Distributed Team Building

Andreas Gies
Principal Architect
http://www.fusesource.com
http://open-source-adventures.blogspot.com

About the Author

- Principal Architect
  PROGRESS - Open Source Center of Competence
- Degree in Computer Science from the University of the German Forces 1992
- Working with middleware like MOMs, CORBA, J2EE, WS and ESBs ever since for Sterling Software, Iona Technologies and PROGRESS
- Specialized on ESB based architectures since 2002

2

About FUSE

- The examples are based on the FUSE releases of Apache ServiceMix and Apache ActiveMQ
- The FUSE community provides
  - **Access to Committers** as many Apache commiters are employed by the FUSE team
  - **Enterprise support** – Open source adaption in the enterprise requires 24x7 reliable support
  - **Increased testing** on a CI environment maintained by the FUSE team
  - **Enterprise qualities of service** – Ensuring sensible Enterprise deployment and backwards compatibility
  - **Documentation and training** for the Apache projects released under the FUSE brand
  - **Backed by large, enterprise company**

3

FUSE products

- FUSE ESB 3
  Based on Apache Service Mix 3
- FUSE ESB 4
  Based on Apache Service Mix 4
- FUSE Message Broker
  Based on Apache ActiveMQ
- FUSE Services Framework
  Based on Apache CXF
- FUSE Mediation Router
  Based on Apache Camel
- FUSE Integration Designer
  Eclipse tooling for implementing EIP flows
- FUSE HQ
  Management and Monitoring of the FUSE infrastructure

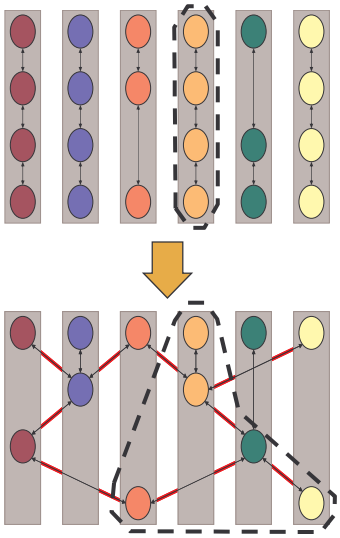
4

Agenda

- A closer look at SOA applications
- Platform components
- Tools enforcing project standards
- Project Lifecycle
- Conclusion

5

Impacts of SOA on development



Silos (with only data integration)

- Each business process is self-contained in a single silo
  - One team has end-to-end responsibility
  - Infrastructure easily mapped to business
  - Policies can apply to the silo only
    - Security, compliance, visibility, etc.
  - App decisions have local impact only
    - Middleware, data formats, standards, etc.

SOA / Shared Services

- Business processes span "silos"
  - ⚠ No one team has end-to-end responsibility or visibility
  - ⚠ Business processes have no direct mapping to silos
    - The same service may serve multiple different business processes
  - ⚠ Policies need to apply to entire process
  - ⚠ App decisions have global impact

6

Definitions: Business application

- A Business application is a collection of modules that solves a given business problem.
- It is:
  - Versioned
  - Documented
  - Tailorable to different runtime environments
  - Specified in terms of business requirements
- A Business application is composed of **Services**
- Examples:
  - Equity Management in Finance
  - Service Provisioning in TelCo
  - Complaint Management in Pharma

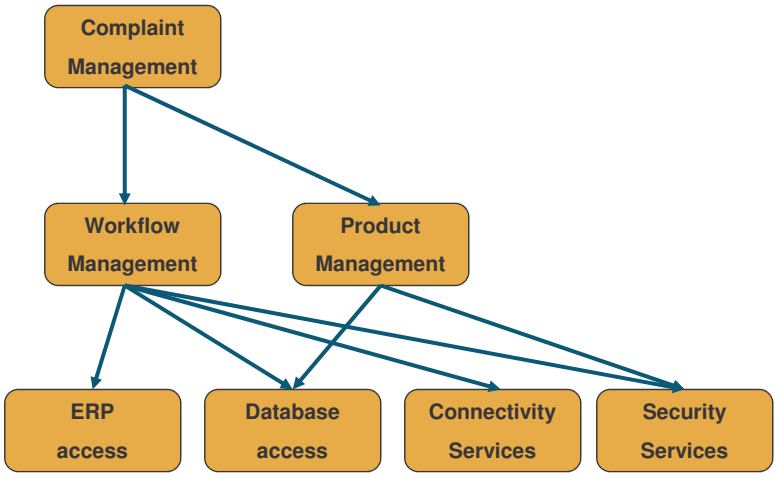
7

Definitions: Service

- A **Service** has a well defined interface and encapsulates a piece of application logic or hides the complexity of a technology used
- It is:
  - Versioned
  - Documented
  - Tailorable to different runtime environments
  - Well specified in terms of interfaces
  - Free of any side effects
- A Service may be stand-alone or be composed of other services
- Examples
  - Sonic ESB® Services
  - Sonic ESB Generic Processes
  - Backend Adapter
  - Customer Database access logic

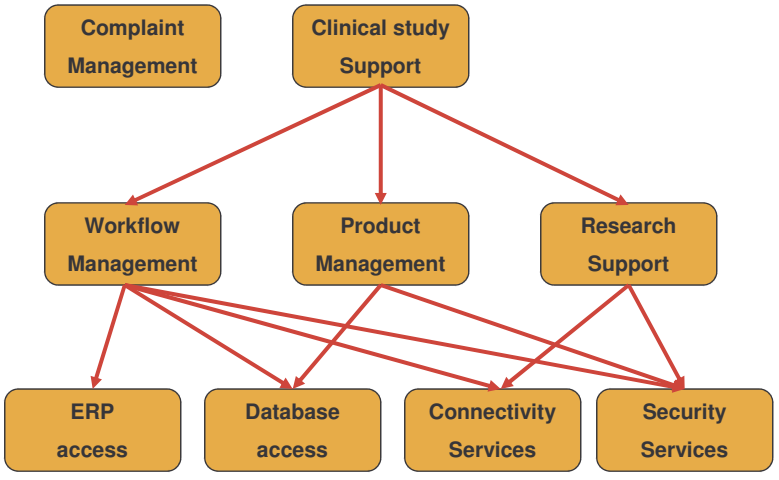
8

An Example

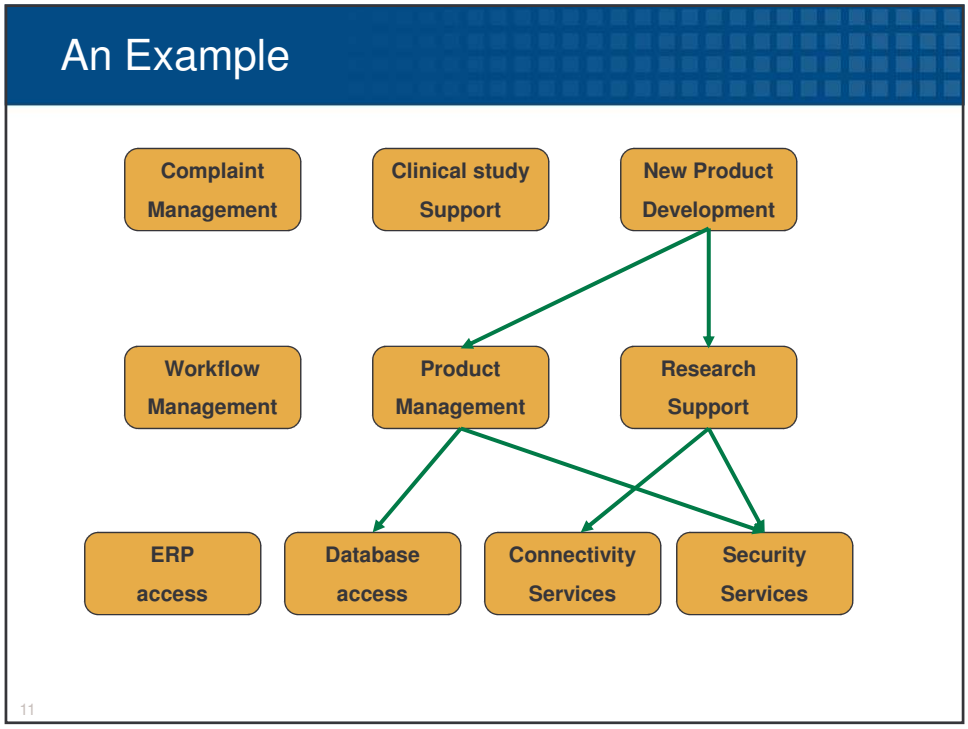


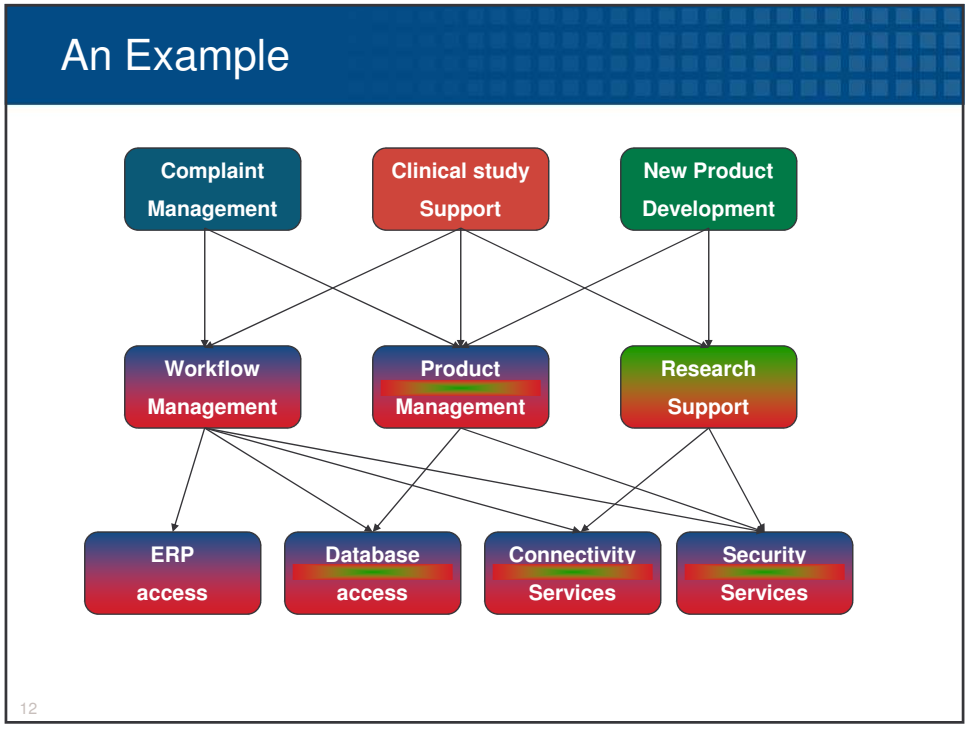
9

An Example

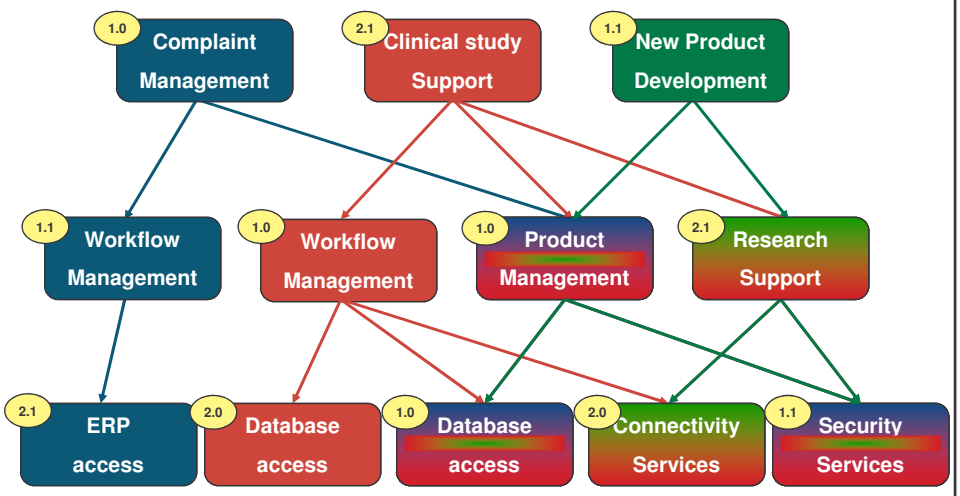


10


An Example
Complaint Management
Clinical study Support
New Product Development
Workflow Management
Product Management
Research Support
ERP access
Database access
Connectivity Services
Security Services
11


An Example
Complaint Management
Clinical study Support
New Product Development
Workflow Management
Product Management
Research Support
ERP access
Database access
Connectivity Services
Security Services
12

Adding versioning to the challenge



13

Team challenges in SOA applications

- Requirements change very fast
- Multiple development teams may exist due to acquisitions or mergers
- Language barriers as teams might be multinational
- Knowledge distribution – not all team members on the same skill level
- Missing trust in each others competences
- Increased overhead handing over components into QA or production

14

Addressing the challenges

- A well defined and distributed development process needs to be established
  - Use a well defined set of tools
  - Define project standards that make the teams life easier rather than hardee
- Encourage regular interaction as early as possible
  - Virtual team meetings (IRC, Webmeetings etc.)
  - Knowledge sharing platforms
- Encourage collaboration rather than competition
- Take different mentalities into account
- Make everyone in the team know his/her function

- A proper development platform can address the technical aspects of a distributed team

15

Requirements development platform

- Services shall be reusable in different business applications
- The developers should be focused on developing their service(s)
- Unit Testing, Integration testing must be part of the development cycle
- Knowledge sharing must be inherent to the proposed solution
- Dependency management must be integral part of the solution
- The packaging process must resolve versioned dependencies.
- The software packages shall be built and tested regularly and automatically using a continuous integration server

16

Continuous Integration

- **Continuous integration** aims to speed up the software delivery by decreasing integration times
- A **code repository** allows multiple developers to work on the same project
- **Build automation** reduces the time to build the software for testing purposes
- **Test automation** allows tests to be run as part of the build process for immediate feedback
- **Automated deployment** enables the staging of the software in Test-, Integration- and Production environments

17

A Continuous Integration platform for SOA

- A CIP provides a version control and dependency management facility for the services
- It also supports build, test and integration automation
- It gives the developer immediate feedback about any issues encountered due to module dependencies
- It provides an automated packaging and distribution mechanism for binaries and documentation

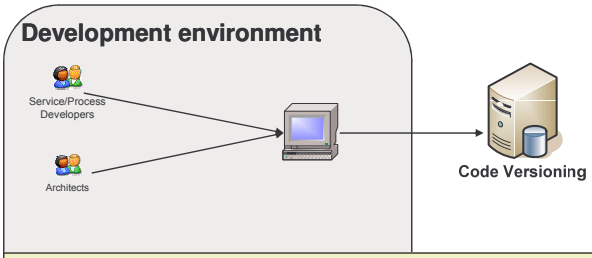
18

Agenda

- A closer look at SOA applications
- Platform components
- Tools enforcing project standards
- Project Lifecycle
- Conclusion

19

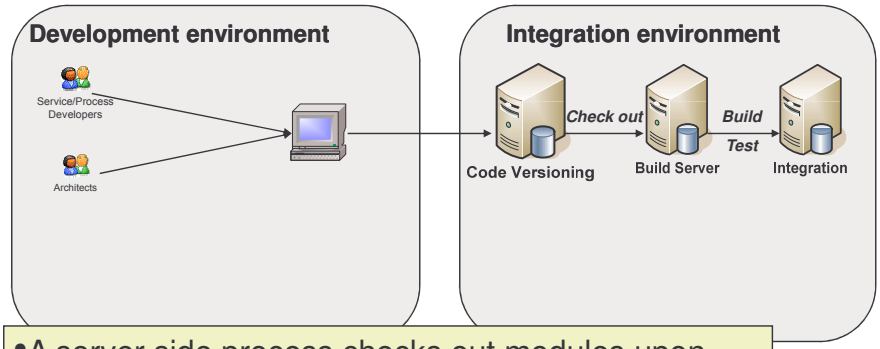
Continuous Integration Platform: Developer



- **The developers and architects work with their usual environment.**
- **Version Control is essential for working in teams.**
- **Unit tests are a mandatory deliverable for each service.**
- **The test framework is usually integrated in the workbench.**
- **Each service must support a headless build.**

20

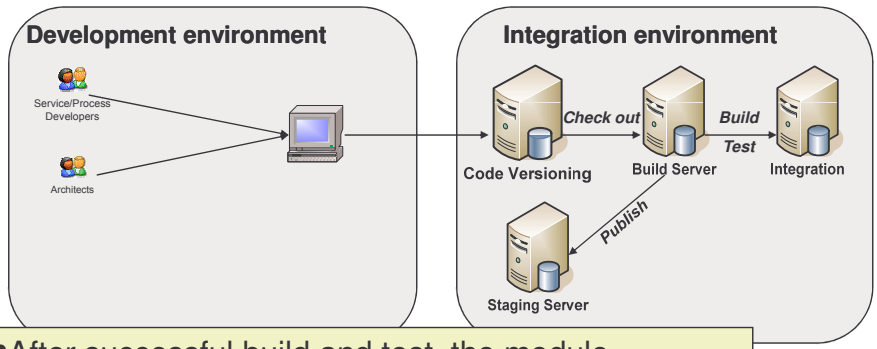
Continuous Integration Platform: Build Server



- A server side process checks out modules upon modification and validates dependencies.
- The changed modules are rebuilt and retested using the unit and integration test specifications.

21

Continuous Integration Platform: Staging



- After successful build and test, the module documentation is published automatically.
- After successful build and test, the module package is published via a module repository.

22

Continuous Integration Platform: QA

- QA can check-out published modules to test
  - Deployment
  - Performance
  - Functionality (in addition to automated tests)
  - Documentation (as published on the Web Server)
  - Problems are handled via the Problem Management process

QA environment

23

Continuous Integration Platform: Production

Development environment

Integration environment

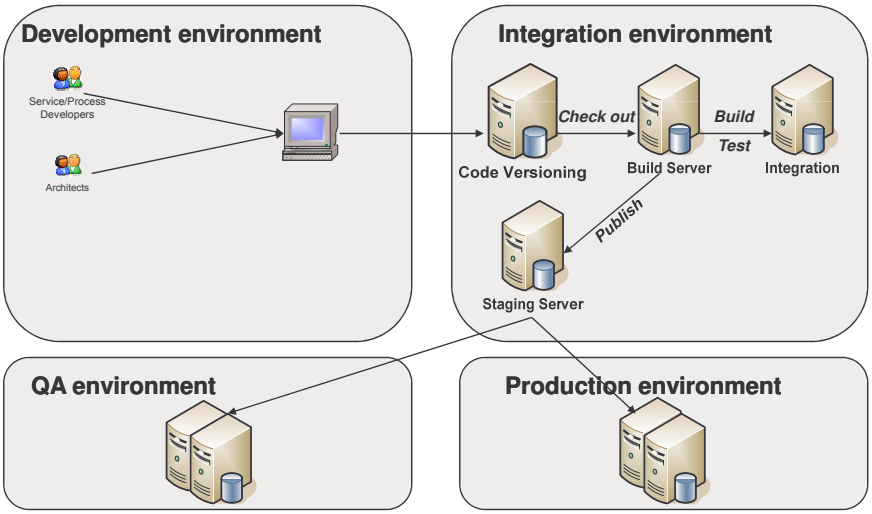
- Release Management can download tested modules and perform the release according to the release procedure
  - Burn the „Golden image“
  - Deploy the application into the productive environment

QA environment

Production environment

24

Continuous Integration Platform: Production



25

Does a build platform address the problems

- The platform allows all team members o work on any component independently of their location
- Test, documentation and development teams work collaboratively on the same artifacts
- Test results, documentation, source code and development metrics are available after each automated build
- Automated build process requires project standards to be set
- A defined project life cycle is required and must be communicated to all team members

26

Agenda

- A closer look at SOA applications
- Platform components
- Tools enforcing project standards
- Project Lifecycle
- Conclusion

27

Project Structuring

http://maven.apache.org/

- Common project elements and build rules in a common master model
  - Version control location
  - Web page location
  - Project infrastructure
- Loosely coupled projects with up-to-date dependencies
  - Repository based build platform
  - Distributed repositories possible

28

Version Control

http://subversion.tigris.org/

- Open Source Version Control System
- Allows also versioning of directories (as opposed to CVS)
- Server available on Windows / Unix platforms
  - Leverages WebDAV protocol provided by Apache to enforce SSL and authentication
- Many clients available
  - Eclipse, Windows Explorer (Tortoise), WinSVN, command line, ANT, Maven etc.

29

Continuous Build component

http://continuum.apache.org

- Rebuilds and Retests registered projects upon committed changes
- Updates the Snapshot repository
- Rebuilds and Re-deploys the Project Web page
- Rebuilds and Retests project dependencies
- Notifies developers upon build errors to take corrective action

30

Documentation Elements

- Project related documentation in Maven format (apt, xdoc, …)
  - Apt is very easy for developers
- Javadoc
- Maven project reports
  - Test report
  - Checkstyle report
  - Test coverage report
  - Source reference

31

QA elements (1)

http://checkstyle.sourceforge.net/

- Checkstyle rules integrated in Maven 2
- Checkstyle violations reported in standard project documentation
  - Should be addressed in code maintenance
- Makes code exchangeable across team members
- Enforces Javadoc documentation
- Checkstyle checker available as Eclipse Plug-in

32

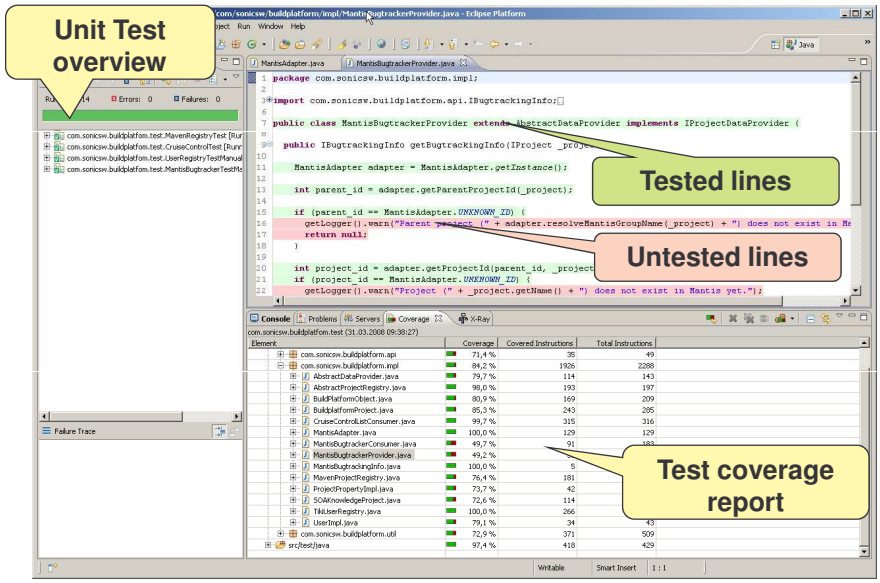
QA elements (2)

http://www.junit.org/index.htm

- Open Source Test Framework for Java™
  - De Facto Standard for Java Testing
- Supported in Eclipse
- Automatically executed by Maven 2
- Generated Test report on Project Web Page
- Test Coverage analysis in Maven 2 by cobertura (http://cobertura.sourceforge.net/)

33

QA integrated into development



34

Agenda

- A closer look at SOA applications
- Platform components
- Tools enforcing project standards
- Project Lifecycle
- Conclusion

35

Project Lifecycle

Development Phase

- Producing
  - Code artifacts including documentation
  - Unit test cases
  - Additional documentation
- Committing
  - Regularly to update Snapshot builds (Share early, Share often)
- Feature Driven
  - Working towards feature completeness before moving to RC1

36

Project Lifecycle

Review Phase (RC1)

- Ideally done by different person
  - Using the tagging mechanism to tag RC1
  - Review & Amend documentation (completeness, quality)
  - Review & Amend test cases (test coverage, execution)
- Commit / Merge changes back to Snapshot branch

37

Project Lifecycle

Release

- Use the tagging mechanism to tag final release
- Rebuild the new release and populate download page with release
- Publish Release Web Page
- Remove Release Version from Continuous build

38

Agenda

- A closer look at SOA applications
- Platform components
- Tools enforcing project standards
- Project Lifecycle
- Conclusion

39

Conclusion

- As SOA moves into practice, a build management system is essential to reliably build reliable Business Applications.
- A properly configured build management system must go hand-in-hand with the developer‘s mind set.
- The build management should impact the single developer only to a minimal degree in terms of effort and to a maximum degree in terms of benefits.
- Reusing versioned components is virtually impossible without a build management system.
- Reusing components grants the ROI for introducing a build management system.
- Built-In communication and sharing minimizes fear and distrust in distributed teams
- Virtual team meetings can be held using the information on the CIP

40

Conclusion ctd.

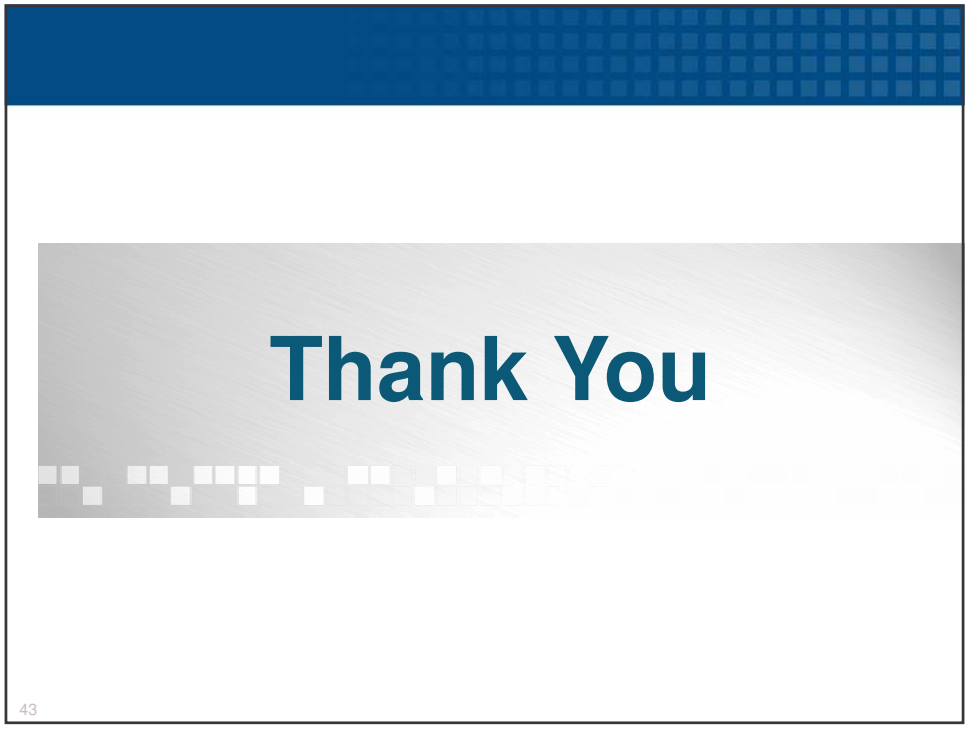
- Reuse of components due to the repository management of the build platform
- Better tested software due to module reusage and more test cases for more scenarios.
- Increased speed of development by standardized view of individual projects.
- Automated deployment into Q&A environments are achievable due to standardization

41

Questions ?

Come and talk to us at http://fusesource.com/forums/index.jspa

42


Thank You
43