Ten Practices of High Performance Teams

Noopur Davis
Davis Systems

Better Software Conference and Expo
Las Vegas, NV
Outline

- Background
- Ten practices of high-performance teams
  1. Self-directed teams
  2. Iterative and incremental development
  3. Openness and transparency
  4. Simplicity
  5. Specific and in-depth how-to training
  6. Focused use of data
  7. Uncompromising commitment to quality
  8. Importance of design
  9. Continuous improvement
  10. Coaching
Most software development work is done by small teams.

Even projects with millions of lines of code are produced by groups of small teams working collaboratively.

To a large degree, the performance of these teams governs the performance of the organization.
Our Background -2

- 25 years as a developer, manager, coach, and consultant.
- Trained, launched, and coached hundreds of teams in dozens of organizations, large and small.
  - Adobe.
  - Intuit.
  - Microsoft.
  - Abb.
  - Amerigroup.
  - Lockheed martin.
  - Computer works.
Methodologies Used

- RUP
- CMM/CMMI
- Team software process (TSP)
- SCRUM (with agile practices such as test driven development, user stories with planning poker)
Observations

- Over the past decade, we have worked with many high-performance software teams.
- Some common characteristics of high-performance teams were observed, regardless of
  - Methodology
  - Organization size
  - Or project duration.
- We will share these key characteristics and best practices.
What Is a Team?

Should have

- 3-15 people
- Multi-disciplined
- Working towards common goals
- Doing inter-dependent work
- Using common work processes

Nice to have

- Co-located
- Largely dedicated to one project
What Are High-performance Teams?

High-performance teams
- Consistently deliver products that delight their customers,
- On predictable schedules,
- With agreed-to functionality,
- And with high quality.

High-performance teams are
- Proud of what they produce,
- Are continuously improving the way they work,
- Are introspective yet open and transparent.
1. Self-directed Teams
Self-directed Teams

- Self-managed, self-directed, or self-organizing?
- What we mean by self-managed
  - Organization is concerned with the “what.”
  - Team is concerned with the “how.”
  - Organization is also concerned with HR, regulatory, compensation, and other such issues.
Organization

- Establishes vision
- Forms teams
- Appoints team leader
- Provides business and product goals
- Performs governance functions
Self-directed Teams

- Estimate their own work
- Self-organize
- Allocate the work amongst team members
- Track and adjust schedule and quality
- Define their own processes (within organization boundaries)
- Provide status to management
Why Self-directed Team?

- Most people, when given the chance, do the right thing.
- The best way to give people that chance is to allow them to
  - Make their own commitments,
  - Determine how they will do their work,
  - How they will plan, track and manage their work.
- Self-directed teams are empowered to do this.
Effective Leadership

- Self-directed teams need leaders just as much as other teams do.

- We have never seen a high-performance team that did not have an effective first-level manager
  - Coached the team,
  - Set high standards,
  - Did not dictate but guided,
  - Believed in quality,
  - Protected the team from the wider organization,
  - Celebrated each victory,
  - Did not panic during difficult times,
  - And made rational decisions based on data.
Motivation

Teams need motivation
- From fellow team members.
- Working towards a common goal.
- Common measure of success.

To motivate a team, management must
- Sets goals.
- Communicates them to the team.
- Allow the team to meet these goals in the best way the team determines.
Example

Koala

Tea

established 2007

Source: Joe Anderson, Amerigroup
A qualified coach guides the team through a defined process to develop its plan and to negotiate that plan with management. The most important outcome is a committed team.
SCRUM Flow

Source: Ken Schwaber
2. Incremental and Iterative Development
Commonly Understood Definitions

- Iterative – planned rework to revise and improve parts of a system.
- Incremental – plan to divide and conquer. Break work into smaller pieces, complete each piece, and then integrate into system.
Incremental Vs. Iterative

“Incremental fundamentally means add onto. Incremental development helps you improve your process.

Iterative fundamentally means re-do. Iterative development helps you improve your product.

Both need improvement. Create your staging, integration and rework strategies accordingly.”

--Alistair Cockburn.
Team Software Process

- Business and product goals
- Iteration Launch
- Lessons, new goals, new requirements, new risk, etc.
- Development Iteration
- Iteration Retrospective
- Release Retrospective
- Release Launch
- Estimates, plans, process, commitment
- Work products, status, metrics, results
Scrum

Source: Mike Cohn
Iterative Development

- Most modern processes embrace incremental development
- High performance teams also focus on iterative development
  - User interface
  - Performance
- Which engineering practices support iterative development?
3. Openness and Transparency
A Story

Consultant: “Wow, the burn-down charts on your wall look great. Looks like things are going well.”

Team: “Well, those charts are for management so they will leave us alone. The real charts are here.”

Consultant: laughs as he relates this story about how teams can get around management when it gets in the way.
An Example

Average Task Hours Per Week

Source: Allied Signal

+57%
Implications

- When given a chance, people will do the right thing.
  - This includes management.
- When a team communicates openly with stakeholders, they
  - Get used to receiving good news and bad.
  - Do not get too excited about the good news.
  - Do not panic about the bad news.
  - Trust the team to manage its work.
4. Simplicity
Simple Process?

http://electronicmuseum.org.uk/2009/02/03/the-problem-with-process/
Simple Process?

Figure 3: Process and project crates with RUP workflow colored folders
The team’s work processes have to be “as simple as possible, but no simpler”.

Work instructions must fit on a page or two.

Review checklists with a dozen or fewer high-impact items.

A few simple metrics collected as the teams do their work.

Processes and measures that get in the way of teams completing their work hinder high-performance teams. Imagine
  – Using a hundred-page coding standard.
  – Adopting twenty process-improvement proposals at once.
Example: Definition of “Done”

A user story is complete when

– Design has been captured and inspected
– Code has been written and inspected
  ● Using team defined, simple coding standard
– Unit test code coverage is at 80% or higher
– Integration is complete
– Acceptance tests are complete
5. Specific and In-depth How-to Training
Training Needs

Teams need both technical and non-technical training.

- planning,
- tracking,
- making commitments,
- use of data to make decisions,
- effective inspections and reviews,
- writing good user stories,
- using static analysis tools,
- using unit test frameworks.
Just in Time

- Frustrating for teams to attend training on processes or procedures that they cannot immediately apply at work.
- Training must be specific and hands-on.
- High-performance teams receive specific, hands-on training, usually just in time to apply at work.
  - What are the implications for an organization?
- Talking about theories is fine.
  - But teams need practical tools and need to learn to apply them to their work.
6. Focused Use of Data
Measurement

- You can’t manage what you can’t measure.

- Measures should be
  - Collected by team members as they do their day-to-day work.
  - Simple.
  - Used to plan, track, adjust, improve, and communicate.
  - Appropriate for small adjustments, and also long-term trends.
Multiple Views

- Customer satisfactions/feature
- Schedule
- Quality
- Cost
Customer Satisfaction

Net Promoter Score

Would you recommend this product or service to others?

Extremely likely | Neutral | Extremely unlikely
---|---|---
10 | 6 | 2
9 | 5 | 1
8 | 4 | 0
7 | 3 |

Promoter | Passive | Detractor

Net Promoter Score = % Promoter minus % Detractor

Source: The Ultimate Question: Fred Reichheld, 2006

Promoters share their delight with their friends, family and business associates creating future promoters

Source: Jim Sartain, Adobe
Features

- Plan vs. Actual stories delivered.
- Business Value delivered.
“If you don’t know where you are going, any road will get you there.” -- Lewis Carrol

“If you don’t know where you are, a map won’t help” – Watts Humphrey
Quality Data For Iteration

Defects Removed by Activity

- Detailed Design Review: 2
- Detailed Design Inspection: 9
- Code: 2
- Code Review: 11
- Code Inspection: 60
- Unit Test: 1
- Integration Test: 1
- System Test: 4
Quality Data – Code Coverage

![Bar chart showing code coverage for Story A, Story B, Story C, and Story D. The coverage ranges from 0% to 100% for each story. The chart indicates that Story C has the highest coverage at 90%, while Story A has the lowest coverage at 50%.]
7. Uncompromising Commitment to Quality
A Conversation

- Team member: “Hey guys, we are working on a VERY tight schedule, so we can not take any shortcuts on quality”.

- Another team member: “We should add a task to update static analysis rules to our plan”.
Quality Principles

- Focus on quality from the beginning of the job.
- To manage quality, measure it.
- Remove defects as soon after they are injected as possible.
- Defects should be removed by the person who injected them.
  This improves the chances of making the correct fix.
- Defects should be measured and analyzed.
  This promotes root cause analysis and defect prevention.

“Quality without numbers is just talk.”
– Watts Humphrey
8. Importance of Design
Why Design?

- Isolate aspects for thinking
  - Focus – get this one aspect right
  - Abstraction – ignore details

- Analysis
  - Learn from the model
  - Possibly verify properties
  - Discern risks
  - Evaluate alternatives

- Communication
  - Drive implementation
  - Evaluation/review

- Planning/Estimating

- Defect Reduction
Controversial Questions

- **Waterfall?**
  - No, high performance teams do not do all the design up front.

- **Refactor instead of design?**
  - Need both
  - Refactoring is not design.
Designed Programs are Smaller

8,100 Programs

Source: SEI
9. Continuous Improvement
High-performance teams don’t rest on their laurels.

They systematically look for areas of improvement:
- Planning
- Process
- Quality
- Technology

They plan and track the improvement activities just as they plan and track other project work.
Retrospectives

- **Quantitative**
  - Planning parameters (velocity, hours available, rate of earning business value)
  - Quality (defect density, coverage)
  - Process (yields, % effort spent in finding and fixing defects)

- **Qualitative**
  - What worked
  - What can be improved and how

- **Focus on one or two improvements at a time.**
10. Coaching
The Need for Coaches

- Even with the best of intentions, it is easy for teams to get too caught up in the “what” and not the “how”.
- Schedule or technical pressure can lead the team to abandon best practices.
- Teams frequently need help with
  - Interpreting data to make decisions,
  - A neutral process expert who works with them on a continuous basis.
  - guides the team along its improvement journey.
Coaching Considerations

- Team leader from team A can coach team B.
- Full-time coaches who coach multiple teams.
- Team members who coach other teams.
- Senior managers should not be coaches.
- Training program for coaches.
Conclusion
## What High Performance Looks Like

<table>
<thead>
<tr>
<th>Measure</th>
<th>Industry Average</th>
<th>High Performance Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Promoter Score</td>
<td>20%</td>
<td>&gt; 70%</td>
</tr>
<tr>
<td>% defects of total injected found by customer</td>
<td>15%</td>
<td>&lt; 2%</td>
</tr>
<tr>
<td>% effort spent in finding and fixing defects</td>
<td>50%</td>
<td>&lt; 10%</td>
</tr>
<tr>
<td>% effort for post-release support</td>
<td>30%</td>
<td>&lt; 5%</td>
</tr>
<tr>
<td>Unit test code coverage</td>
<td>Varies</td>
<td>&gt;= 80%</td>
</tr>
<tr>
<td>Post release defect density</td>
<td>7.5 defects/KLOC</td>
<td>&lt; 0.5 defects/KLOC</td>
</tr>
</tbody>
</table>
Conclusion

- It is not the methodology
  - Maturity ratings
  - Process conformance
  - How agile are we?
  - Can’t do this, it is not <Agile, TSP, RUP, XP, Lean>

- Ultimately, what matters most is performance.
For More Information

- Visit us on the web at
  http://www.DavisSys.com

- Latest copy of this presentation available at
  http://www.davissys.com/reldoc.htm

- Noopur Davis – NDavis@DavisSys.com