An Introduction to Flexible Product Development

Building Agility for Changing Markets

Bob Becker
March 20, 2008

Understanding Flexibility

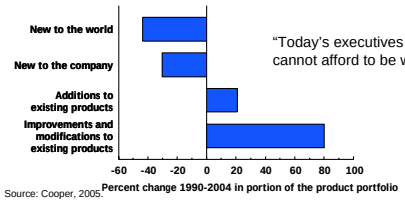
It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change.

- Charles Darwin

Definition: The ability to make changes in the product being developed or how it is developed, even relatively late in development, without being too disruptive. The later one can make changes, the more flexible the process is. The less disruptive the changes are, the more flexible the process is.

Understanding Flexibility

The more innovative the product, the more likely you are to make changes during its development.



Source: Cooper, 2005.

"Today's executives simply cannot afford to be wrong."

Understanding Flexibility

Why change?

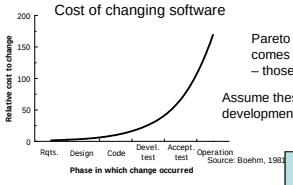
- Customer requirements
- Market shift changes
- Technology driven changes
- World events
- Corporate network (partners, suppliers)
- Organizational / people changes

What can you do about changes?

- Move faster (like agile sw development)
- Plan better (hope to anticipate change – 6 sigma, phase gate)
- Build process and apply tools and approaches that accommodate or embrace change (what Smith advocates in his book)
  - Apply flexibility where you must be innovative!
  - Flexibility can be expensive – use with discretion (think cost-benefit)

Understanding Flexibility

Cost of changing software



Pareto principle applies: 80% of cost of change comes from 20% of the most disruptive changes – those with system-wide impact.

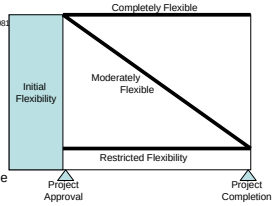
Assume these findings generally apply to developments other than software.

3 Levels of Managing Flexibility in a Development Project

A project managed for flexibility will look like the moderately flexible zone.

Decisions are made when necessary, often by progressively tightening up tolerances on variables.

Beware of unnecessary change simply because the system is now more tolerant of it.



Understanding Flexibility

MOVING FROM SOFTWARE TO OTHER PRODUCTS:

Software is a special medium that lends itself to agile approaches. Some of software's characteristics that agilists have exploited are:

- Object technologies, which allow clean modularization to isolate change and allow substitution of modules (we cover this in chapter 3)
- The low cost of an automated build, which makes frequent and early testing feasible
- The logic basis of software, which allows relatively fast automated checking for many types of errors
- (For IT projects) Customers that are easy to find and involve in development
- In general, the malleability of the software medium, which makes it relatively easy to make changes

- *Mechanical systems can lend themselves to physical prototypes*
- *Electrical systems often have programmable components*

Agile SW is a motivating model but many agile practices don't translate directly to non-sw products.

Customers and Product Requirements

"Your most unhappy customers are your greatest source of learning."
- Bill Gates

The Fallacy of Frozen Requirements
Best practice – Development team should thoroughly understand the customer, write down exactly what the needs are in a spec and design to that.

Reinertsen study of >1000 pd managers:
How often did the requirements remain stable during the entire design? Never (0%)
How many developers had complete specs before starting design work? 5%
On average what portion of requirements were specified before design commenced? 54%

*It is not that specs seldom remain constant during development; they **never** do.*

Changing Requirements

Changing requirements have a bad reputation
- usually mean added requirements from marketing or management
- project grew while the schedule and resources stay the same.

In effective flexible development, a requirement is as likely to be removed as added, a new one could be easier or more difficult than what it replaces. It is not "scope creep".
The objective is to provide customer value, not make the product bulletproof against anything a competitor could add.
Ideally, the change occurs before much is invested in the changes' requirements.
Each change involves cross-functional deliberation regarding pros and cons to provide maximum value from available resources.

Customers and Product Requirements

How do you handle customers and requirements that increase flexibility?

Specify at a higher level
- Less likely to change
- Could be:
  - more of a product vision giving guiding principles
  - Personas
  - Use cases
  - User stories

Anticipate changes in requirements
- Not forecasting the exact future
- Anticipate areas most likely to require change

Modular Product Architectures

"Minimizing connections between modules also minimizes the path along which changes and errors can propagate into other parts of the system, thus eliminating disastrous "ripple" effects, where changes in one part cause errors in another, necessitating additional changes elsewhere, giving rise to new errors."
- Larry Constantine

Modular versus Integral - Each has advantages

Modular:
+ Ease of changing the design
+ Independent testing
+ Reusable portions
- Planning time
- Performance weaknesses
- Integration burden

Integral:
+ Cheaper to make
+ Lighter, more compact
- Harder to change
- Late testing

Can shift boundaries to emphasize change – hw to sw, mechanical/hydraulics to electronics

Set-based Design

"When you come to a fork in the rod, take it."
- Yogi Berra

Designers typically practice *point-based design* – they select what they perceive to be the best course of action and concentrate on it
…in contract to *set-based design* which involves maintaining sets of options or a range of possibilities
- this creates/preserves flexibility
- this defers decisions – keeping the cost of change low
- American name for one of the key techniques used at Toyota

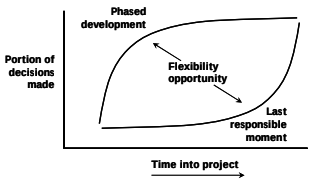
Set-based design emphasizes constraints (any path might still be ok as long as no cliff or swamp – what's in/out of bounds)
Point-based design emphasizes choices (which path of the fork)

Basic process involved in set-based design is pruning – look for the weakest solutions and eliminate them by imposing constraints
- early on pruning is obvious with numerous opportunities
- later weak spots are no to clear, more judgment is used
- judgment in managing convergence is key – prune too quickly and eliminate excellent options – prune too slowly and you prolong convergence to the optimal solution

At Toyota, always have at least one feasible solution – if they had to decide today, they have something that works
At Toyota, use progressive decisions to eliminate weaker options and tighten tolerances

Improving Decision Making

The Last Responsible Moment
The most important point about making decisions flexibly is *not* to make them until they must be made.
Phased development processes attempt to specify everything as completely as possible before development starts, as this helps to ensure that tasks are not skipped and nothing is left to chance.
This tendency toward completeness also limits options unnecessarily right from the beginning



The last responsible moment is the earliest of:
• an important option expiring
• the decision goes onto the critical path
• the expense of carrying the decision rises greatly
• delaying increases risk to project success