

Distributed Development across the Lifecycle with Jazz™

Harold Ossher

IBM Thomas J. Watson Research Center

Jazz is a trademark of IBM Corp. in the US and other countries.

Contents

- Jazz
- Requirements in distributed projects
- Jazz demo
 - with an emphasis on support for requirements engineering

Jazz

- Joint project: IBM Rational and Research
- Team Collaboration Platform
 - Support software development teams in everything they do
 - Eclipse focuses on individual developers
 - Jazz focuses on teams.
 - Make team development fun
 - Across the software lifecycle
 - Full spectrum of stakeholders and artifacts
 - *Artifacts stored in a common repository*

jazz.net

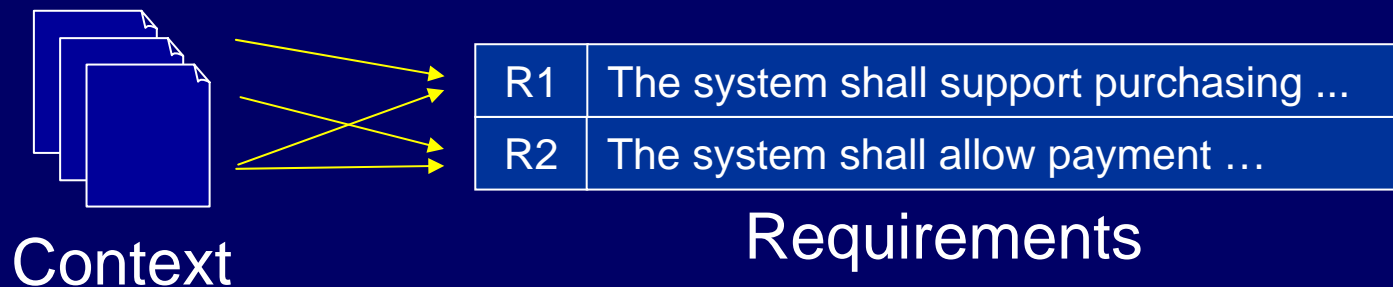
Jazz

- Includes:
 - Repository, links, process guidance
 - SCM, work items, planning, project health measurement and visualization
 - Team awareness, chat, screen sharing, ...
 - Requirements, Java coding and testing, build, ...
- Open platform
 - Open points for new tools and extensions
 - *Open commercial software*

jazz.net

Requirements in Context

- RE is inherently collaborative
- “Requirements are useless, but the requirements-engineering process is invaluable!”
- Can’t make requirements completely freestanding
 - Must depend on *context*
 - Stakeholder conversations
 - Source documents with embedded ideas
- *Harder in distributed projects* → challenge for tools



Requirements Downstream

- Requirements are needed downstream
 - Project managers doing planning
 - Developers implementing and testing
 - ...
- But requirements usually are not:
 - Integrated with their tools
 - Tied to the artifacts they work with
 - Tied to tasks being performed
- Need to be made *visible* and *relevant*

Agile

- Stories/use cases/scenarios are key, but:
- Agile projects seldom use requirements tools
 - Why not?
- Process focused on production of quality code
- Hypothesis: Support for requirements that is
 - Light-weight
 - Well integrated with process and tools
 - Makes requirements artifacts structured enough to be managed like bugs and milestone plans

will help them to produce quality code that meets the right requirements, even in distributed projects

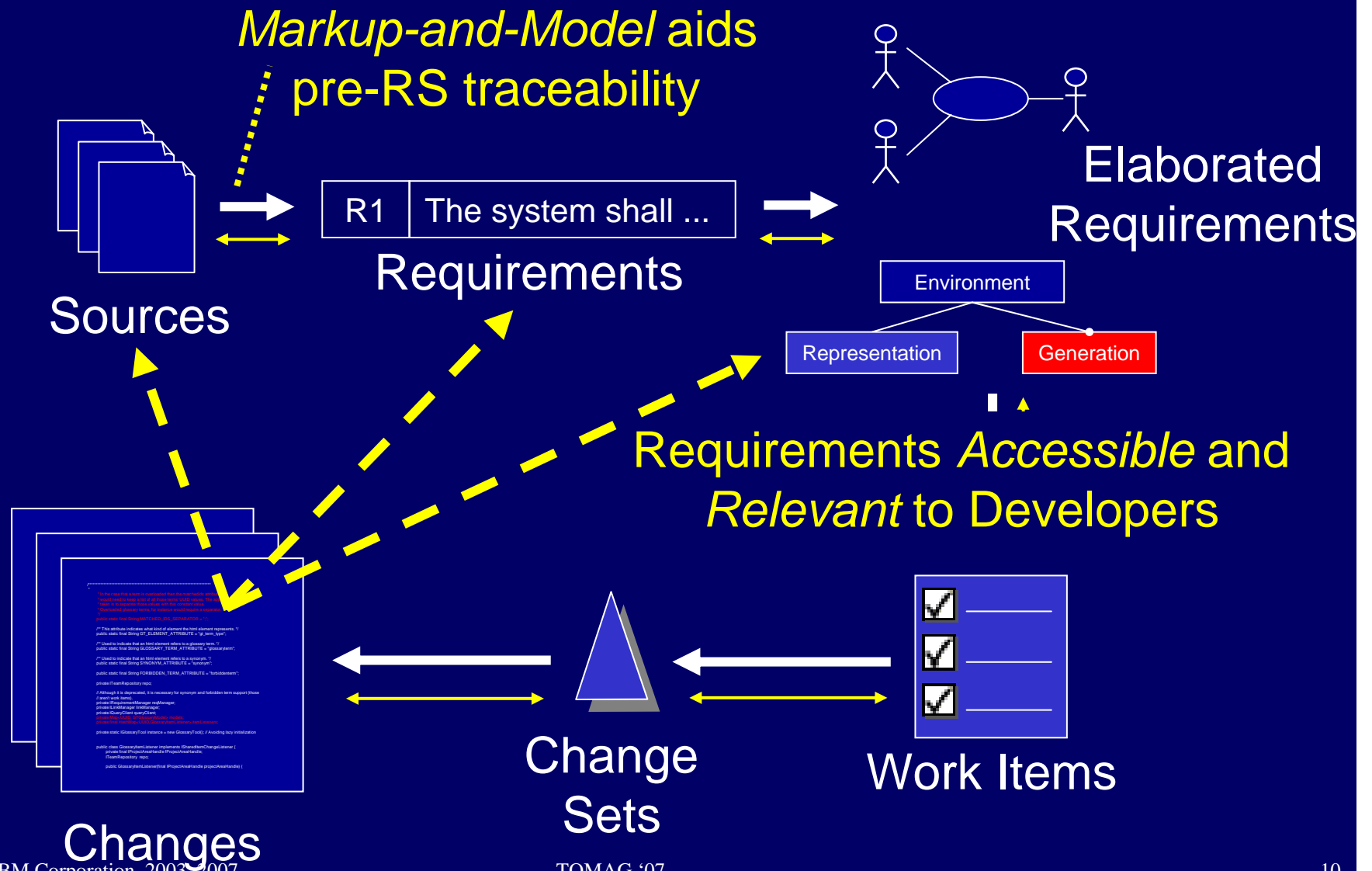
Task-Based Traceability

- Need traceability without manual maintenance
- Many development tasks involve related artifacts
- Tools can infer some traceability links during “normal development”
- Tasks are often managed using *work items*
 - E.g., open source projects using Bugzilla
 - Can aid in recording traceability links

Demo

See <http://jazz.net/pub/learn/videos/videos.jsp>
for a selection of Jazz demo videos.

Task-Based Traceability



Thank you!

Questions?