Suggested List of Risks

Contents

[Purpose 2](#_Toc424308606)

[Reference 2](#_Toc424308607)

[List of Risks 2](#_Toc424308608)

[Schedule Creation 2](#_Toc424308609)

[Organization and Management 2](#_Toc424308610)

[Development Environment 2](#_Toc424308611)

[End-Users 3](#_Toc424308612)

[Customer 3](#_Toc424308613)

[Requirements 3](#_Toc424308614)

[Product 3](#_Toc424308615)

[External environment 4](#_Toc424308616)

[Personnel 4](#_Toc424308617)

[Design and Implementation 5](#_Toc424308618)

[Process 5](#_Toc424308619)

[Infrastructure 5](#_Toc424308620)

# Purpose

This document provides guidance to identify all the risks that may materialize during the execution of the project.

# Reference

Rapid Development-Taming Wild software Schedules, by, Steve McConnell

# List of Risks

## Schedule Creation

* Schedule, resources, and product definition have all been dictated by the customer or upper management and are not in balance
* Schedule is optimistic, "best case" (rather than realistic, "expected case") Schedule omits necessary tasks
* Schedule was based on the use of specific team members, but those team members were not available
* Cannot build a product of the size specified in the time allocated
* Product is larger than estimated
* Effort is greater than estimated (per line of code, function point, module etc.)
* Re-estimation in response to schedule slips is overly optimistic or ignores project history
* Excessive schedule pressure reduces productivity
* Target date is moved up with no corresponding adjustment to the product scope or available resources
* A delay in one task causes cascading delays in dependent tasks
* Unfamiliar areas of the product take more time than expected to design and implement

## Organization and Management

* Project lacks an effective top-management sponsor
* Management or marketing insists on technical decisions that lengthen the schedule
* Management makes decisions that reduce the development team's motivation
* Nontechnical third-party tasks take longer than expected (budget approval, equipment purchase approval, legal reviews, security clearances, etc.)
* Project plans are abandoned under pressure, resulting in chaotic, inefficient development
* Management places more emphasis on heroics than accurate status reporting, which undercuts its ability to detect and correct problems

## Development Environment

* Facilities are not available on time
* Facilities are available but inadequate (e.g., no phones, network wiring, furniture, office supplies, etc.)
* Facilities are crowded, noisy, or disruptive
* Development tools are not in place by the desired time
* Development tools do not work as expected; developers need time to create workarounds or to switch to new tools
* Development tools are not chosen based on their technical merits and do not provide the planned productivity
* Learning curve for new development tool is longer or steeper than expected

## End-Users

* End-user insists on new requirements
* End-user ultimately finds product to be unsatisfactory, requiring redesign and rework
* End-user does not buy into the project and consequently does not provide needed support
* End-user input is not solicited, so product ultimately fails to meet user expectations and must be reworked

## Customer

* Customer insists on new requirements
* Customer review/decision cycles for plans, prototypes, and specifications are slower than expected
* Customer will not participate in review cycles for plans, prototypes, and specifications or is incapable of doing so—resulting in unstable requirements and time-consuming changes
* Customer communication time (e.g., time to answer requirements-clarification questions) is slower than expected
* Customer insists on technical decisions that lengthen the schedule
* Customer micro-manages the development process, resulting in slower progress than planned
* Customer-furnished components are a poor match for the product under development, resulting in extra design and integration work
* Customer-furnished components are poor quality, resulting in extra testing, design, and integration work and in extra customer-relationship management
* Customer-mandated support tools and environments are incompatible, have poor performance, or have inadequate functionality, resulting in reduced productivity
* Customer will not accept the software as delivered even though it meets all specifications

## Requirements

* Requirements have been baselined but continue to change
* Requirements are poorly defined, and further definition expands the scope of the project
* Additional requirements are added
* Vaguely specified areas of the product are more time-consuming than expected

## Product

* Error-prone modules require more testing, design, and implementation work than expected
* Unacceptably low quality requires more testing, design, and implementation work to correct than expected
* Pushing the computer science state-of-the-art in one or more areas lengthens the schedule unpredictably
* Development of the wrong software functions requires redesign and implementation
* Development of the wrong user interface results in redesign and implementation
* Development of extra software functions that are not required (gold-plating) extends the schedule
* Meeting product's size or speed constraints requires more time than expected, including time for redesign and reimplementation
* Strict requirements for compatibility with existing system require more testing, design, and implementation than expected
* Requirements for interfacing with other systems, other complex systems, or other systems that are not under the team's control result in unforeseen design, implementation, and testing
* Operation in an unfamiliar or unproved software environment causes unforeseen problems
* Operation in an unfamiliar or unproved hardware environment causes unforeseen problems
* Dependency on a technology that is still under development lengthens the schedule

## External environment

* Product depends on government regulations, which change unexpectedly Product depends on draft technical standards, which change unexpectedly

## Personnel

* Hiring takes longer than expected
* Task prerequisites (e.g., training, completion of other projects) cannot be completed on time
* Poor relationships between developers and management slow decision making and follow through
* Team members do not buy into the project and consequently do not provide the level of performance needed
* Low motivation and morale reduce productivity
* Lack of needed specialization increases defects and rework
* Personnel need extra time to learn unfamiliar software tools or environment
* Personnel need extra time to learn unfamiliar hardware environment
* Personnel need extra time to learn unfamiliar programming language
* Permanent employees leave before project is complete
* New development personnel are added late in the project, and additional training and communications overhead reduces existing team members' effectiveness
* Team members do not work together efficiently
* Conflicts between team members result in poor communication, poor designs, interface errors, and extra rework
* Problem team members are not removed from the team, damaging overall team motivation
* The personnel most qualified to work on the project are not available for the project
* The personnel most qualified to work on the project are available for the project but are not used for political or other reasons
* Personnel with critical skills needed for the project cannot be found
* Key personnel are available only part time
* Not enough personnel are available for the project
* People's assignments do not match their strengths
* Personnel work slower than expected

## Design and Implementation

* Overly simple design fails to address major issues and leads to redesign and reimplementation
* Overly complicated design requires unnecessary and unproductive implementation overhead
* Poor design leads to redesign and reimplementation
* Use of unfamiliar methodology results in extra training time and in rework to fix first-time misuses of the methodology
* Product is implemented in a low-level language (e.g., assembler), and productivity is lower than expected
* Necessary functionality cannot be implemented using the selected code or class libraries; developers must switch to new libraries or custom-build the necessary functionality
* Code or class libraries have poor quality, causing extra testing, defect correction, and rework
* Schedule savings from productivity enhancing tools are overestimated
* Components developed separately cannot be integrated easily, requiring redesign and rework

## Process

* Amount of paperwork results in slower progress than expected
* Inaccurate progress tracking results in not knowing the project is behind schedule until late in the project
* Upstream quality-assurance activities are shortchanged, resulting in time-consuming rework downstream
* Inaccurate quality tracking results in not knowing about quality problems that affect the schedule until late in the project
* Too little formality (lack of adherence to software policies and standards) results in miscommunications, quality problems, and rework
* Too much formality (bureaucratic adherence to software policies and standards) results in unnecessary, time-consuming overhead
* Management-level progress reporting takes more developer time than expected Half-hearted risk management fails to detect major project risks

## Infrastructure

* Loss of Data due to hardware failure cause the team to recreate the same.