Rational. software

From Eclipse to Jazz

Erich Gamma, John Wiegand
IBM Distinguished Engineers, IBM Rational

IBM Rational Software Development Conference 2007

What keeps me **Rational**?

IBM

What is Jazz?

Innovation

A major investment by IBM to create a scalable, extensible team collaboration platform for seamlessly integrating tasks across the software lifecycle

the topic of this talk

Tooling the Eclipse Way

A commercial project led by the IBM team that brought you the Eclipse Platform, tooling the agile practices of this proven open collaborative model

Innovative Software Engineering

Community

Jazz.net – Jazz project venue for open commercial development of Jazz platform and Jazz-based products and an extension of the world wide Eclipse ecosystem

Vision

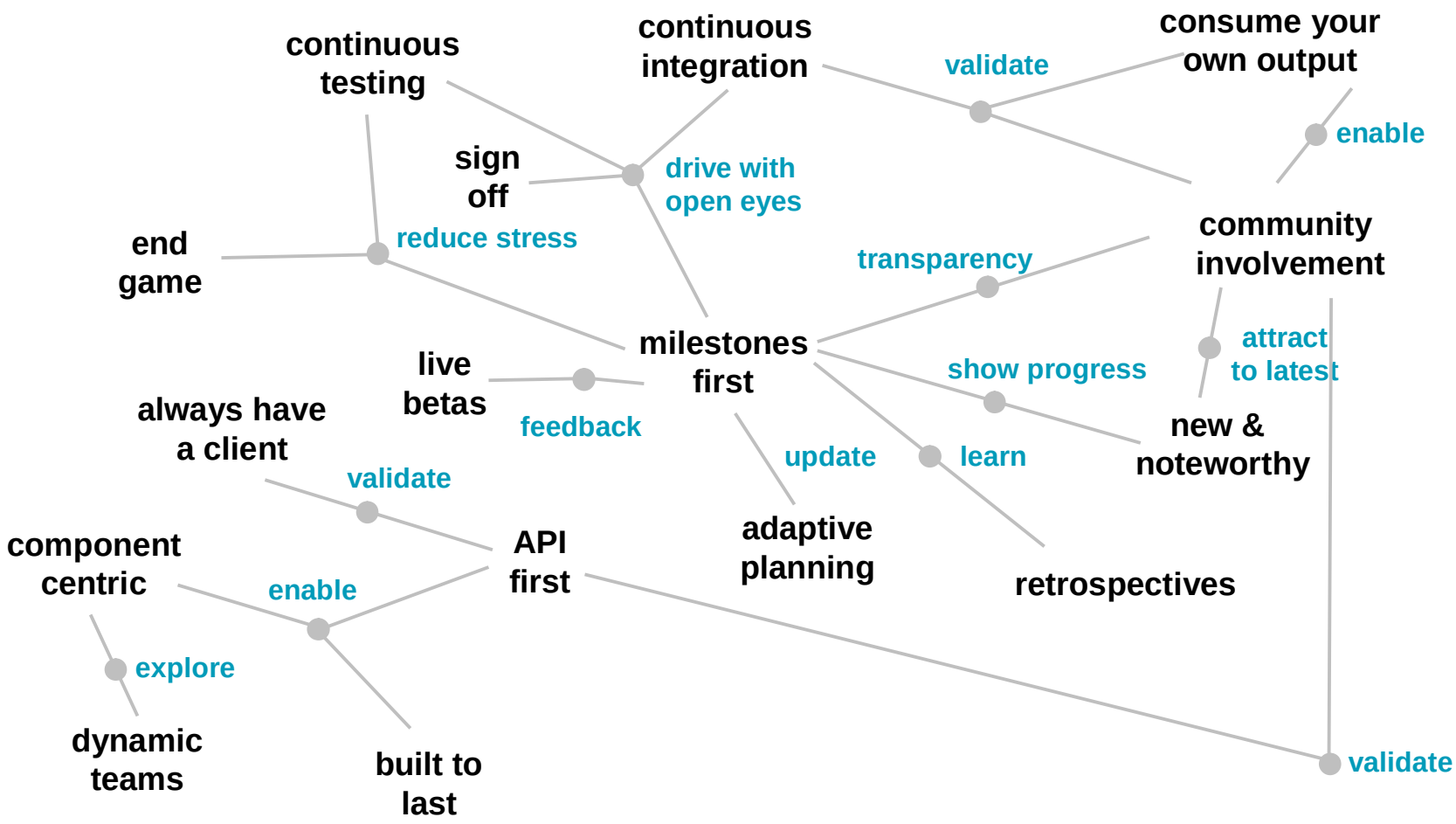
A vision for the value and experience that future Rational products can bring to software and systems delivery teams

Rational.

From Eclipse to Jazz

- Pre-Eclipse Way
  - **Culture**: shipping matters, delivering quality on time
- Eclipse
  - 6 years shipping on time
  - Reflecting on process
  - **Describing** the process ⇒ the Eclipse Way
- Jazz
  - **Tooling** the process

The Eclipse Way Practices



It is about being Continuous

- **Continuous** iterative and adaptive planning
- **Continuous** design/refactoring
- **Continuous** integration
- **Continuous** testing
- **Continuous** listening
- **Continuous** demos
- **Continuous** consumption of our own output
- **Continuous** feedback
- **Continuous** learning
- **Continuous health**
- **Continuous progress**

What is behind the Eclipse Way

- Practices underpinned with **values**
  - ship quality on time
- **Used**, **developed** and **improved** over time
  - Worked for us (and others)
- Practices are from all kinds of sources
  - XP, Scrum, Crystal Clear, RUP, …
  - Patterns - Organizational Patterns of Agile Software Development – Coplien
- It is **not low ceremony**
  - Approvals, verifications, reviews
- It is **agile**: incremental, iterative, collaborative, transparent, customizable
  - And it scales up

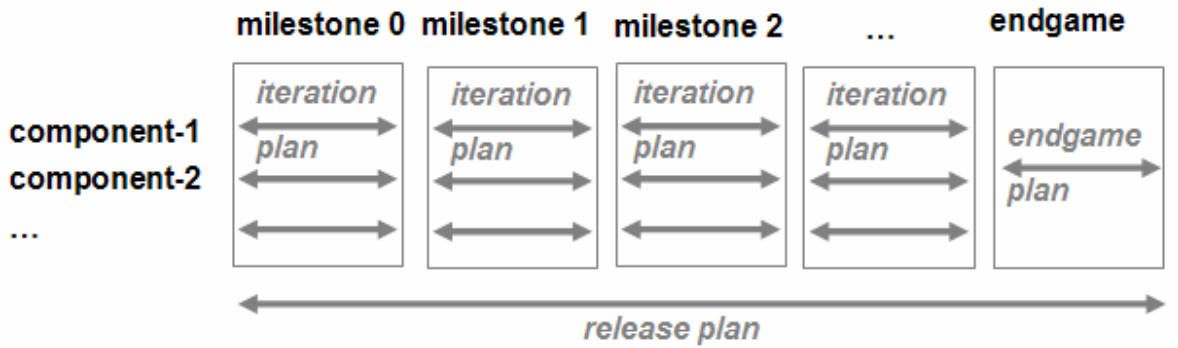
The Eclipse Way is Collaborative

Example “top architectural issues” work product

1. Project leadership team **initiates**
   - collection of architectural issues
2. Component teams **provide input**
   - architectural issues of own component
   - architectural concerns about other components
3. Project leadership team **consolidates**
   - top five architectural issues list (work product)

The Eclipse Way – Scales up

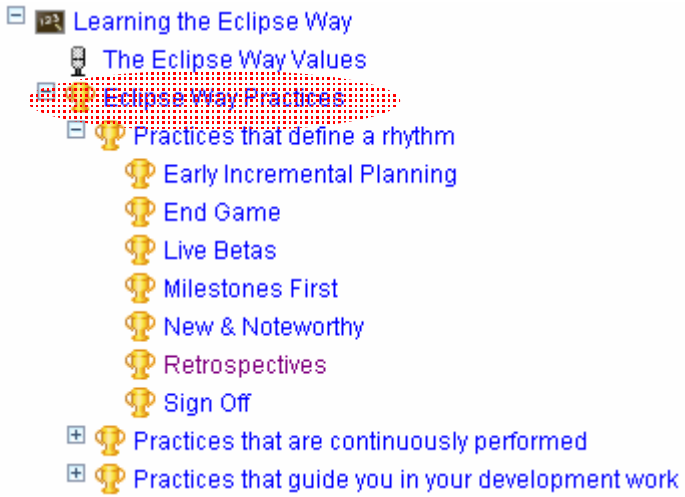
- Many practices working together
- It is **not** just about **code**
  - release plans, iteration plans, test plans, end game plans…
- **Interdisciplinary** – we play many roles
  - developer
  - project lead
  - tester
  - business analyst
  - customer support
  - system admin
  - release engineering
- **Component** centric
  - a **team** is responsible for one or more **component**
  - dependencies through **APIs**
  - plans per component

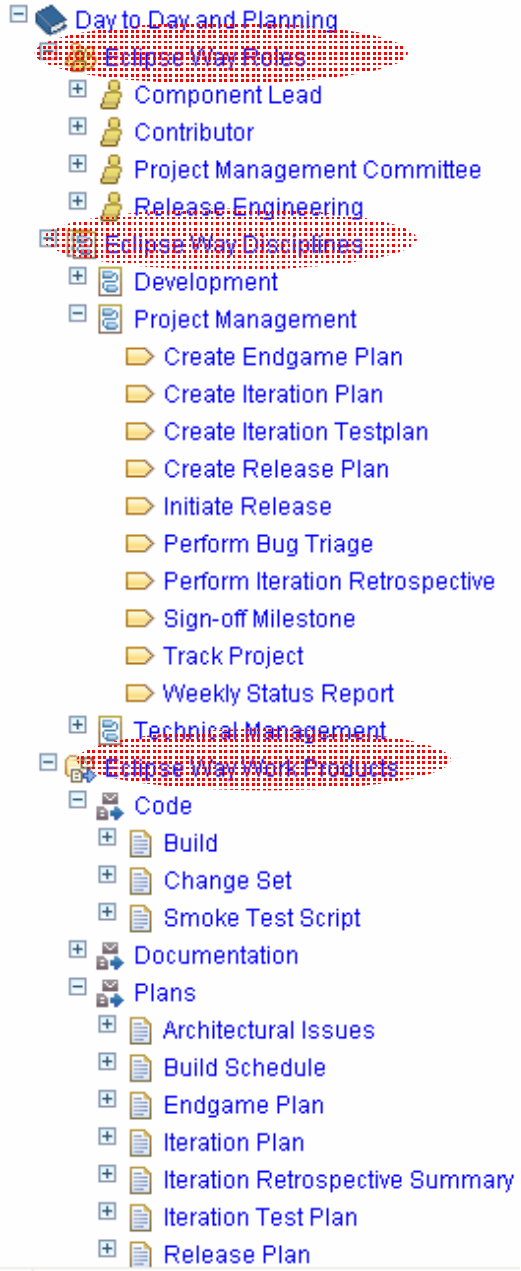


Describing the Eclipse Way

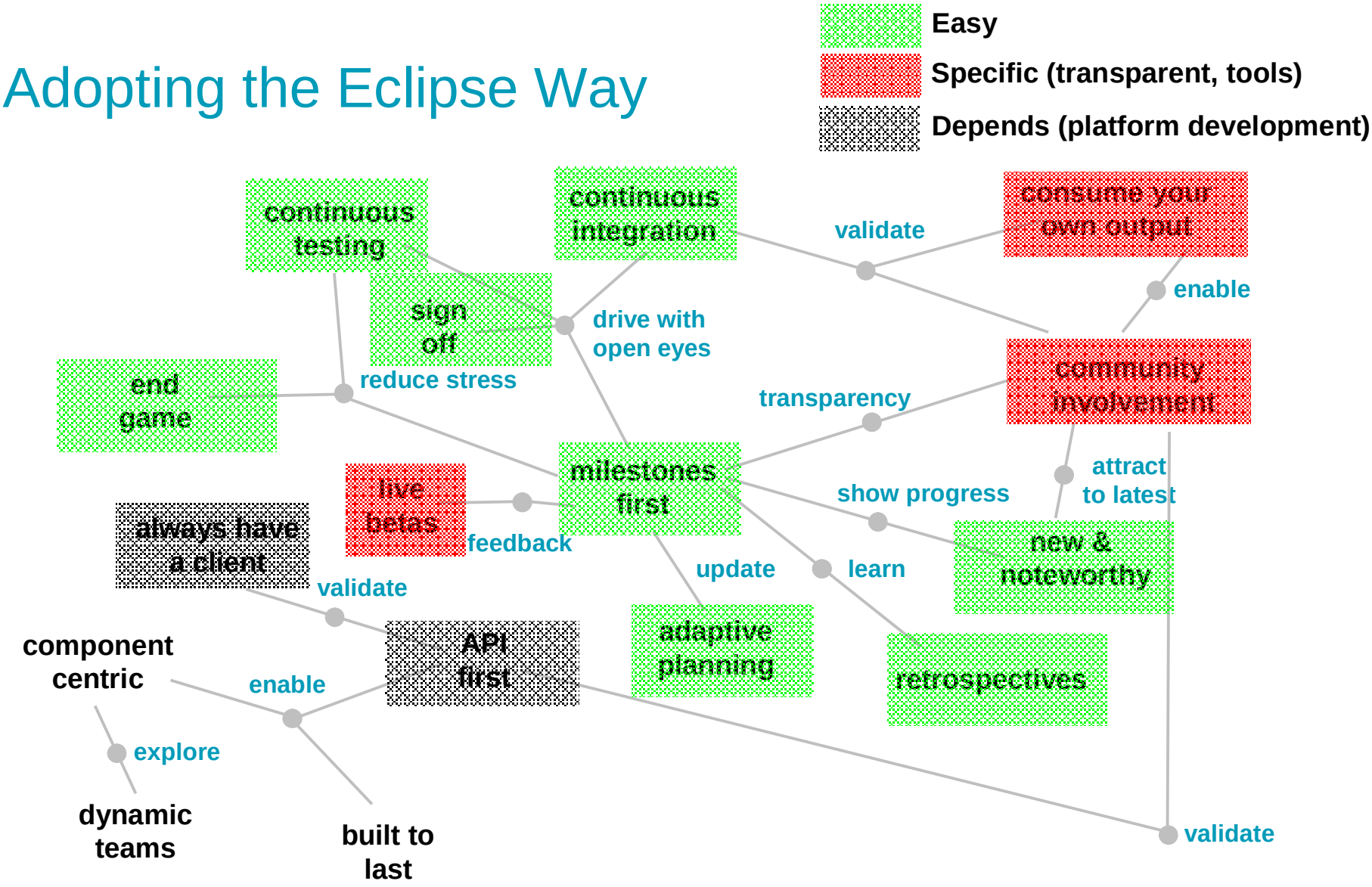
- Support learning the eclipse way
  - values
  - practices
- Minimal
  - focus on specifics
  - as concrete as possible

Ø Use RMC/EPF





Adopting the Eclipse Way



Martin Lippert: *The Eclipse Way - Adopting the Process*, JAX 2007

Is a Process Description Sufficient?

- Nope!
  - …but it can help new team members
- Existing team members read only minimal documentation
  - they want support to apply a practice not read about it
  - help with the boring stuff so that developer can focus on the creative challenging stuff
  - help with the checking and fixing

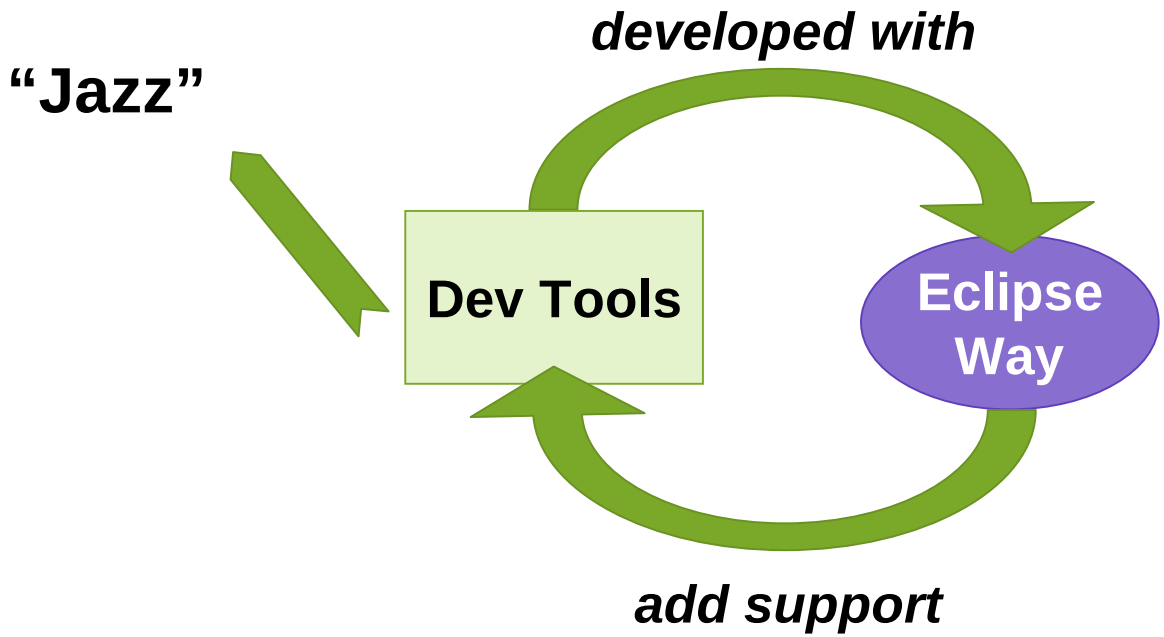
Ø **Practice** and **process aware** tools

The Agile Manifesto

http://agilemanifesto.org

Tooling the Eclipse Way

§ Iterative improvement, adoption



From individual productivity to team productivity

An analogy…

- Refactoring
  - Helps with the **boring** steps
    - semantic preserving changes
  - Empowers developer ⇒ **individual** productivity

- Process awareness
  - Helps with **boring** steps
    - follow team's process
    - checking and fixing
  - Empowers team ⇒ **team** productivity

Our Pain Points…

- ü joining a team
- ü get my environment configured to be productive
- ü what is happening in my team
- ü collecting progress status
- ü following the team's process
- ü ad hoc collaboration/sharing of changes
- ü starting an ad hoc team

Team awareness

- ü is the fix in the build?
- ü run a personal build
- ü tracking a broken build
- ü why is this change in the build?
- ü reconstructing a context for a bug/build failure

Build awareness

- ü interrupting development due to a high priority bug fix
- ü working on multiple releases concurrently
- ü tracking the code review of a fix
- ü referencing team artifacts in discussions
- ü how healthy is a component?
- ü collecting project data/metrics?
- ü keeping plans up to date

Project awareness

Boring and painful

Why “Jazz”?

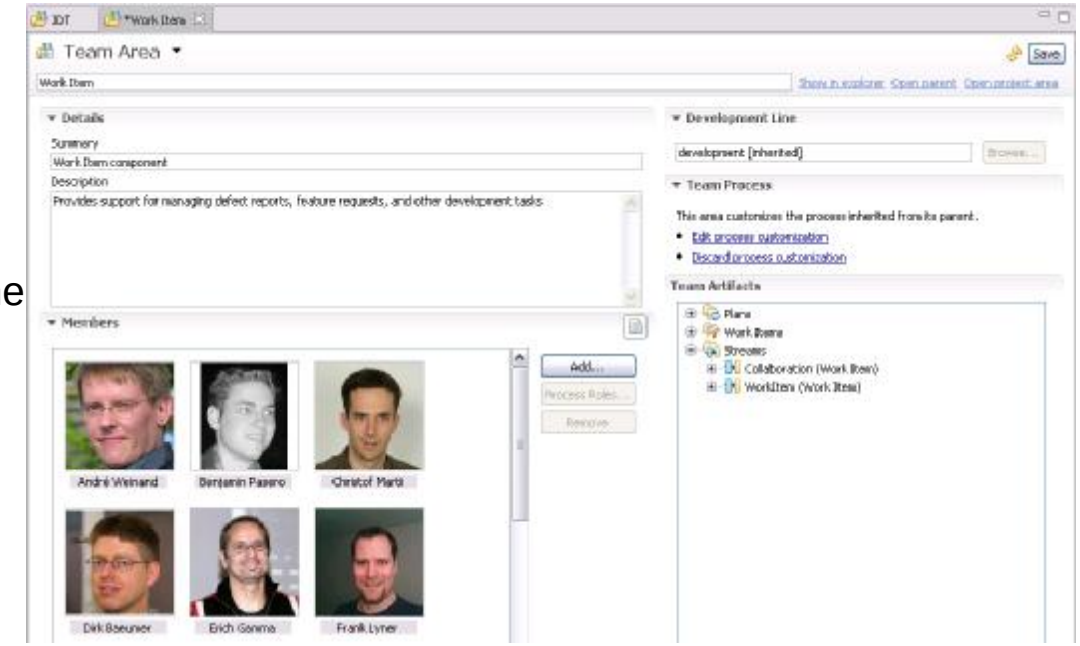
- Software development on a brightly lit stage
- Like playing in an orchestra
- Everyone knows what everyone else is doing.

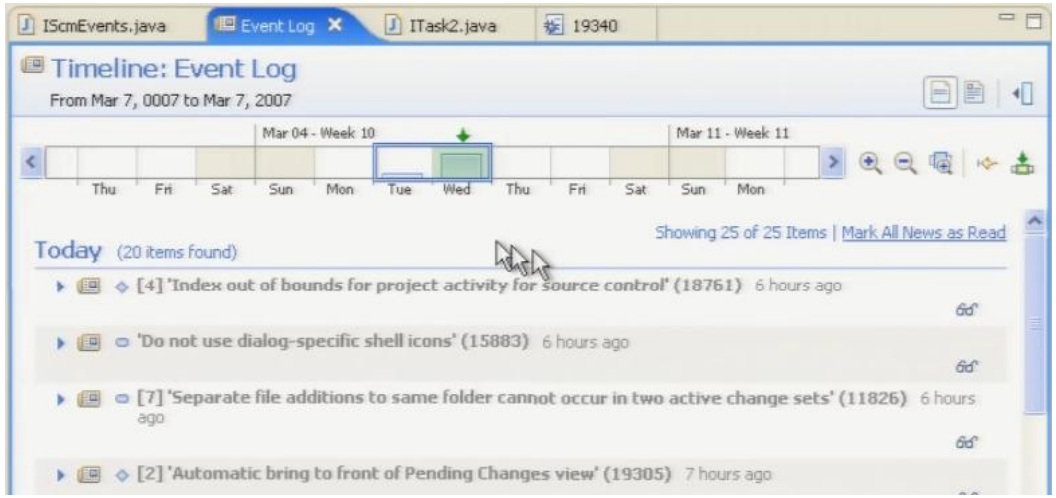
Key Themes

- Collaboration
  - Team first
- Process awareness
- Transparency

Team First

- Explicit team structure and roles
  - Contributor can belong to more than one team
- Team awareness
  - event feeds per team
  - chat in context
  - presence indication
- Team information at your finger tips
- Team autonomy
  - team owns its process
  - team owns components
  - team has its plans
  - team has its build

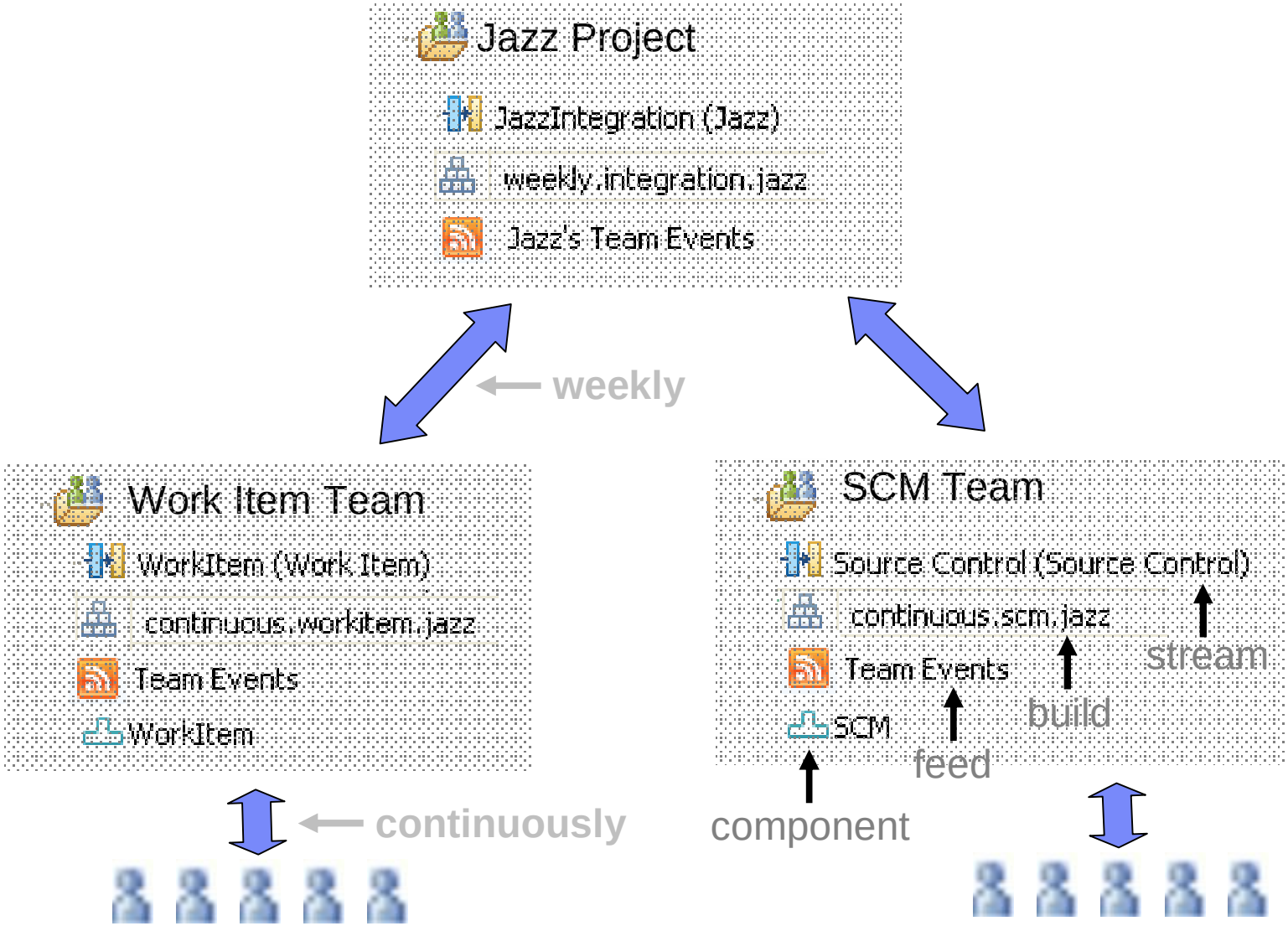




Tool support for Scaling up Agility

- First: working in the **big soup**
  - problems:
    - private worlds for developers only ⇒ not for teams
    - only product builds but no builds for an individual team
  - Larger teams must take more care ⇒ exceptional events become commonplace
- Then: **component-centered** design
  - Architectural approach to improve development effectiveness
- Next: **team-aware tooling**
  - Tools understand component - support collaborating on components
    - private worlds for teams, dynamic teams, and individuals
    - builds per team
  - Tools approach to improve a team's effectiveness collaborating on components

Team First: Scaling Up – Teams of Teams



Process/Practice Awareness

- Support many **different** practices and processes
- Each team has its **own dynamic**
  - Team owns its process and practices
- Tool **understands** how a team works
  - A project is associated with a process
  - Process controls how the tool behaves
  - Reduce team member mistakes
  - Not only establish rules, but help follow them
- Process must be highly **tweakable** to be **tolerable**
- **Jazz is Process neutral**

Process Customization

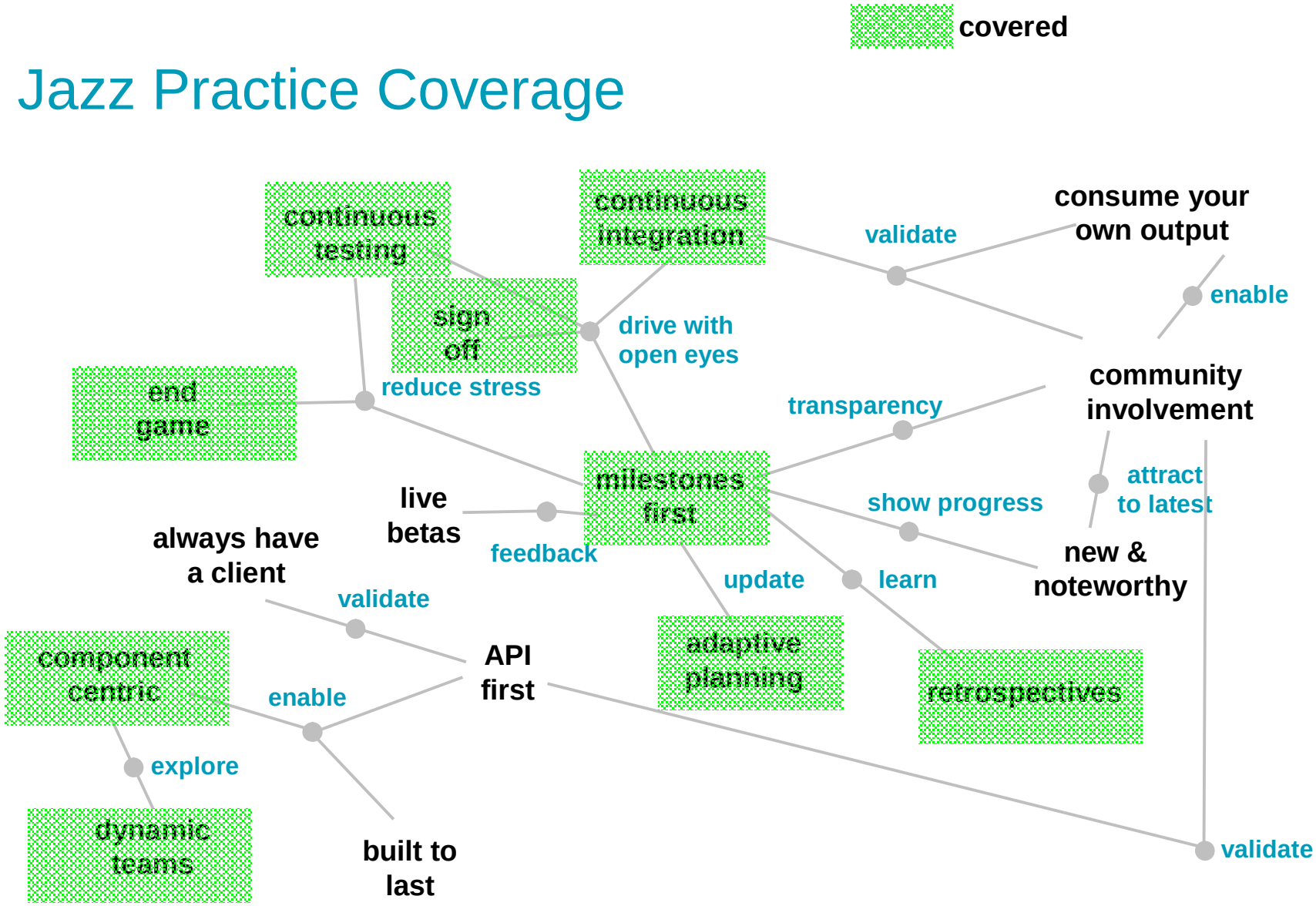
- Configuration data
  - work item types
  - work flow
- Operation Advisors
  - changes associated with work item
  - changes reviewed
- Events
  - work item changed state
  - build failed

Transparency

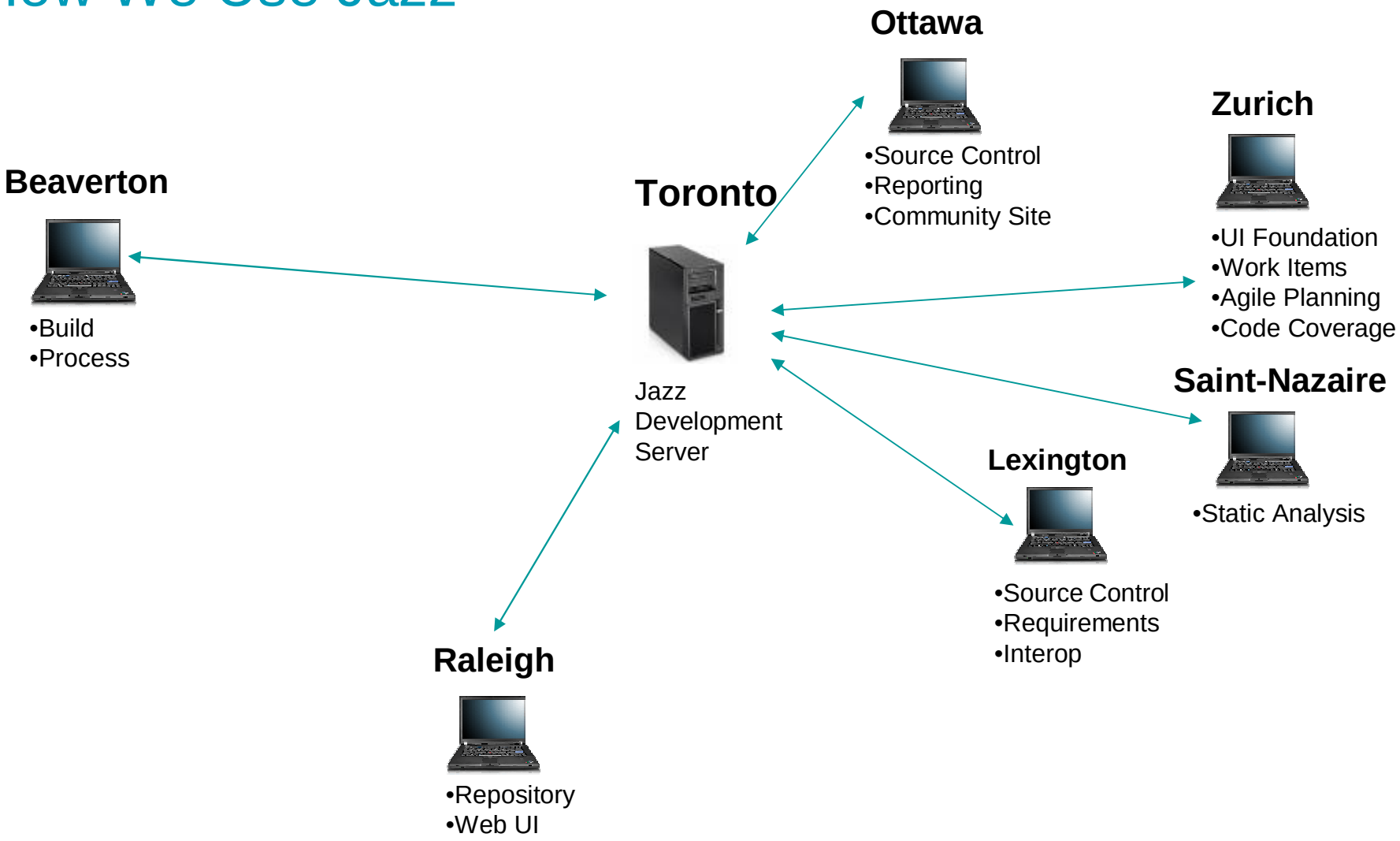
- transparency in planning
  - dynamic plans
- transparency in development
  - automatic linking
  - build results/reports
  - project health
- transparency in the end game
  - code reviews
  - verification
- transparency in process
  - team structure
  - team roles

What keeps me **Rational**?

Jazz Practice Coverage



How We Use Jazz



How Has Jazz Changed Us

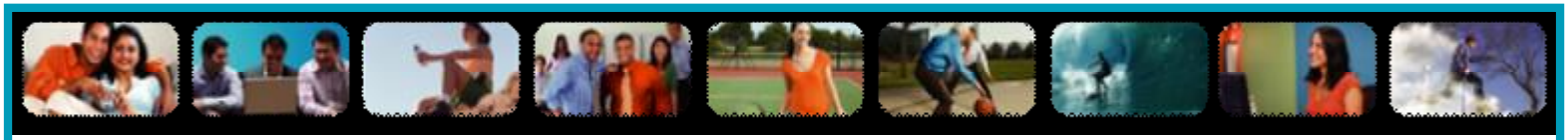
- We have become **more continuous**
  - If it is **simple** to do ⇒ do it more **frequently** ⇒ do it **continuously**
    - Initiate a build, private builds, exchange a change set
    - Coverage, static analysis ⇒ you do not want to run this on the desktop
- Things important to us we make **explicit**
  - custom work item types (RFS, plan items)
- Increased **awareness**
  - Linking, build awareness, team awareness
- Improved **practices**
  - Retrospectives ⇒ leverage project health data
  - Testing ⇒ coverage

Ø **Raised** our **expectations** with regard to tool support

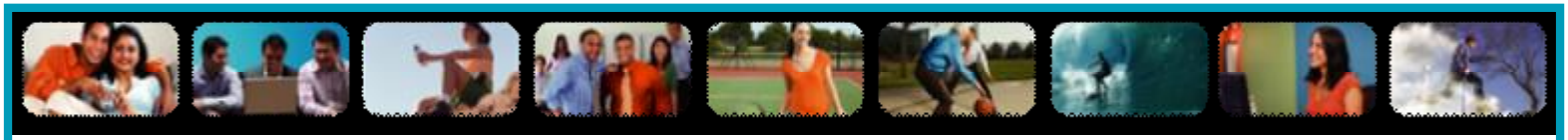
Summary

- **Absorb** practices that work
  - Don't overwhelm all at once
- **Tune** Jazz to your practices
- **Your** process, **your** practices, **your** work style, **your** team, **your** culture

➢ **Your Way**



Questions



Thank You