Risk Management

Andrea Polini

Software Project Management
MSc in Computer Science
University of Camerino

Step 6: Identify activity risks

1. Identify and quantify activity based risks
2. Plan risk reduction and contingency measures where appropriate
3. Adjust overall plans and estimate to take account of risks

Risks

__First__, risk concerns future happenings. Today and yesterday are beyond active concern, as we are already reaping what was previously sowed by our past actions. The question is, can we, therefore, by changing our actions today, create an opportunity for a different and hopefully better situation for ourselves tomorrow? This means, __second__, that risk involves change, such as in changes of mind, opinion, actions, or places. . . . [__Third__,] risk involves choice, and the uncertainty that choice itself entails. Thus paradoxically, risk, like death and taxes, is one of the few certainties of life. R. Charette

Risk Management

Risk management is one of the most important activity in project planning. It has to do with

- anticipating risks that might affect the project schedule, budget, or quality of the the software being developed (identify cause-effect relations)
- taking actions to “handle” the risks
- documenting the risks in the plan

Effective risk management makes it easier to cope with problems and to ensure that these do not lead to unacceptable effects. Indeed every plan is based on assumptions and risk management tries to plan for and control the situations where those assumptions become incorrect

Examples

- Causes:
  - staff inexperience
  - lack of top management commitment
  - new technology
  - users uncertain of requirements
- Effects:
  - testing takes longer than planned
  - planned effort and time for activities exceeded
  - project scope increases
  - time delays in getting changes to plans agreed

Risk categories

- Project risks: affect the project schedule or resources
  - Staff turnover: experienced staff will leave the project before it is finished
  - Management change: There will be a change of organizational mgmt with different priorities
  - Requirements change: There will be a larger number of changes to the requirements than anticipated
- Product risks: affect the quality of the of the software being developed
  - Specification delays: Specification of essential interfaces are not available on schedule
  - CASE tool underperformance: CASE tools which support the project do not perform as anticipated
- Business risks: affect the business perspectives for the organization developing or procuring the software
  - Technology change: The underlying technology on which the system is built is superseded by new technology
  - Product competition: A competitive product is marketed before the system is completed

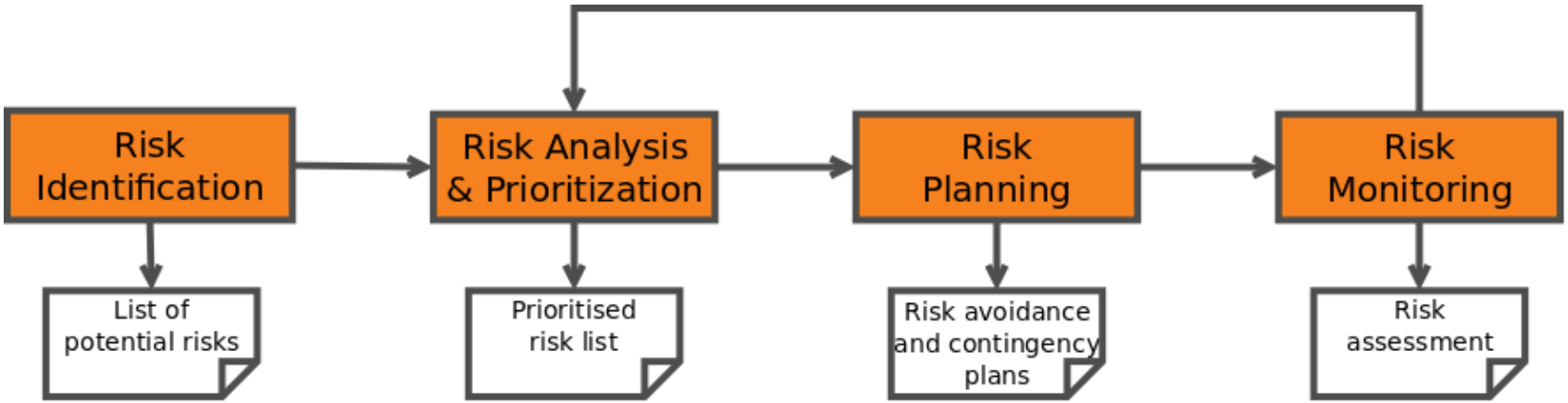
Clearly there is not a clear cut among the different categories

Socio-technical view

- Actors: all the people involved in the development of the software
- Structure: management structure and systems
- Technology: refers both to the technology to support the manufacturing and to the one included in the product itself
- Tasks: relates to the plan of the activities

Risk Management Process

- Risk identification
- Risk Analysis and Prioritization
- Risk Planning
- Risk Monitoring



Risk identification

Two main techniques:

- checklists
- brainstorming

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	Risk reduction techniques
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to require-ments	Stringent change control procedures; high change thresh-old;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agree-ments; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development techni-cally too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Checklist

Barry Boehm proposed the following list of 10 most recurring risks for software dev.:

Risk	**Risk reduction techniques**
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental develop.; recording and analysis of previous project; standardiz. of methods
Developing the wrong software functions	Improved software evaluation; formal specification; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold;incremental development
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost-benefit analysis; prototyping; staff training and development

Brainstorming

Representatives of the main stakeholders meet together, and on the base of their individual knowledge identify problems that might occur

Classification of risks can help:

- Technology risks
  - e.g.: Reused software components contain bugs
- People risks
  - e.g.: impossible to recruit staff with the skills required
- Organizational risks
  - e.g.: change in management, organizational financial problems and budget cut
- Tools risks
  - e.g.: Code generated by tools are inefficient
- Requirements risks
  - e.g.: Customers fail to understand the impact of requirements changes
- Estimation risks
  - e.g.: size of software is underestimated, time for development is underestimated

Brainstorming

Representatives of the main stakeholders meet together, and on the base of their individual knowledge identify problems that might occur

Classification of risks can help:

- Technology risks
  - e.g.: Reused software components contain bugs
- People risks
  - e.g.: impossible to recruit staff with the skills required
- Organizational risks
  - e.g.: change in management, organizational financial problems and budget cut
- Tools risks
  - e.g.: Code generated by tools are inefficient
- Requirements risks
  - e.g.: Customers fail to understand the impact of requirements changes
- Estimation risks
  - e.g.: size of software is underestimated, time for development is underestimated

Questions to drive brainstorming

1. Have top software and customer managers formally committed to support the project?
2. Are end users enthusiastically committed to the project and the system/ product to be built?
3. Are requirements fully understood by the software engineering team and its customers?
4. Have customers been involved fully in the definition of requirements?
5. Do end users have realistic expectations?
6. Is the project scope stable?
7. Does the software engineering team have the right mix of skills?
8. Are project requirements stable?
9. Does the project team have experience with the technology to be implemented?
10. Is the number of people on the project team adequate to do the job?
11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

If any one of these questions is answered negatively, mitigation, monitoring, and management steps should be instituted without fail.

Risk Analysis

List of risks could easily become unmanageable $\rightarrow$ it is necessary to distinguish likely and damaging risks

Risk Exposure

It tries to provide an economical impact (time, money, . . .) for a risk:

- $risk\ exposure = potential\ damage \times probability\ of\ occurrence$

  e.g. recruiting staff from competitors to cover required skills
- Calculated on the base of an accumulated probability

Top ten

Boehm suggests to derive and consider only the 10 top most dangerous risks

4×4 approach

A common way of categorizing risks is using a 4×4 matrix

- Columns: probability that the risk will concretize
  - Low: Less then 10% chance of happening
  - Moderate: 10% - 29%
  - Significant: 30% - 50%
  - High: Greater then 50% chance of happening
- Rows: impact produced on the project if the risk become concrete
  - Insignificant: Less then 10% above budget
  - Tolerable: within 10% and 19%
  - Serious: within 20% and 29%
  - Catastrophic: More then 30% above budget

The project manager select a number between 2 and 8. To be taken into account the sum of indexes for an identified risk should be greater then the defined value.

Risk analysis should be continuously reconsidered and some of the risks will vanish after a while

4×4 approach

A common way of categorizing risks is using a 4×4 matrix

- Columns: probability that the risk will concretize
  - Low: Less then 10% chance of happening
  - Moderate: 10% - 29%
  - Significant: 30% - 50%
  - High: Greater then 50% chance of happening
- Rows: impact produced on the project if the risk become concrete
  - Insignificant: Less then 10% above budget
  - Tolerable: within 10% and 19%
  - Serious: within 20% and 29%
  - Catastrophic: More then 30% above budget

The project manager select a number between 2 and 8. To be taken into account the sum of indexes for an identified risk should be greater then the defined value.

Risk analysis should be continuously reconsidered and some of the risks will vanish after a while

Risk Planning

Risk Planning has to do with the definition of strategies to manage the identified risks. In particular the following choices are possible:

- risk acceptance: do-nothing
- risk avoidance: when possible put in place actions so that the risk condition cannot occur
- risk reduction and mitigation
- risk transfer: transfer the risk to another person or organization (e.g.: outsourcing)

Reduction

- Reduction: put in place actions to reduce the probability that the risk condition occurs
  e.g.: data lost $\rightarrow$ make a back-up copy

Deciding on risk reduction actions

Counter-measure to reduce risks are take when they are cost effective

$$Risk\ Reduction\ Leverage = (RE_{before} - RE_{after})/\ cost\ of\ risk\ reduction$$

The ratio should be greater than 1

Mitigation

- Mitigation: put in place actions so that the impact is reduced
  e.g.: back-up copies to reduce the impact of possible data corruption

Examples of mitigation strategies:

Risk	Strategy
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each others jobs
Requirements changes	Derive traceability information to more easily assess the impact of changes
Organizational re-structuring	Prepare a briefing document for senior Mgmt showing how the project is making a very important contribution to the goals of the business
Underestimated development time	Investigate use of COTS and/or program generator

Mitigation

- Mitigation: put in place actions so that the impact is reduced
  e.g.: back-up copies to reduce the impact of possible data corruption

Examples of mitigation strategies:

Risk	**Strategy**
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each others jobs
Requirements changes	Derive traceability information to more easily assess the impact of changes
Organizational re-structuring	Prepare a briefing document for senior Mgmt showing how the project is making a very important contribution to the goals of the business
Underestimated development time	Investigate use of COTS and/or program generator

Mitigation

- Mitigation: put in place actions so that the impact is reduced
  e.g.: back-up copies to reduce the impact of possible data corruption

Examples of mitigation strategies:

Risk	**Strategy**
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each others jobs
Requirements changes	Derive traceability information to more easily assess the impact of changes
Organizational re-structuring	Prepare a briefing document for senior Mgmt showing how the project is making a very important contribution to the goals of the business
Underestimated development time	Investigate use of COTS and/or program generator

Mitigation

- Mitigation: put in place actions so that the impact is reduced
  e.g.: back-up copies to reduce the impact of possible data corruption

Examples of mitigation strategies:

Risk	**Strategy**
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each others jobs
Requirements changes	Derive traceability information to more easily assess the impact of changes
Organizational re-structuring	Prepare a briefing document for senior Mgmt showing how the project is making a very important contribution to the goals of the business
Underestimated development time	Investigate use of COTS and/or program generator

Mitigation

- Mitigation: put in place actions so that the impact is reduced
  e.g.: back-up copies to reduce the impact of possible data corruption

Examples of mitigation strategies:

Risk	**Strategy**
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each others jobs
Requirements changes	Derive traceability information to more easily assess the impact of changes
Organizational re-structuring	Prepare a briefing document for senior Mgmt showing how the project is making a very important contribution to the goals of the business
Underestimated development time	Investigate use of COTS and/or program generator

Contingency Plan

A contingency plan reports the actions to take in order to cope with the concrete emergence of a risk. It is generally developed for risks with a particularly high impact

Risk Mgmt - Risk register

RISK RECORD					
Risk id		Risk Title			
Owner		Date raised		Status	
Risk Description					
Impact description					
Recommended risk mitigation					
Probability/Impact values					
	Probability	Impact			
		Cost	Duration	Quality	
Pre-mitigation					
Post-mitigation					
Incident/action history					
Date	Incident/Action	Actor	Outcome/comment		

Risk Monitoring

Monitoring requires to regularly assess each of the identified risks to decide whether or not the risk is becoming more or less probable and whether the effects of the risk have changed. Often factors are difficult to observe directly, so look for other indicators.

Example of possible general indicators:

Risk type	**Potential Indicator**
Technology	Late delivery of hardware or support software, many reported technology problems
People	Poor staff morale, poor relationships amongst team members, job availability
Organizational	Organizational gossip, lack of action by senior management
Tools	Reluctance by team members to use tools, complaints about CASE tools, demand for higher-powered workstations
Requirements	Many requirements change requests, customer complaints
Estimation	Failure to meet agreed schedule, failure to clear reported defects

Evaluating risks to the schedule

PERT - Program Evaluation and Review Technique

This is a technique that takes into account the uncertainties in the duration of the activities within a project in order to define the schedule

An interesting drawback of using method like PERT is that in practice there is a tendency for developers to work on the schedule even if they could end the activity before

PERT

The PERT strategy requires three different estimation for each activity in order to derive the an estimation for the expected duration (t_e):

- Most likely time (m)
- Optimist time (a)
- Pessimistic time (b)

The expected duration is then computed as:

$$t_e = \frac{a+4m+b}{6}$$

Example

Let's reconsider the example of a set of activities on which precedence relation is defined as follows:

	Activity	**Duration (weeks)**	**Precedents**
A	Hardware selection	6	
B	System Configuration	4	
C	Install Hardware	3	A
D	Data Migration	4	B
E	Draft Office Procedures	3	B
F	Recruit Staff	10	
G	User Training	3	E,F
H	Install and Test System	2	C,D

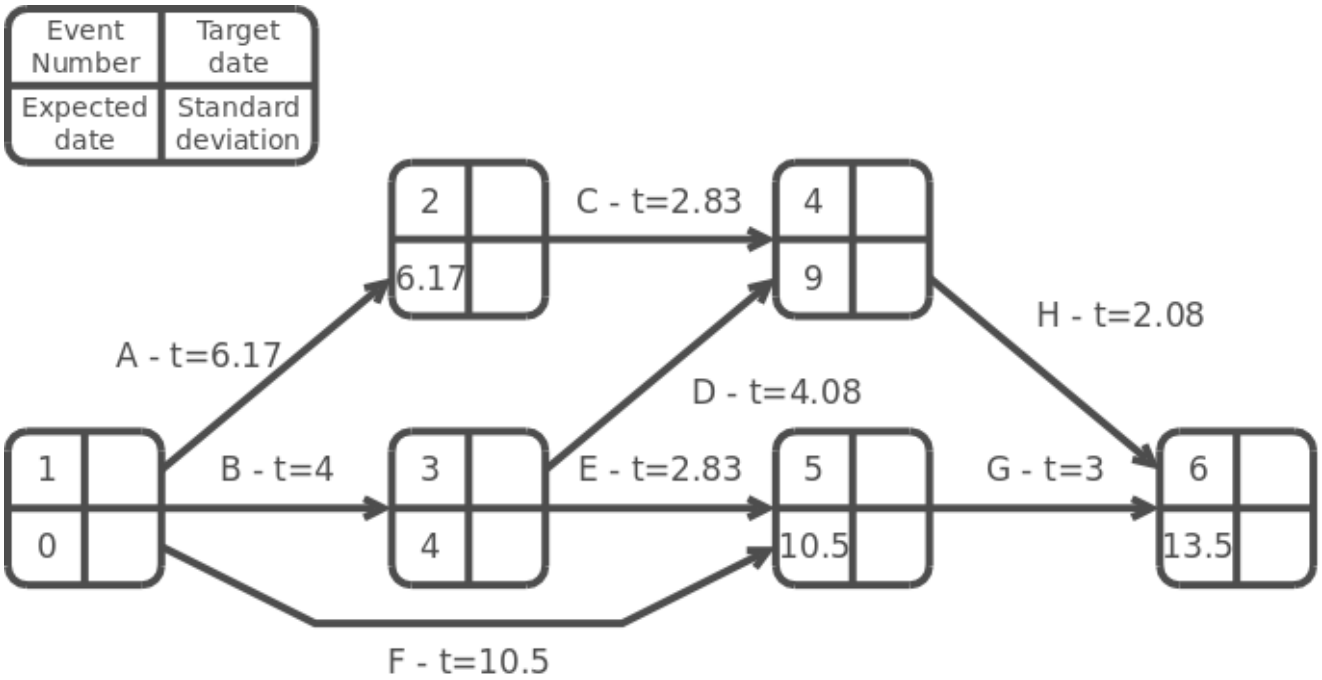
Example - PERT version

Now let's add information to apply the PERT approach:

Activity (Precedents)	Activity Durations (weeks)				
	Optimistic (a)	Most likely (b)	Pessimistic (b)	Expected te	Standard deviation (s)
A	5	6	8	6,16	
B	3	4	5	4	
C (A)	2	3	3	2,83	
D (B)	3,50	4	5	4,08	
E (B)	1	3	4	2,83	
F	8	10	15	10,5	
G (E,F)	2	3	4	3	
H (C,D)	2	2	2,5	2,08	

PERT network

The following PERT network results from the considered data after a "forward pass" step:



Now we say "we expect to complete the project by . . . "

Activity standard deviation

Probability Theory

- What is a Random variable?
- What is a distribution function for a Random variable?
- What is the expectation for a Random Variable?
- What is the standard deviation?

Computing the standard deviation (s) it is possible to derive a quantitative measure of uncertainty. This is given by the formula:

$$s = \frac{b-a}{6}$$

s can be used as a ranking measure of the degree of uncertainty or risk for each activity

Example - Standard deviations

Now let's add information for the *s* values:

Activity (Precedents)	Activity Durations (weeks)				
	Optimistic (a)	Most likely (b)	Pessimistic (b)	Expected te	Standard deviation (s)
A	5	6	8	6,16	0,5
B	3	4	5	4	0,33
C (A)	2	3	3	2,83	0,16
D (B)	3,50	4	5	4,08	0,25
E (B)	1	3	4	2,83	0,5
F	8	10	15	10,5	1,16
G (E,F)	2	3	4	3	0,33
H (C,D)	2	2	2,5	2,08	0,08

s can be used to rank activities according to their degree of risk. In our case activity F needs particular attention while C should cause little concern

Uncertainties in meeting targets

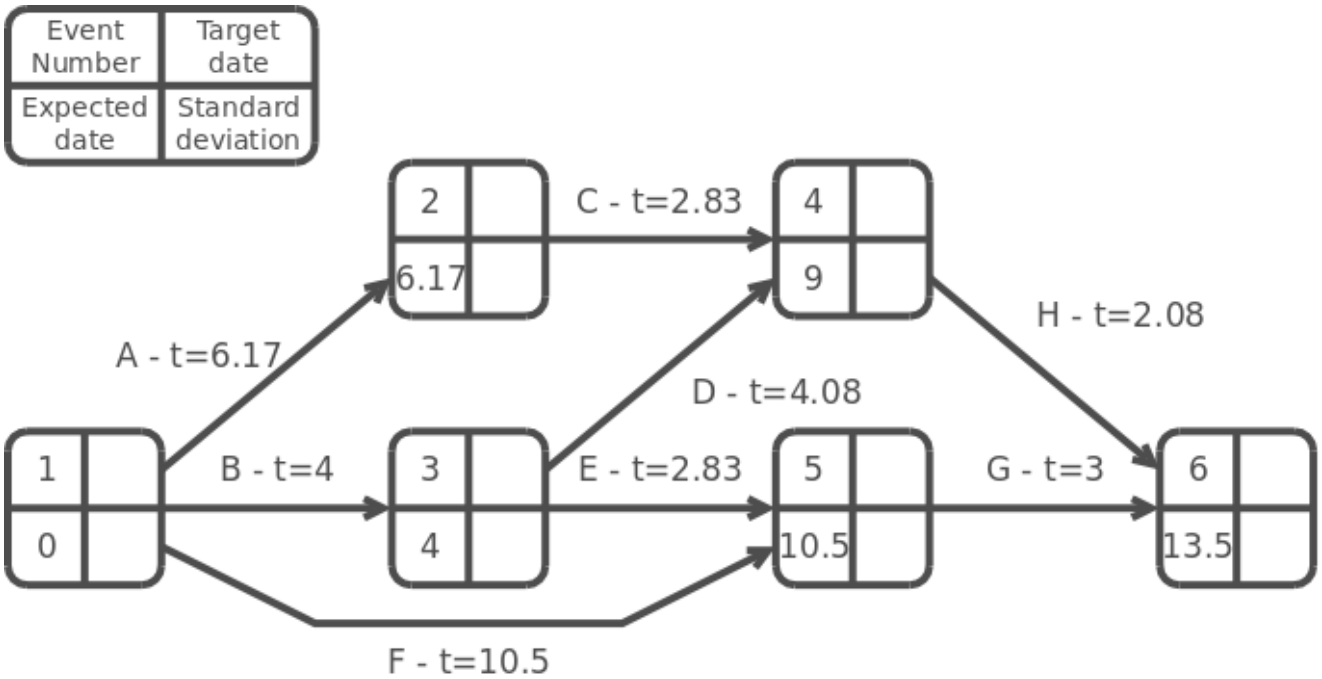
PERT diagrams include information permitting to derive the likelihood of meeting a target date. This is related to the inclusion of standard deviation values in the network.

Assessment can be done using a three step procedure:

- calculate the standard deviation of each project event
- calculate the z value for each event that has a target date
- convert z values to a probabilities

PERT Network with s values

The following PERT network results from the data after a forward pass for s values:

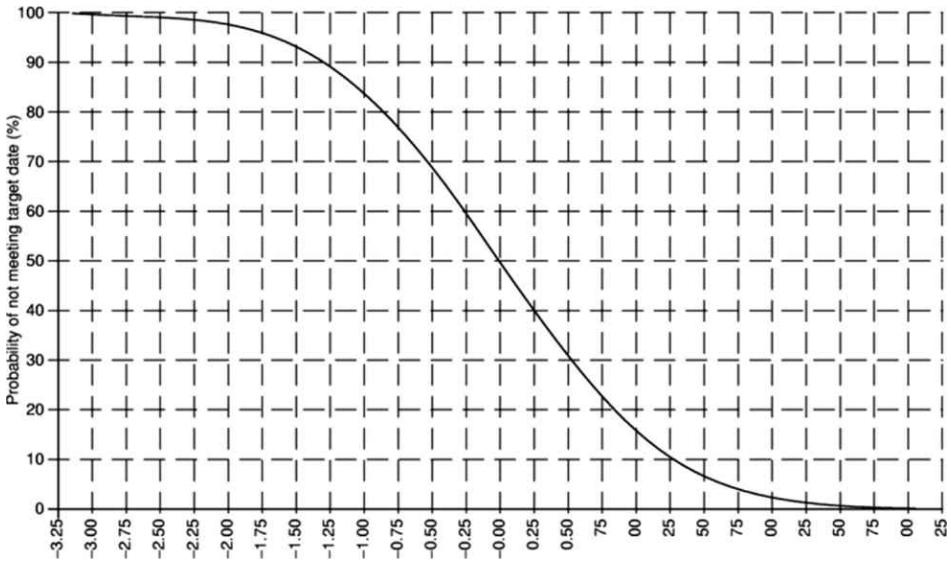


z

The z value is calculated for each node that has a target date. It is calculated using the formula:

$$z = \frac{T - t_e}{s}$$

Given the value of z there exist graphs that permit to transform it in a probability of not meeting the target date



In order to derive a more reliable estimation you sketch a set of workpackages (WPs) and tasks needed in order to complete the project. WPs and tasks are detailed in the table that includes information concerning the dependencies among the tasks.

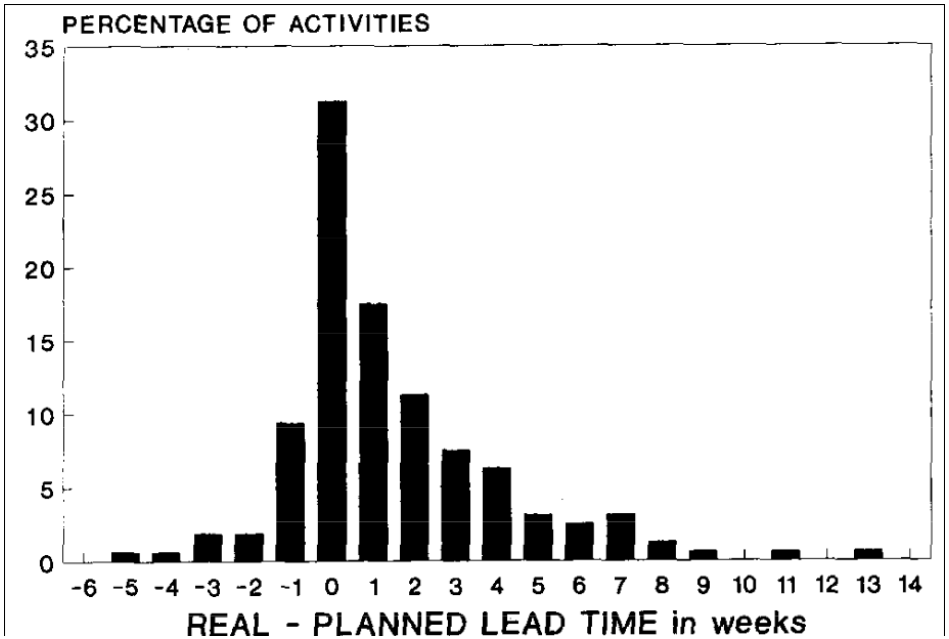
Activity (Precedents)	Activity Durations (weeks)				
	Optimistic (a)	Most likely (m)	Pessimistic (b)	Expected te	Standard deviation (s)
T1.1	1	2	4	2,17	0,5
T1.2	1	2	4	2,17	0,5
T1.3	4	8	13	8,17	1,5
T2.1 (T1.1, T1.2)	8	12	17	12,17	1,5
T2.2 (T1.2, T1.3)	3	4	4,5	3,92	0,25
T2.3 (T1.1, T1.3)	2	4	6	4	0,66
T3.1 (T2.1,T2.2)	7	8	9	8	0,33
T3.2 (T2.1,T2.2)	4	12	10	10,33	1
T3.3 (T2.3,T3.1)	8	12	13	11,5	0,83
T3.4 (T2.3)	14	20	25	19,83	1,83
T4.1 (T3.2,T3.3)	3	4	5	4	0,33
T4.2 (T3.2,T3.1)	3	4	5	4	0,33
T4.3 (T4.1,T4.2)	6	8	11	8,17	0,83
T4.4 (T3.4)	16	20	24	20	1,33

Apply the PERT approach to:

- Compute the duration of the project
- Provide the probability of successfully terminating task T4.2 within 35 weeks, according to the dependencies and the corresponding path ending with activity 4.2
- Provide the probability of finishing the project after one year (52 weeks)
- Provide the probability of finishing the project after one year (56 weeks)
- Provide the ordered list of activities belonging to the critical path where the value for the time is the one specified by t_e.

Observation Respect of Deadline

van Genuchten studied the execution of 160 complex projects and it resulted that the real closing time of activities with respect to planned ones are distributed according to the following graph:



Michiel van Genuchten, "*Why is Software Late? An Empirical Study of Reasons For Delav in Software Development*", IEEE Transaction on Software Engineering, vol. 17 n. 6 pp. 582-590

Distribution is not in accordance to a Gauss one. This fact Can be considered as a confirmation of the Parkinson's Law – *"work expands so as to fill the time available for its completion"*

Critical Chain

The critical chain approach tries to solve the issue, considering as target date the one in which it is estimated that the probability to finish the activity is 50%.
Safe planning could consider the probability of getting the activity by at least 0.95. In some case smaller probability values are considered.

In order to derive a more reliable estimation you sketch a set of workpackages (WPs) and tasks needed in order to complete the project. The table also includes information concerning the dependencies among the tasks.

Activity	Activity Durations (weeks)				x
(Precedents)	Optimistic (a)	Most likely (m)	Pessimistic (b)	Expected te	Standard deviation (s)
T1.1	1	2	5	2,33	0,66
T1.2	1	2	4	2,17	0,5
T1.3 (T1.1,T2.1)	4	8	13	8,17	1,5
T2.1	8	12	17	12,17	1,5
T2.2 (T1.1, T1.2)	3	4	4,5	3,92	0,25
T2.3 (T1.1, T1.2)	2	4	6	4	0,66
T3.1 (T2.1,T2.2)	7	8	9	8	0,33
T3.2 (T2.1,T2.3)	4	12	10	10,33	1
T3.3 (T2.2,T2.3,T3.1)	8	12	13	11,5	0,83
T4.1 (T3.2,T3.3)	3	4	5	4	0,33
T4.2 (T3.1,T3.3)	3	4	5	4	0,33
T4.3 (T4.1,T4.2)	6	8	12	8,33	1

Apply the PERT approach to:

- Provide the probability of successfully terminating task T3.2 within 22 weeks, according to the dependencies and the corresponding path ending with activity 3.2
- Provide the probability of finishing the project after 44 weeks
- make a plan with the 0.95 probability of respecting the deadlines

Use of latest start dates

The Critical chain considers tow value for the duration of an activity at the one with at least 0.5 accumulated probability, and a safe one:

- the most likely estimate can be defined according to a PERT analysis or otherwise derived (e.g. proposed by developers, experts etc.), the safe deadline can correspond to an accumulated probability of 95% or more on the PERT, or again otherwise derived
- comfort zones are generally defined as the difference between the two values
- Activities are moved ahead to end of the period (latest start).
  - people are not distracted or moved on other activities
- a project buffer is inserted at the end of the project.
  - It will last 50% of the removed comfort zones
- Subsidiary chains of activities that feed into the critical chain are closed with feeding buffers
  - The feeding buffer is set to 50% of the removed comfort zones from the subsidiary chain

Let's consider the previous example with comfort zone equal to 95% of probability of getting the activity done

Use of latest start dates

The Critical chain considers tow value for the duration of an activity at the one with at least 0.5 accumulated probability, and a safe one:

- the most likely estimate can be defined according to a PERT analysis or otherwise derived (e.g. proposed by developers, experts etc.), the safe deadline can correspond to an accumulated probability of 95% or more on the PERT, or again otherwise derived
- comfort zones are generally defined as the difference between the two values
- Activities are moved ahead to end of the period (latest start).
  - people are not distracted or moved on other activities
- a project buffer is inserted at the end of the project.
  - It will last 50% of the removed comfort zones
- Subsidiary chains of activities that feed into the critical chain are closed with feeding buffers
  - The feeding buffer is set to 50% of the removed comfort zones from the subsidiary chain

Let's consider the previous example with comfort zone equal to 95% of probability of getting the activity done

Resources

Study material can be found here:

Bob Hughes and Mike Cotterell
Software Project Management, 5th Ed.
McGraw-Hill, 2009

- Chapter 7 - Risk Management

Ian Sommerville
Software Engineering 10^{th} Ed.
Addison Wesley 2016.

- Sect. 5.4 - Risk Management (introductory notions)