Sources of Scope Risk

Scope risks are most numerous in the Project Experience Risk Information Library (PERIL) database, representing more than one third of the data. Even more important, risks related to scope accounted for nearly half of the total schedule impact. The two broad categories of scope risk in PERIL relate to *changes* and *defects*. By far the most damage was due to poorly managed change (two thirds of the overall scope impact and almost a third of all the impact in the entire database), but all the scope risks represented significant exposure for these projects. While some of the risk situations, particularly in the category of defects, were legitimately "unknown" risks, quite a few of the problems could have been identified in advance and managed as risks. The two major root-cause categories for scope risk are separated into more detailed subcategories:

Scope Root-Cause Subcategories	Definition	Count	Cumulative Impact (Weeks)	Average Impact (Weeks)
Changes: Creep	Any nonmandatory scope change	77	676	8.8
Changes: Gap	Legitimate scope requirements discovered late in project	87	731	8.4
Defects: Software	System or intangible deliverable problems that must be fixed	42	306	7.3
Defects: Hardware	Tangible deliverable problems that must be fixed	38	261	6.9
Defects: Integration	Program-level defects that require scope shifts in projects	13	87	6.7
Changes: Dependency	Scope changes necessary because of external dependencies	13	53	4.1

Scope changes due to gaps were the most frequent, but scope creep changes were the most damaging on average. A Pareto chart of overall impact by type of risk is summarized in Figure 3-1, and a more detailed analysis follows.

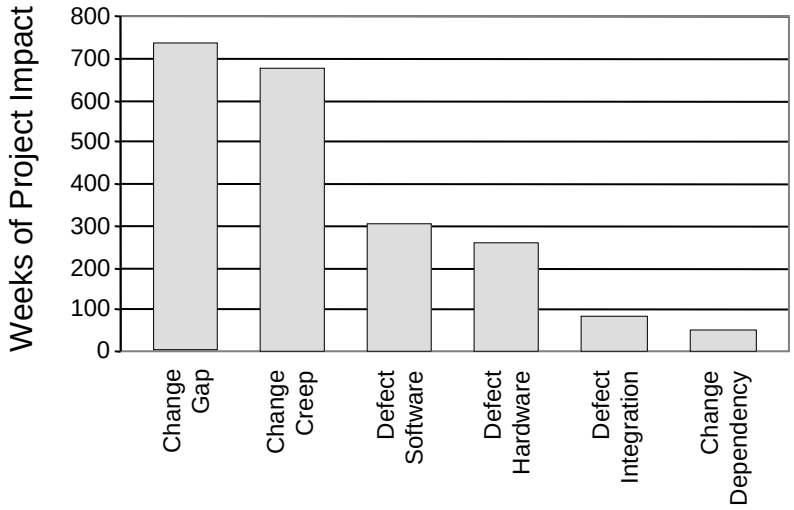

Figure 3-1: Total Project Impact by Scope Root-Cause Subcategories

Change risks

Change happens. Few if any projects end with the original scope intact. Managing scope risk related to change relies on minimizing the loose ends of requirements at project initiation and having (and using) a robust process for controlling changes throughout a project. In the PERIL database, there are three categories of scope change risks: *scope creep*, *scope gaps*, and *dependencies*.

Scope creep was the most damaging type of change risk, resulting in an average schedule slip of nearly 9 weeks. Scope gaps were only slightly less damaging, at well over 8 weeks of slippage, and were also both more common and had greater total impact. Each of these subcategories individually represented about one sixth of all the problems in the PERIL database.

Scope gaps are the result of committing to a project before the project requirements are complete. When legitimate needs are uncovered later in the project, change is unavoidable. Some of the overlooked requirements were a consequence of the novelty of the project, and some were due to customers, managers, team members, or other project stakeholders who were not available at the project start. While some of the scope gaps are probably unavoidable, in most of the cases these gaps were due to incomplete or rushed analysis. A more thorough scope definition and project work breakdown would have revealed the missing or poorly defined portions of the project scope.

Scope creep plagues all projects, especially technical projects. New opportunities, interesting ideas, undiscovered alternatives, and a wealth of other information emerges as the project progresses, providing a perpetual temptation to redefine the project and to make it "better." Some project change of this sort may be justified through clear-eyed business analysis, but too many of these nonmandatory changes sneak into projects because the consequences either are never analyzed or are drastically underestimated. To make matters worse, the purported benefits of the change are usually unrealistically overestimated. In retrospect, much of scope creep delivers little or no benefit. In some particularly severe cases, the changes in scope delay the project so much that the ultimate deliverable has no value; the need is no longer pressing or has been met by other means. Scope creep represents unanticipated additional investment of time and money, because of both newly required effort and the need to redo work already completed. Scope creep is most damaging when entirely new requirements are piled on as the project runs. Such additions not only make projects more costly and more difficult to manage, they can also significantly delay delivery of the originally expected benefits. Managing scope creep requires an initial requirements definition process that thoroughly considers potential alternatives, as well as an effective process for managing specification changes throughout a project.

Scope creep can come from any direction, but one of the most insidious is from inside the project. Every day a project progresses you learn something new, so it's inevitable that you will see things that were not apparent earlier. This can lead to well-intentioned proposals by someone on project team to "improve" the deliverable. Sometimes, scope creep of this sort happens with no warning or visibility until too late, within a portion of the project where the shift seems harmless. Only after the change is made do the real, and sometimes catastrophic, unintended consequences emerge. Particularly on larger, more complicated projects, all changes deserve a thorough analysis and public discussion, with a particularly skeptical analysis of all alleged benefits. Both scope creep and scope gaps are universal and pervasive issues for technical projects.

Scope dependencies are due to external factors that affect the project and are the third category of change risk. (Dependency risks that are primarily due to timing rather than requirements issues are characterized as schedule risks in the database.) Though less frequent in the PERIL database compared with other scope change risks, scope dependencies did represent an average slippage of over a month. Admittedly, some of the cases in the database involved situations that no amount of realistic analysis would have uncovered in advance. Other examples, though, were a result of factors that should not have come as complete surprises. While legal and regulatory changes do sometimes change without notice, a little research will generally provide advance warning. Projects also depend on infrastructure stability, and periodic review of installation and maintenance schedules will reveal plans for new versions of application software, databases, telecommunications, hardware upgrades, or other changes that the project may need to anticipate and accommodate.

Defect risks

Technical projects rely on many complicated things to work as expected. Unfortunately, new things do not always operate as promised or as required. Even normally reliable things may break down or fail to perform as desired in a novel application. Defects represent about a third of the scope risks and about one seventh of all the risks in the PERIL database. The three categories of defect risks related to *software*, *hardware*, and *integration.*

Software problems and hardware failures were the most common types of defect risk in the PERIL database, approximately equal in frequency. They were also roughly equal in impact, with software defects slightly exceeding seven weeks of delay on average and hardware problems a bit less than seven weeks. In several cases, the root cause was new, untried technology that lacked needed functionality or reliability. In other cases, a component created by the project (such as a custom integrated circuit, a board, or a software module) did not work initially and had to be fixed. In still other cases, critical purchased components delivered to the project failed and had to be replaced. Nearly all of these risks are visible, at least as possibilities, through adequate analysis and planning.

Some hardware and software functional failures related to quality or performance standards. Hardware may be too slow, require too much power, or emit excessive electromagnetic interference. Software may be too difficult to operate, have inadequate throughput, or fail to work in unusual circumstances. As with other defects, the definition, planning, and analysis of project work will help in anticipating many of these potential quality risks.

Integration defects were the third type of defect risk in the PERIL database. These defects related to system problems above the component level. Although they were not as common in the database, they were quite damaging. Integration defects caused an average of nearly seven weeks of project slip. For large programs, work is typically decomposed into smaller, related subprojects that can progress in parallel. Successful integration of the deliverables from each of the subprojects into a single system deliverable requires not only that each of the components delivered operates as specified but also that the combination of all these parts functions as a system. All computer users are very familiar with this failure mode. Whenever all the software in use fails to play nicely together, our systems lock up, crash, or report some exotic “illegal operation.” Integration risks, though relatively less common than other defect risks in the PERIL database, are particularly problematic, as they generally occur very near the project deadline and are never easy to diagnose and correct. Again, thorough analysis relying on disciplines such as software architecture and systems engineering is essential to timely identification and management of possible integration risks.

Black swans

The worst of the risks from each category in the PERIL database deserve more detailed attention. Defining the worst 20 percent of the risks based on schedule impact as “black swans,” we’ll explore these “large-impact, hard-to-predict, rare events.” Each of the

"black swan" risks represented at least three months of schedule slip, so they certainly qualify as large impact. They are rare; the PERIL database has an intentional bias in favor of the most serious risks, which are (or at least we hope are) not risks we expect to see frequently. The purpose of this section and the discussions in Chapters 4 and 5 is to make some of these "black swans" easier to predict.

Of the most damaging 127 risks in the PERIL database, 64—just over half—were scope risks. In the database as a whole, the "black swans" accounted for slightly more than half of the total risk impact. The top scope risks exceeded this with nearly 60 percent of the aggregate scope risk impact. The details are:

Scope Risks		Total Impact (Weeks)	"Black Swan" Impact (Weeks)	"Black Swan" Percentage
Changes	Creep	676	427	63%
	Dependency	53	26	49%
	Gap	731	480	66%
Defects	Hardware	261	137	52%
	Integration	87	26	30%
	Software	306	155	51%
Totals		2,114	1,251	59%

As the table shows, the "black swan" scope risks were dominated by change risk, both in terms of quantity and impact. Changes caused about three quarters of the "black swan" risks, by both quantity and total impact. When major change risks occur, their effects are very painful. "Black swan" defect risks were less common as well as somewhat less damaging overall, because recovery from these risks is generally more straightforward.

There were 47 "black swan" scope risks associated with change, dominated by scope gaps with a total of 25. Examples of these were:

- Project manager expected the solution to be one item, but it proved to be four.
- New technology required unanticipated changes in order to function.
- Development plans failed to include all of the 23 required applications.

- End users were too little involved in defining the new system.
- Requirements were understood differently by key stakeholders.
- Scope initially proposed for the project did not receive the upper management sign-off.
- Some countries involved provided incomplete initial requirements.
- Fit/gap analysis was poor.
- The architect determined late that the new design plan would be considerably more complex than expected.
- Lack of consensus on the specifications resulted in late project adjustments.
- When a survey required for the project was assigned to several people in different countries, each assumed someone else would do it but no one did.
- A midproject review turned up numerous additional regulations .
- Manufacturing problems were not seen in the original analysis.

Most of the rest of the change risk "black swans" were attributable to scope creep. Among these 21 risks were:

- Scoping for the project increased substantially after the award was won.
- New technology was introduced late in the project.
- The project team agreed to new requirements, some of which proved to be impossible.
- Late changes were poorly managed.
- The contract required "state of the art" materials, which changed significantly over the project's two-year duration.
- Volume requirements increased late in project, requiring extensive rework.
- Midproject feature revisions had a major impact on effort and schedule, making the product late to market.
- A merger occurred during a company-wide software refresh of all desktops and laptops, adding more systems and hugely complicating the project.

- A system for expense analysis expanded into redesign of most major internal systems.
- One partner on a Web design project expanded scope without communicating or getting approval from others.
- Late change required new hardware and a second phase.
- Application changed midproject to appeal to a prospective Chinese customer (who never bought).
- Project specifications changed, requiring material imported from overseas.

There was a single "black swan" change risk caused by an external dependency: in a pharmaceutical project, a significant study was unexpectedly mandated.

There were fewer "black swans," 17 total, in the scope defect categories. Software and hardware defects each caused eight, and one was a consequence of poor integration. Examples of scope defect risks included:

- Redesign was required in a printer development project that failed to meet print quality goals.
- The system being developed had 20 major defects and 80 additional problems that had to be fixed.
- In user acceptance testing, a fatal flaw sent the deliverable back to development.
- During unit testing, performance issues arose with volume loads.
- Contamination of an entire batch of petri plates meant redoing them all.
- Server crashed with four months of information, none of it backed up, requiring everything to be reentered.
- Hardware failed near the end of a three-month final test, necessitating refabrication and retest.
- Purchased component failed, and continuing the project depended on a brute force, difficult-to-support workaround.
- System tool could not be scaled to a huge Web application.
- A software virus destroyed interfaces in two required languages, requiring rework.

- A purchased learning management system had unanticipated modular complexity.

Identifying scope risks similar to these can expose many potential problems. Reviewing these examples and the additional scope risks from the PERIL database listed in the Appendix can be a good starting point for uncovering possible scope-related problems on your next project.