Topic PM
Project Management

DAAD project

"Joint Course on Software Engineering"

**Humboldt University of Berlin, University of Novi Sad, University of Plovdiv,
University of Skopje, University of Belgrade, University of Niš, University of Kragujevac**

**Parts of this Chapter use material from the textbook
E. J. Braude, "Software-Engineering: an Object-Oriented perspective", Wiley, 2001**

Version: Dec. 14, 2002 (D xx. xx, xxxx)

PM. Project Management

- a) Introduction
- b) Variables of project management
- c) People management
- d) Risk management
- e) Final issues

What is project management?

Project management is the discipline of organizing and managing resources in such a way that these resources deliver all the work required to complete a project within defined scope, time, and cost constraints

- Project
  → *A project is a temporary and one-time endeavor undertaken to create a unique product or service*
- Process
  → *Processes, or operations, in contrast, are permanent or ongoing work to create the same product or service over-and-over again*
- Solution
  → *The management of these two systems is often very different and requires varying technical skills and philosophy, hence requiring the development of project management*

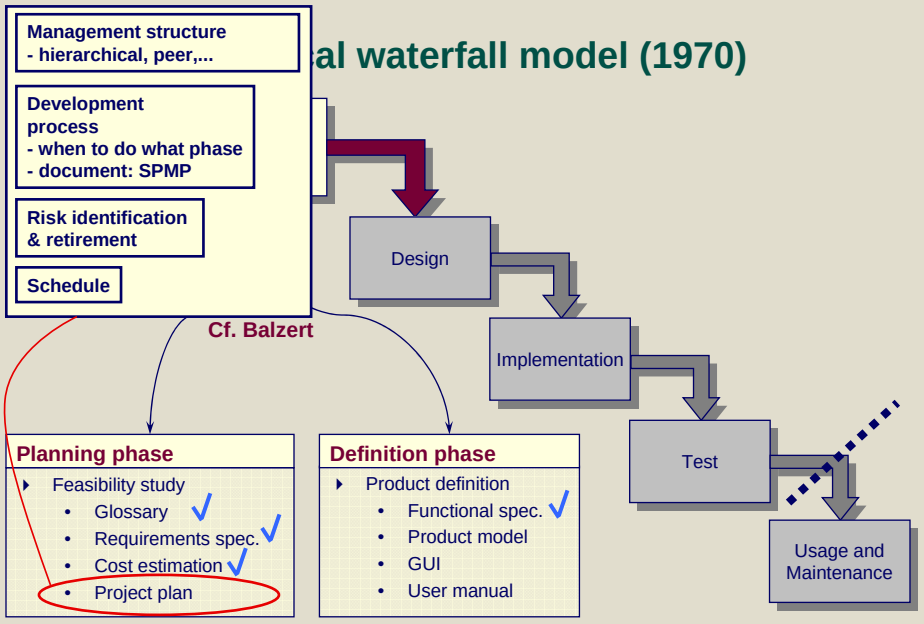
The project manager

- *Project management is often the responsibility of an individual project manager*
- *The individual seldom participates directly in the activities that produce the end result*
- *He rather strives to maintain the progress and productive mutual interaction of various parties in such a way that overall risk of failure is reduced*

Challenges of project management

- The first challenge
  → *Ensuring that a project is delivered within the defined constraints*
- The second challenge
  → *Optimized allocation and integration of the inputs needed to meet those pre-defined objectives*
  → *More ambitious*
- Carefull selection of activities
  → *To optimaly use resources like*
  → *time, money, people, communication, quality, risk, etc*

al waterfall model (1970)



D. Design patterns

a) Introduction

b) Variables of project management

c) People management

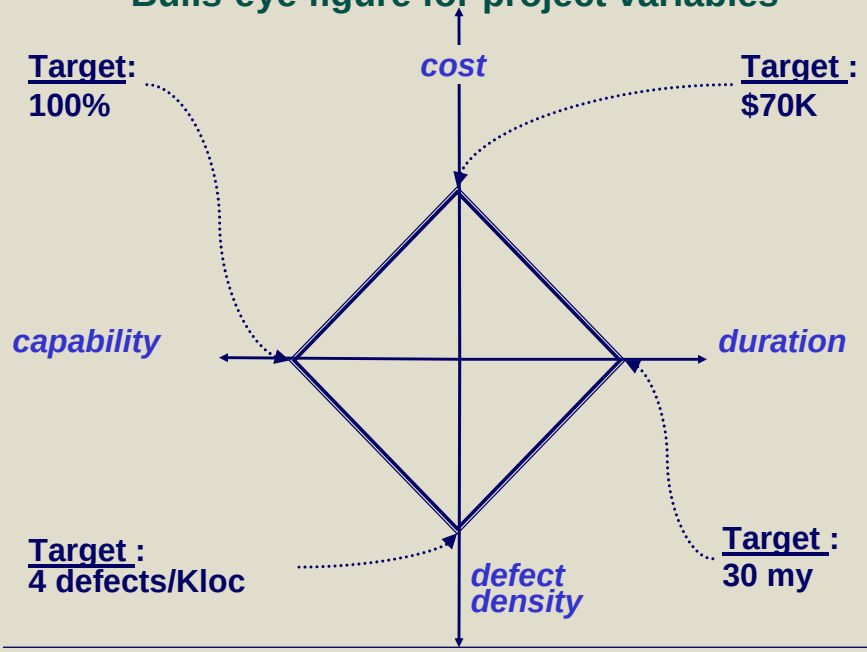
d) Risk management

e) Final issues

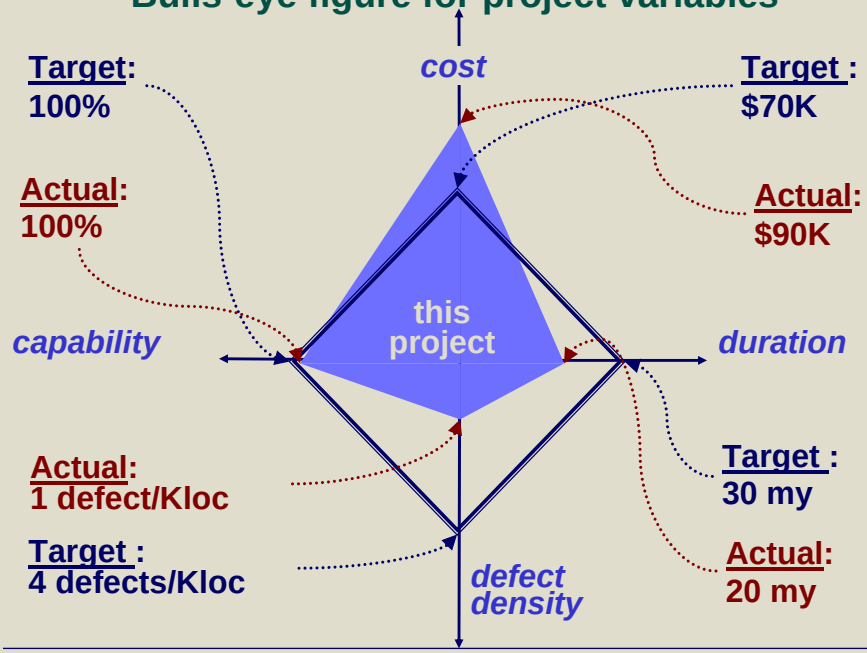
Variables of project management

- The total cost of the project,
  - e.g., increase expenditures
- The capabilities of the product,
  - e.g., subtract from a list of features
- The quality of the product,
  - e.g., increase the mean time between failure
- The date on which the job is completed.
  - e.g., reduce the schedule by 20%
  - e.g., postpone project's completion date one month

Bulls-eye figure for project variables



Bulls-eye figure for project variables



Road map for project management

1. **Understand project content, scope, & time frame**
2. **Identify development process**
   (*methods, tools, languages, documentation and support*)
3. **Determine organizational structure**
   (organizational elements involved)
4. **Identify managerial process**
   (responsibilities of the participants)
5. **Develop schedule**
   (*times at which the work portions are to be performed*)
6. **Develop staffing plan**
7. **Begin risk management**
8. **Identify documents to be produced**
9. **Begin process itself**

D. Design patterns

a) Introduction

b) Variables of project management

c) People management

d) Risk management

e) Final issues

People in the project management

- People are an organisation's most important assets
- The tasks of a manager are essentially people oriented. Unless there is some understanding of people, management will be unsuccessful
- Software engineering is primarily a cognitive activity. Cognitive limitations effectively limit the software process

Management activities

- Problem solving (using available people)
- Motivating (people who work on a project)
- Planning (what people are going to do)
- Estimating (how fast people will work)
- Controlling (people's activities)
- Organising (the way in which people work)

Truths and “truths” about people management

- Brooks, “Mythical Man-Month”
  - Putting more people on a failing software project may not help and even make it worse
- DeMarco, Lister, “Peopleware”
  - If people could manage software project over again, they would do it differently
  - Take the worker’s estimate and double it.
  - Keep pressure on.
  - Don’t let people work at home, they’ll goof off.

People management

- Enterprise perspective
  - General, business oriented
- Management perspective
  - Mixture of business concerns and people interests
  - Ensure that that technical efforts are directed appropriately
  - Leadership
- Engineers’ perspective

Motivation

- An important role of a manager is to motivate the people working on a project
- Motivation is a complex issue but it appears that their are different types of motivation based on
  - Basic needs (e.g. food, sleep, etc.)
  - Personal needs (e.g. respect, self-esteem)
  - Social needs (e.g. to be accepted as part of a group)
- Motivations depend on satisfying needs

Need satisfaction

- Social
  - Provide communal facilities
  - Allow informal communications
- Esteem
  - Recognition of achievements
  - Appropriate rewards
- Self-realization
  - Training - people want to learn more
  - Responsibility

Personality types

- Task-oriented.
  - The motivation for doing the work is the work itself
- Self-oriented.
  - The work is a means to an end which is the achievement of individual goals - e.g. to get rich, to play tennis, to travel etc.
- Interaction-oriented
  - The principal motivation is the presence and actions of co-workers. People go to work because they like to go to work

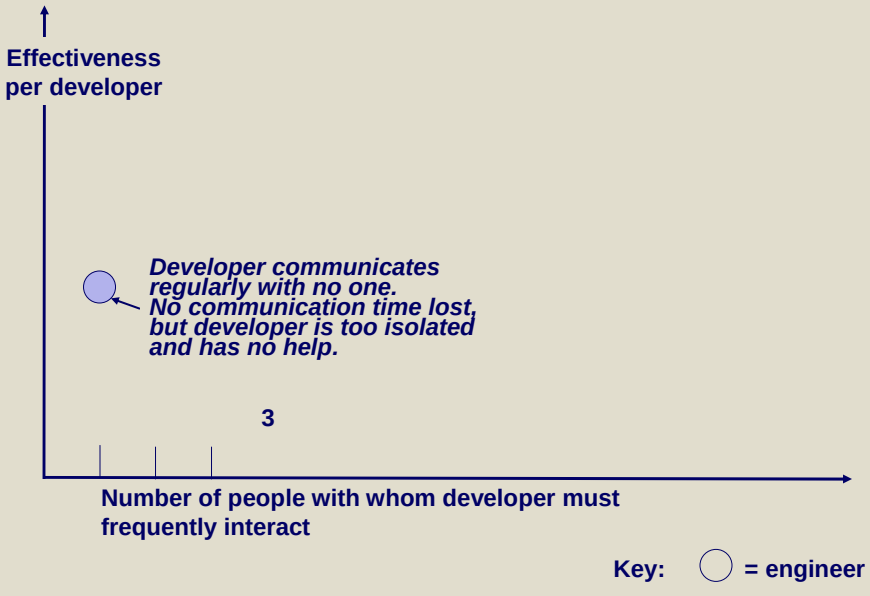
Group working

- Most software engineering is a group activity
  - The development schedule for most non-trivial software projects is such that they cannot be completed by one person working alone
- Group interaction is a key determinant of group performance
- Flexibility in group composition is limited
  - Managers must do the best they can with available people

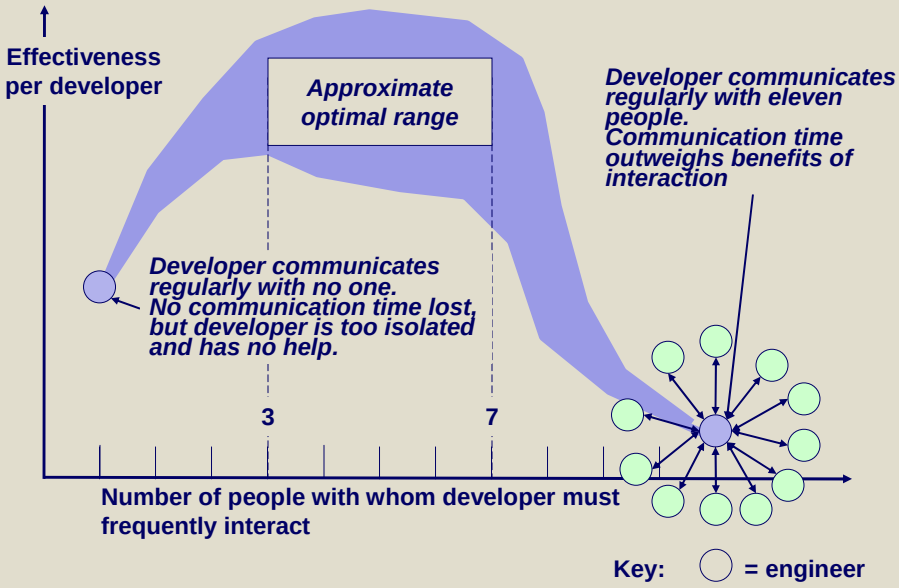
Group composition

- Group composed of members who share the same motivation can be problematic
  - Task-oriented - everyone wants to do their own thing
  - Self-oriented - everyone wants to be the boss
  - Interaction-oriented - too much chatting, not enough work
- An effective group has a balance of all types
- Can be difficult to achieve because most engineers are task-oriented
- Need for all members to be involved in decisions which affect the group

Optimal Size for Interaction (Approximate)



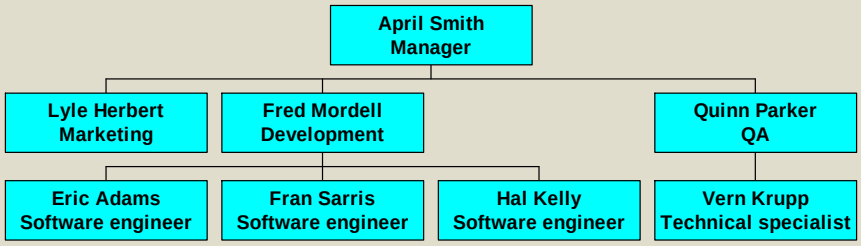
Optimal Size for Interaction (Approximate)



Group leadership

- Leadership depends on respect not titular status
- There may be both a technical and an administrative leader
- Democratic leadership is more effective that autocratic leadership
- A career path based on technical competence should be supported

Hierarchical Project Management Organizations



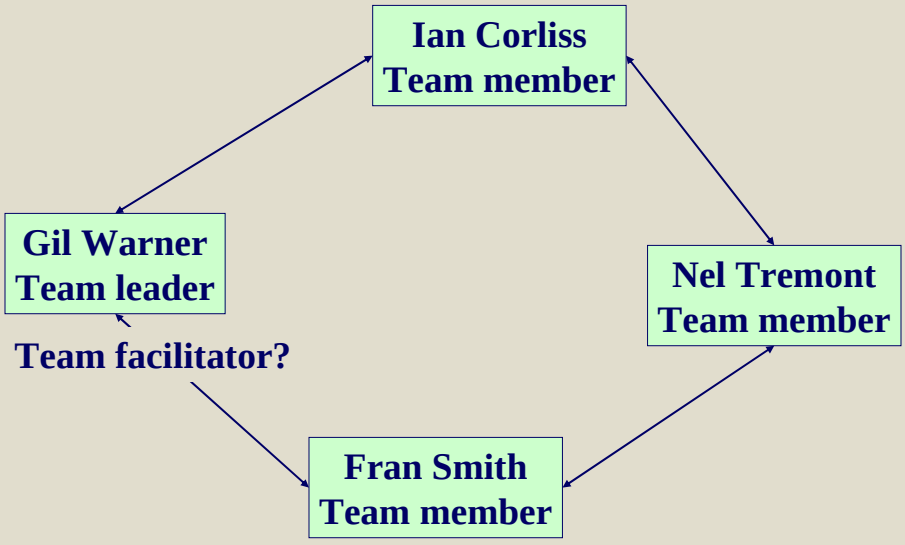
Smaller Projects:

No separate Marketing?
No separate QA organization?

Larger Projects:

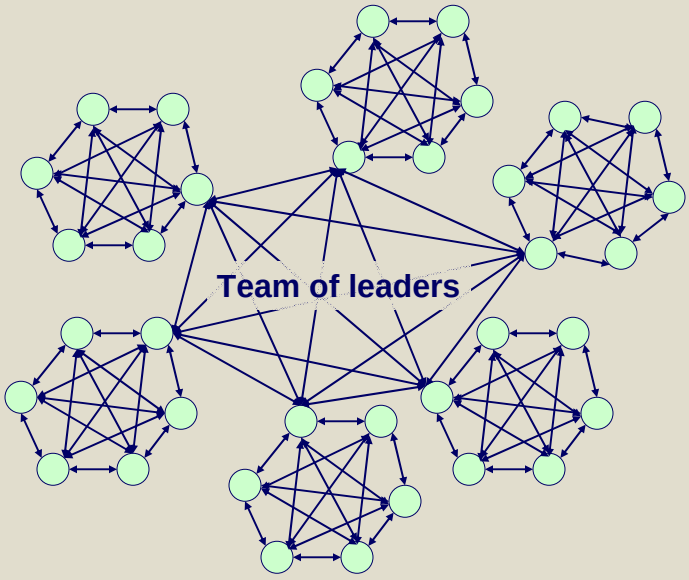
Subdivide QA into testing, …?
Subdivide Engineering into system engineering, …?

Horizontal Project Management Organizations



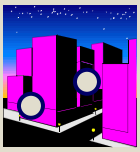
Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Peer Organizations for Larger Projects



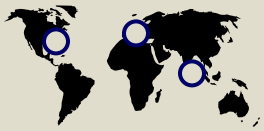
Remote Team Options

- Same office area
  - + ideal for group communication
  - - labor rates sub-optimal



- Same city, different offices
  - communication fair
- Same country, different cities
  - - communication difficult
  - + common culture

- Multi-country
  - - communication most difficult
  - - culture issues problematical
  - + labor rates optimal



One way to organize teams

1. **Select team leader: responsibilities:**
   - **ensure all project aspects active**
   - **fill all gaps**
3. **Designate leader roles & document responsibilities**
   - **team leader: *proposes and maintains… SPMP***
   - **configuration management leader: *... SCMP***
   - **quality assurance leader: *... SQAP, STP***
   - **requirements management leader: *... SRS***
   - **design leader: *... SDD***
   - **implementation leader: *... code base***
2. **Leaders' responsibilities:**
   - **propose a strawman artifact (e.g. SRS, design)**
   - **seek team enhancement & acceptance**
   - **ensure designated artifact maintained & observed**
   - **maintain corresponding metrics if applicable**
4. **Designate a backup for each leader**

Table 2.1 Matrixed organization

		Project			
		Airline reserv. project	**Bank accountg. project**	**Molecular analysis project**	**Fluid mechanics project**
Functional Unit	**Project management department**	Al Pruitt Full time	Quinn Parker Half time	Ruth Pella Full time	Fred Parsons Full time
	Marketing department	Oscar Mart Full time	Pete Merrill Full time	Sue More Half time	Elton Marston Full time
	Engineering department	Hal Egberts	Ben Ehrlich	Mary Ericson	Len Engels .....

One way to organize meetings

- * Distribute start time, end time, and agenda with approximate times
  - list important items first
- * Ensure “strawman” items prepared
- Start on time
- Have someone record action items‡
- Have someone track time & prompt members
- Get agreement on the agenda and timing
- Watch timing throughout, and end on time
  - allow exceptions for important discussion
  - stop excessive discussion; take off line
- Keep discussion on the subject
- ** E-mail action items & decision summary.

*** in advance of meeting ‡ actions members must perform ** after meeting**

One way to specify agendas

- Get agreement on agenda & time allocation
- Get volunteers to … :
  - … record decisions taken and action items
  - … watch time and prompt members
- Report progress on project schedule -- *10 mins*
- Discuss strawman artifact(s) -- *x mins*
- Discuss risk retirement -- *10 mins*
- <MORE ITEMS>
  - metrics and process improvement?
- Review action items -- *5 mins*

D. Design patterns

a) Introduction
b) Variables of project management
c) People management
d) Risk management
e) Final issues

Four risk activities

- **Identification**
  - **Mindset: *try to continually identify risks***
- **Retirement planning**
- **Prioritization**
- **Retirement or mitigation**

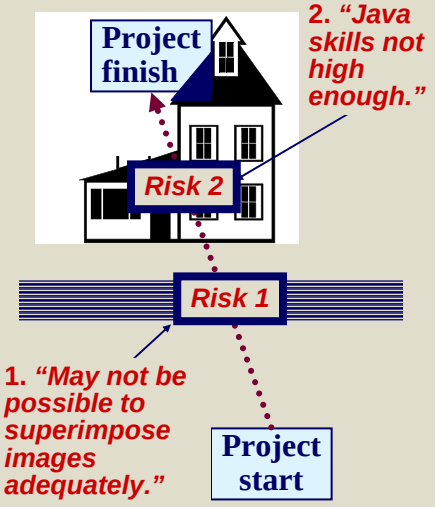
Risk sources (by importance)

Keil, Cule, Lyytinen, Schmidt

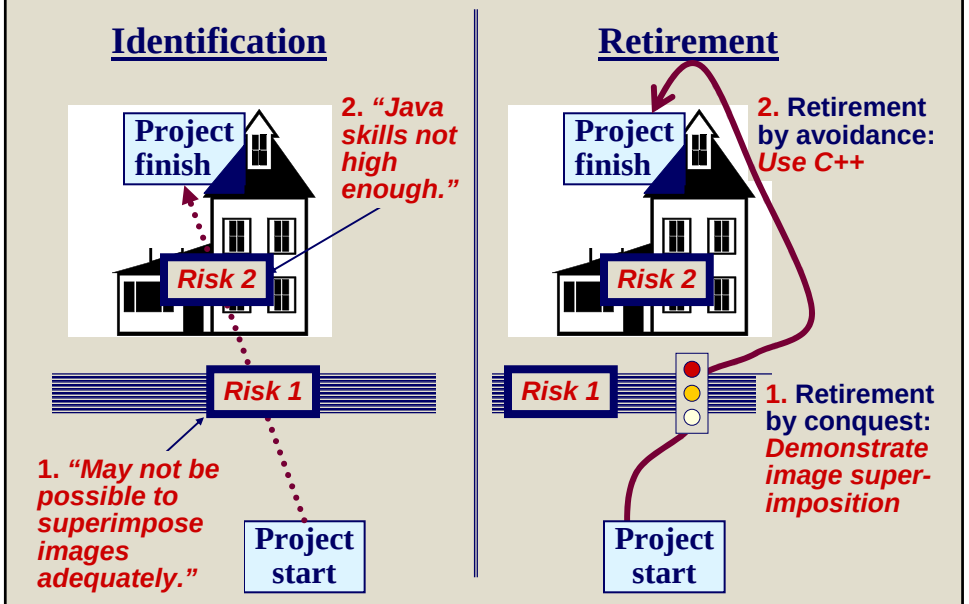
- Lack of top management commitment
- Failure to gain user commitment
- Misunderstanding of requirements
- Inadequate user involvement
- Failure to manage end-user expectations
- Changing scope and/or objectives
- ….

The Risk Management Mindset

Identification



The Risk Management Mindset



A way to compute risk priorities

	Likelihood 1-10 *1 = least likely*	Impact 1-10 *1 = least impact*	Retire-ment cost 1-10 *1 = lowest retiremen t cost*	Priority computation	Resultin g priority *Lowest number handled first*
The highest priority risk	**10 (most likely)**	*10 (most impact)*	1 (lowest retiremen t cost)	(11-**10**) *(11-*10*) *1	**1**
The lowest priority risk	**1 (least likely)**	*1 (least impact)*	10 (highest retiremen t cost)	(11-**1**) *(11-*1*) *10	**1000**

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Table 2.3 Sample Risk Analysis for *Encounter* Case Study

Risk title (details given above)	Like-lihood 1-10	Im-pact 1-10	Retire-ment cost 1-10	Priority	Retirement / mitigation plan	Responsible engineer	Target completion date
Superimposing images	3	10	1	8	Experiment with Java images.	P. R.	2/1/99
Deficient Java skills	9	6	8	80	H.T., K.M., V.I. and L.D. to attend training course beginning 1/5/99 at Ultra Training Corp, obtain Java level 2 certification by 3/1/99 and level 3 certification by 4/15/99	H. L.	4/15/99
Alan Gray may be pulled off this project	3	7	9	288	Susan Ferris to inspect all of Alan's work	S.F.	Continual
...	...	...	...	...	...	...	...

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Identify and Retire Risks

1.* Each team member spends 10 mins. exploring his or her greatest fears for the project's success
2.* Each member specifies these risks in concrete language, weights them, writes retirement plans, (see format above) & e-mails to the team leader
3.* Team leader integrates and prioritizes results
4.M Group spends 10 mins. seeking additional risks
5.M Team spends 10 mins. finalizing the risk table
   - Designates responsible risk retirement engineers

6.** Responsible engineers do risk retirement work
7.M Team reviews risks for 10 mins. at weekly meetings
   - responsible engineers report progress
   - team discusses newly perceived risks and adds them

*:in advance of first meeting M: at meeting **: between meetings

D. Design patterns

a) Introduction

b) Variables of project management

c) People management

d) Risk management

e) Final issues

Example of Build vs. Buy Decision-making: Game Graphics engine

	Build cost (in thousands)	*Buy cost* (in thousands)	*Comments* multi-year costs not accounted for
Supplies	$ 0	$40	Purchase Ajax engine
First-person perspective	$ 5	$ 0	Ajax has this feature
3-D	$10	$ 1	Customize Ajax application
Light reflection	$15	$10	Customize Ajax application
TOTALS	$30	$51	

Build, do not buy

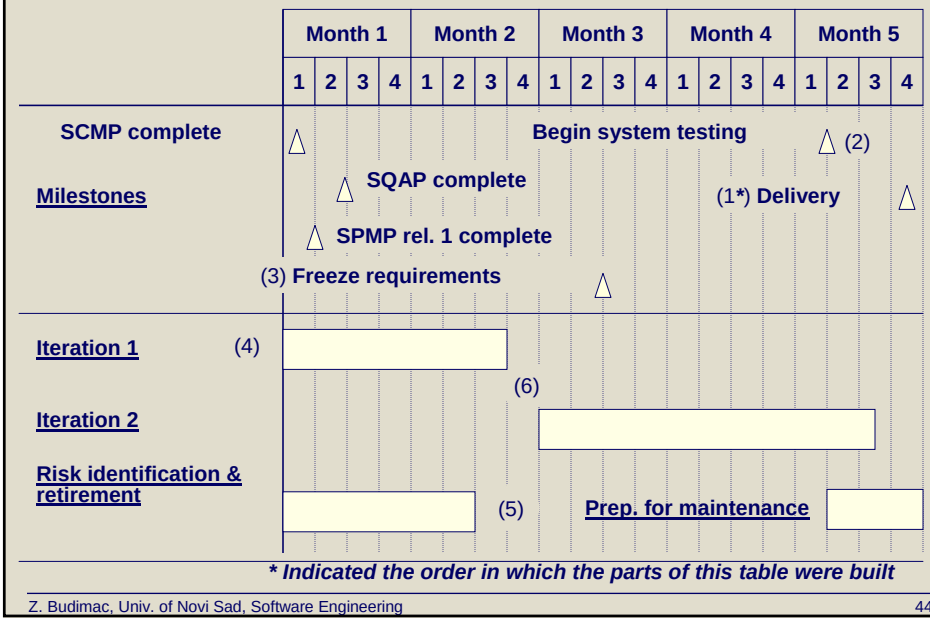
Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Example of method for deciding language choice

Factor	Weight *(1-10)*	Benefit of Language 1 *1 to 10=best*	Benefit of Language 2 *1 to 10=best*
Internet-friendly	*3*	8	2
Familiarity to development team	*8*	3	9
Compilation speed	*5*	2	8
Runtime speed on processor *p*	*1*	7	3
Score		***3***8 + **8***3 + **5***2 + **1***7 = 65*	***3***2 + **8***9 + **5***8 + **1***3 = 121*

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

High Level Task Chart with Fixed Delivery Date: Order of Completion



** Indicated the order in which the parts of this table were built*

One way to create an initial schedule

1. Indicate the milestones your *must* observe
   - usually includes delivery date
2. Back these up to introduce the milestones you *need*
   - e.g., begin system testing well before delivery
3. Designate a time at which all requirements frozen

The remaining steps depend on the process used. We will assume an iterative process.

4. Show first iteration: establishes minimal capability
   - usually: keep it *very modest, even trivial, in capability*
   - benefit: exercises the development process itself
5. Show task of identifying & retiring risks
   - starting from project inception
6. Show unassigned time (e.g., week) near middle?
7. Complete the schedule

Level Labor Allocation for Fixed Labor Total

	Month 1				Month 2				Month 3				Month 4				Month 5			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Milestones									Freeze requirements △								Complete testing △			
						Karen vacation											Release to production △			
Iteration 1	2	2	2	3	2	2	3													
													Hal vacation							
Iteration 2									4	4	4	3	3	4	4	4	4	4	4	4
Risk ID & retire	2	2	2	1	1	1		4	To be assigned											
Given team size:	*4*	*4*	*4*	*4*	*4*	*3*	*3*	*4*	*4*	*4*	*4*	*3*	*3*	*4*	*4*	*4*	*4*	*4*	*4*	*4*

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Table 2.5 Defects by Phase

Defects detected per ... ... 100 requirements, or ... design diagram, or ... KLOC **This project** / *norm*		**Phase in which defects detected**		
		Detailed require-ments	Design	Implemen-tation
Phase containing defects	Detailed requirements	**2** / *5*	**0.5** / *1.5*	**0.1** / *0.3*
	Design		**3** / *1*	**1** / *3*
	Implementa-tion			**2** / *2*

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Five Process Metric Examples

Compare each of the following with company norms averaged over similar processes.

1. *Number of defects per KLOC detected within x weeks of delivery*
2. *Variance in schedule on each phase*

$$\frac{\text{actual duration - projected duration}}{\text{projected duration}}$$

3. *Variance in cost*

$$\frac{\text{actual cost - projected cost}}{\text{projected cost}}$$

4. *Total design time / total programming time*
   should be at least 50% (Humphry)
5. *Defect injection and detection rates per phase*
   e.g. “1 defect per class in detailed design phase”

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

One way to gather process metrics

1. Identify & define metrics team will use by phase;
   include ... time spent on 1. research, 2. execution, 3. review
   … size (e.g. lines of code)
   … # defects detected per unit (e.g., lines of code)
   include source
   … quality self-assessment of each on scale of 1-10
   maintain bell-shaped distribution
2. Document these in the SQAP
3. Accumulate historical data by phase
4. Decide where the metric data will be placed
   - as the project progresses SQAP? SPMP? Appendix?
5. Designate engineers to manage collection by phase
   - QA leader or phase leaders (e.g., design leader)
6. Schedule reviews of data for lessons learned
   - Specify when and how to feed back improvement

Requirements Document: 200 detailed requirements	Meeting	Research	Execution	Personal Review	Inspection
Hours spent	0.5 x 4	4	5	3	6
% of total time	10%	20%	25%	15%	30%
% of total time: norm for the organization	*15%*	*15%*	*30%*	*15%*	*25%*
Self-assessed quality 1-10	2	8	5	4	6
Defects per 100	N/A	N/A	N/A	5	6
Defects per 100: organization norm	*N/A*	*N/A*	*N/A*	*3*	*4*
Hours spent per detailed requirement	0.01	0.02	0.025	0.015	0.03
Hours spent per detailed requirement: organization norm	0.02	0.02	0.04	0.01	0.03
Process improvement	Improve strawman brought to meeting		Spend 10% more time executing		
Summary	Productivity: 200/22 = 9.9 detailed requirements per hour Probable remaining defect rate: 6/4´[organizational norm of 0.8 per hundred] = 1.2 per 100				

Table 2.6 Project Metric Collection for phases

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Project Management Plan
(IEEE 1058.1-1987 SPMP Table of Contents)

1. **Introduction**
   1. Project overview
   2. Project deliverables
   3. Evolution of the SPMP
   4. Reference materials
   5. Definitions and acronyms
2. **Project organization**
   1. Process model
   2. Organizational structure
   3. Organizational boundaries and interfaces
   4. Project responsibilities
3. **Managerial process**
   1. Managerial objectives & priorities
   2. Assumptions, dependencies & constraints
   3. Risk management
   4. Monitoring & controlling mechanisms
   5. Staffing plan
4. **Technical process**
   1. Methods, tools & techniques
   2. Software documentation
   3. Project support functions
5. **Work packages, schedule & budget**
   1. Work packages
   2. Dependencies
   3. Resource requirements
   4. Budget & resource allocation
   5. Schedule

Table 2.7 Example of Process Comparison

Motor control applications	Process		
	Waterfall	Spiral, 2-4 iterations	Spiral, 5-10 iterations
Company average -- Defects per thousand source lines of code at delivery time injected at ...			
... requirements time	4.2	3.2	2.4
... architecture time	3.1	2.5	3.7
... detailed design time	1.1	1.1	2.2
... implementation time	1.0	2.1	3.5
TOTAL	9.4	8.9	11.8

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Feed Back Process/Project Improvement

1. **Decompose the process or sub-process being measured into *Preparation*, *Execution* and *Review***
   - **include *Research* if learning about the procedure**
2. **Note time taken, assess degree of quality for each part on a 1-10 scale, count defects**
   - **try to enforce a curve**
3. **Compute** *quality / (percent time taken)*
4. **Compare team's performance against existing data, if available**
5. **Use data to improve next sub-process**
   - **note poorest values first e.g., low *quality/(percent time)***

Table 2.8 Measuring Team Phase Performance

	For each part ...		
	Preparation	**Execution**	**Review**
% time	*45*	*30*	*25*
Quality (0 to 10)* ***If low, investigate***	*6*	*2* *investigate*	*6*
Quality/(% time) ***If low, investigate***	0.13 *investigate*	0.07 *investigate*	0.24
Typical?	*No* *Joe lost specs*	*Yes*	*Yes*
Action		*Schedule 20% more time for execution, taken equally from other phases*	

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.