Software Development
Best Practices
Artifact Reuse Supported by Modern Team Communication Tools and Patterns

Andrew Bose
Principal
(979) 777-1431
andrew.bose@parivedasolutions.com

Pariveda Solutions, Inc.
Houston, TX
Table of Contents

- Difficulties Facing Custom Software Development
- Introduction to Software Development Best Practices
- Team Communication
- Artifact Reuse
- Assessing Current and Planning New Best Practices
Half of every dollar spent on software development is currently spent on defect mitigation

- The software industry has the highest failure rate of any so-called engineering field” – Capers Jones
- Other resources echo Jones’ claims. In 2011, a Computer World study found
  - 21% of software projects fail completely
  - Another 42% were over budget, late, or did not meet requirements
- In another study, Capers Jones found
  - 75% of software projects were late
  - 35% of software projects were cancelled altogether
- Technology research firm, IDC conducted a survey of 139 companies in North America
  - Ranged in sizes from 250 to 10,000 employees
  - Over 50% had over 1000 employees
- IDC reported that these companies had annual defect resolution costs of
  - $5.2M (midpoint-mean 100 IT employee organization)
  - $22M (midpoint-mean 416 IT employee organization)
Best Practices bring a level of control and predictability to software engineering that increases the probability of completing a successful project

- Best practices are methodologies or measures a team can take to give a reasonable level of confidence of a successful outcome; they help a team bring order to chaos.

- In order to choose best practices, it is best to look to industry organizations and individuals who have been involved with hundreds or even thousands of projects:
  - Capers Jones, Fred Brooks, Mike Cohn, Martin Fowler, Laurie Williams
  - Institute of Electrical and Electronics Engineers (IEEE)
  - The Software Engineering Institute

- These software development experts have captured metrics and done the science to prove the benefits of the best practices they advocate.

- It is not necessary for a team to invent all of its own best practices, it is, however, important to keep project metrics so that improvements can be measured:
  - Defect density
  - Source repository statistics
  - Size of project

- A development shop should choose best practices based on organizational culture and the product(s) they are building.
Although the benefits of specific best practices may vary between organizations, most companies will benefit from Artifact reuse and Team communication

  - Although the benefits of specific best practices may vary between organizations, most companies will benefit from:

**Artifact Use**
- Team Communication
  - Team communication and artifact reuse have a powerful interaction
  - Team communication builds a sense of community within the organization necessary for artifact reuse
  - Artifact reuse encourages cross-team, project, and domain communication building trust between members of the organization
  - Artifact reuse is most effective when it is built on a strong foundation of communication tools and processes driven by a culture that encourages and values cross-team knowledge sharing

- Having good communication within a team and reusing artifacts are two best practices that can dramatically increase development output

- Artifact reuse is a fantastic way for a development team to:
  - Increase velocity
  - Improve consistency of design across the enterprise
  - Bring more value to a project than otherwise would have been possible
Often times, internal software development teams are not communicating optimally.

<table>
<thead>
<tr>
<th>Commonly Encountered</th>
<th>Best Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Real Time</strong></td>
<td></td>
</tr>
<tr>
<td>Face to Face / Phone / Email</td>
<td>Integrated Communication Tool</td>
</tr>
<tr>
<td>Developers communicate in person or by phone often resulting in interruptions. Interruptions result in a break in concentration, and reduce team velocity.</td>
<td>Developers communicate in person or via IC with a view of each other’s schedules and current availability. This prevents interruptions during meetings or blocks of work time. Teams use signals and cues to indicate whether or not they are available for conversation.</td>
</tr>
</tbody>
</table>

| **Asynchronous**      |               |
| Email                | Message Board System |
| Developers communicate asynchronously over email. Conversations are sometimes saved, but accessible only by those participating in the original conversation. | Developers communicate via message boards. Conversations are persisted indefinitely, and accessible by all those interested in the same subject matter. |

| **Persistent**        |               |
| Documents / Email / Share | Wiki |
| Team members save project documentation in a document repository, or are satisfied that documents have been emailed between team members. Documents last indefinitely, but are obsolete almost as soon as they are created. Documents are rarely updated, and there are sometimes arguments over who has the latest version. | Project documentation is stored in a wiki. Because the barrier to update is so low, documents are updated often. Links between message boards, source control, and feature trackers allow updating persistent documentation seamless and easy. |

| **Feature Tracking**  | Purpose-Built Application |
| Email / Spreadsheet | Features and bugs are stored in a purpose-built feature tracking tool. This tool integrates closely with source control, can easily link to persistent documentation, and be linked to from message boards. Features and bugs can be updated in real time by any team member with changes viewable by any other team member. |

Information on features and bugs are shared over email. Sometimes features and bugs are cataloged in a spreadsheet, but there are often questions surrounding who has the latest version. Some spreadsheets may be uploaded to a document repository, but these are often out of date.
To be successful, a development team should have in place, and use, a plan and modern tools for communication

- A sufficient communication plan should include not only a description of how a team communicates internally, but also how it will communicate to
  - End users
  - Stakeholders
  - Other teams
  - Infrastructure
  - Vendors

- Processes described in the plan may include routines such as daily standup, iteration reviews, weekly status reports, weekly manager meetings, etc.

- Organized communication between teams becomes essential as a team grows sufficiently large to experience the “Network Effect,” where communication lines increase exponentially as the number of team members increases linearly

- A development team should be sufficiently enabled to communicate in the following ways:
  - Real Time
  - Asynchronous
  - Persistent
Artifact reuse is the number one way to increase development team productivity, and should be one of the first targeted best practices for software development

- Some software engineering experts believe that in the future it will be possible to build software out of large pre-existing components in much the same way that projects are completed in the physical world
- Other experts think this will never happen as software engineering is inherently different than its physical-world counterparts
- The truth is somewhere in the middle
  - Certainly, we could be doing more to develop and catalog reusable artifacts
  - Reusable components become available as specific requirements and features become more common place
  - Currently it is very difficult to even approach the component re-use utopia suggested by Jones
- It is difficult for an artifact reuse program to thrive unless a sound foundation of communication tools is already in place and consistently used by the enterprise
Artifact reuse exists in a pyramid of evolution; while internally achieving the highest levels of evolution may be difficult, all enterprises are currently engaged in at least some level of reuse right now.

- **Service**
  - Generalized data and algorithms for service-level reuse
  - Similar to idea of SOA

- **Library**
  - Code that can be used as a component of library with minimal configuration
  - Challenges around upgrades and bug fixes

- **Source**
  - Sample code
  - Source code from previous projects
  - Requires good communication and a reliable community

- **Architecture**
  - Pattern decisions
  - Common codebase organizations
  - Technical organization of solution

- **Design**
  - Tool choices
  - Usage documentation

- **Concept**
  - Technology stack decisions
  - High-level documents
The following is a plan to assess the current state of communication and reuse and explore how an organization can add best practices to their processes in order to increase the probability of project success.

**Establishing Reuse Best Practices Within the Enterprise**

**Preliminary Work**
- Preliminary Analysis
  - Interview project sponsor
  - Interview subset of project teams
  - Interview subset of users / business leaders
- Assess High-Level Maturity
  - Assess state of communication
  - Assess state of reuse
- ROI Estimated
  - Assess potential benefit
  - High level ROI assessment
  - Automated source analysis
- Plan Phase I
  - High-level Plan
  - Proposed participants and projects

**Initial Analysis**
- Initial Analysis
  - Interview Steering committee
  - Identify team members
  - Segment Organization
  - Divide by Functional Units
  - Divide by Geography
  - Select representatives
  - Appoint Leads
  - Appoint lead curator
  - Appoint change manager
  - Identify candidates
  - Onboard candidates
  - Develop Plan
  - Set cadence
  - Set precedence for time commitment

**Cultural Analysis**
- Cultural Analysis
  - Interview steering committee
  - Interview team members
  - Interview SMEs (technical and business)
- Tool Analysis
  - Review tools used within technical teams
  - Review tools used by

**Reuse Maturity Workshops**
- Reuse Maturity Workshops
  - Plan and organize workshops
  - Conduct workshops across business units or geographies
  - Define current and desired maturity

**Team Engagement**
- Team Engagement
  - Obtain access to project documentation and artifacts
  - Attend iteration planning and review meetings

**Gap Analysis**
- Gap Analysis
  - Analyze current and desired maturity for each business unit or geography
  - Document gaps, identify changes needed to close gaps

**Opportunity Identification**
- Opportunity Identification
  - Identify technical overlap with existing or other in-flight projects
  - Identify technical overlap with future (planned) projects

**Tool and Process Update**
- Tool and Process Update
  - Recommend new tools and processes based on gap analysis

**Next Phase Planning**
- Next Phase Planning
  - Identify next projects, business units, or geographies to target for reuse
  - Plan communication and change management
  - Plan specific reuse enablement engagement(s)

**Project Identification**
- Project Identification
  - Identify projects through which to pilot preliminary reuse recommendations

**Additional Recommendations**
- Additional Recommendations
  - Document and present software development best practice recommendations beyond reuse:
    - Conceptual
    - Testing
    - Etc.