



The Eclipse Way: Adopting the Process

Martin Lippert, martin.lippert@it-agile.de

A few words about me...



- Martin Lippert
 - Senior IT consultant at akquinet it-agile GmbH
 - martin.lippert@akquinet.de
- Focus
 - Agile software development
 - Refactoring
 - Eclipse technology
- Equinox incubator committer

The Eclipse Way

- “The secret of the success of the Eclipse team“
- A collection of agile software development practices
- Used, developed and improved over time by the Eclipse SDK team
- Central ideas and concepts:
 - “people and interactions over processes and tools“
 - Feedback and shipping
 - Continuous project health: get healthy, stay healthy
 - Continuous *

The success of the practices

- The Eclipse team is shipping high quality software on-time for many years now
 - Weekly integration builds on-time
 - Six week milestones on-time
 - Yearly releases on-time

- A healthy project
 - Works on this high-level over years
 - Continuously improving the process

What makes “The Eclipse Way” special?

- Used successfully for many years:
 - Developed within a real project
- Fully transparent:
 - Everybody can observe the team using the process
 - Transparent progress, transparent planning, transparent quality
- Used by a large team
- Used by a distributed team

Why adopting?

- Wouldn't it be great to use this process for general in-house projects?
 - Always deliver on time
 - Always produce high quality
- Who is not dreaming of such a project?
;-)

Experiences

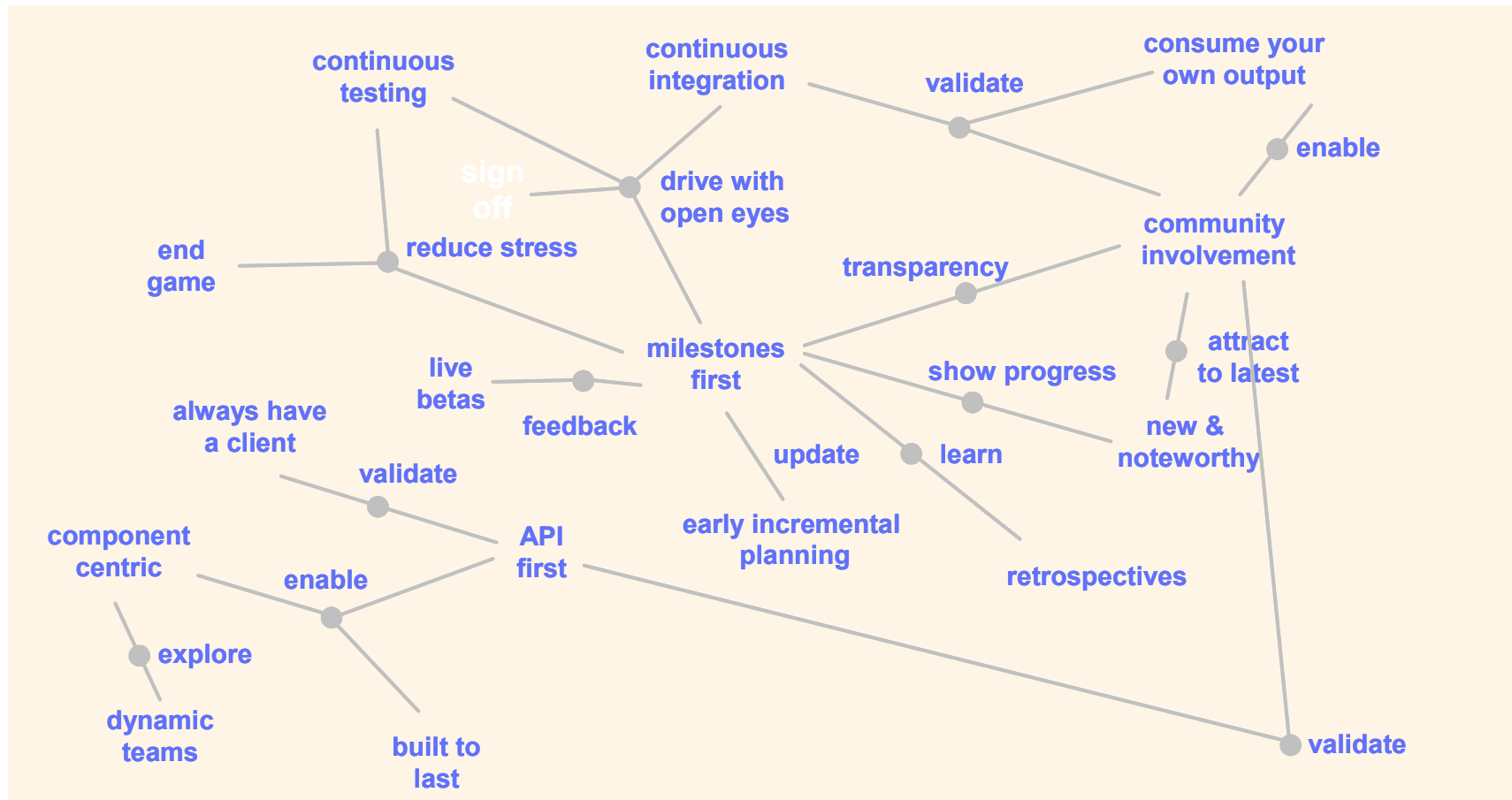
- Others are already starting to adopt the process
 - Internal IBM projects, Jazz team, ...
- My own experiences from a project:
 - ... average of 10 developers
 - ... 3 domain experts (customer role)
 - ... 1 build- and test-manager
 - ... 1 requirements manager
 - ... using the process for more than 2,5 years now
 - ... produced 4 releases

Adopting: It's not just about some practices

- The values of the Eclipse team:
 - **Quality:** ship high-quality software
 - **Predictability:** ship on time
 - **Transparency:** no secrets about ship readiness
 - **Feedback:** are we ready to ship?

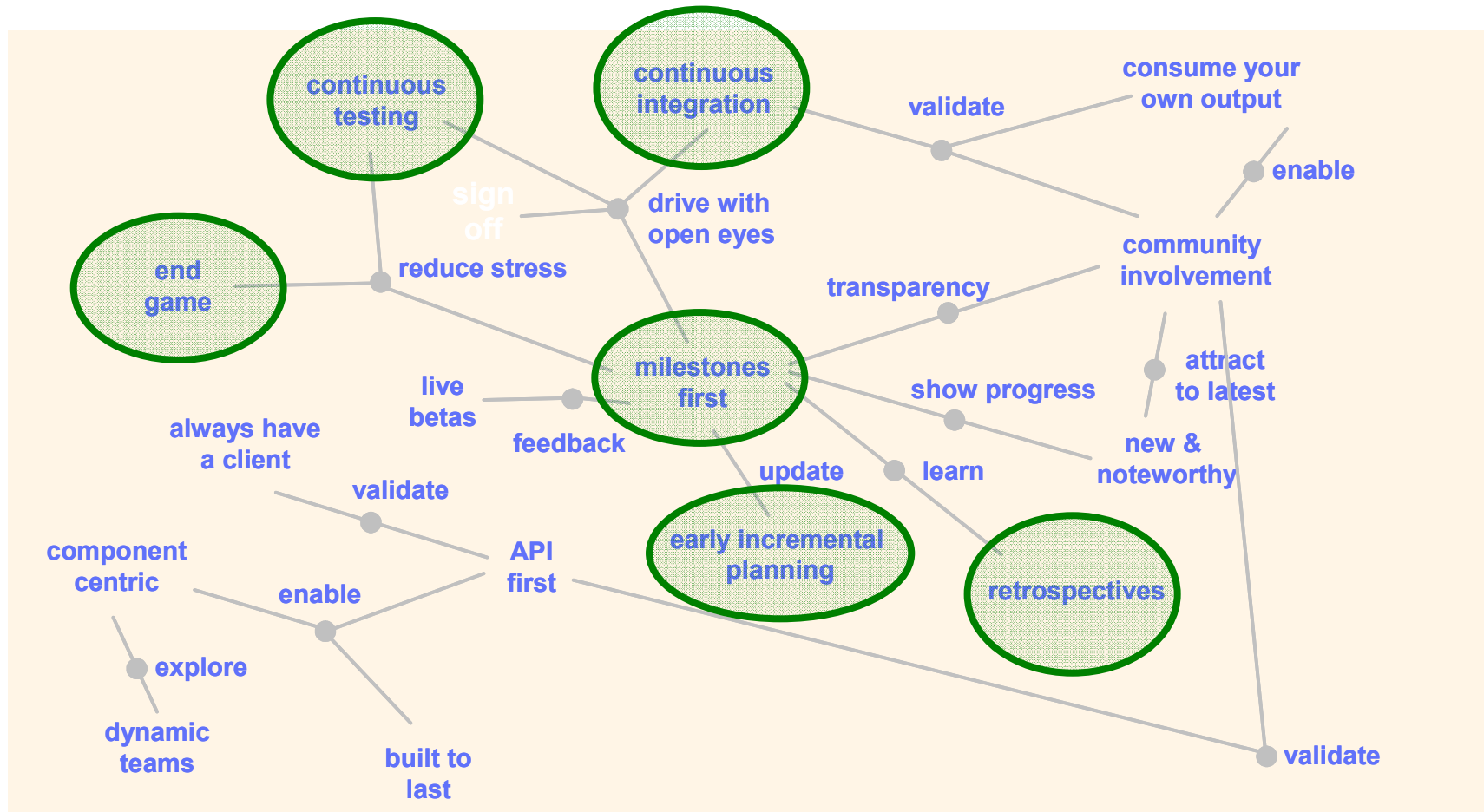
- The team needs to adopt those values
 - This is not trivial within some companies or teams
 - Often needs organizational and social development
 - Support from management necessary
 - Skilled enthusiasts

Adopting: the practices



from: JAX 2005, The Eclipse Way – Part 1: The Eclipse Way explained, Tobias Widmer, Copyright by IBM

Easy to adopt



Easy to adopt

- Continuous testing, Continuous integration
 - Essential for many agile processes
- Milestones first
 - Small cycles, maybe less than six weeks
- Early incremental planning
 - Essential for many agile processes
- Endgame
 - Stabilizing the product at the end of the release cycle - no feature adding
- Retrospectives
 - Essential to improve the process over time

Our way of adopting

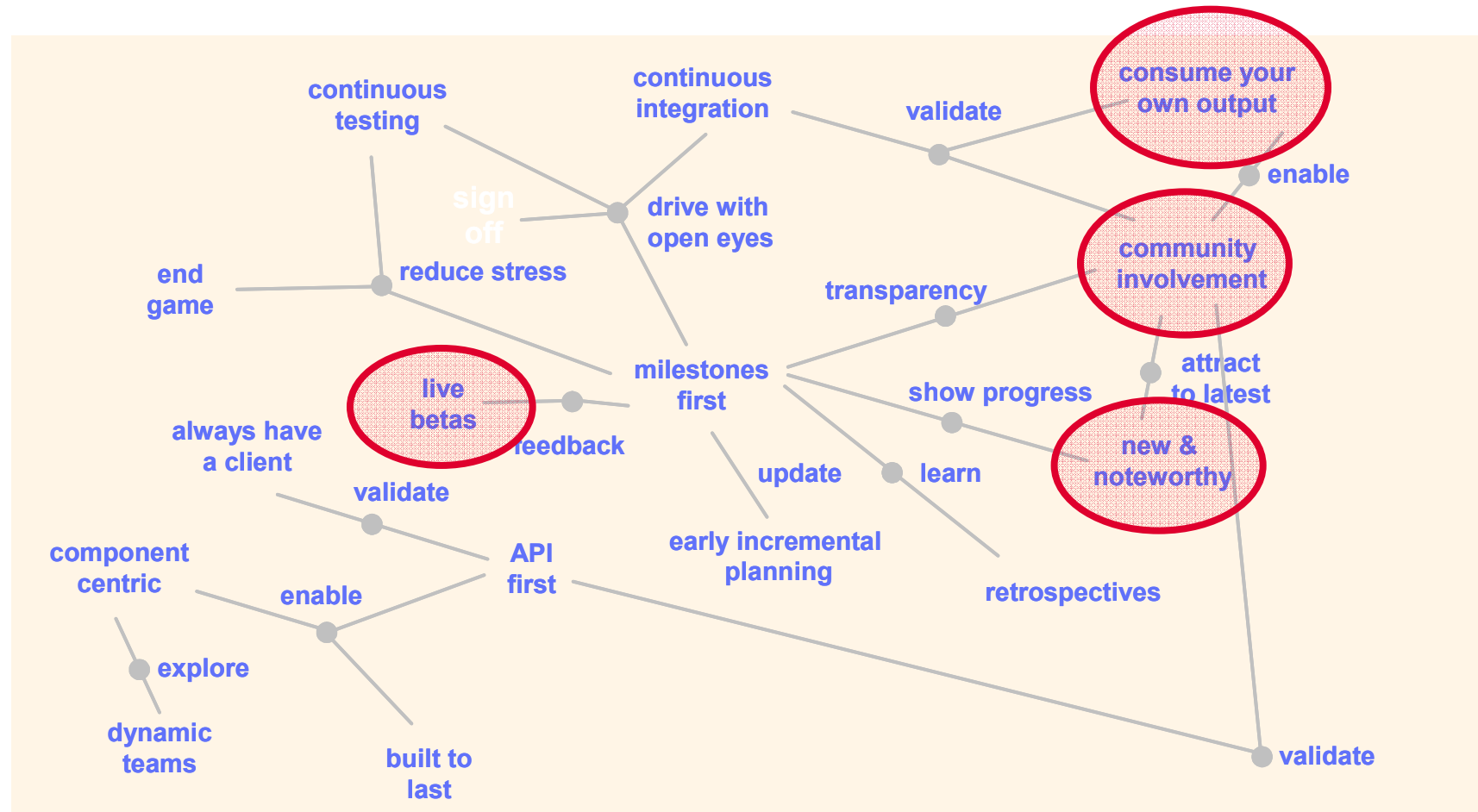
- Absolutely essential in our projects:
 - Committing code several times a day
 - Having a good state in HEAD all the time (not just for nightly builds)
 - Having the unit tests running all the time (not just for nightly builds)

- We use milestones (not only but also) for planning
 - It's a good planning unit
 - Domain experts prioritize and decide directly about the contents of a milestone

Our way of adopting

- Endgame
 - Means for us also a feature freeze
 - Length depends on the contents of the release and the backend systems we are depending on
 - Re-prioritization happens every day
 - Very close domain expert collaboration
- We do retrospectives more frequently during the beginning of a project
 - Team building
 - Get feedback about the process as quick as possible

Harder to adopt



Consume your own output

- We seldom implement software for software developers
 - Instead for people who are experts in a specific business domain

- How to adapt?
 - Integrate the domain experts into the team as close as possible
 - Let them play with the system all the time
 - Let them play with real data
 - Request feedback early and often

Live betas

- Could mean that customers use milestone builds for daily use
- Why not?
 - DB schema differences between last production release and current beta
 - Law changes
 - Revision safety
 - Collaboration between different versions
- How to adapt?
 - Working with different database schema versions hard to solve
 - Working in test mode on updated copy of production system
 - Using different stages to come as close as possible to a live beta

Community Involvement + New & Noteworthy

- Who is your community?
 - Typically you have a limited number of users
 - But few users providing good feedback are better than many users providing no feedback

- How to adapt?
 - Shorten release cycles (more Extreme Programming)
 - Strengthen customer involvement (integrate them into your team)
 - Be open, transparent and honest to them, show them that they have influence on the software
 - New & Noteworthy could help!!!

Our way of adopting

- Domain experts represent real customer within the project team
 - They are integrated into the development process closely
 - They write, prioritize and test features all the time
 - Developers work together with the experts on the features
 - Requesting feedback all the time
 - While implementing the feature
 - While testing a milestone
 - While testing the release in the endgame
- Shorter release cycles
 - 2-3 releases a year
 - Trying to get more releases out

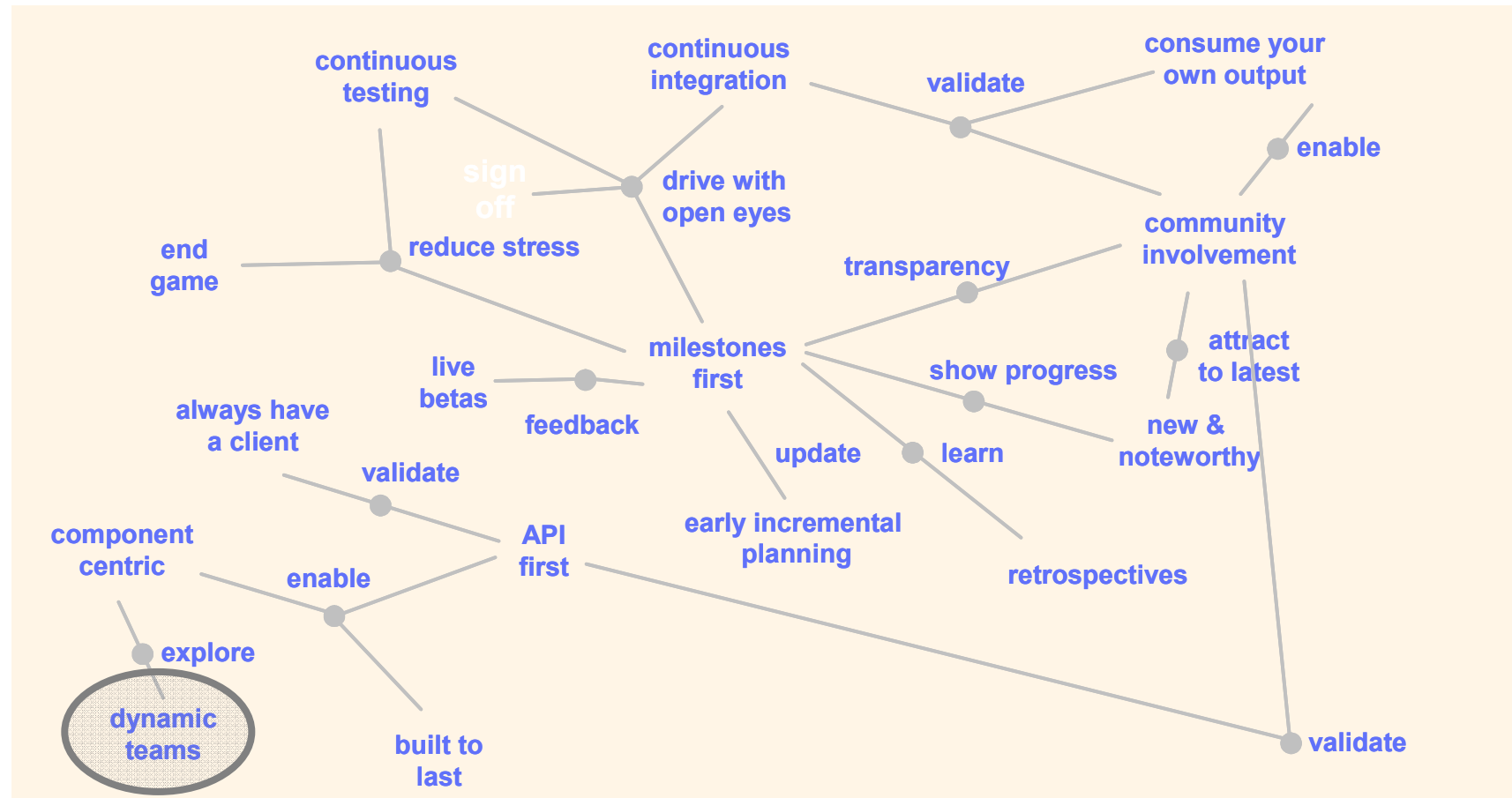
Our way of adopting

- Evolutionary database design:
 - Many small changes
- Staged database migration:
 - Daily work: developers have their own database schema
 - Daily work: one integration machine with an integrated schema
 - Weekly work: migrated production copy for developer testing
 - Milestones: migrated production copy for domain expert testing

At the end...

- **It's all about starting the feedback loop**

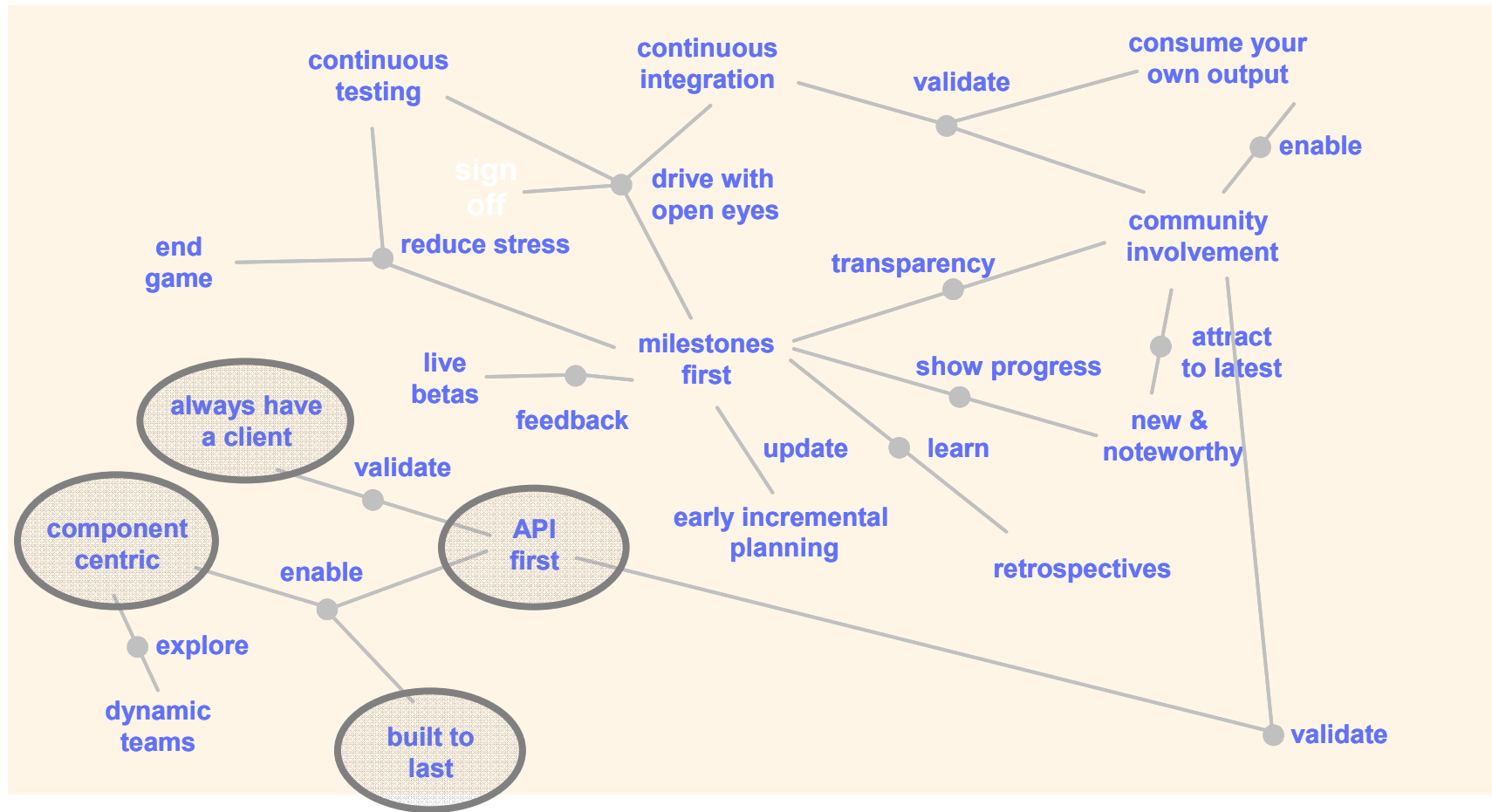
More stuff to adopt



Our way of adopting

- Creating a small team for a special task
 - For example for a complicated feature to realize
 - The small team is responsible for implementing the feature
 - The small team is responsible for discussing and evaluating the feature with the domain experts
- We did this with a team of 3 people with great success
- Also possible:
 - For tasks that crosscut component boundaries
 - For tasks that crosscut distributed team boundaries

Even more stuff to adopt



Building platforms

- Sounds like this is mostly interesting for
 - ... larger teams
 - ... distributed development
 - ... platform-based development

- But even small design decisions benefit from this
 - Better APIs between components
 - Better extensibility and maintainability

Platform-based development

- Is really interesting for in-house projects
 - Supports building a uniform architecture
 - Supports wider reuse of components
 - Allows building pluggable and highly integrated applications built over time and by different teams

- But it's not for free
 - Higher investment (API first, build to last)
 - More organizational questions coming up

Our way of adopting

- We built a platform for in-house insurance applications
 - The platform contains core functionality, abstract and concrete concepts as well as UIs for those concepts
 - Concrete insurance apps can be build upon that platform
- Clear separation between
 - ... the platform and the apps
 - ... the apps under each other
- The development:
 - We use different workspaces for platform and apps
 - We use a platform build as target environment

Challenges

- The infrastructure is essential
 - You need a fluent automated build process
 - You need a easy to use issue tracking system

- A well-rehearsed team is essential
 - Each team member is responsible for playing well
 - Each team member needs to behave nicely within the process
 - Huge knowledge differences between team members becomes difficult

- Don't forget: **It's about people and interactions**

Experiences

- The process works great for in-house projects !!!
- Fast feedback is essential
- Continuous build is essential
 - Including unit-testing and other reports
- Milestone builds are a good backbone
 - Good as a planning unit
 - Needs to be tested by customers intensively
- Live-Betas
 - Only if you trust your customers and take them seriously you will get real feedback

Thank you for your attention

Questions always welcome!

Visit us at the it-agile booth!!!



martin.lippert@it-agile.de

Special thanks to Tobias Widmer!!!



Schulung verlängerte Werkbank
agile Softwareentwicklung

Festpreisprojekte Coaching

RCP **Systemintegration** Eclipse

h3270 Hostintegration

Scrum Refactoring testgetriebene Entwicklung

Hibernate SAP-Netweaver **OpenSource**

Ajax JBoss/JEMS Groovy



eXtreme Programming