Construx®
Software Development Best Practices

Software Development's Classic Mistakes

Steve McConnell
stevemcc@construx.com

www.construx.com

Classic Mistakes Background

- ❖ Original list of classic mistakes was in Steve McConnell's 1996 Book, *Rapid Development*
- ❖ Original list consisted of 36 classic mistakes
- ❖ "Classic Mistakes" provide one view into the most common project risks
  - ◆ Attempt is not to capture all mistakes, just the classic ones

Need for Update

- 11 Years, lots of changes in industry
- Construx work with hundreds of clients (~1000 in the last 5 years)
- List updated in 2007 by Construx consultants

Update to Classic Mistakes

- Additions:
  - Confusing estimates with targets
  - Excessive multi-tasking
  - Assuming global development has a negligible impact on total effort
  - Unclear project vision
  - Trusting the map more than the terrain
  - Outsourcing to reduce cost
  - Letting a team go dark (replaces the previous "lack of management controls")
- 2007: Total of 42 Classic Mistakes

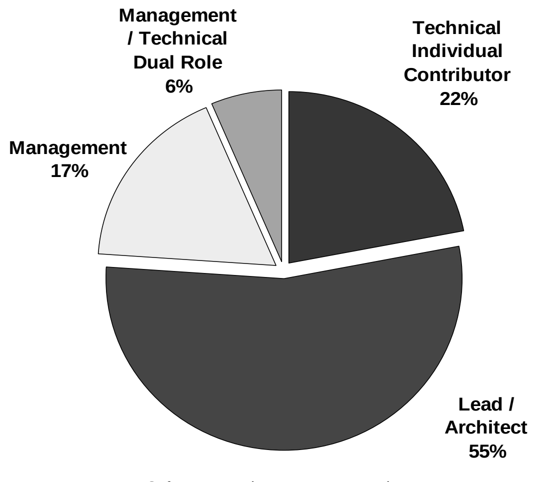
Need for Data

- ❖ Just how "classic" are these classic mistakes?
- ❖ Does the "programming public" think these mistakes are as severe as we do?
- ❖ Survey conducted May 2007-August 2007 to assess frequency of occurrence and impact

Construx

Survey Background

- ❖ Slightly >500 survey respondents



Construx

Construx®
Software Development Best Practices

Survey Results

- ❖Most Frequent Mistakes
- ❖Highest Impact Mistakes
- ❖Worst Overall Classic Mistakes
- ❖Conclusions

Construx®
Software Development Best Practices

Most Frequent Mistakes

Survey Response Key—Frequency

Possible Responses:

- ❖ Almost Always (75%+)
- ❖ Often (50-74%)
- ❖ Sometimes (25-49%)
- ❖ Rarely (<25%)
- ❖ Don't know / N/A

Most Frequent ("Almost Always")

Rank	Classic Mistake	"Almost Always"
1	Noisy, crowded offices	45%
2	Overly optimistic schedules	40%
3	Shortchanged quality assurance	40%
4	Unrealistic expectations	40%
5	Confusing estimates with targets	36%
6	Excessive multi-tasking	34%
7	Insufficient risk management	34%
8	Feature creep	32%
9	Wishful thinking	30%
10	Omitting necessary tasks from estimates	26%

Most Frequent (“Almost Always” or “Often”)

Rank	Classic Mistake	“Almost Always” or “Often”
1	Overly optimistic schedules	77%
2	Unrealistic expectations	73%
3	Excessive multi-tasking	71%
4	Shortchanged quality assurance	70%
5	Noisy, crowded offices	69%
6	Feature creep	69%
7	Wishful thinking	68%
8	Insufficient risk management	68%
9	Confusing estimates with targets	65%
10	Omitting necessary tasks from estimates	61%

Most Frequent-- Modes of “Almost Always” or “Often”

Rank	**Classic Mistake**
1	Overly optimistic schedules
2	Unrealistic expectations
3	Shortchanged quality assurance
4	Noisy, crowded offices
5	Confusing estimates with targets
6	Excessive multi-tasking
7	Feature creep
8	Wishful thinking
9	Insufficient risk management
10	Omitting necessary tasks from estimates
11	Abandoning planning under pressure
12	Shortchanged upstream activities
13	Heroics
14	Lack of user involvement
15	Insufficient planning
16	Planning to catch up later

Most Common Answer is “Almost Always”

Most Common Answer is “Often”

Construx®
Software Development Best Practices

Highest Impact Mistakes

Survey Response Key—Impact

Possible Responses:

- “Catastrophic Impact”
- “Serious Impact”
- “Moderate Impact”
- “Hardly any Impact”
- “Don’t know / N/A”

Highest Impact (mode)

- Zero classic mistakes had a modal impact of "Catastrophic"
- 35 classic mistakes had a modal impact of "Severe"
- 7 classic mistakes had a modal impact of "Moderate"
- Zero classic mistakes had a modal impact of "Hardly any"
- *Survey Limitation:* Fuzzy definitions of catastrophic, severe, moderate, and hardly any

Highest Impact (by number of "Catastrophic" responses)

Rank	Classic Mistake	Percent "Catastrophic" Responses
1	Unrealistic expectations	32%
2	Lack of automated source control	32%
3	Weak personnel	27%
4	Wishful thinking	26%
5	Lack of project sponsorship	26%
6	Outsourcing to reduce cost	25%
7	Politics placed over substance	24%
8	Friction between dev & customers	24%
9	Overly optimistic schedules	24%

Highest Impact (by number of “Catastrophic” and “Serious” responses)

Rank	Classic Mistake	Percent “Catastrophic” and “Serious” Responses
1	Unrealistic expectations	83%
2	Weak personnel	78%
3	Overly optimistic schedules	78%
4	Wishful thinking	76%
5	Shortchanged quality assurance	72%
6	Inadequate design	72%
7	Lack of project sponsorship	71%
8	Confusing estimates with targets	71%
9	Excessive multi-tasking	71%
10	Lack of user involvement	70%

Highest Impact

- 35 of 42 classic mistakes were listed as having “catastrophic” or “serious” impact by more than 50% of respondents

Construx®

Software Development Best Practices

Classic Mistakes Rogues Gallery

Risk Exposure (RE)

- ❖ RE = Severity * Impact
- ❖ Statistically this is the “expected value”
- ❖ RE rankings are approximate at best, given our survey methodology

Clear #1: Unrealistic Expectations

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
73%	83%

"One of the most common causes of friction between developers and their customers or managers is unrealistic expectations. Often customers simply start with unrealistic expectations (which is probably just human nature). Sometimes project managers or developers ask for trouble by getting project approval based on optimistic estimates. A Standish Group survey listed realistic expectations as one of the top five factors needed to ensure the success of an in-house business-software project."

Sources of "Unrealistic Expectations"

Upper Management	69%
Sales & Marketing	51%
Customers	46%
Project / Program Managers	37%
Development Team	15%

Sources of “Unrealistic Expectations”

These responses were consistent across respondents who identified themselves as technical staff, management staff, and the respondents who identified themselves as playing both technical and management roles

#2: Overly Optimistic Schedules

“Often” & “Almost Always” Responses	“Catastrophic” & “Severe” Responses
77%	78%

“The challenges faced by someone building a three-month application are quite different than the challenges faced by someone building a one-year application. Setting an overly optimistic schedule sets a project up for failure by underscoping the project, undermining effective planning, and abbreviating critical upstream development activities such as requirements analysis and design. It also puts excessive pressure on developers, which hurts developer morale and productivity. ”

#3: Shortchanged Quality Assurance

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
70%	72%

"Projects that are in a hurry often cut corners by eliminating design and code reviews, eliminating test planning, and performing only perfunctory testing. It is common for design reviews and code reviews to be given short shrift in order to achieve a perceived schedule advantage. This often results in the project reaching its feature-complete milestone but then still being too buggy to release."

#4: Wishful Thinking

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
68%	76%

"Wishful thinking isn't just optimism. It's closing your eyes and hoping something works when you have no reasonable basis for thinking it will. Wishful thinking at the beginning of a project leads to big blowups at the end of a project. It undermines meaningful planning and can be at the root other problems."

#5: Confusing Estimates with Targets

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
65%	71%

"Some organizations set schedules based purely on the desirability of business targets without also creating analytically-derived cost or schedule estimates. While target setting is not bad in and of itself, some organizations actually refer to the target as the 'estimate,' which lends it an unwarranted and misleading authenticity as a foundation for creating plans, schedules, and commitments."

#6: Excessive Multi-Tasking

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
71%	71%

"When software developers are assigned to more than one project, they must 'task switch' as they change their focus from one project to another. They must get out of 'flow' on one project and into 'flow' on another. Task switching can be a significant factor—some studies have said that each task switch in software development can incur a 5-30 minute downtime as a developer works out of flow on one project and works into flow on the other."

#7: Feature Creep

“Often” & “Almost Always” Responses	“Catastrophic” & “Severe” Responses
69%	61%

“The average project experiences about a 25-percent change in requirements over its lifetime. Such a change produces at least a 25-percent addition to the software effort and schedule, which is often unaccounted for in the project's plans and unacknowledged in the project's status reports.”

#8: Noisy, Crowded Offices

“Often” & “Almost Always” Responses	“Catastrophic” & “Severe” Responses
69%	51%

“About 60 percent of developers report that their work environments are neither sufficiently quiet nor sufficiently private. For many developers, this can prevent concentration and prevent achieving a state of ‘flow’ that is helpful in achieving high levels of productivity. Workers who occupy quiet, private offices tend to perform significantly better than workers who occupy noisy, crowded work bays or cubicles.”

#9: Abandoning Planning Under Pressure

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
59%	67%

"Projects make plans and then routinely abandon them when they run into schedule trouble. This would not be a problem if the plans were updated to account for the schedule difficulties. The problem arises when the plans are abandoned with no substitute, which tends to make the project slide into code-and-fix mode."

#10: Insufficient Risk Management

"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
68%	60%

"Some mistakes have been made often enough to be considered classic mistakes. Other potential problems need to be identified project-by-project through risk management. The most common problem with risk management is not doing any risk management at all. The second most common problem with risk management is not doing *enough* risk management."

Summary of Worst Classic Mistakes

Rank	Classic Mistake	"Often" & "Almost Always" Responses	"Catastrophic" & "Severe" Responses
1	Unrealistic expectations	73%	83%
2	Overly optimistic schedules	77%	78%
3	Shortchanged quality assurance	70%	72%
4	Wishful thinking	68%	76%
5	Confusing estimates with targets	65%	71%
6	Excessive multi-tasking	71%	71%
7	Feature creep	69%	61%
8	Noisy, crowded offices	69%	51%
9	Abandoning planning under pressure	59%	67%
10	Insufficient risk management	68%	60%

Construx®
Software Development Best Practices

Conclusions

Conclusions

- Some classic mistakes are indeed *classic*—with more than half the respondents reporting numerous mistakes that are committed “Almost Always” or “Often”
- Most classic mistakes are indeed *mistakes*—with 35/42 having impacts that are mostly “Catastrophic” or “Severe”
- Some mistakes are made much more frequently than others
- Some mistakes are much more severe than others
- There is a top tier of mistakes that are both common and severe
- Bottom line: “Classic Mistakes” are still a useful way to identify common project problems—and, hopefully, to avoid them!

Contact Information

Construx®
Software Development Best Practices

- **Seminars**
- **Consulting**
- **Software Tools**

stevemcc@construx.com
construx.com
(425) 636-0100