Construx®
Delivering Software Project Success

The 10 Best Ideas in Software Development

www.construx.com

Most Key Ideas Are Not New

Q: What are the most exciting/promising software engineering ideas or techniques on the horizon?

A: I don't think that the most promising ideas are on the horizon. They are already here and have been here for years but are not being used properly.

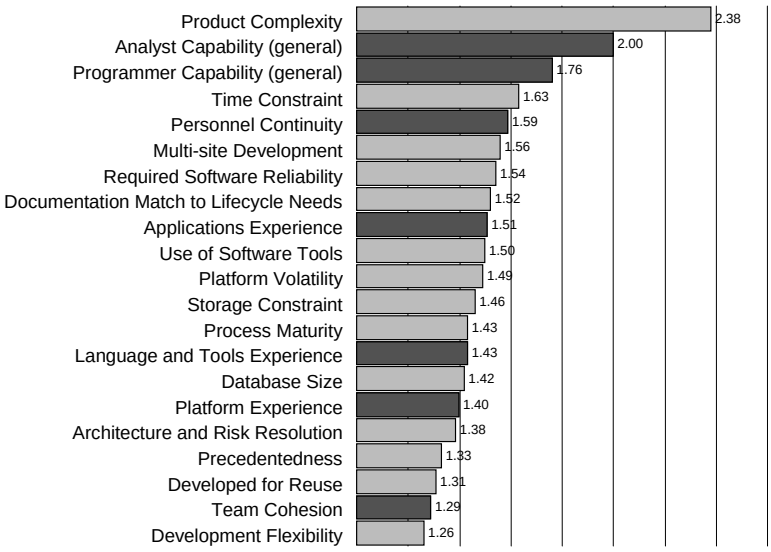
— David L. Parnas

Construx®
Delivering Software Project Success

#1

Software Development Work is Performed by Human Beings

Cocomo II's View of Software Project Influences



Importance of Human Influences

- Human Influences make a 14x difference in total project effort & cost, according to Cocomo II
- Capability factors alone make a 3.5x difference
- Experience factors alone make a 3.0x difference
- Consensus of studies is that similarly-experienced developers vary by about 20x in productivity and quality of work

Where do Variations Exist?

Researchers have found 20:1 variations in:

- Coding speed
- Debugging speed
- Defect-finding speed
- Percentage of defects found
- Bad-fix injection rate
- Design quality
- Amount of code generated from a design
- Teamwork
- Etc.

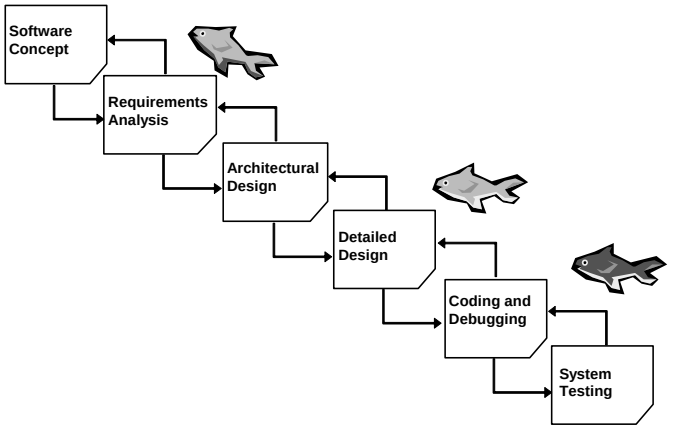
Some Implications

- Success of Google, Amazon, Microsoft, etc.
- Matching workers and workstyles
- Value of retention programs
- Importance of staff development

Construx®
Delivering Software Project Success

Creating software practices based on the assumption that developers are omniscient…

… i.e., the Waterfall Model



What About This?

From the Agile Manifesto

We value individuals and interactions over processes and tools.

Construx®
Delivering Software Project Success

#2

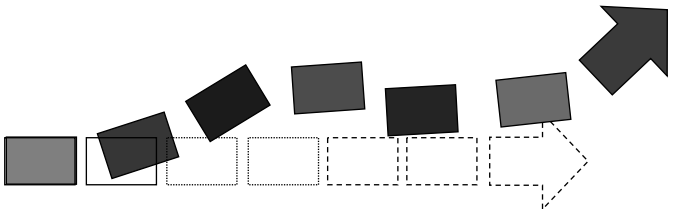
Iteration & Incrementalism are Essential

Incrementalism

- Definition: “The use of a series of regular additions or contributions”
- An “incremental” approach is one that is done a little bit at a time.
- The final result may be completely mapped out in advance

Iteration

- Definition: “Recital or performance a second time; repetition”
- An “iterative” approach is one that converges to a solution a little bit at a time
- The result is not known in advance



Iteration & Incrementalism

- Iterative approaches are incremental
- Incremental approaches are not necessarily iterative
- The waterfall model is neither incremental nor iterative
- Staged delivery is incremental but not iterative
- Evolutionary prototyping is both incremental and iterative
- Some practices derive their power from iteration, some from incrementalism, and some from both

Examples of Iteration & Incrementalism

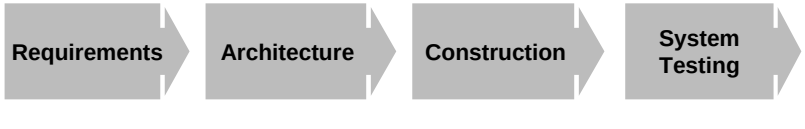
Iterative Approaches

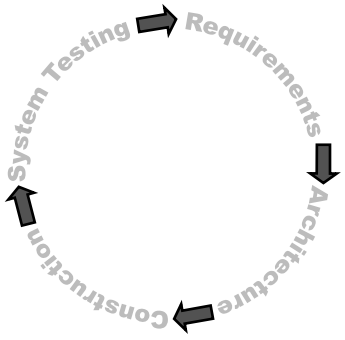
- Short development cycles (e.g., XP, Scrum)
- UI prototyping
- Evolutionary prototyping
- Requirements reviews
- Design reviews
- Project estimation
- Project planning

Incremental Approaches

- Staged delivery
- Design to schedule
- Incremental integration
- Daily builds
- Project planning

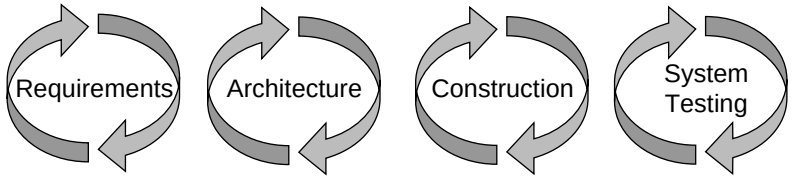
The only two development options in existence are "Iterate Everything" and "Iterate Nothing" (i.e., the Waterfall Model).





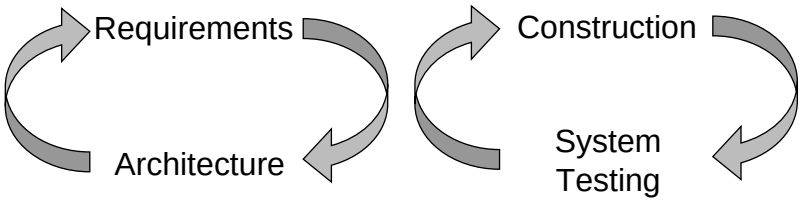
Iteration is a More Flexible Concept

- You can iterate *within* phases...



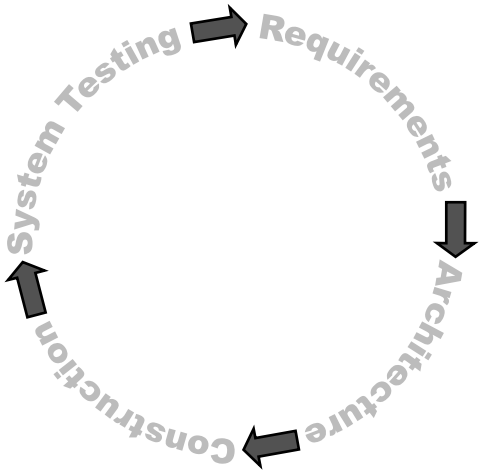
Iteration is a More Flexible Concept

- You can iterate *across* phases...



Iteration is a More Flexible Concept

- You can iterate across *entire dev cycles*



Iteration Applies Both in the Large and in the Small

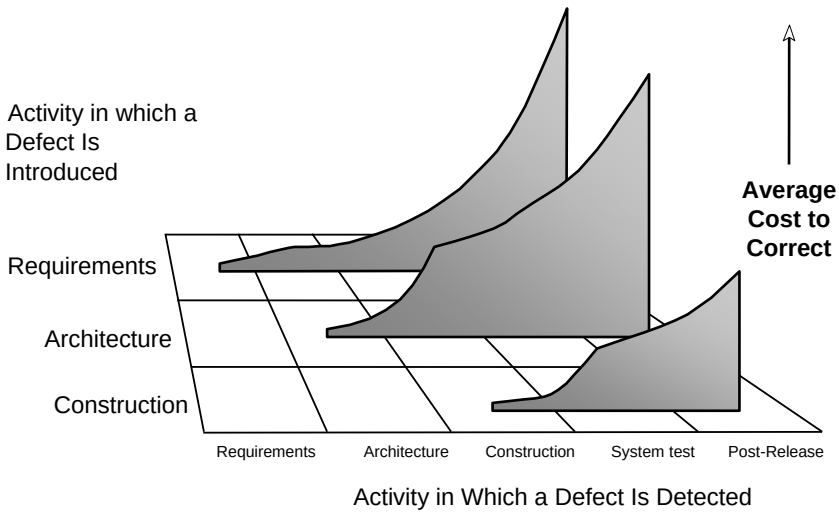
- The degree of iteration can vary from practically 0-100% either *within* or *across* activities

Construx®
Delivering Software Project Success

#3

The Cost To Fix A Defect Increases Over Time

Defect Cost Increase (DCI)



Construx®
Delivering Software Project Success

Agile projects are immune to DCI dynamics.

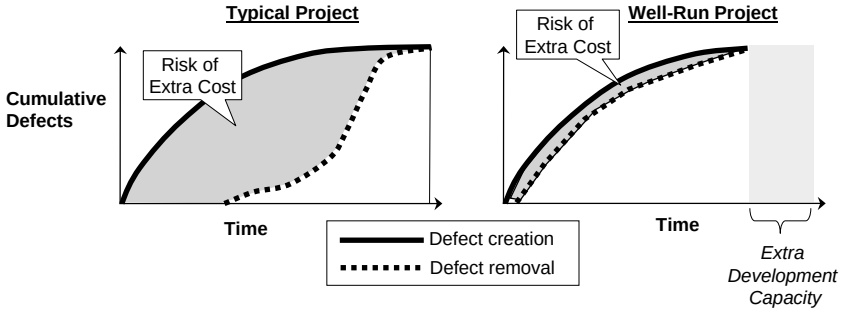
Fortunately, Some Agile Advocates Recognize DCI

"DCI is one of the few empirically verified truths about software development: the sooner you find a defect, the cheaper it is to fix."

–Kent Beck,
Creator of Extreme Programming

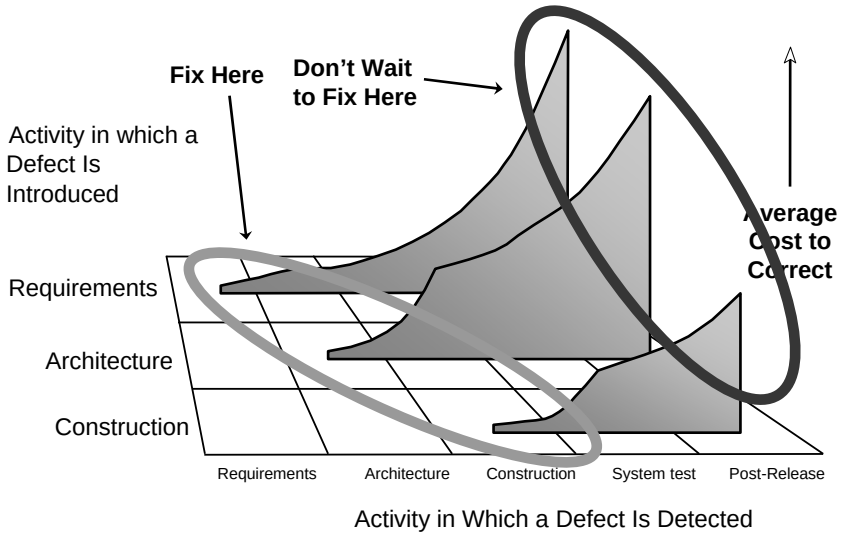
General Principle

- Defect creation is a function of effort
- Defect detection is a function of QA activities
- Goal is to minimize gap between defect insertion and defect detection/correction

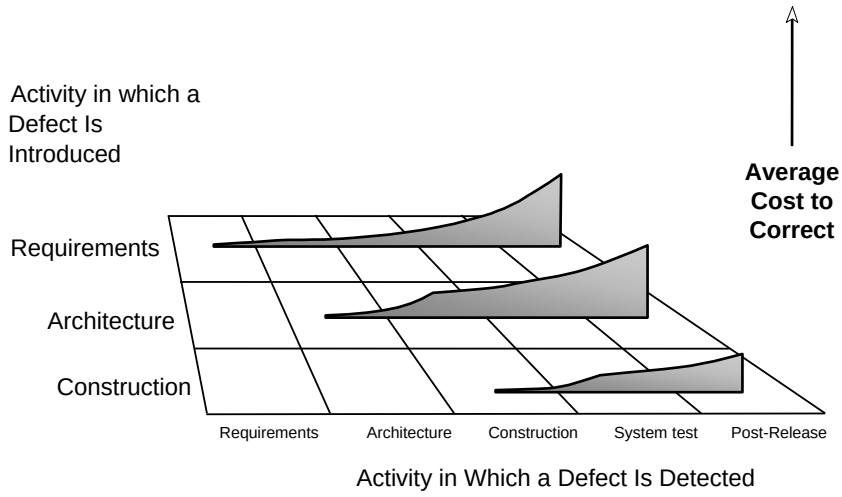


One Solution:

Fix Defects Earlier!



Another Solution:
Reduce Defect Cost Increase!

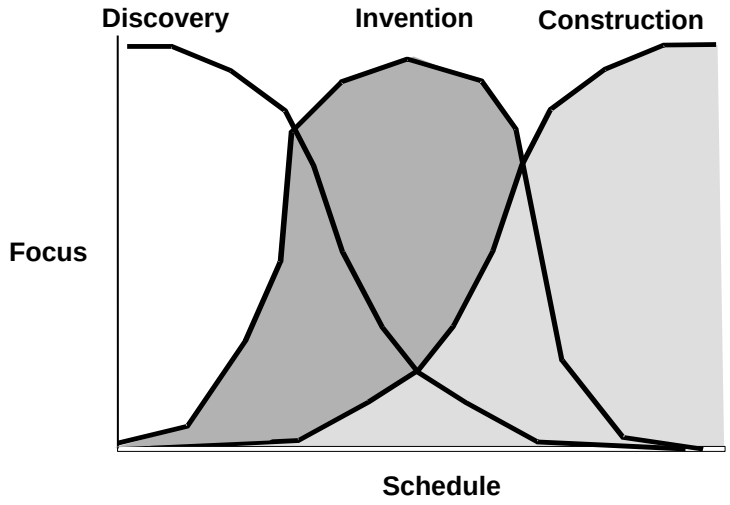


Construx
Delivering Software Project Success

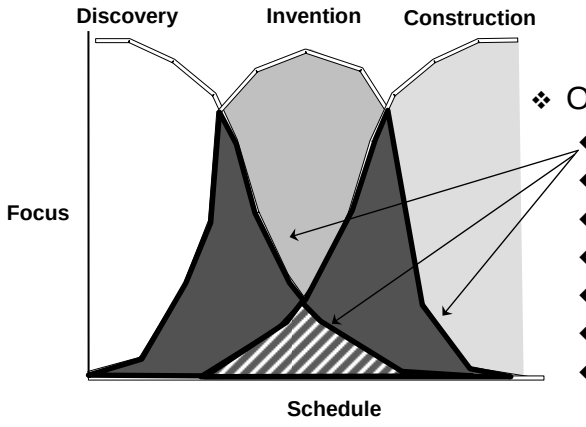
#4

Software Projects Tend to Follow a Predictable Intellectual Flow (Intellectual Phases)

Intellectual Phases



Cost of Overlapping Intellectual Phases



- Overlap =
  - Dependencies
  - Uncertainty
  - Risk
  - Rework
  - Higher costs
  - Longer schedules
  - Lower quality

Construx®
Delivering Software Project Success

Since software projects are "wicked problems," we just have to accept that wickedness.

Construx®
Delivering Software Project Success

Requirements are always changing, and our development practices should be based on the inevitablity of change.

Construx
Delivering Software Project Success

Requirements can be "gathered" (or they just drop out of the sky like manna from heaven)

A Successful Approach to Requirements

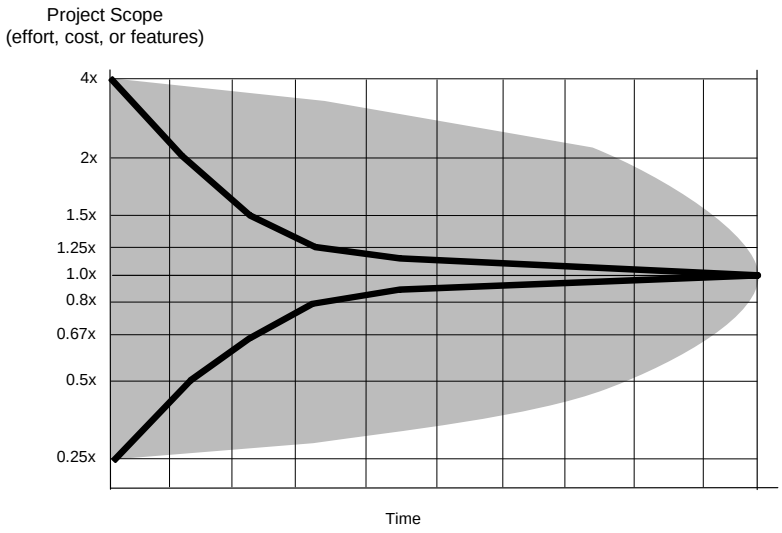
- Active approaches to requirements are *active* not passive
  - "Elicitation"
  - "Discovery"
  - "Investigation"
- The fact that it is possible to discover good requirements does not mean that it's *easy*

Construx®
Delivering Software Project Success

#5

Ability to Create Accurate Software Estimates Can be Improved Over Time (The Cone Of Uncertainty)

Cone of Uncertainty



Implications of the Cone of Uncertainty

- Estimation must be iterative
- Project planning must be incremental
- Estimates should contain descriptions of their inaccuracy

Construx
Delivering Software Project Success

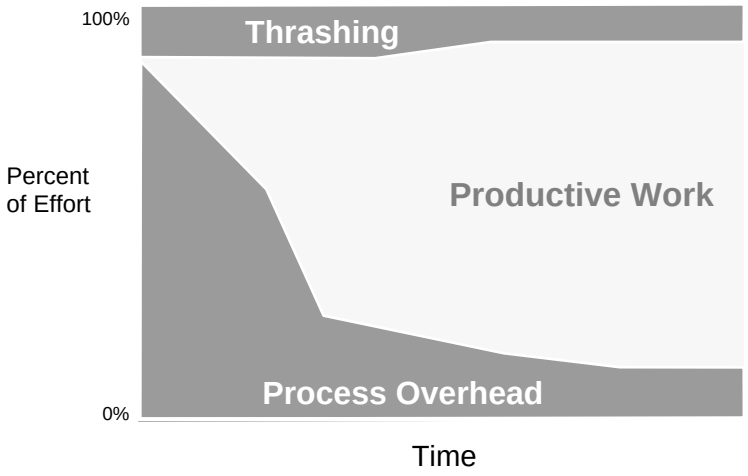
#6

The Most Powerful Form of Reuse is *Full* Reuse

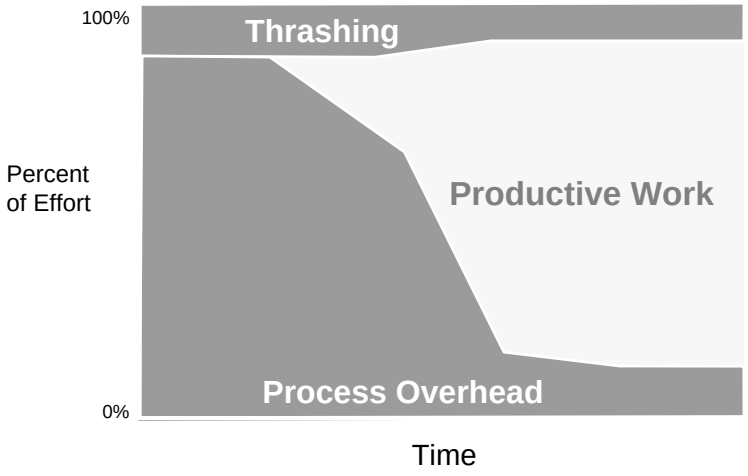
History of Reuse

- First idea was to reuse code
- Later idea was to reuse code + design
- Current idea is to reuse as much as possible, including processes and plans

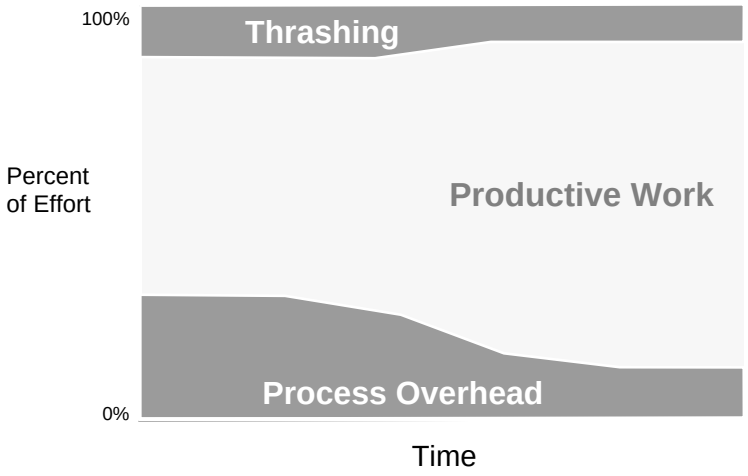
Effect of Adding Process the First Time (What I Wrote in *Software Project Survival Guide*)



Effect of Adding Process the First Time (What I Think Now)



Effect of Adding <u>Reused</u> Processes



What All Can Be Reused?

- Coding standards
- Change control policies
- Estimation procedures
- Formats & outlines of project plan, requirements doc, design docs, QA plan, test plan, etc.
- Checklists for plans, estimates, change control, inspections, QA, etc.
- Roles & responsibilities
- Training
- What are the most prominent current examples of *process* reuse?

Construx®
Delivering Software Project Success

#7

Risk Management Provides Critical Insights into Many Core Software Development Issues

Risk Management Type 1: Extrinsic

- Added on to the project primarily for purposes of risk management
- Examples of Extrinsic Risk Management
  - Top 10 Risks list
  - Risk management plans
  - Risk officer
  - Etc.

Risk Management Type 2: Intrinsic

- Built into the project for other reasons; risk reduction is an additional benefit
- Examples of intrinsic risk management
  - Active project tracking
  - UI Prototyping
  - End-user involvement
  - Incremental delivery
  - Upstream technical reviews
  - Etc.

Risk Insights

- Risk is the key to many tough decisions in project management:
  - What is the best lifecycle model?
  - How much requirements work is enough?
  - How much design work is enough?
  - Can you use junior staff instead of senior staff?
  - Should you do design reviews? Code reviews?
  - How much schedule buffer do you need?

A View of Software Risk Reduction

Typical Relationship between Planned Work and Variable Work:

Planned Work | **Unplanned, Variable Work (typically >50% of total)**

Better Relationship:

Planned Work | **Planned "Overhead"** | **Unplanned, Variable Work**

Construx®
Delivering Software Project Success

Entrepreneurial companies can't be afraid of risk.

Construx®
Delivering Software Project Success

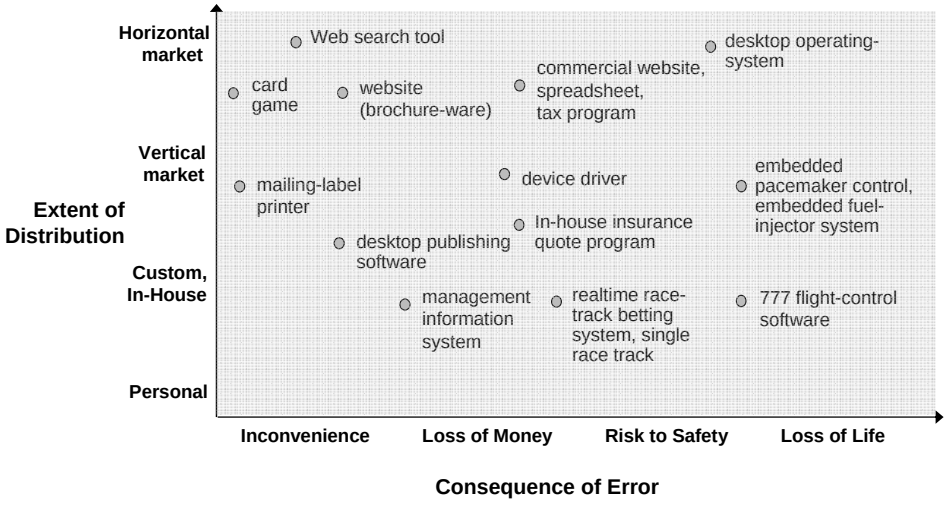
#8

Different Kinds of *Software* Call For Different Kinds of *Software Development* (The "Toolbox")

Construx
Delivering Software Project Success

There is one single development approach that will work best for all projects.

How Could Any One Development Approach Possibly Work?



Construx
Delivering Software Project Success

#9

Software Engineering Body of Knowledge (SWEBOK)

The SWEBOK
(Software Engineering Body of Knowledge)

- Software Configuration Management
- Software Construction
- Software Design
- Software Engineering Management
- Software Engineering Process
- Software Maintenance
- Software Quality
- Software Requirements
- Software Testing
- Software Tools and Methods

What SWEBOK Supports

- Defined, reusable software development processes
- Academic curriculums
- Career development
- Professional certification
- Employment interviewing
- Technical skills inventory

And we're just getting started!

Is the SWEBOK the Ultimate Answer?

To organize something is to understand it.
– Aristotle

"Truth will sooner come out of error than from confusion."
– Francis Bacon

Construx®
Delivering Software Project Success

Conclusions

Construx®
Delivering Software Project Success

- Training
- Consulting
- Tools

sales@construx.com
www.construx.com
+1 (425) 636-0100