Software Factory & Continuous integration

Cours LPSIL
2013

Questions ?

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

Agenda

- Introduction:
  - Software Factory
  - Maturity Levels
- Initial
- Repeatable
- Defined
- Managed
- Optimized
- Conclusion

Introduction

- What you always do to develop a software ?

Introduction

- What you always do to develop a software ?
  - Write source code
  - Compile/Build source code
  - Test your code
  - Package your code (to a product)
  - Change, update, correct, improve your code
  - Deliver your product to customers
  - Help & support your customers

Introduction
Software Factory

- Being manual or automated, these steps are the ones you always have to perform to create and deliver a product to a customer

- The Software Factory discipline will focus on industrializing steps from the production of **code** up to the delivery of a **product**

Introduction
Software Factory

- Main goals of the Software Factory

Introduction
Software Factory

- Main goals of the Software Factory
  - Allow collaborative development
  - Ensure robust development process
  - Increase confidence in the deliverable
  - Ensure expected level of quality
  - Reduce costs (of poor quality)
  - Fasten time-to-market
  - Finally ensure customer satisfaction
  - Guarantee traceability and repeatibility

Introduction
Maturity Levels

- CMMI
  - Capability Maturity Model Integration
  - A model to assess how organizations are effective in their processes
  - And also, to provide them with elements to improve these processes

Introduction
Maturity Levels

- CMMI Maturity levels
  - Initial
  - Repeatable
  - Defined
  - Managed
  - Optimizing

Introduction
Maturity Levels

- Why is it relevant for us ?

- We will show tools and processes for operating the software factory
  - Some are very naive, some are very advanced
  - It does make sense to pick a consistent set
  - **Choice will depend on maturity** of your organization
  - **Choice will evolve over time** when organization grows / needs arise / investment is made

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

Initial

- Works for very small team (1 or 2 people)
- Works for new development project
- Works for Lab/Inception/Prototype/Small product
- Works for small customer base

Initial

- Source Configuration Management
  - None ?
  - Shared drive
  - Email
  - Zip files
  - Dropbox
  - other…?

Initial

- Build Management
  - From the IDE
  - Or local command line (javac)
  - Manual when existing

Initial

- Tests Management
  - Manual testing
  - Low level of testing
  - Small set of scenario and use cases verified
  - Focus on « Happy path »
  - Short list of platforms verified

Initial

- Defects Management
  - To-do list
  - Emails
  - Other ways ? What do you do yourself ?

Initial

- Packaging
  - Archive created manually
  - Jar, zip, tar, rar …

Initial

- Changes Management
  - Discuss them at the office
  - Take decision quickly
  - Implement them quickly

Initial

- Delivery Management
  - Send him the zip file by email ?

Initial

- Support Management
  - None ?
  - Ad-hoc / on demand

Initial

- Exercise
- What are the limitations of this model ?

Initial

Limitations (non-exhaustive list)

- Hard to manage/support several versions
- Limited collaboration with team players
- Compilation can be dependent of your local environment
- Test coverage is low
- Hard to reproduce customer environment
- Lack of traceability
- Reactive to Events

- Heroic, chaotic, not repeatable
- « Works on my machine » syndrom

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

Repeatable

- Source Configuration Management
  - Put in place a central repository (SVN or similar)
  - Start working with several branches
  - Ensure code is backuped, versioned, secured
  - Allows several members to contribute

Repeatable

- Build Management
  - Write a script !
  - Using Ant (or similar) for example
  - One full build script
  - Developed and maintained by one build « guru » member

  - Ensure everyone will build the same way

Repeatable

- Tests Management
  - Extended manual testing
  - Start managing your tests
    - Define use scenario to cover
    - Write test plans / test cases
    - Save/Store them somewhere in a document (word or similar) available to everyone

Repeatable

- Defects Management
- and Changes Management
  - Maintain the list somewhere
    - HTML changelog page
    - Wiki page
    - Other…?
  - Be able to provide the list to customers

Repeatable

- Delivery Management
  - Provide the customer with your zip file
  - Also provide the customer with
    - Changelog
      - Resolved bugs list
      - New features list
    - User documentation
      - Installation
      - Deployment
      - Use cases / How to…

Repeatable

- Support Management
  - Support your customers using emails/mailinglist
  - Start having a « single point of contact »

Repeatable

- Exercise
  - What happens if the « Build Guru » is off ?

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

Defined

- Governance
  - Start writting processes
  - Share them with the whole organization
  - Source Code Management practices
    - Well known active branches
    - Coding rules and best practices
  - How to compile code
    - Using which techno ?
    - Within which architecture ?
  - How to deliver to customer
    - What can the customer expect ?

Defined

- Source Configuration Management
  - Start being more detail-oriented when you commit
  - Think about others
  - Think about when you are not there !

  - Add a comment for each commit : Mandatory

Defined

- Build Management
  - What about a script by module ?
  - Componentize your code and build
  - Start using a continuous build system to automate builds
  - Trigger automated builds more often in a reference environment

Defined

- Defects Management
  - Use a bug/defect management system
  - Qualify
  - Prioritize
  - Track
  - Close / Verify

Defined

- Tests Management
  - Manual and documented test plans
  - But also Unit tests to avoid regression
  - Tests executed by the Continuous Integration server for each change

  - Put in place a dedicated Test infrastructure
    - Performance tests
    - Functional tests
    - …

Defined

- Packaging
  - What about providing a real installer to the customer ?
    - InstallAnywhere
    - Install Shield
    - Installation Manager
    - Etc …
  - Far better user-experience than extracting a zip file, no ?

Defined

- Changes Management
  - Several stakeholders would be interested in changes (Developers, Product Marketing, Release Management, Testers, etc…)
  - Would probably need to put in place processes to review and approve (or reject !) the changes
  - Need to assess the impacts, the risks…
  - Log the proposed changes somewhere (wiki page ?) and take the decision with all relevant stakeholders
  - Track the status of the changes

Defined

- Delivery Management
  - Provide the customer with the Product Installer
  - Provide the customer with the full release changelog
  - Provide the customer with Product Documentation
  - Get customer acceptance

Defined

- Support Management
  - Provide the customer a ticket system to log any issue he should have
  - Provide support more quickly (time of response)
  - Defined in the customer contract ? (SLA=service level agreement)

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

Managed

- Governance
  - Follow the defined processes
  - Capture KPI (key performance indicators)
  - Track your performance
  - Enforce predictability

Managed

- Source Configuration Management
  - In addition to the comment associated to your commit
  - Link a defect number / bug ID
  - It will provide more context

  - Report bug fixes on several branches (even on already shipped versions)

Managed

- Build Management (1)
  - Needs multiple build plans
  - 1 by module/team
  - 1 by branch

  - Needs a more advanced dependency management
  - Should allow dev team to work even if the module of the other team is « RED » (i.e Broken, does not compile)

  - **Automated Build Plan for everything**:
    - Product modules
    - Installers (integration)
    - « Globalization » (if required)

Managed

- Build Management (2)
  - Needs parallel builds
  - Needs larger infrastructure
  - Needs more configuration available (various OS, JVM, DB, Appservers, browsers)
  - Needs advanced Continuous Integration tools
    (ex: Jenkins, Bamboo, etc)
  - Needs a dependency management repository for easier integration
    (ex Maven + Nexus, Artifactory, etc…)

Managed

- Tests Management
  - Run more and more tests, automate more and more…
    - Integration
    - Platforms
    - UI
    - Scalability
    - Security
    - Installation

Managed

- Defects Management
  - Priority management
  - Cost estimate
  - Risk analysis
  - Planning

Managed

- Packaging
  - Nightly installer generation for early internal testing
  - Milestones installers (iteration, beta, release candidate)
  - Automated installation testing
  - Implement Fixpack / Update installers

Managed

- Changes Management
  - See dedicated course
  - Ticket system
  - Requirements management
  - Approval mechanism
  - Release plan
  - …

Managed

- Delivery Management
  - Provide customers 24x7 download server
  - Replicated servers
  - Customer tracking (how many download ?)
  - Entitled area for customers
  - Manage licenses, users rights

Managed

- Support Management
  - Ticket system
  - Online FAQ / Knowledge base
  - Live chat assistance
  - Phone line that covers every timezones
  - Support in various languages …

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

Optimizing

- Processes
  - Documented
  - Monitored
  - Revised given performance

  - Certified ? -> selection criteria for choosing a supplier
    - CMMI
    - ITIL
    - PMI
    - ISO
    - Sox / Sarbane Oxley
    - etc

Optimizing

- Source Configuration Management
  - Advanced features, such as:
  - Links between defect or work item or build and code revision
  - Local branch management
  - Triggers upon commits:
    - Check for defect patterns
    - Auto formatting
    - Adding copyright headers

Optimizing

- Build Management
  - Capture advanced QA metrics at build time
    - Code coverage
    - Static analysis
    - Bug detection
    - Duplicated code detection
  - Scalable build infrastructure
    - Virtualized
    - On the cloud ?
  - Multiplatform

Optimizing

- Defects Management
  - Proactive defect management
    - Graphs showing defect intake and take-down
    - Defect reduction objectives depending on dev. Phase
  - Strategy on which bugs to fix
  - Open and proactive communication with customers

Optimizing

- Tests Management
  - All types of tests
  - Vastly automated
  - Virtualized
  - Strategy
    - Where to test?
    - What to re-test?
    - Platform gaps

Optimizing

- Packaging / Installer
  - Incremental updates
  - Automatic updates
  - Better user experience

Optimizing

- Changes Management
  - This deserves a specific session. Will be handled the week of Oct 16th.

Optimizing

- Support / Maintenance
  - Ticket system
  - Online FAQ (self-serve support)
  - Proactive patch push

Agenda

- Introduction
- Initial
- Repeatable
- Defined
- Managed
- Optimizing
- Conclusion

State of the art & Best Practices

- Continuous integration :
  - Take a look at what **Martin Fowler** says : http://martinfowler.com/articles/continuousIntegration.html
  - "Continuous Integration is a software development practice where members of a team **integrate their work frequently**, usually each person integrates at least daily – leading to multiple integrations per day. **Each integration is verified by an automated build** (including test) to detect integration errors as quickly as possible."

State of the art & Best Practices

- Key best practices
  - Commit your changes often (one or more time a day at least !)
  - Build, Test, Integrate continuously
  - Keep the build « Green », Fix quickly otherwise…
  - Get feedback, Monitor reports continuously
  - Do not accept regressions !
  - Track, log, document and follow-up changes

Conclusion

- Version control is key
- Traceability is key
- Repeatibility is key

Tomorrow

- Scalability through the Cloud?
  - Automated on-demand delivery of standardized test environments
  - Traceability across the whole chain: dev teams, test teams, operational (IT) teams

Tomorrow

- DevOps
  - Development and Operations working together
  - Continuous Delivery
    - Continuous Testing
    - Continuous Intregration
    - Continuous Deployment
    - Continuous Monitoring

Questions ?