Managing .NET Software Factory Using Task Based SCM tools

Krzysztof Kniaz

WeightWatchers.com

IT Architect Regional Conference 2008

Agenda

- Problem Background
- Software Factory Requirements
- SCM Process
- SCM and EAI
- Integration with QA life cycle
- Team metrics

Problem: change control for the large global app

- Application Name: WeightWatchers.com
- #1 Weight Management destination on the Internet
- Global presence (8 subscription sites – US, CA, UK, AU, DE, SE, Fr, NL +15 content sites)
- The product has mix of content and functionality
- Diet business is very dynamic; product and marketing teams need the ability to respond to change fast
- How to stay competitive while maintaining control over quality of the software?

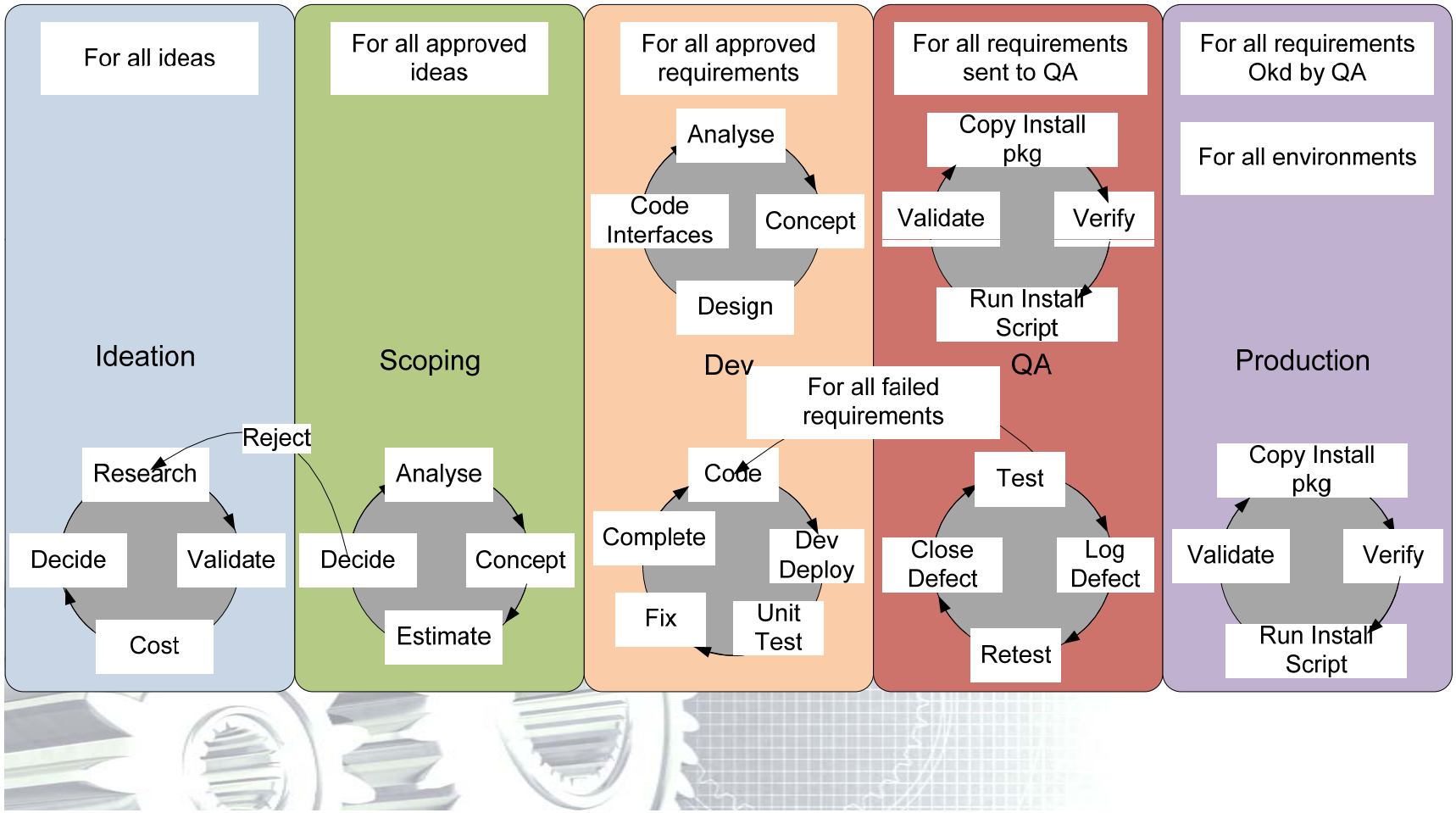
Problem background

- Most organizations start with rudimentary dev process
  - Symptom: Developers are permitted to deploy the app to QA and /or Production
  - Symptom: Handoff between the Dev and IT teams is manual
  - Symptom: The dev team can't identify production baseline
- Most organizations start with short term architecture
  - Symptom: Small changes require a release
  - Symptom: Initial Services (messages / schemas, operations) are always narrow in scope and functionality
  - Symptom: No ownership of the EAI/Feed Architecture
- Development Process and Architecture are usually interdependent so to maximize quality Architect needs to address both

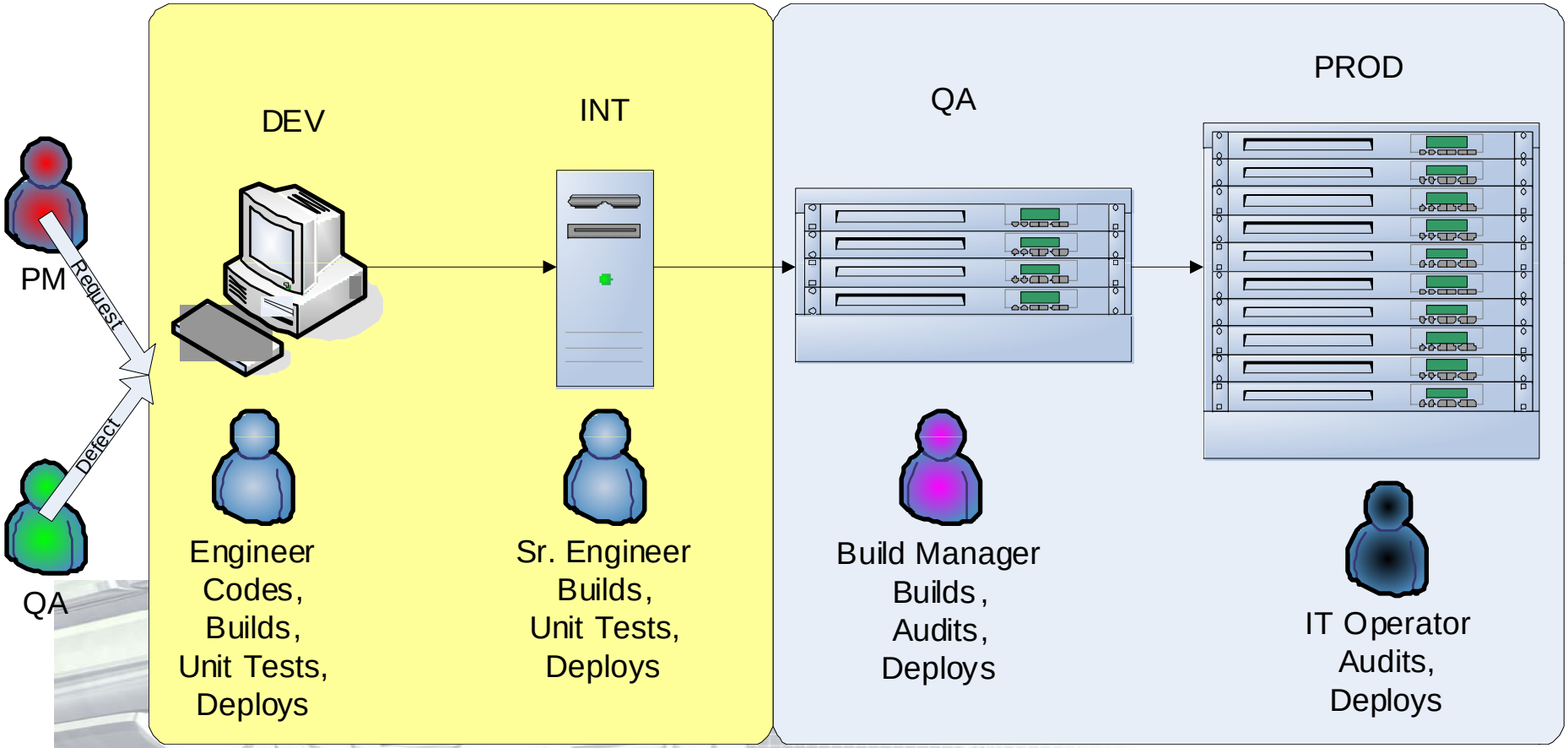
Process Roadmap

- Analyze processes inherent to all participants of the Software Development
- Refactor the app to reduce dependencies and for maximum flexibility
- Define Principles of the development process
- Select best tools and automate processes using the tools for each team
- Define interfaces and integrate lifecycles (e.g. Dev and QA teams)
- Define metrics and build process reporting

Process Analysis



Code and Product Flow



Software Factory Requirements

- Automation
  - Builds must be automated
  - Builds must run often
- Scale out (Lab to Pilot Plant to Production Facility)
  - Run the same build and deployment process on any environment (Developer Desktop, Integration Server, QA server)
    - UI and MW code as well as data base code
- Testable
  - Support automated unit tests

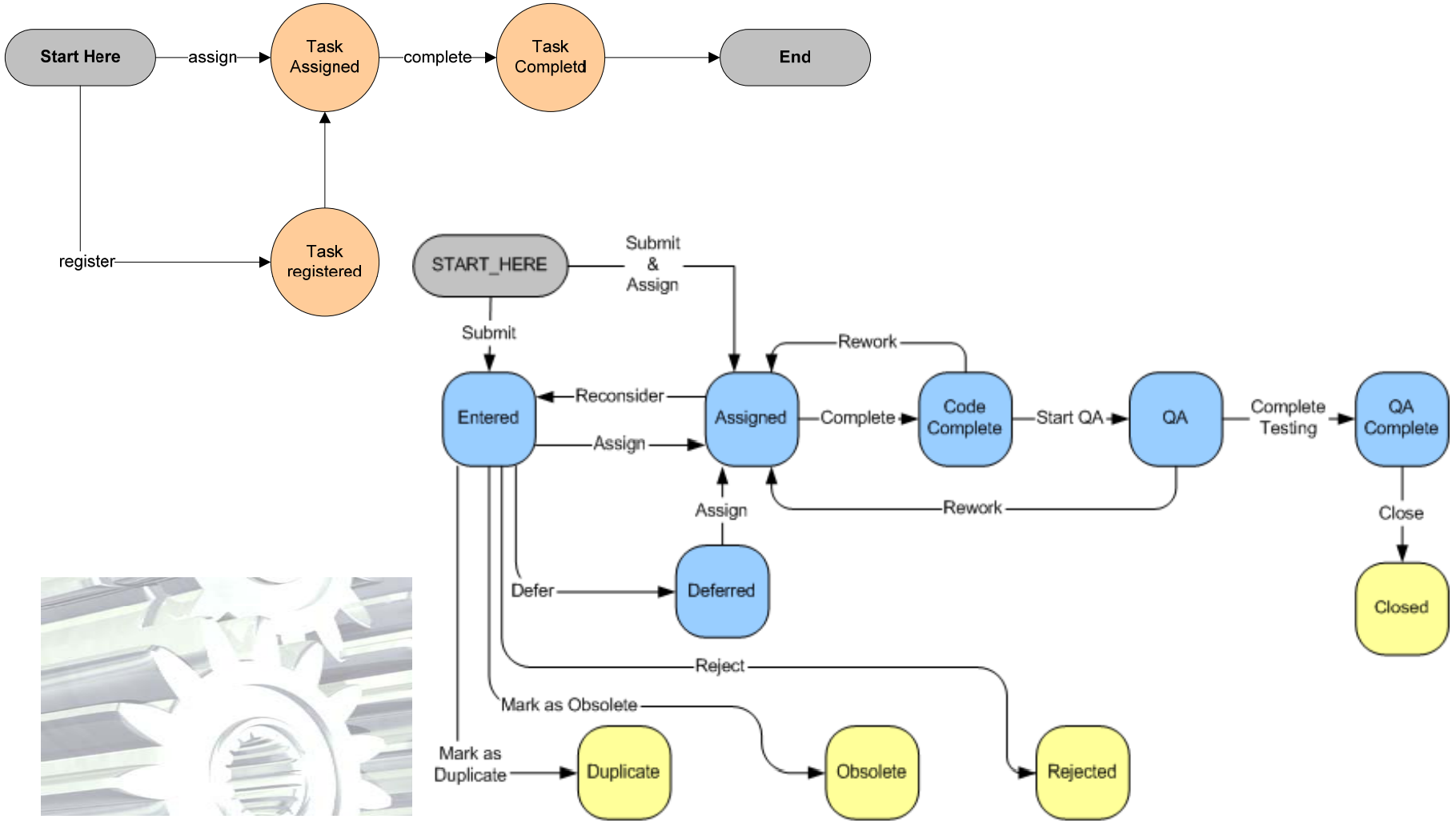
Software Factory Requirements

- Verifiable
  - Knowledge of what code (to the file level) was deployed to production in a given release
  - Trace code change to developer that changed it and change request or defect that precipitated the work.
    - Link binary code to the source code
- Support concurrent and parallel development
  - Concurrent: development of the major and minor release at the same time
  - Parallel: development of many major releases at the same time

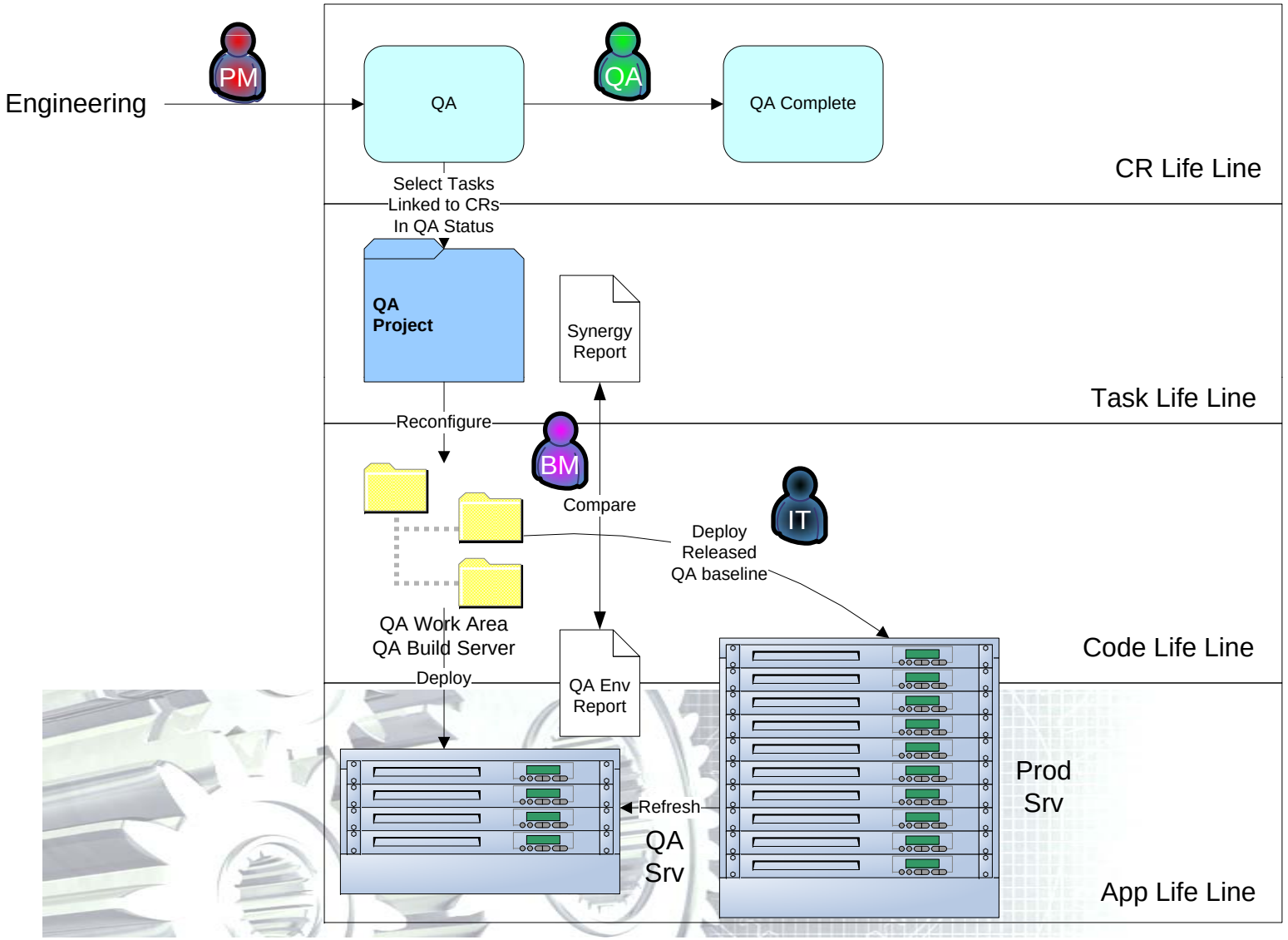
Tool Selection

- Portfolio and planning: Custom built db, Superdecisions for priority analysis
- Requirements : Custom built db, Excel for data entry
- Change: Synergy Change
- SCM: Synergy CM
- QA: Mercury QC
- Deployment: Blade Logic, NAnt scripts

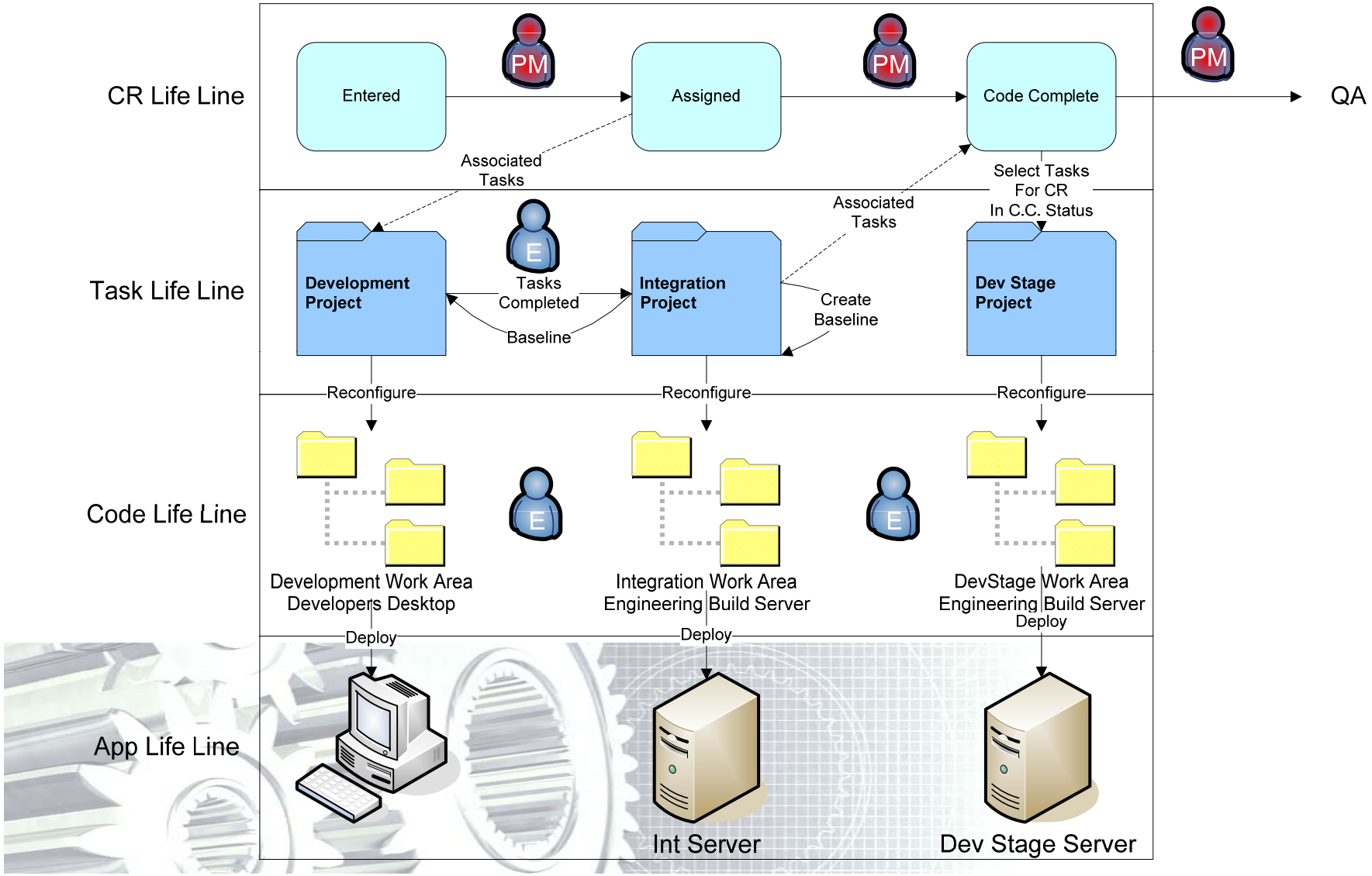
Change Request & Task Flows



QA and Prod Process



Engineering Process



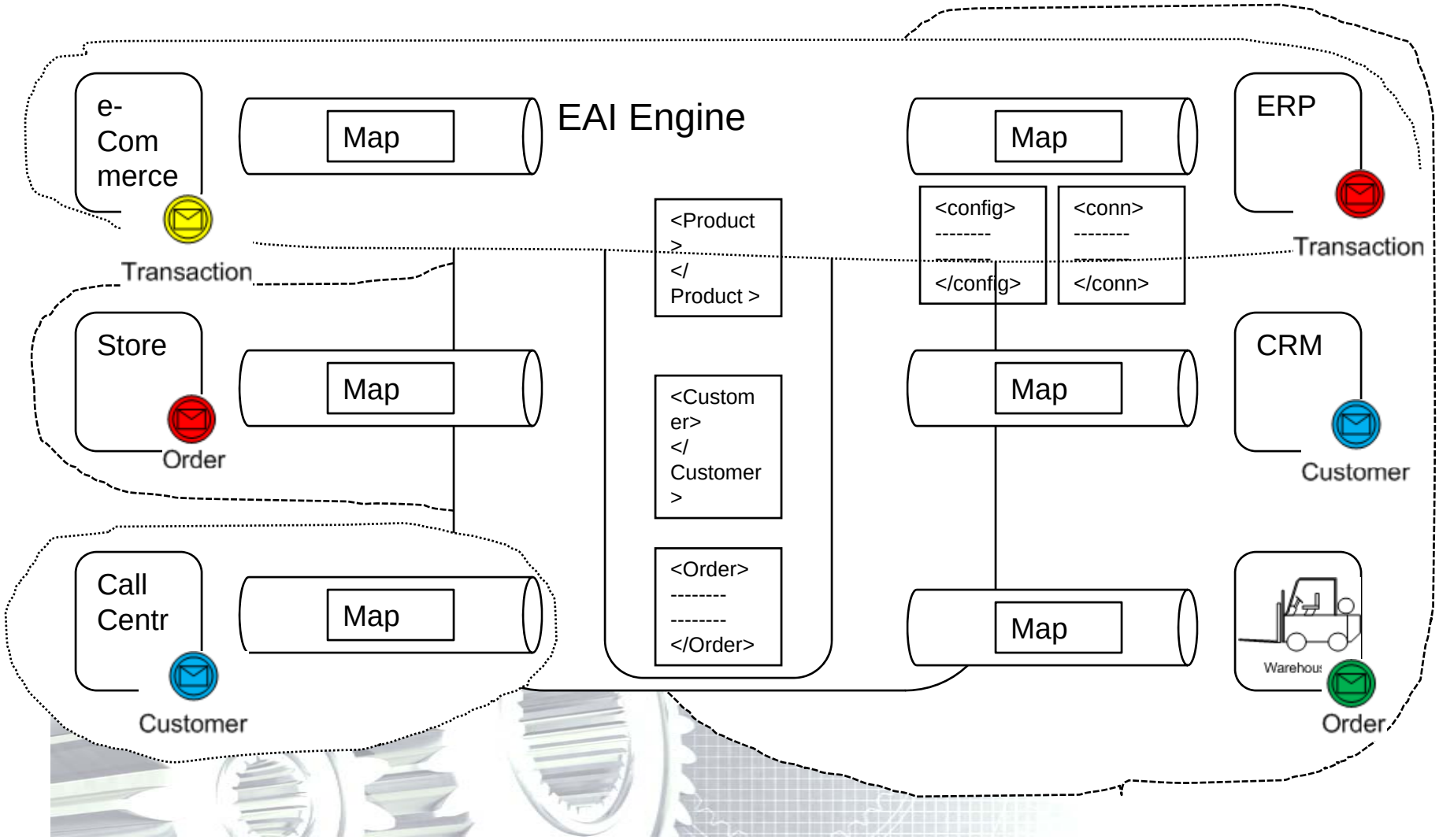
Automated Build Process

- NAnt scripts (http://nant.sourceforge.net)
- Re-runable NUnit tests (http://nunit.org)
- Continuous Integration Server: CruiseControl.NET
  - WW.com team wrote Synergy Plug-in for CruiseControl.NET (published Sep 2005)
- Environment (or locale) specific settings are stored in the configuration files that are dynamically created by the build process

Auditing

- Link binary products to source code
  - Select tool that has “Bill of Materials” (BOM) concept
  - If you can’t use the concept out of the box make sure your tool has an open interface so you can code to your requirements
  - In Synergy out of the box utility works only for C files compiled via make, not particularly useful for .NET files, assemblies, NAnt
- Compare deployed code to the released baseline
- Passed SOX audit for the build and deployment process

EAI and SCM

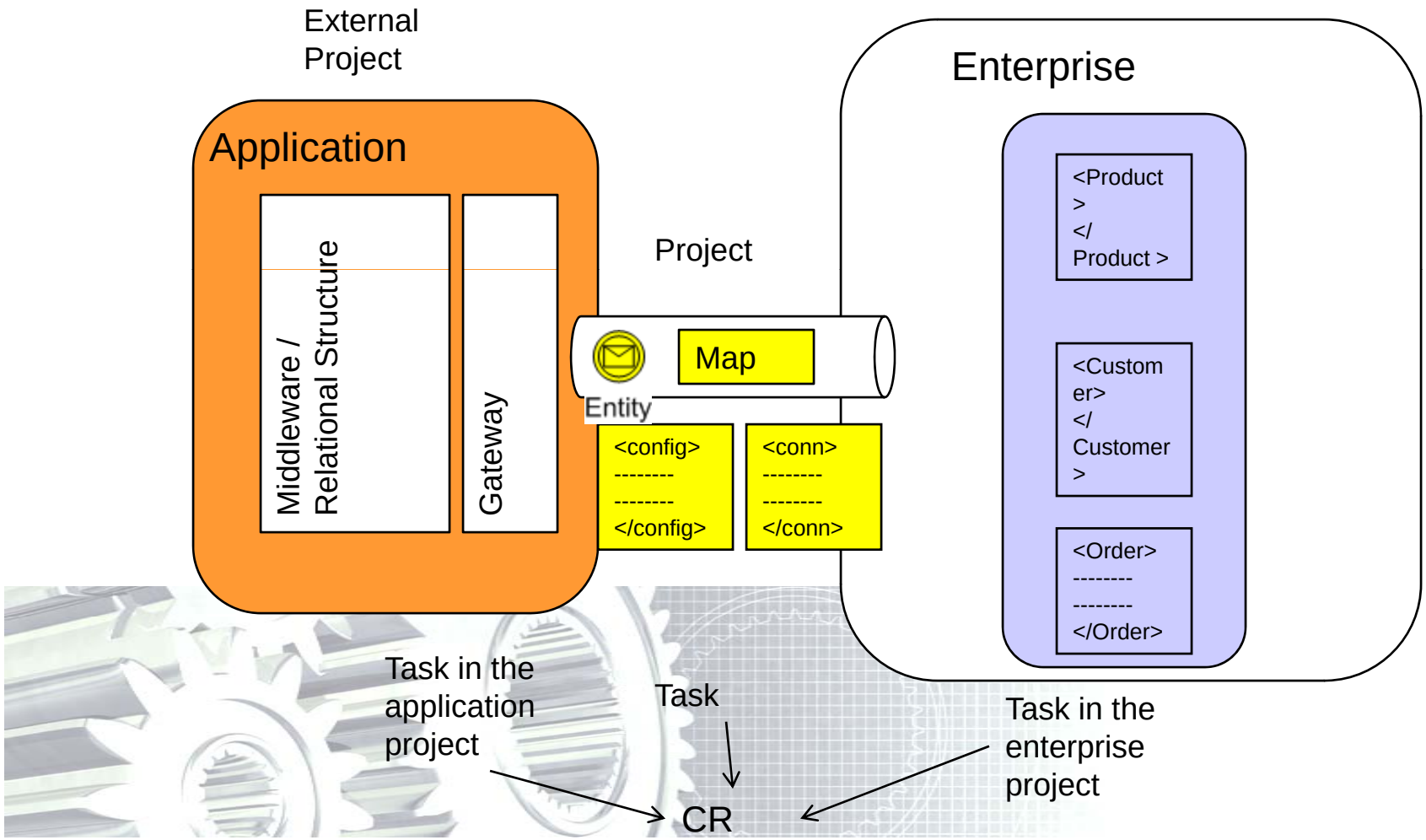

e-
Com
merce
Transaction
Map
EAI Engine
<Product
>
</
Product >
Map
ERP
Transaction
<config>

</config>
<conn>

</conn>
Store
Order
Map
<Custom
er>
</
Customer
>
Map
CRM
Customer
Call
Centr
Customer
Map
<Order>

</Order>
Map
Warehous
Order

EAI and SCM

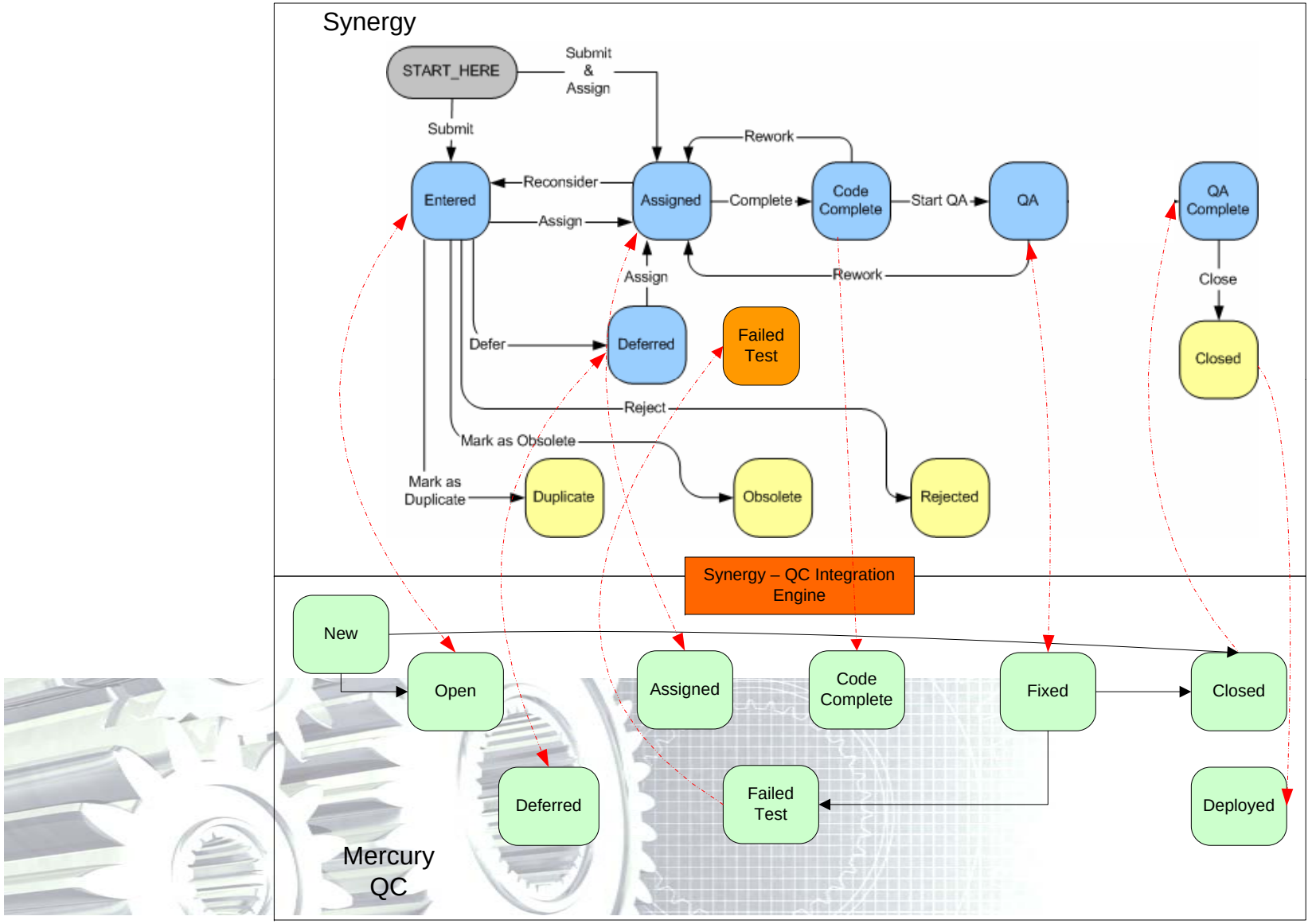

External
Project
Application
Middleware /
Relational Structure
Gateway
Project
Map
Entity
<config>

</config>
<conn>

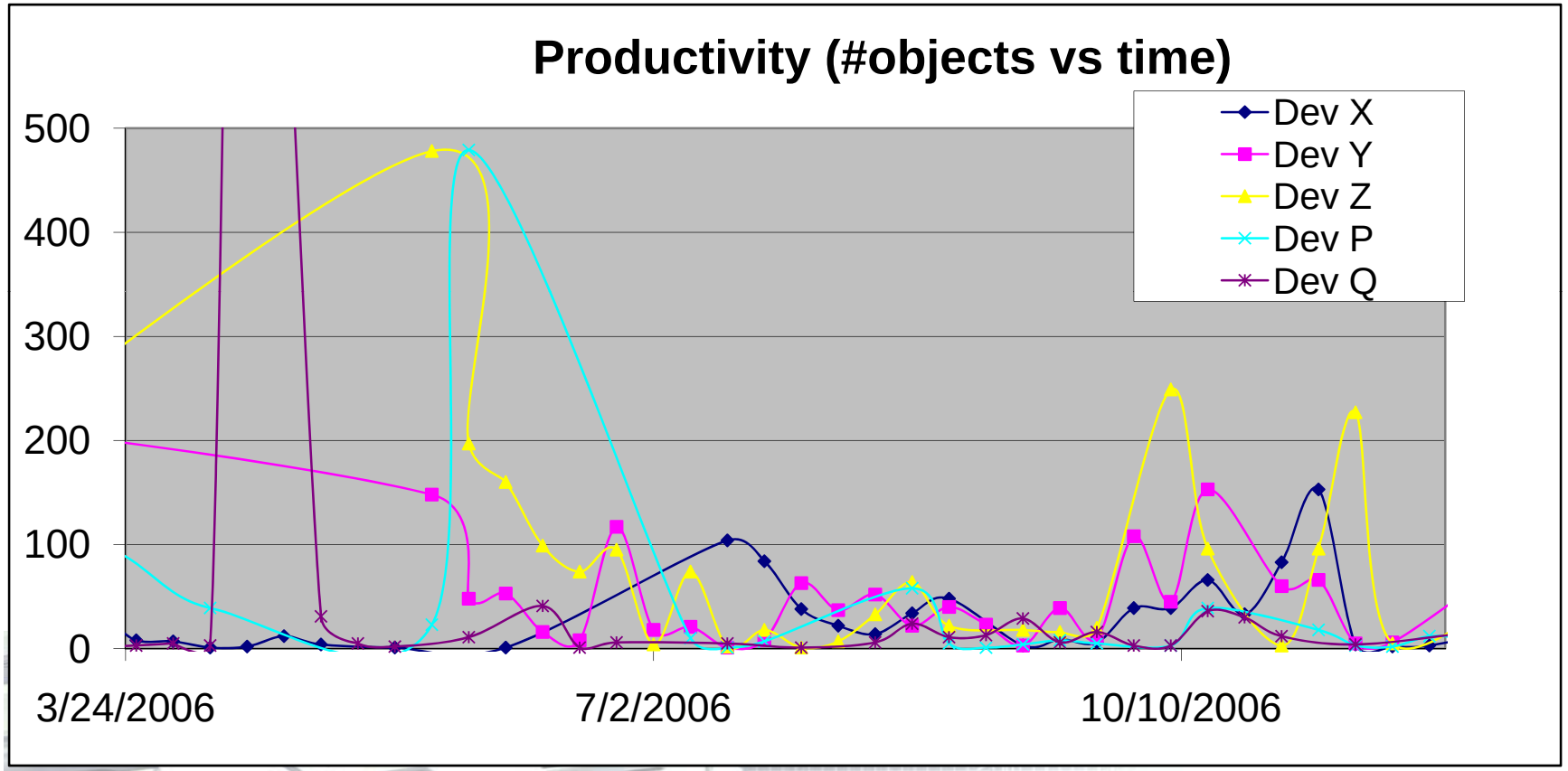
</conn>
Enterprise
<Product
>
</
Product >
<Custom
er>
</
Customer
>
<Order>

</Order>
Task in the
application
project
Task
Task in the
enterprise
project
CR

QA – DEV integration



Metrics


Productivity (#objects vs time)
500
400
300
200
100
0
3/24/2006
7/2/2006
10/10/2006
Dev X
Dev Y
Dev Z
Dev P
Dev Q

Conclusions

- Benefits
  - Integrated development lifecycle increases true agility (vs. perceived agility)
  - Automation saves time and money
  - Reporting capabilities showing true cost of the project and quality of the staff
- Drawbacks
  - High cost of implementation
  - Long learning curve for the “average developer”