QSPIN

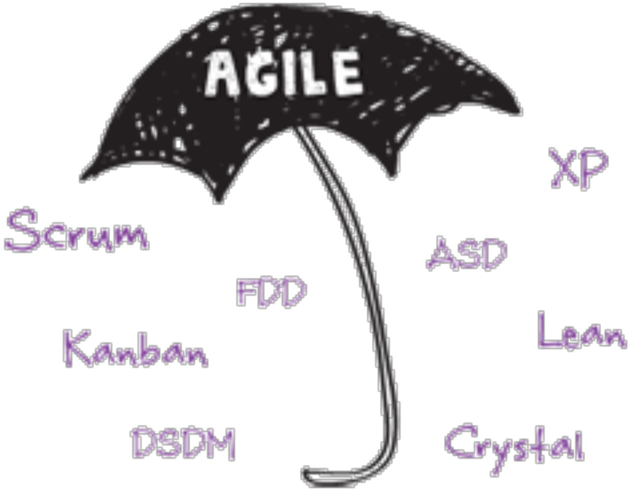
EMBED & IMPROVE

Qgile - code review

Improving software quality

SDLC - software development life-cycle

- Agile methodologies…
- Scrum
- XP (extreme programming)
- Kanban
- Lean
- Crystal
- FDD (Feature Driven Development)
- DSDM (Dynamic Systems Development Method)



Agile Development Methodologies

These practices all help teams work better together!
None of these are the best choice, select one and adapt to your team, in the past I have used methods from different standards…

Agile - pros & cons

- Frequent delivery of working software (reduce time-to-market)
- Close customer collaboration
- Respond quickly to change (changing requirements)
- Higher productivity & quality
- Cross functional team (self-organizing)

- Hard to predict timelines
- Time consuming
- Budget (hard to put a fixed price)
- Less documentation

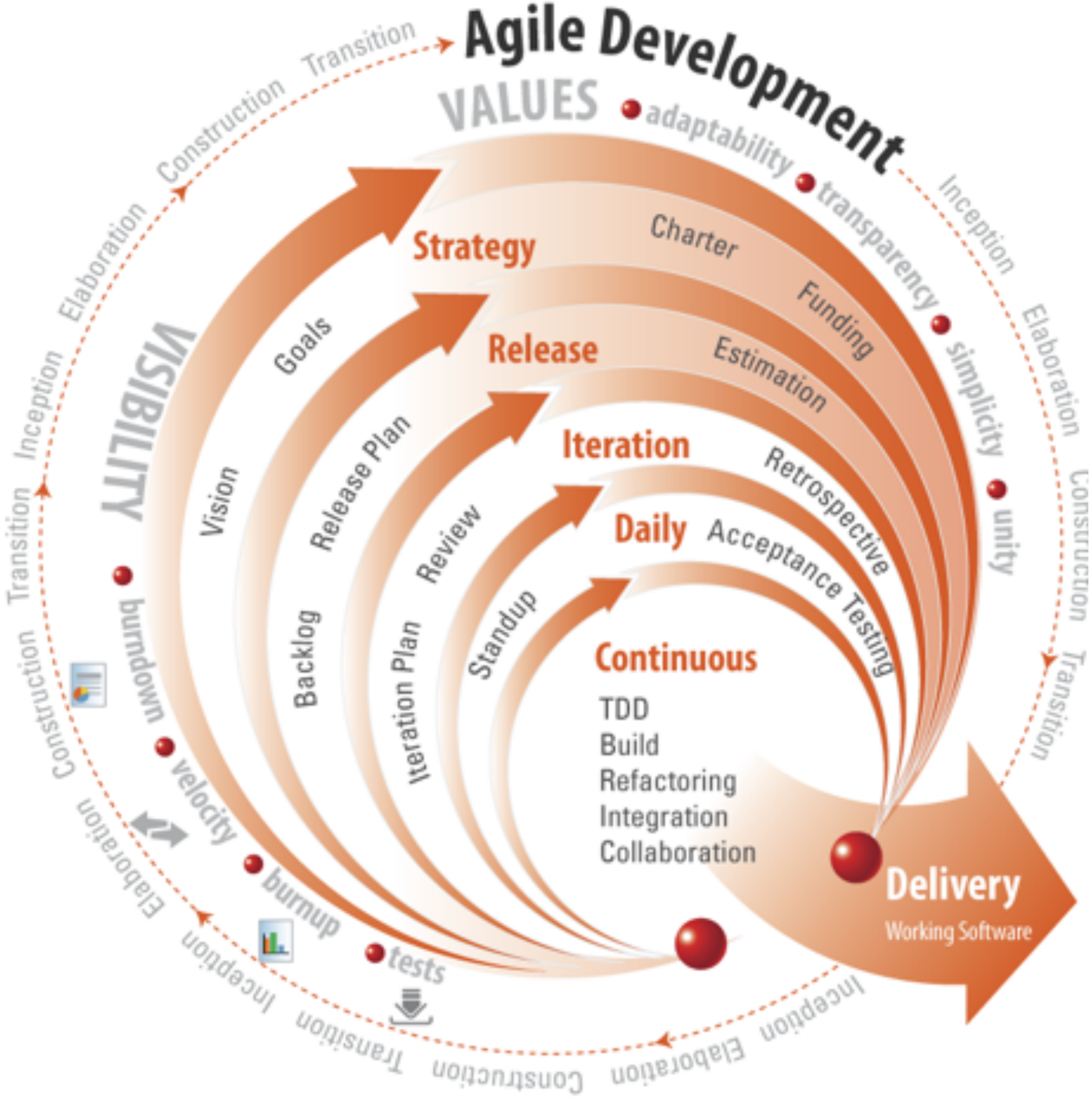
Some key practices in Agile

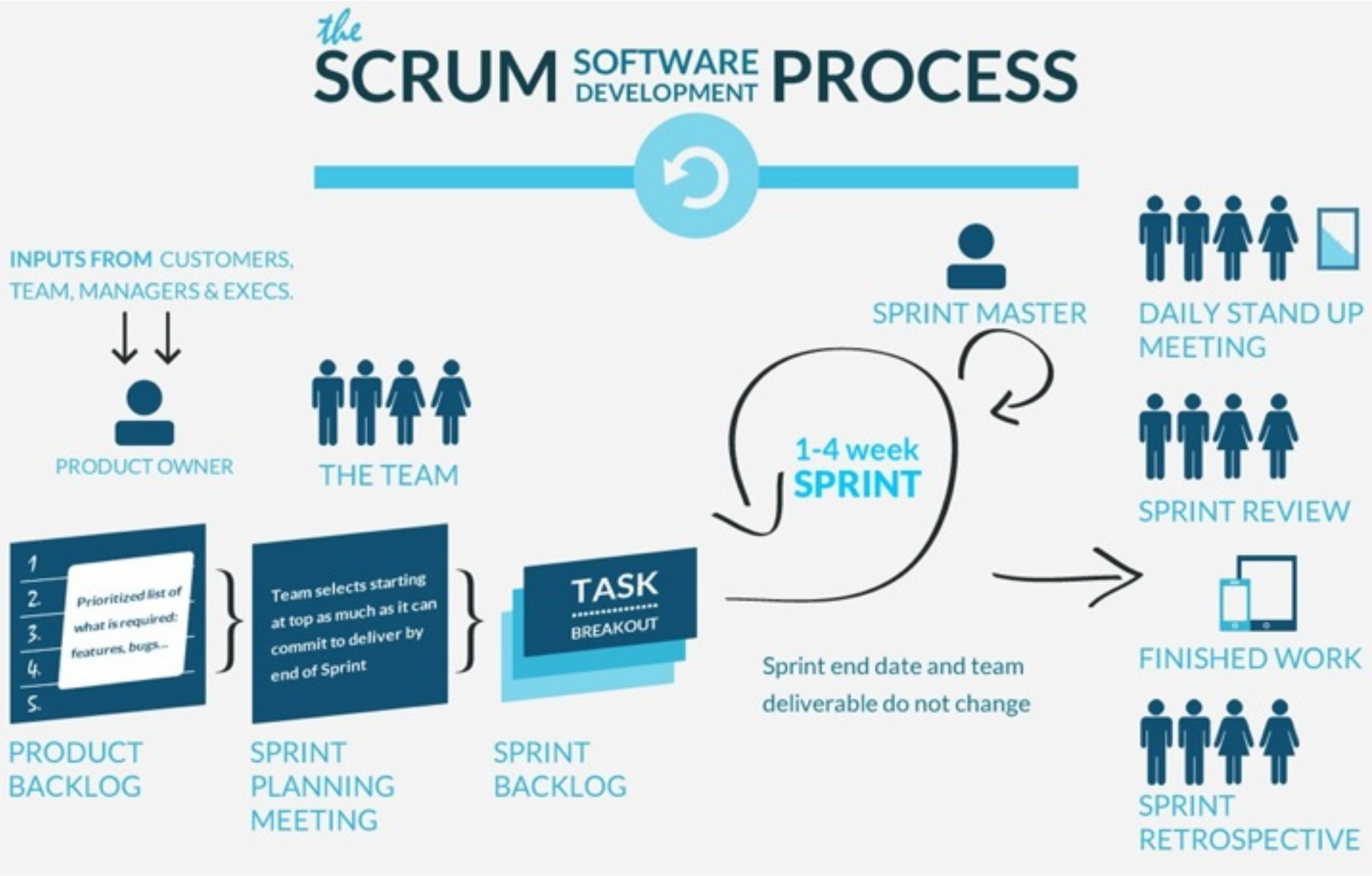
- Collective code ownership
- User story design
- Personas
- Continuous integration
- Refactoring
- Test-Driven development
- Pair programming (peer review)
- Customer acceptance testing
- Small releases
- Coding standards
- Workflow (visual scrumboard)
- Limit WIP (work in progress limit)
- **Code review & inspection**
- Code quality
- Code productivity
- MoSCoW rule (Must, Should, Could, Won't)
- Regular & fast building
- Configuration management
- Developing by feature
- Documentation
- Demo's

These practices are taken from different agile methodologies
but can be applied on every software project

The Agile Manifesto - a stement of values

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan


Agile Development
VALUES
adaptability
transparency
simplicity
unity
VISIBILITY
burndown
velocity
burnup
tests
Strategy
Charter
Funding
Goals
Vision
Release
Estimation
Release Plan
Backlog
Iteration
Retrospective
Review
Iteration Plan
Daily
Acceptance Testing
Standup
Continuous
TDD
Build
Refactoring
Integration
Collaboration
Delivery
Working Software
Inception
Elaboration
Construction
Transition



the SCRUM SOFTWARE DEVELOPMENT PROCESS

INPUTS FROM CUSTOMERS, TEAM, MANAGERS & EXECS.

PRODUCT OWNER

THE TEAM

SPRINT MASTER

1-4 week SPRINT

PRODUCT BACKLOG

1.
2.
3.
4.
5.

Prioritized list of what is required: features, bugs...

SPRINT PLANNING MEETING

Team selects starting at top as much as it can commit to deliver by end of Sprint

SPRINT BACKLOG

TASK BREAKOUT

Sprint end date and team deliverable do not change

DAILY STAND UP MEETING

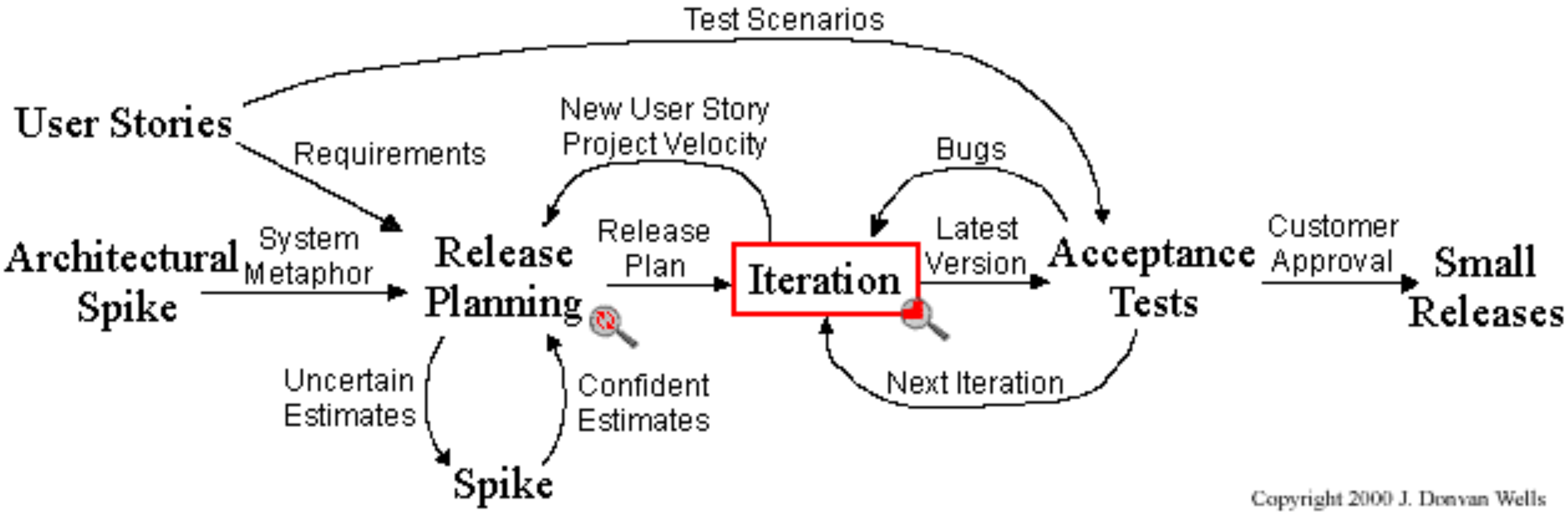
SPRINT REVIEW

FINISHED WORK

SPRINT RETROSPECTIVE

Extreme Programming

Extreme Programming Project



Copyright 2000 J. Donvan Wells

Typical selection of tools

- SCM (source control management)
- Development language
- Unit testing framework
- IDE (integrated development environment)
- Issue management
- Code analyse & quality
- Continuous integration
- **Code review**

- Release notes & changes
- Documentation & management
- Team collaboration
- Database
- Story writing
- Acceptance test criteria
- Personas

A lot of tools and practices need to be selected, again none is the best but search what is the best fit for the software project

Code review

- Share knowledge & experience
- Learning about the code base
- Finding bugs
- Mentoring by senior developers
- Examine and improve design
- Team discussions
- Working with distributed teams

- Higher code quality
- Approve or reject changes
- Ensuring code standards & guidelines
- Maintainability (poor programming practices e.g. copy paste)
- Check unit tests
- Find areas of improvement
- Verify requirements
- Reduce complexity

Code review - features

- easy to use
- understandable
- user friendly
- simple user interface
- fast
- adding comment (inline code snippets)
- approve or reject changes
- see differences between commits
- multiple repositories
- compare revisions
- request review (by mail)
- see statistics
- easily adapt the tool
- integration in other tools (e.g. Continuous integration, issue management, ...)

Code review - tools

- Phabricator
- Gerrit
- Review board
- Rietvield
- Barkeep
- Upsource (jetbrains)
- Crucible (atlassian)

Based on the features a code review should have I selected Barkeep as a tool for doing code review, it is also easy to adapt (open source - ruby)

Qgile - github & dropbox

- dropbox: https://www.dropbox.com/home/QSpin/Qgile
- github: https://github.com/qspin/tools

Some more documentation regarding Qgile and presentations can be found on the QSpin dropbox or on the github site for QSpin

Useful links

- http://www.extremeprogramming.org
- www.versionone.com/Agile101/Agile-Development-Methodologies-Scrum-Kanban-Lean-XP/
- http://www.mountaingoatsoftware.com/agile/scrum
- http://www.ganssle.com/inspections.pdf
- http://getbarkeep.org
- https://github.com/ooyala/barkeep

Qgile

Improving software quality