Software Executive Report

Notes from the Executive Council for Software Excellence

May 2008

Managing Core Development

"Core" code always refers to code that is in some way more central than other product code. There are several variations on this theme:

- Most companies use "Core" to refer to architecture and functionality that multiple groups depend on. Reusable architectures, engines, toolsets, platforms, and frameworks are all examples of software that companies think of as core.
- "Core" can refer to code that has exceptionally high quality requirements.
- "Core" can refer to software components that provide competitive advantage.
- One company defines Core as anything that affects the user experience. In this company's business, this seems to be a variation on the theme of "Core" referring to code that has exceptionally high quality requirements and that provides competitive advantage.
- In some cases, when data is of central importance to a company (e.g., product info for a web company), "Core" can also include data and data access.
- Parts of the software that require governance are also sometimes considered to be core.

Core is also known variously as "Platform," "Application Architecture," "Infrastructure", and other terms.

Why Set up a Core Group?

The core team's responsibility is to provide an easy path for other groups to use leading technology, i.e., make it easier for non-core developers to do a good job.

Core groups are sometimes set up to tap into specialized skills of a group that are needed commonly across products. For example, a company that produces scientific software might have a core group that consists of science Ph.D.s who write core code to implement key company algorithms.

When the core is managed well, it can accelerate development by providing tools that make other groups more efficient and effective in the short term, reduce the support burden in the long term, or both.

Differences between Core Development and Product Development

Companies report several common differences between product development and core development:

- **Quality assurance tends to be more rigorous**, because problems in the core can adversely affect multiple products.
- **Management of the core tends to be more rigorous** for the same reason—schedule problems in the core can adverse affect schedules for multiple projects.
- When the core changes, change impacts on all teams need to be considered.
- Support obligations for the core are nettlesome. How long will the core group sup-

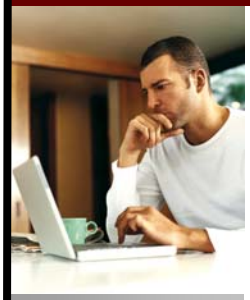
June ECSE Meetings

Organizational Improvement Strategies

How do you balance between *improving* and *doing*? How do you organize improvement efforts? Through a PMO, an SEPG, project post mortems, or some other organization or practice? Our June meeting will explore these questions.

Bellevue Meeting: June 9, 5:00-7:00 pm.
Dial-in Meeting: June 13, 8:00-9:00 am,

Construx®
www.construx.com



Register Now!

2008 Software Executive Summit

Construx's 5th Annual Software Executive Summit will be held this year from **October 27-29** at the Grand Hyatt in Seattle, Washington. In addition to the small group discussions, this year's Summit features the following presentations:

- **Steve McConnell**, author of *Software Estimation* and *Code Complete*, "Secrets of World Class Software Organizations"
- **Martin Fowler**, author of *Refactoring* and *Patterns of Enterprise Application Architecture*, "Cultivating Great Architects and Designers"
- **Ken Schwaber**, co-creator of Scrum, "Scaling Scrum"
- **Travis McElfresh**, VP Technology, "Driving Employee Satisfaction, Morale, and Productivity at MSNBC.com"
- **Matt Peloquin**, CTO, Construx Software, "Technical Lessons from the Software Wild"

Register Before June 15 and receive the 2007 price of $3000 and a credit of $2000 toward Construx's public seminars.

Register now at http://www.construx.com/Page.aspx?nid=231 or see more information

(Continued from page 1)

port each version of the code that it produces? Supporting several versions across each of several products can quickly become an unmanageable support obligation.

Staffing the Core Group

A common practice is to staff the core group with the strongest staff. Working on the core is usually perceived as a desirable assignment.

However, **most companies report that it's important not to treat core development as either better or worse than product development**—it's just different. Otherwise an us vs. them mentality develops. One way to split product responsibility vs. core responsibility is by people who like to work on tangibles vs. people who like to do work that's more abstract.

One company that uses short iterations (~1 month) reports that it **embeds core architects in the product teams on a per-iteration basis**. Architects are included in all team meetings and are responsible for deliverables on that iteration. Other architects are brought in as-needed.

The percentage of effort that goes into the core varies quite a bit. 20-25% is the most often reported percentage of total development, but companies report wide variances in this number.

Support obligations for the core are nettlesome. How long will the core group support each version of the code that it produces?

Does the Core Lead or Follow?

The core usually does some leading and some following. Product teams need to be free to innovate and deliver faster than a pure-core development strategy will allow, which puts the core into a following posture for some functionality. In other cases the core team needs to be able to pursue deeper, longer-term innovations that aren't possible on a schedule-bound project team, which puts the core into a leading posture for that functionality.

A common arrangement is to have product teams develop the bulk of the code first and then migrate selected product-line code into the core. This approach can help with core-staffing issues—the person who originally wrote the product-line code moves into the core group while doing the work to migrate the product code into the core. Companies report that when you rotate people through the core the people you rotate in tend not to want to rotate out. Still,

(Continued on page 3)

About the *Software Executive Report*

The *Software Executive Report* is published monthly by Construx Software, 10900 NE 8th Street, Suite 1350, Bellevue, WA 98004. To subscribe to the *Software Executive Report* or for copies of past editions, please contact us at ecse@construx.com or by telephone at +1 (866) 296-6300.

Construx®
www.construx.com

(Continued from page 2)
most companies report that some amount of rotation through the core team is healthy.

Migrating code into the core is not trouble free—the level of redesign, re-implementation, and retesting can be significant. The work to generalize code that was originally developed for the field is a project in itself. But if the code isn't originally developed for the field, it's very difficult to get real requirements and keep the core group grounded.

One company had a problem with product development teams dumping new functionality on the core group, which the core group was then expected to maintain. That was demotivating to the core team. The company decided to give the core team the power to reject code with low quality—it then became the product team's responsibility to maintain the code they'd written if the core team didn't see enough value in their code.

Another company "open sourced" non-strategic functionality, which freed up in-house developers to focus on more innovative software.

A third company practices a purely internal version of open sourcing core code. Product teams can check code into the core; they track how much each code unit is used, and they invest more deeply in the code that's used the most (i.e., assign dedicated staff to work on that code).

Pull or Push?

Companies reported different philosophies related to requiring product teams to use core code. Most commonly companies mandate use of the core because the core is critical to achieving business critical software requirements (e.g., security or reliability).

The work to generalize code that was originally developed for the field is a project in itself.

Other companies report that pull vs. push is a moot point – the core functionality is so vast and so central to the company's business that it isn't practical for individual project teams to replicate core functionality at the individual team level.

A few companies report that the core is a "pull" operation—product teams request functionality from the core group. These companies also seemed to be the companies in which the distinction between "core" and "product code" was the least clear.

What Not to Put Into the Core

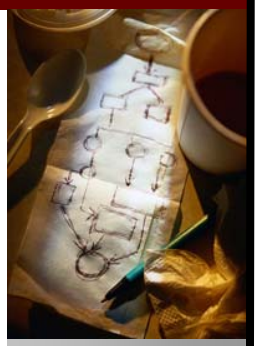
It's important to have a vision for the core—rather than just handling service requests from the product teams. If the core loses its conceptual integrity, the distinction between core development and product development will become blurry, and the development practices that are appropriate to core development can end up being applied to develop software that really isn't core—which creates more bureaucracy than necessary.

Most companies also report that areas in which the market is moving quickly are not well suited for core development—core-style development can't keep up.

About the Executive Council for Software Excellence (ECSE)

The ECSE is an executive discussion group hosted by Construx Software. Meeting monthly since 2002, the ECSE's goal is to share, analyze, and evaluate members' experiences facing enterprise-level software development challenges. ECSE members are executives with multi-project, enterprise-level responsibility for software development. The typical member oversees activities of 100 or more software personnel. The ECSE has a few members who oversee smaller staffs.

If you are interested in joining the ECSE or if you know someone who would be interested, please contact the ECSE host, Steve McConnell, at stevemcc@construx.com or +1(866) 296-6300.



10900 NE 8th Street
Suite 1350
Bellevue, WA 98004
+1 (866) 296-6300
ecse@construx.com

Construx®
www.construx.com

Communicating What's in the Core and Tracking How Much of the Core is Actually Used

Active communication about what's going into the core and what's available in the core is important. The core group needs to facilitate lots of communication going both ways—in and out of the core.

One company's core group publishes a technology roadmap, which is reviewed by the product teams. There is then a long process of getting closure—resolving competing interests and so on. After that point, buy-in in using the core is high.

Another company provides architects from the core group to work embedded in the product development teams. This helps ensure the core is used in the way intended, provides quick answers to questions about using the core, and helps the teams understand what they need to do in product code to support reusability.

Risks of Core Development

Some groups report that the core slows down innovation, slows down development, and introduces bureaucracy. Other groups report avoiding these problems, so it seems clear both that this is a risk, and that this is a risk that can be effectively managed.

Companies also report that too many demands on the core group is a common problem. Resolving conflicting demands on the core can be challenging.

ECSE Calendar 2008

June	Organizational Improvement Strategies: Balancing "Doing" with "Improving"
July	Supporting Innovation
August	*Summer break*
September	Issues in Test Management
October	Compensation Updates
November	Improving Productivity
December	*To be announced*

Core teams can lose touch with customers and product needs if core development is allowed to become too isolated from product development. Rotating staff through the core seems to help with this issue.

It can be a challenge to justify the amount of budget allocated to the core since the core doesn't contribute as directly to the business as product line software development does. Solutions can be as drastic as simply not exposing the existence of the core development group above the director level—burying funding in other groups' budgets. A more transparent approach is to impose a "core tax" on the budgets of the product line teams. In these cases, "no taxation without representation" is a relevant mantra—product teams need to have a seat at the core group's table to ensure the core group stays responsive to their needs. ■