scale)
- Large development groups may require more structures and documents
- If maintainers are not the people that developed the code, good documentation is necessary
- Generic design may be necessary to enable expected future development

SCRUM

- "Scrum is not an acronym. It's an event in the game of rugby where like-minded people get together and politely discuss ownership of a ball."

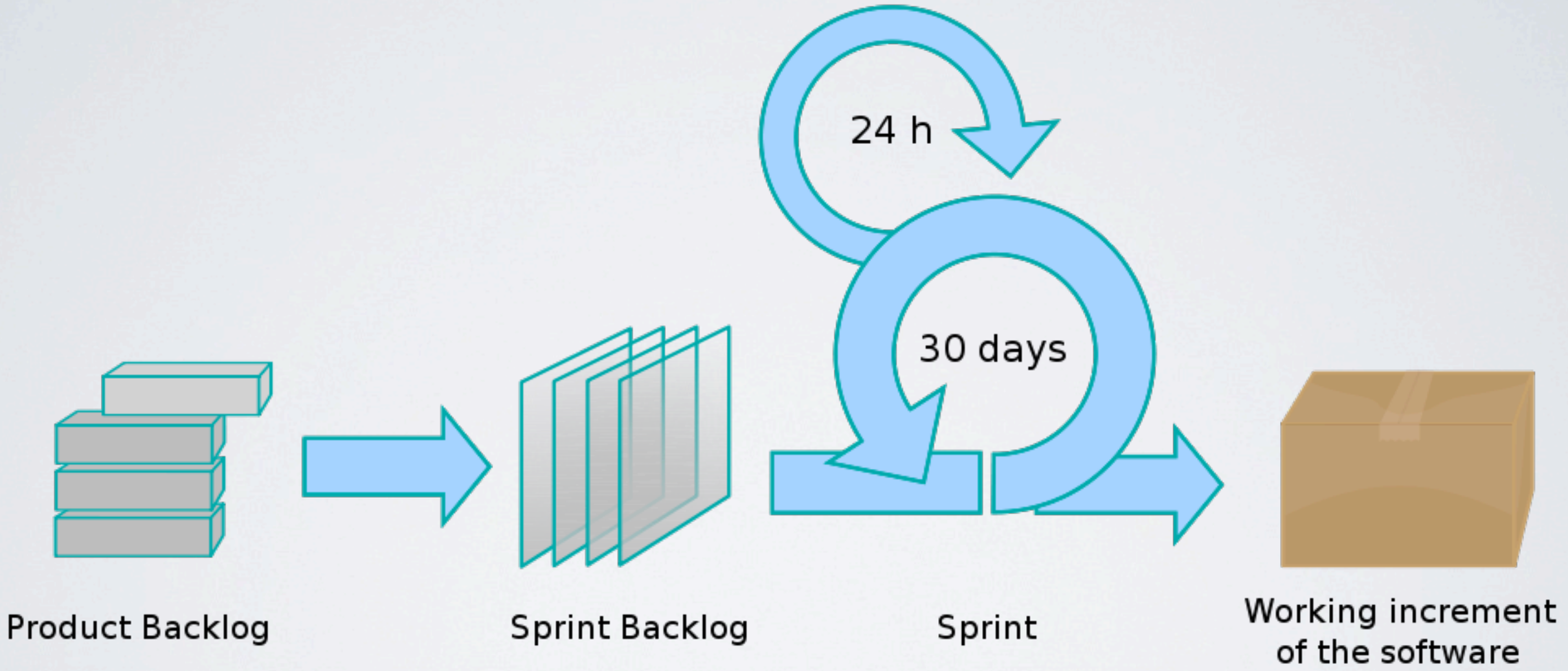
- Rugby metaphor introduced by Takeuchi and Nonaka in their 1986 paper The New New Product Development Game that reported on innovative processes being used by big companies for new product development (e.g., cars, cameras, copiers.)

- Video: Ken Schwaber, Google Tech Talks

  - http://video.google.ca/videoplay?docid=-7230144396191025011&q=agile+development

  - Intro (2m00s to 9m27s).

SCRUM PROCESS



SOFTWARE DEVELOPMENT SCRUM

- A collection of practices, reacting against high-formality processes
- Key ideas:
  - Timeboxes with deliverables
  - Small, self-managed, cross-functional teams of 3-9 people
  - Daily meetings
  - Achieve quality through transparency and monitoring, rather than processes

SCRUM MEETINGS

- Daily, less than 15 minutes
- Each team member asked
  - What have you done since last meeting?
  - What has impeded your work?
  - What will you do next?
- Try to resolve impediments quickly, or schedule smaller meetings immediately following

SCRUM SPRINT

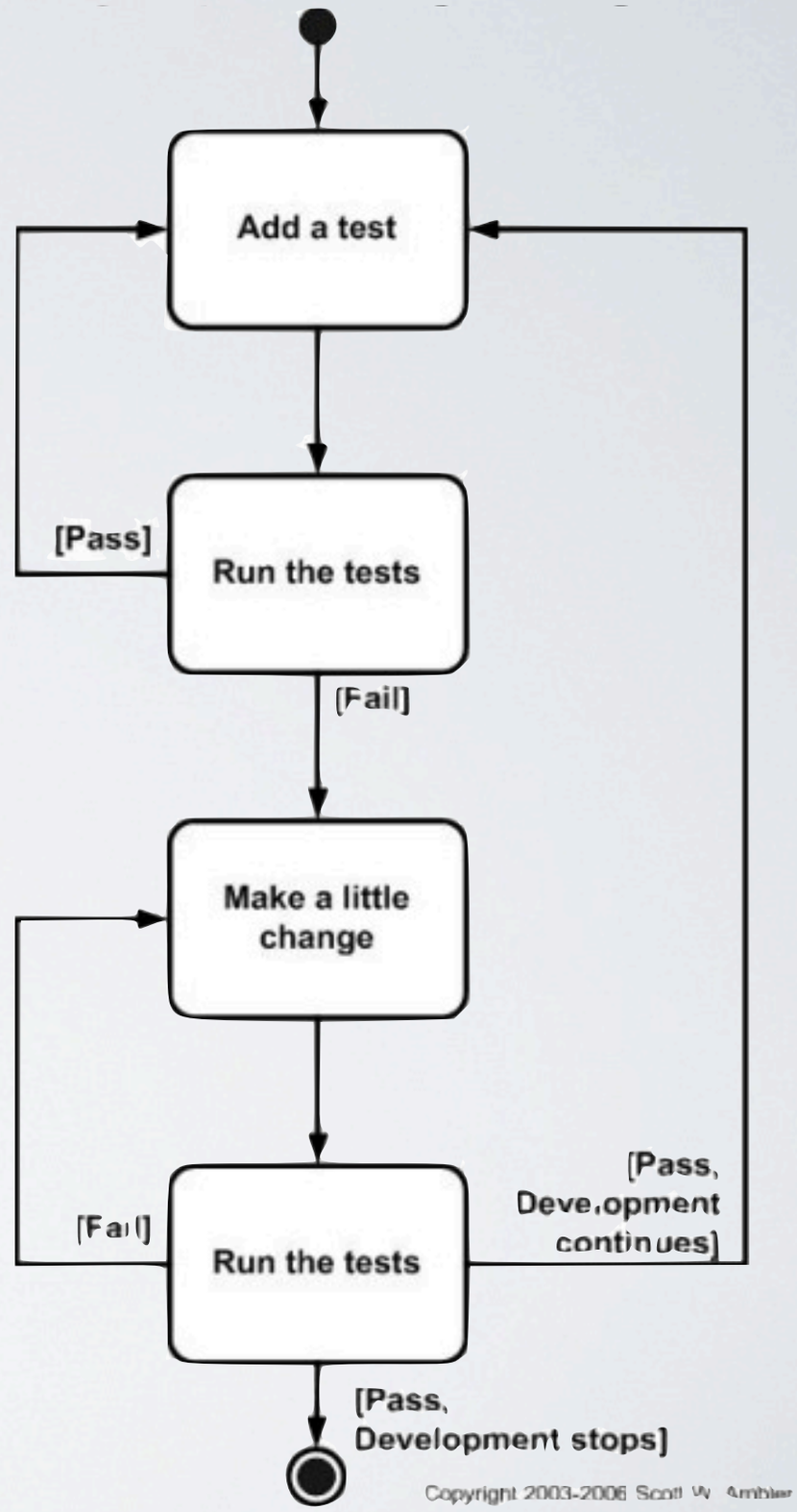
- A short development interval, e.g. 3 weeks
- Well-defined goal: what's going to be implemented, what's going to be demoed at the end of the spring
- Keep track of *Backlog*
  - Prioritized list of work to be done (features, stories, requirements)
  - Rough estimate of ideal number of days required to implement each item
  - Before each sprint, a planning session is held to decide what items from the backlog will be addressed by the sprint

END OF SPRINT

- End of sprint:
  - Demo
  - Collect feedback
  - Discuss performance, process
  - Plan next sprint

TDD

- Write tests before code
- Code is complete when tests pass
- This might sound crazy, but if you try it I bet you would be surprised by how nice this feels in practice
- Encourages 'testable' code
- Validates APIs before they are written



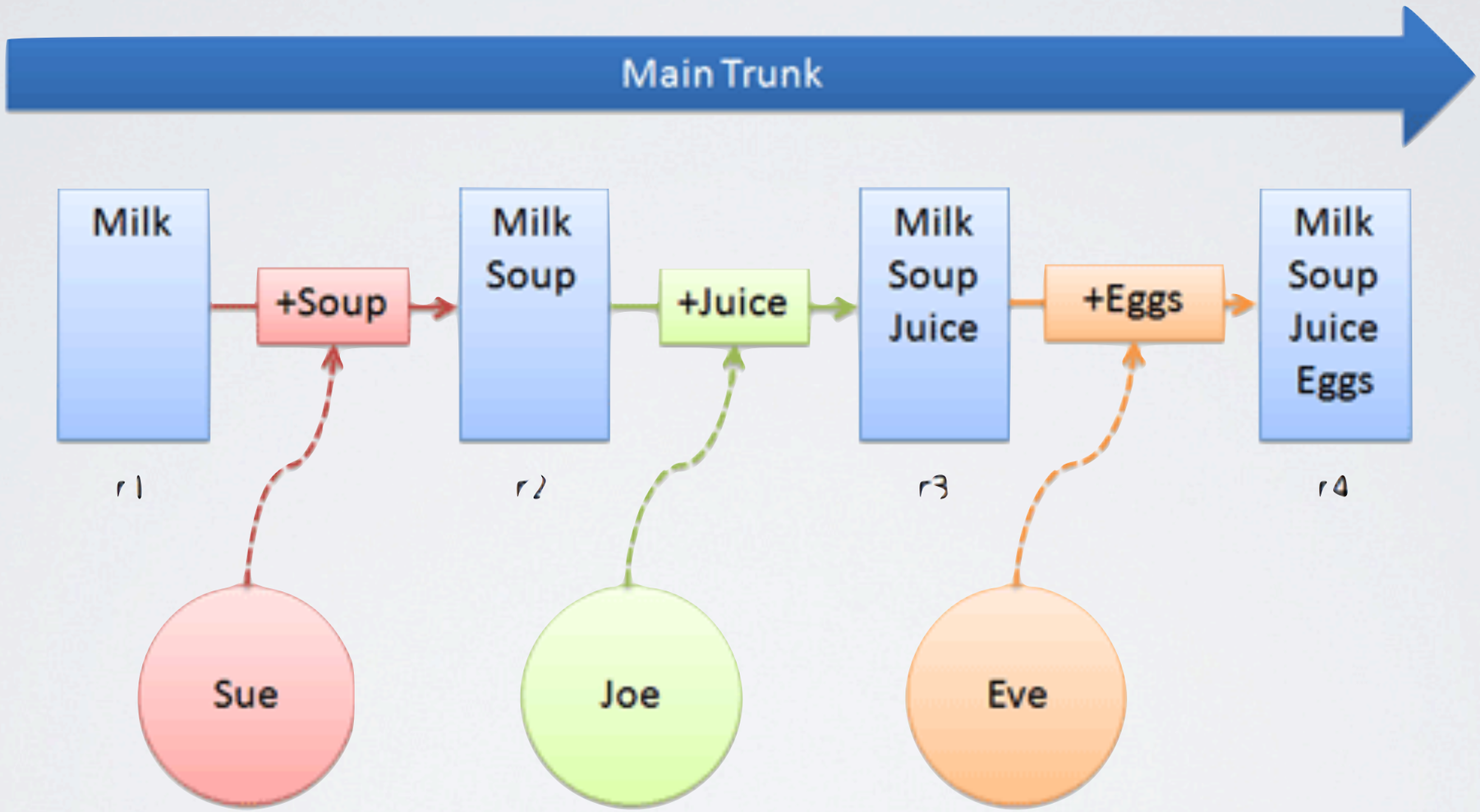
TOOLS

- Agile methods rely heavily on tools:
  - version control
  - automated test suites
  - continuous integration
  - refactoring
- The key agile guys were also great builders, which is why these tools are so prevalent today

VERSION CONTROL

- Enable teams to work together effectively by enabling changes to files to be easily shared among the team
- Permit development to ‘roll back’ to a past version if a poor change is made.
- Allow developers to try ‘risky’ changes in their own development space without putting the code base in peril
- Two flavours:
  - CVCS: (centralized)
  - DVCS: (distributed)

CVCS



[http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/]

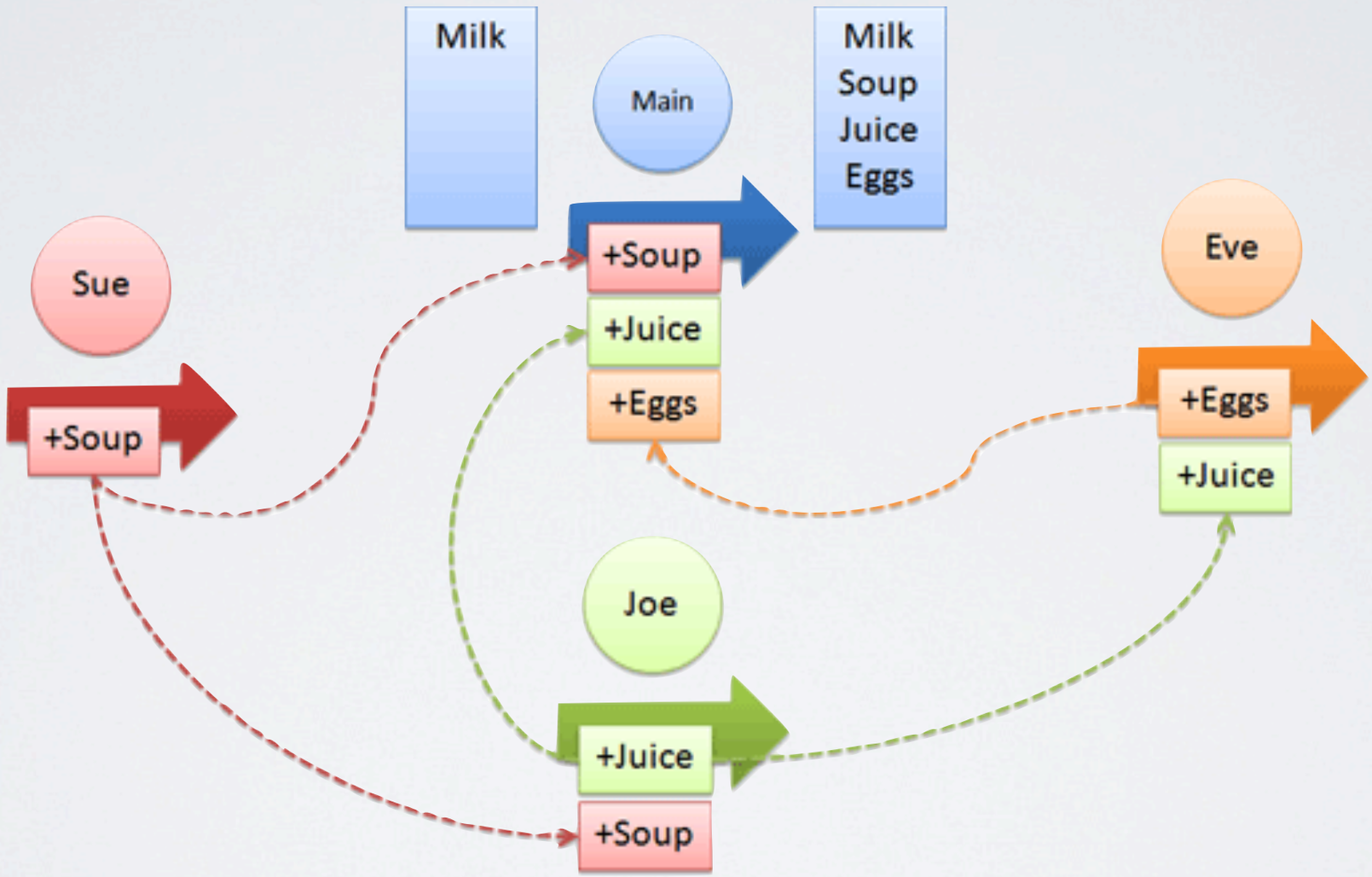
KEY TERMS (CVCS)

- Repository: data store containing controlled resources
- Checkout: download a repository
- Add: put a file in a repository
- Commit: update file(s) in the repo
- Update: retrieve remote changes from repo
- Revision: what version a file is (e.g., v23)
- Tag: put a logical name on a specific revision
- Branch: create an independent copy for some special purpose

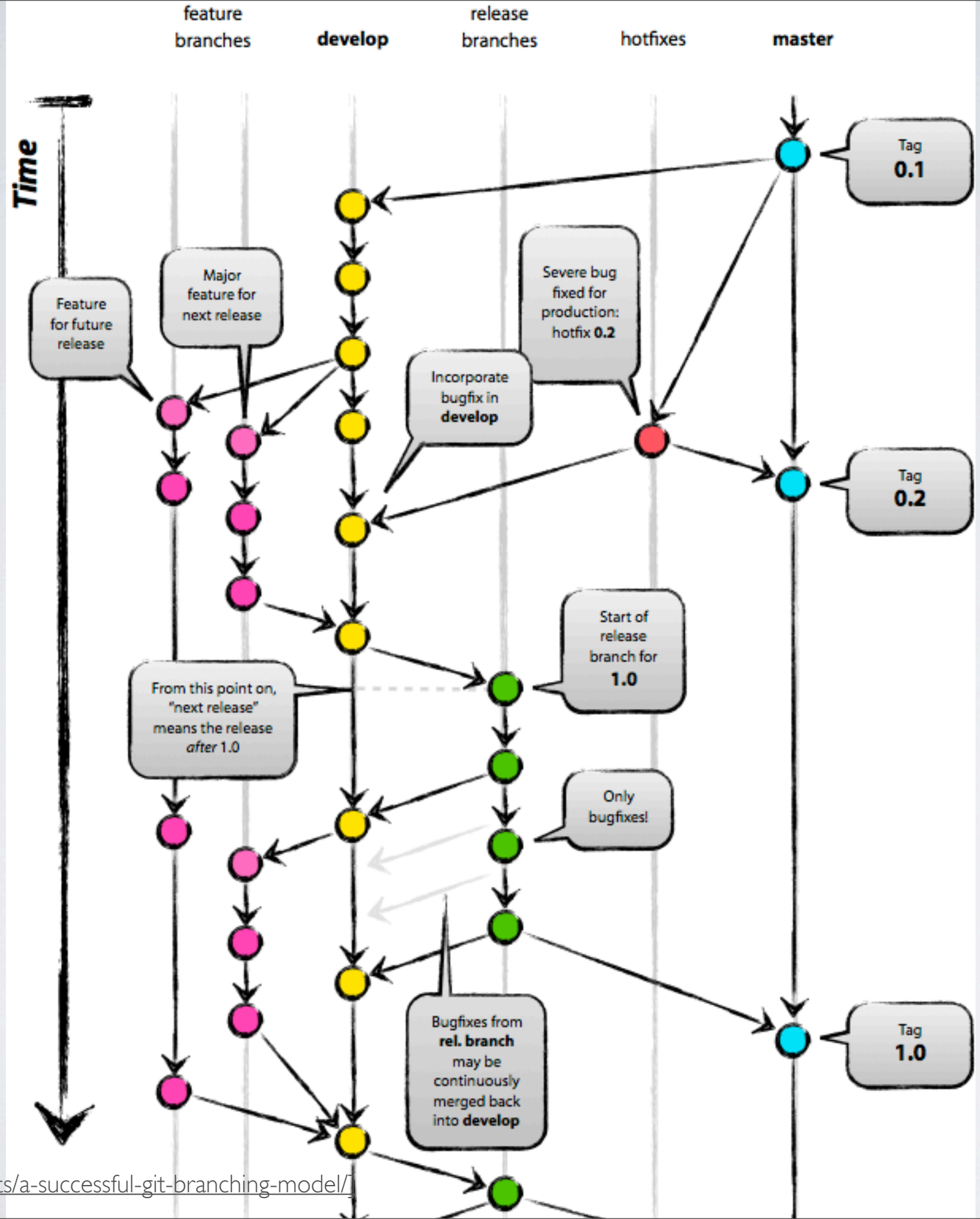
CVCS

- Shortcomings:
  - Branches / tags are heavyweight (e.g., copies)
  - Online operations only
  - Merging challenging
- Benefits:
  - Simple
  - Constraints keep you from getting into trouble

DVCS



[http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/]



[http://nvie.com/posts/a-successful-git-branching-model/]

DVCS

- Benefits of DVCS:
  - No central repository
  - Each 'working copy' contains entire change history
  - Offline commits (all operations are local)
  - Lightweight branching / tagging. Easy merging.
- Downsides of DVCS:
  - Systems give you more than enough flexibility to get yourself in trouble.

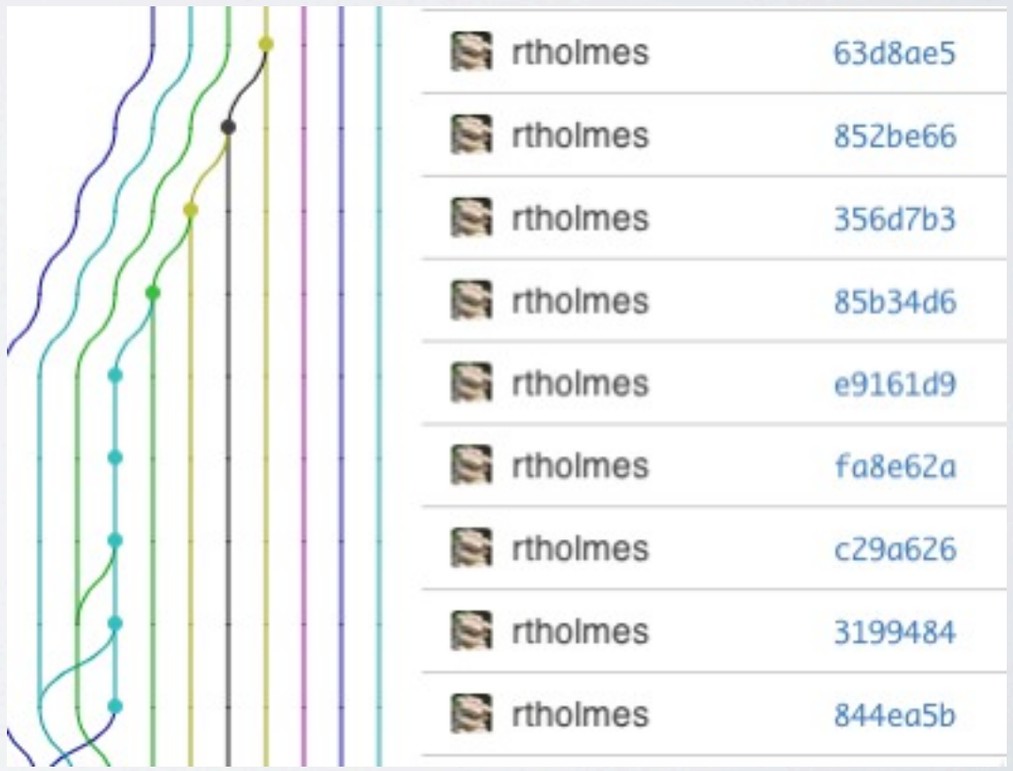
DVCS TERMINOLOGY

- Repository: every working copy contains the complete repo
- Clone: create a local repo from a remote one
- Commit: add changes to local repo
- Push: send local changes (can be many) to remote repo
- Pull: download remote changes to local repo

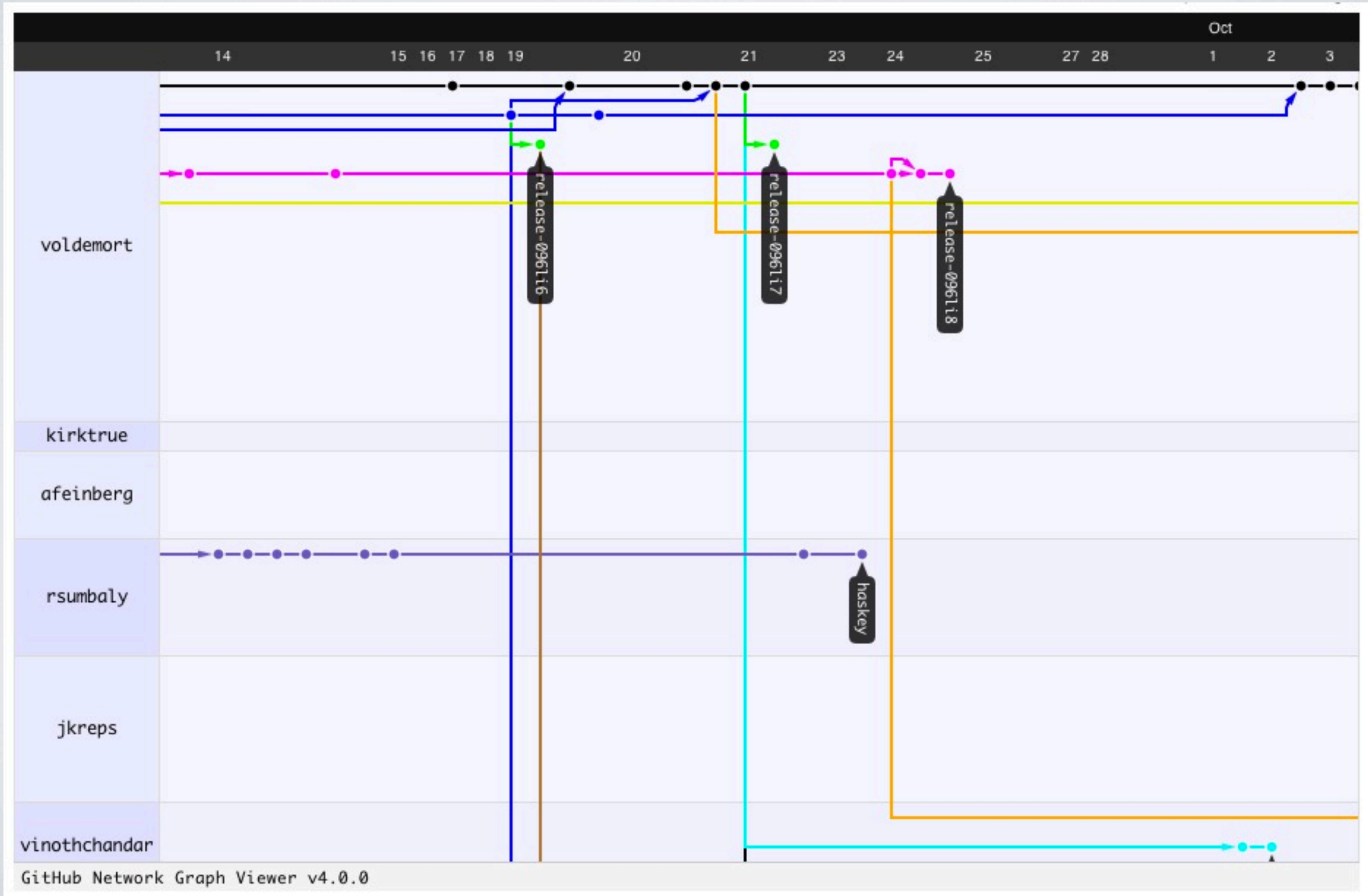
GITHUB / BITBUCKET "SOCIAL CODING"

- DVCSs offer a lot of flexibility but make it harder to keep track of what is happening in a project
- Github / Bitbucket greatly simplify working with DVCSs

- Visible public forks
- Pull requests
- Integrated issue tracking
- Code reviews



GITHUB - PUBLIC BRANCHES



GITHUB - REVIEW SUPPORT