Software Requirements and Architecture (SENG404)

Matthias Galster

Lecture 6 – Requirements analysis

March 16, 2023

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Schedule 2023

Lecture	Week	Date	Topic
1	1	February 22	Kick-off; Introduction
2	1	February 23	Instead of May 3; Requirements and requirements engineering processes
3	2	March 1	Requirements elicitation (part 1)
4	2	March 2	Instead of May 17; Requirements elicitation (part 1); Requirements elicitation (part 2)
5	3	March 8	Requirements elicitation (part 2); Requirements documentation
6	3	March 9	Matthias away
7	4	March 15	Requirements documentation
8	4	March 16	Requirements documentation; Requirements analysis
9	5	March 22	Assignment 1
10	6	March 29	
			Term break
11	7	April 26	
12	8	May 3	Matthias away
13	9	May 10	
14	10	May 17	Matthias away
15	11	May 24	Assignment 2: presentations + report
16	12	May 31	
		TBD	**Final exam**

Assignment 1

Student(s)	Topic
Saskia van der Peet	Use of design thinking in requirements engineering
April Clarke	Influence of social factors on requirements engineering
Jonathan Tomlinson + Danish Jahangir	Software requirements elicitation techniques
Michael Wilson	Architecture recovery and recovery techniques
Jamie Thomas	Usefulness in requirements prioritization techniques
Lisa Lu + Joshua Egan	Towards Understanding Software Architectural Patterns
Andrew Cook	Security requirements engineering

- Before you start
  - What would a practitioner want to know?
  - Source(s)?
  - Key message(s) – build blog post around it
  - Provide facts and evidence; if you share opinions mark them as such

Previous lecture

1. Requirements documentation – overview

2. Natural language requirements

3. Formal languages

4. Requirements modeling

5. Characteristics of requirements

Reading for this session

- R. Britto, E. Mendes, and J. Borstler. *An empirical investigation on effort estimation in agile global software development.* In 10th International Conference on Global Software Engineering (ICGSE), 2015, pp. 38-45, doi: 10.1109/ICGSE.2015.10

Questions and lessons

Reading for next session

- L. Montgomery, D. Fucci, A. Bouaffa, L. Scholz, and W. Maalej. *Empirical research on requirements quality: a systematic mapping study.* Requirements Engineering, 2022, pp. 183-209, https://doi.org/10.1007/s00766-021-00367-z

Agenda

1. Requirements analysis – overview

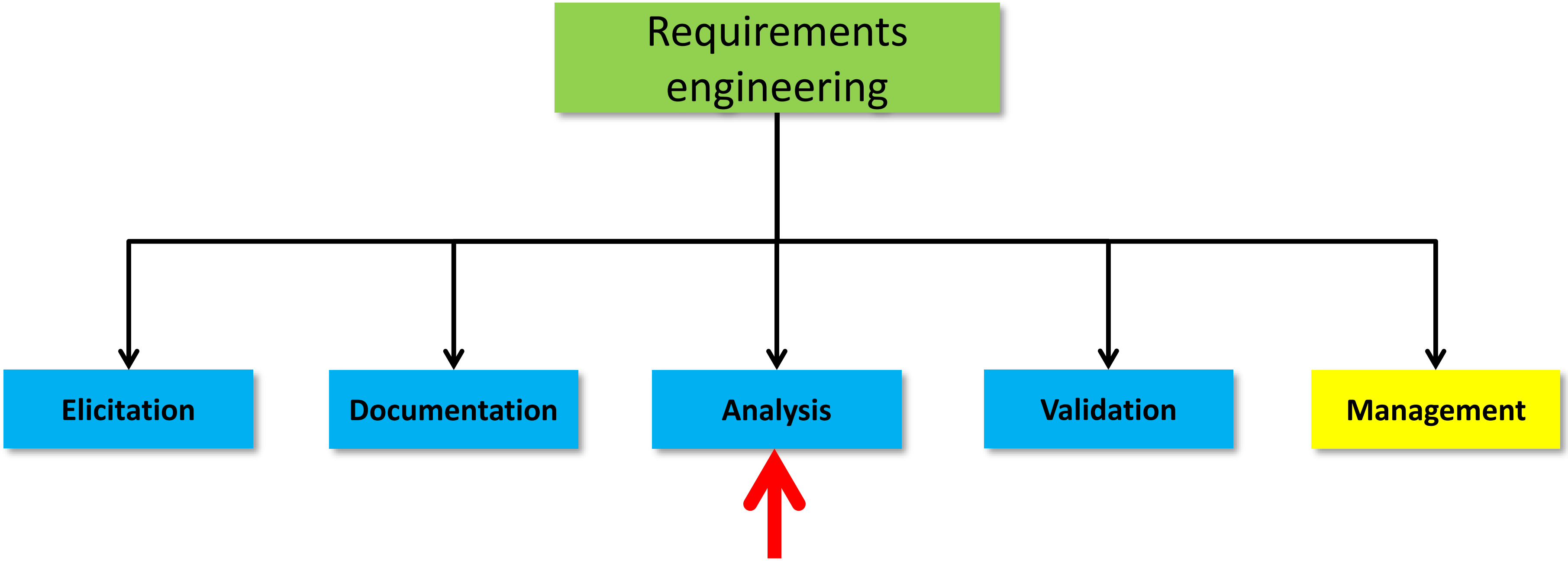
2. Estimation

3. Prioritization

Agenda

1. Requirements analysis – overview
2. Estimation
3. Prioritization

Generic RE activities



Requirements analysis

- Process and activities related to
  - refining
  - user‘s needs and constraints
  - by estimating and prioritizing requirements

Agenda

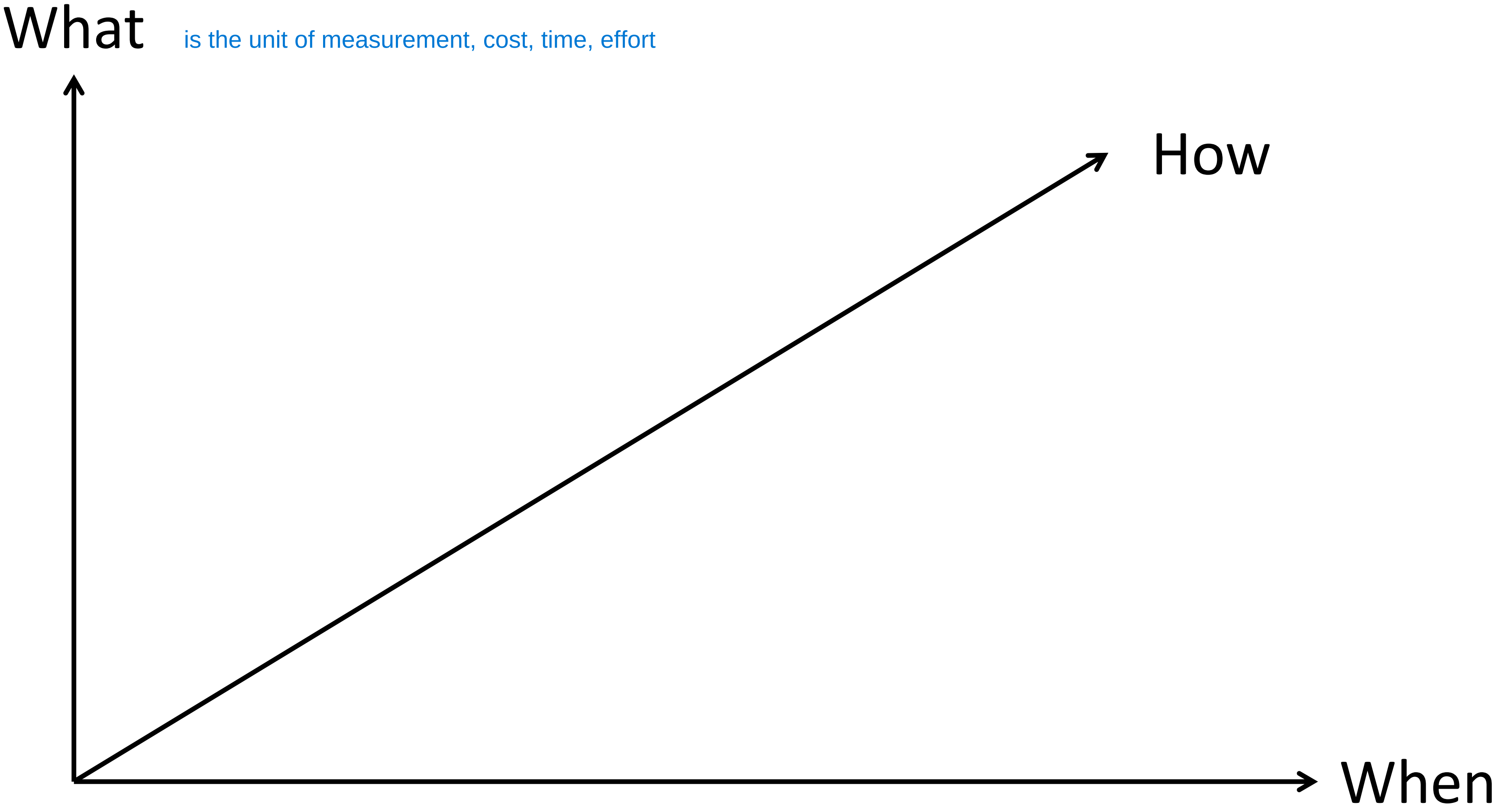
1. Requirements analysis – overview

2. Estimation

3. Prioritization

Estimation

the more high level the more difficult to estimate



is the unit of measurement, cost, time, effort

Estimation – overview

why is there no one set way to estimate. they all have pro and con and hopefully they can compliment each other. and depending on environment

- Different units of analysis, i.e., “requirements” (“what”)
  - Features, user stories, textual requirements in document, etc.
  - Sometimes hierarchical relationships, e.g., feature packages
  - For comparison: estimated units should be at same level

need to be at the same level of abstraction

- Different units of estimation (“how”)
  - Time, money, function points, story/use case points, LoC, etc.
  - Sometimes multiple units, depending on what is estimated
- At different stages (“when”)
  - Current day planning, iteration, release, project bidding, etc.
  - Could estimate several times (same or different things)

Typical estimation techniques (1)

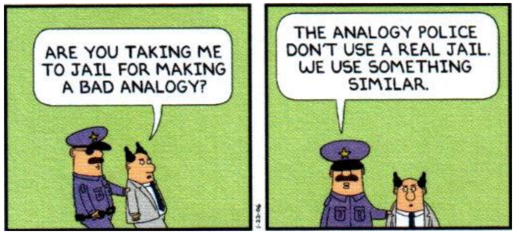
- Expert judgement
  - "Experts" on software development + domain are consulted

- Estimation in groups
  - Each estimate is arrived at by consensus (in theory)
  - Example: Delphi
    - Estimate-Talk-Estimate (ETE)
  - Example: Planning Poker
    - Using story points (0, 1, 2, 3, 5, 8, 13, 20, 40, 100)
    - Relative values matter (e.g., 2 points twice as much 1 point)

lot easier than having estimates per person.

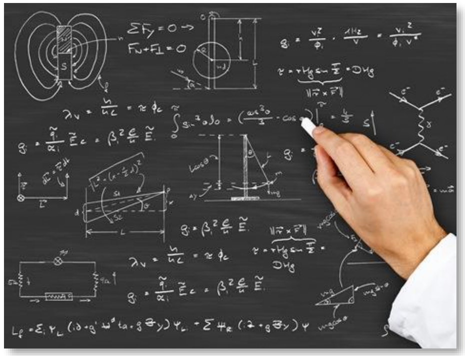
Typical estimation techniques (2)

- Estimation by analogy
  - Effort estimated by analogy with completed projects


ARE YOU TAKING ME TO JAIL FOR MAKING A BAD ANALOGY?
THE ANALOGY POLICE DON'T USE A REAL JAIL. WE USE SOMETHING SIMILAR.

Typical estimation techniques (3)

- Effort modeling
  - Historical information which relates some metric to effort
  - Mathematical estimation, prediction or regression models
    - See also “Big data” and “Data science”
  - Examples
    - Function points (including different flavors, e.g., IFPUG, COSMIC*)
    - CoCoMo (Constructive Cost Model)
    - Putnam model (empirical estimation model)
    - Organization-specific models



* IFPUG: International Function Point User Group; COSMIC: Common Software Measurement International Consortium

Example of effort modeling: function points

- Measurement to express the amount of business functionality
  - "Synthetic" metric derived from the following five attributes
    - Inputs/input types
    - Outputs/output types
    - InquiriEs/inquiry types
    - Number of Logical files
    - Number of interFaces

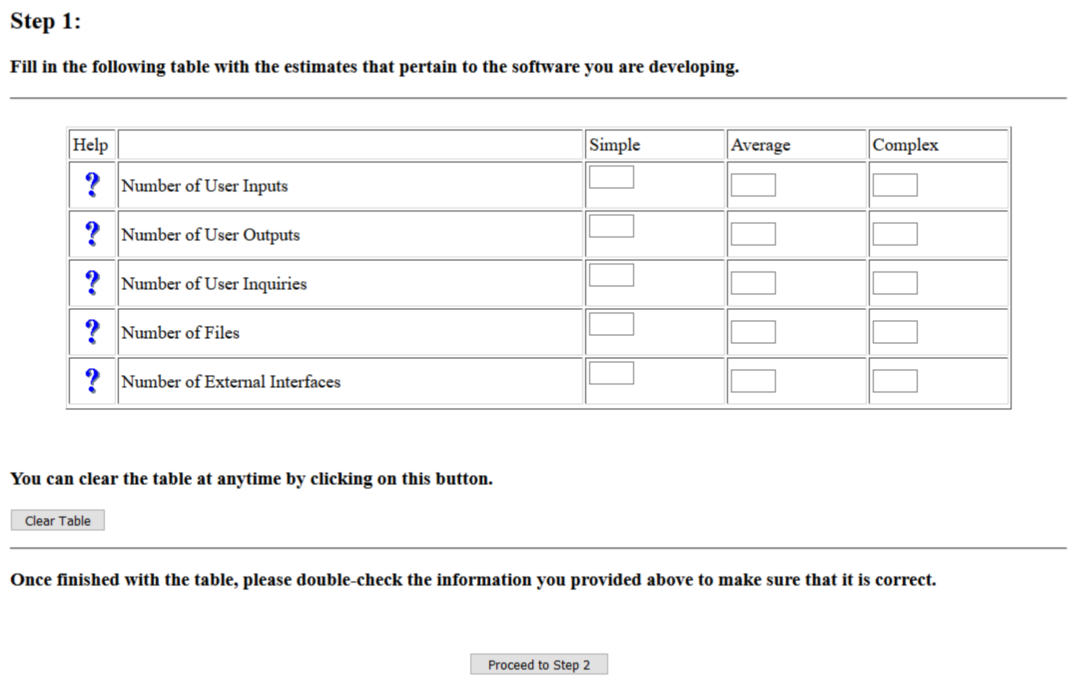
- UFP (unadjusted function points) = 4I + 5O + 4E + 10L + 7F
  - Can be configured

may not allow for uncertainty

  - FP = UFP x TCF
    - **T**echnical **C**omplexity **F**actor = .65 + .01 x DI
    - **D**egree of **I**nfluence (app characteristics, e.g., operational/installation ease)

Can be applied at different levels of abstraction

Example tool* – step 1



Step 1:

Fill in the following table with the estimates that pertain to the software you are developing.

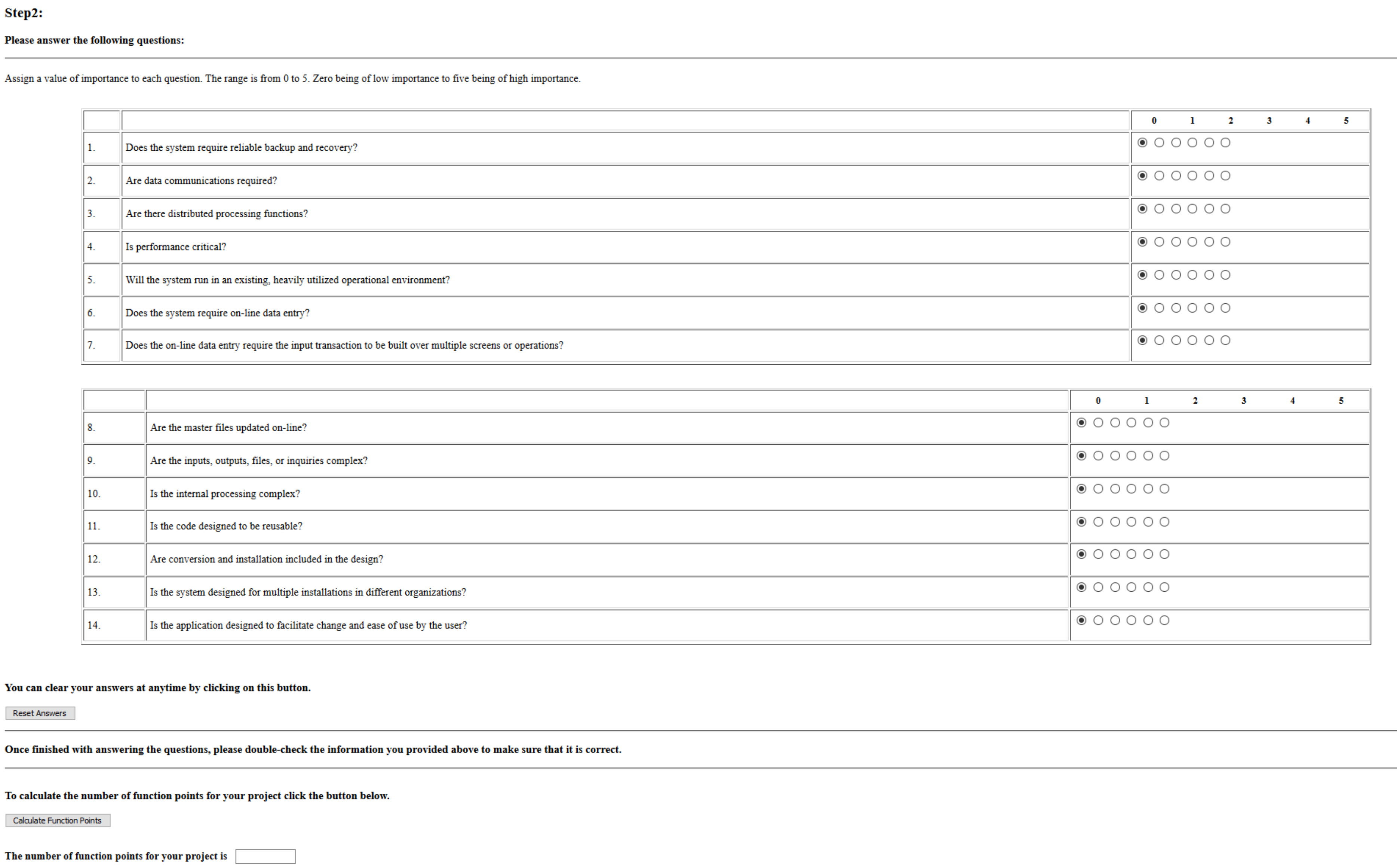
Help		Simple	Average	Complex
?	Number of User Inputs	□	□	□
?	Number of User Outputs	□	□	□
?	Number of User Inquiries	□	□	□
?	Number of Files	□	□	□
?	Number of External Interfaces	□	□	□

You can clear the table at anytime by clicking on this button.

Once finished with the table, please double-check the information you provided above to make sure that it is correct.

* http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/fp_leader.html

Example tool* – step 2



Step2:

Please answer the following questions:

Assign a value of importance to each question. The range is from 0 to 5. Zero being of low importance to five being of high importance.

		0	1	2	3	4	5
1.	Does the system require reliable backup and recovery?	●	○	○	○	○	○
2.	Are data communications required?	●	○	○	○	○	○
3.	Are there distributed processing functions?	●	○	○	○	○	○
4.	Is performance critical?	●	○	○	○	○	○
5.	Will the system run in an existing, heavily utilized operational environment?	●	○	○	○	○	○
6.	Does the system require on-line data entry?	●	○	○	○	○	○
7.	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	●	○	○	○	○	○

		0	1	2	3	4	5
8.	Are the master files updated on-line?	●	○	○	○	○	○
9.	Are the inputs, outputs, files, or inquiries complex?	●	○	○	○	○	○
10.	Is the internal processing complex?	●	○	○	○	○	○
11.	Is the code designed to be reusable?	●	○	○	○	○	○
12.	Are conversion and installation included in the design?	●	○	○	○	○	○
13.	Is the system designed for multiple installations in different organizations?	●	○	○	○	○	○
14.	Is the application designed to facilitate change and ease of use by the user?	●	○	○	○	○	○

You can clear your answers at anytime by clicking on this button.

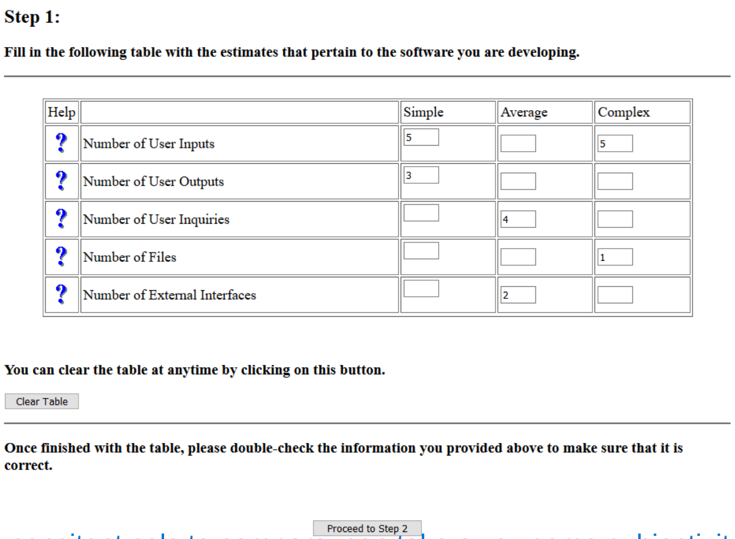
Once finished with answering the questions, please double-check the information you provided above to make sure that it is correct.

To calculate the number of function points for your project click the button below.

The number of function points for your project is

* http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/fp_leader.html

Example tool*



Step 1:

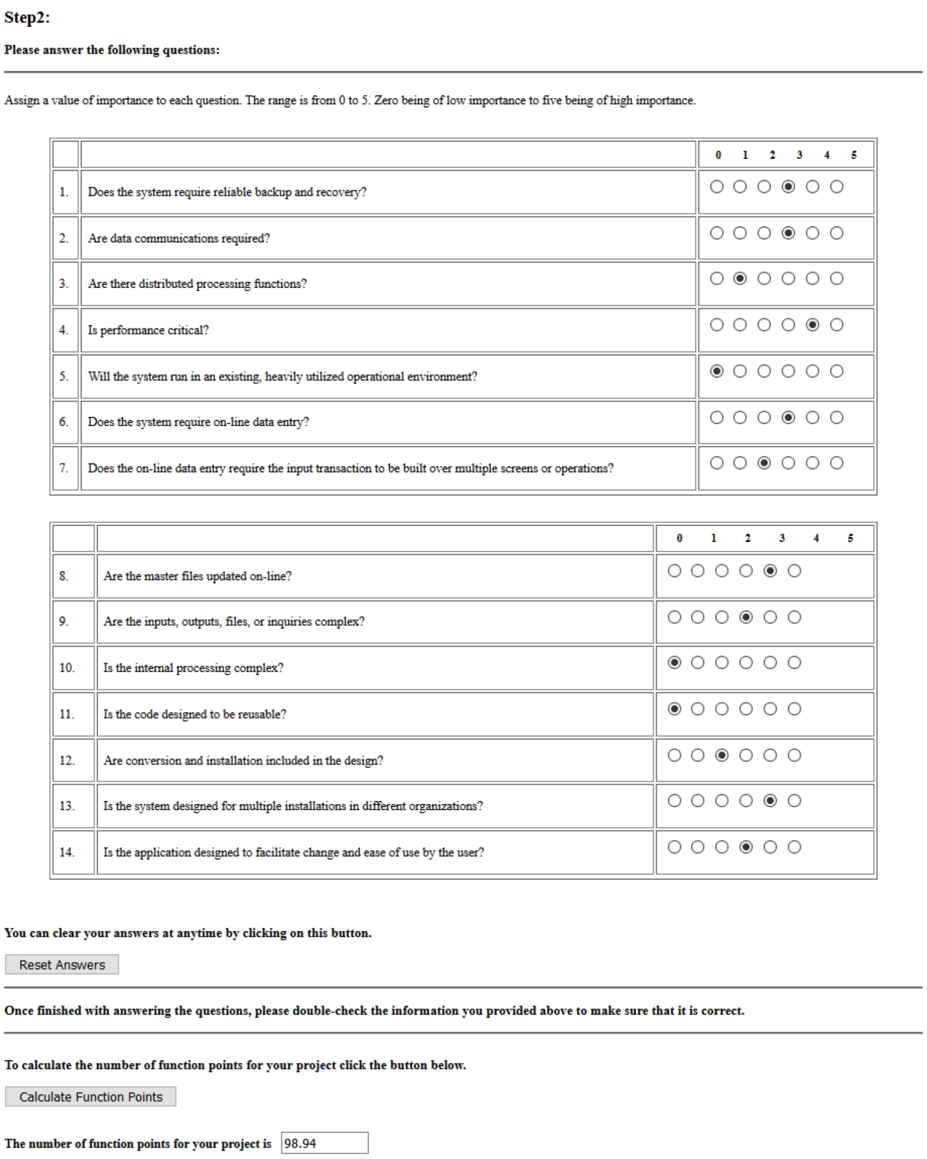
Fill in the following table with the estimates that pertain to the software you are developing.

Help		Simple	Average	Complex
?	Number of User Inputs	5		5
?	Number of User Outputs	3		
?	Number of User Inquiries		4	
?	Number of Files			1
?	Number of External Interfaces		2	

You can clear the table at anytime by clicking on this button.

Once finished with the table, please double-check the information you provided above to make sure that it is correct.

consitent calc to compare, can take away some subjectivity



Step2:

Please answer the following questions:

Assign a value of importance to each question. The range is from 0 to 5. Zero being of low importance to five being of high importance.

		0	1	2	3	4	5
1.	Does the system require reliable backup and recovery?	○	○	○	●	○	○
2.	Are data communications required?	○	○	○	●	○	○
3.	Are there distributed processing functions?	○	●	○	○	○	○
4.	Is performance critical?	○	○	○	○	●	○
5.	Will the system run in an existing, heavily utilized operational environment?	●	○	○	○	○	○
6.	Does the system require on-line data entry?	○	○	○	●	○	○
7.	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	○	○	●	○	○	○

		0	1	2	3	4	5
8.	Are the master files updated on-line?	○	○	○	○	●	○
9.	Are the inputs, outputs, files, or inquiries complex?	○	○	○	●	○	○
10.	Is the internal processing complex?	●	○	○	○	○	○
11.	Is the code designed to be reusable?	●	○	○	○	○	○
12.	Are conversion and installation included in the design?	○	○	●	○	○	○
13.	Is the system designed for multiple installations in different organizations?	○	○	○	○	●	○
14.	Is the application designed to facilitate change and ease of use by the user?	○	○	○	●	○	○

You can clear your answers at anytime by clicking on this button.

Once finished with answering the questions, please double-check the information you provided above to make sure that it is correct.

To calculate the number of function points for your project click the button below.

The number of function points for your project is 98.94

* http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/fp_leader.html

Example of effort modeling: basic COCOMO

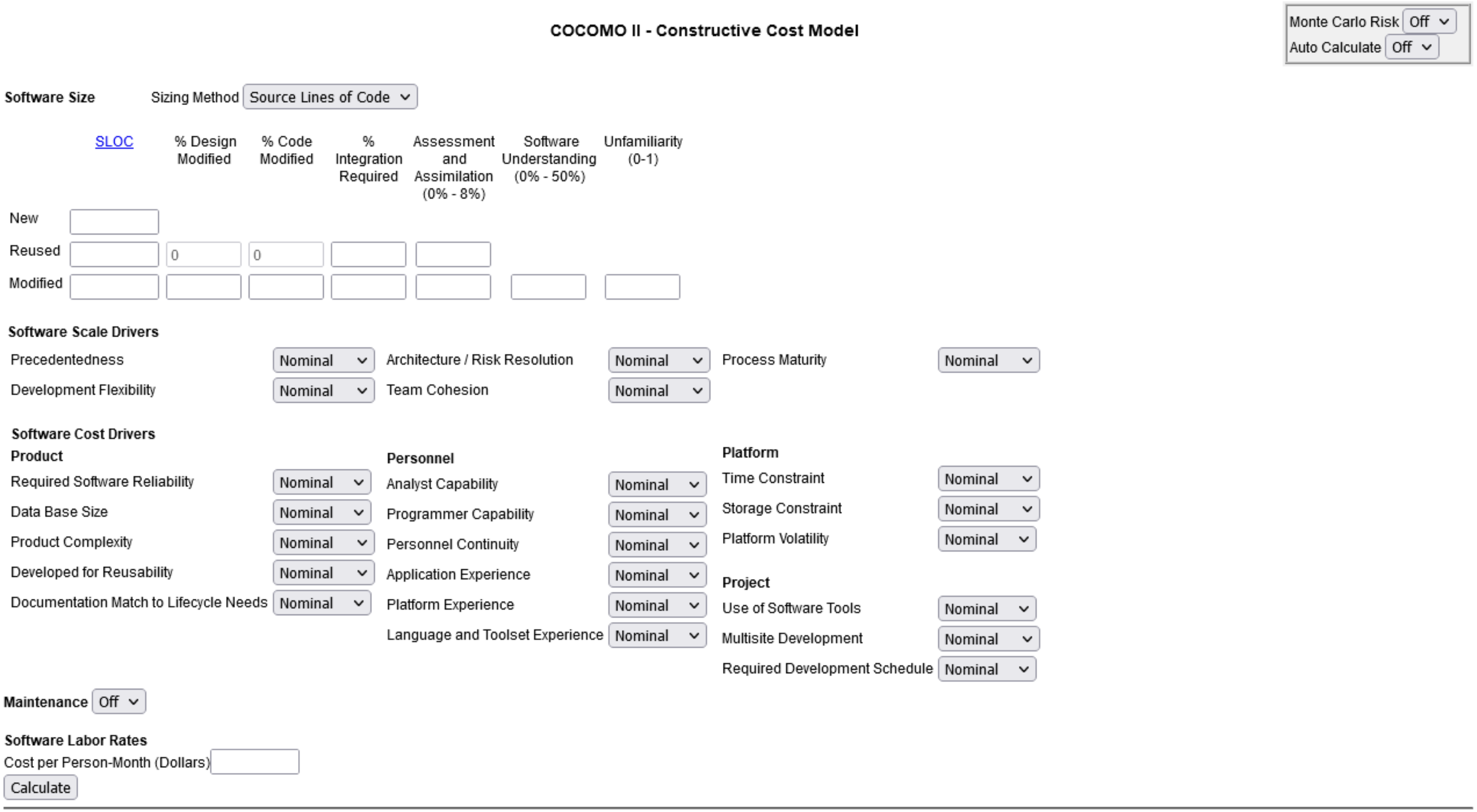
- E (staff months) = a x size^b x C
  - size: estimated KLOC
  - a, b: constants that change according to the estimate required
    - a = 2.4 (organic), 3.0 (semi-detached), 3.6 (embedded)
    - b = 1.05 (organic), 1.12 (semi-detached), 1.20 (embedded)
  - C (optional): adjustment factor based on 16 factors
    - E.g., product attributes, personnel attributes, project attributes

Can be applied at different levels of abstraction

Classes of projects in COCOMO

- Organic
  - Relatively small team develops software in known environment; people involved generally have a lot of experience with similar projects in their organization, are able to contribute early, since no initial overhead; projects of this type will seldom be very large
- Embedded
  - Product embedded in environment which is very inflexible and poses severe constraints; e.g., air traffic control
- Semi-detached
  - Intermediate form; team may include mix of experienced and inexperienced people, project may be fairly large, but not excessively large

Example tool*



COCOMO II - Constructive Cost Model

Monte Carlo Risk: Off
Auto Calculate: Off

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New							
Reused		0	0				
Modified							

Software Scale Drivers

Precedentedness	Nominal	Architecture / Risk Resolution	Nominal	Process Maturity	Nominal
Development Flexibility	Nominal	Team Cohesion	Nominal		

Software Cost Drivers

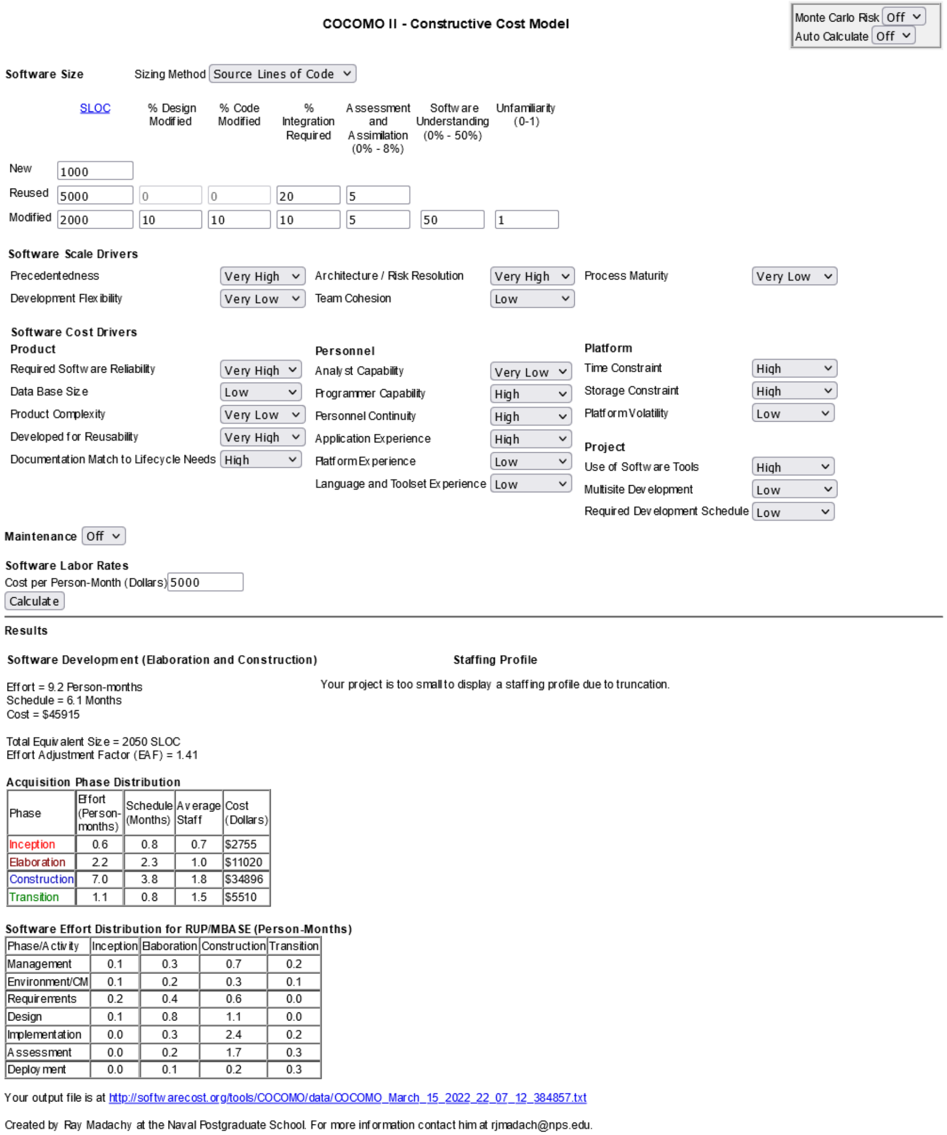
Product		**Personnel**		**Platform**	
Required Software Reliability	Nominal	Analyst Capability	Nominal	Time Constraint	Nominal
Data Base Size	Nominal	Programmer Capability	Nominal	Storage Constraint	Nominal
Product Complexity	Nominal	Personnel Continuity	Nominal	Platform Volatility	Nominal
Developed for Reusability	Nominal	Application Experience	Nominal	**Project**	
Documentation Match to Lifecycle Needs	Nominal	Platform Experience	Nominal	Use of Software Tools	Nominal
		Language and Toolset Experience	Nominal	Multisite Development	Nominal
				Required Development Schedule	Nominal

Maintenance Off

Software Labor Rates
Cost per Person-Month (Dollars)

* http://softwarecost.org/tools/COCOMO/

Example tool*



COCOMO II - Constructive Cost Model

Monte Carlo Risk: Off
Auto Calculate: Off

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	1000						
Reused	5000	0	0	20	5		
Modified	2000	10	10	10	5	50	1

Software Scale Drivers

Precedentedness	Very High	Architecture / Risk Resolution	Very High	Process Maturity	Very Low
Development Flexibility	Very Low	Team Cohesion	Low		

Software Cost Drivers

Product		**Personnel**		**Platform**	
Required Software Reliability	Very High	Analyst Capability	Very Low	Time Constraint	High
Data Base Size	Low	Programmer Capability	High	Storage Constraint	High
Product Complexity	Very Low	Personnel Continuity	High	Platform Volatility	Low
Developed for Reusability	Very High	Application Experience	High	**Project**	
Documentation Match to Lifecycle Needs	High	Platform Experience	Low	Use of Software Tools	High
		Language and Toolset Experience	Low	Multisite Development	Low
				Required Development Schedule	Low

Maintenance Off

Software Labor Rates
Cost per Person-Month (Dollars) 5000

Results

Software Development (Elaboration and Construction)

Effort = 9.2 Person-months
Schedule = 6.1 Months
Cost = $45915

Total Equivalent Size = 2050 SLOC
Effort Adjustment Factor (EAF) = 1.41

Staffing Profile

Your project is too small to display a staffing profile due to truncation.

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.6	0.8	0.7	$2755
Elaboration	2.2	2.3	1.0	$11020
Construction	7.0	3.8	1.8	$34896
Transition	1.1	0.8	1.5	$5510

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.3	0.7	0.2
Environment/CM	0.1	0.2	0.3	0.1
Requirements	0.2	0.4	0.6	0.0
Design	0.1	0.8	1.1	0.0
Implementation	0.0	0.3	2.4	0.2
Assessment	0.0	0.2	1.7	0.3
Deployment	0.0	0.1	0.2	0.3

Your output file is at http://softwarecost.org/tools/COCOMO/data/COCOMO_March_15_2022_22_07_12_384857.txt

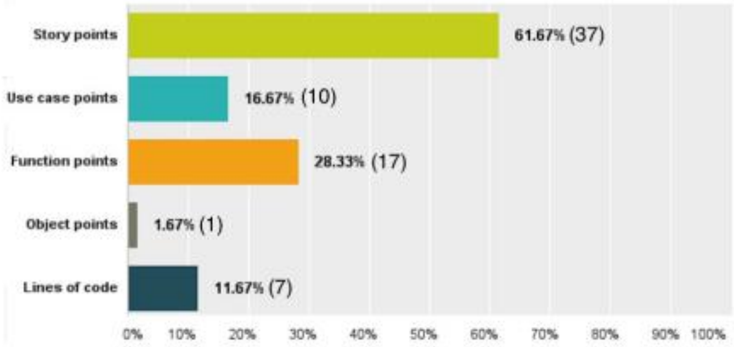
Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu.

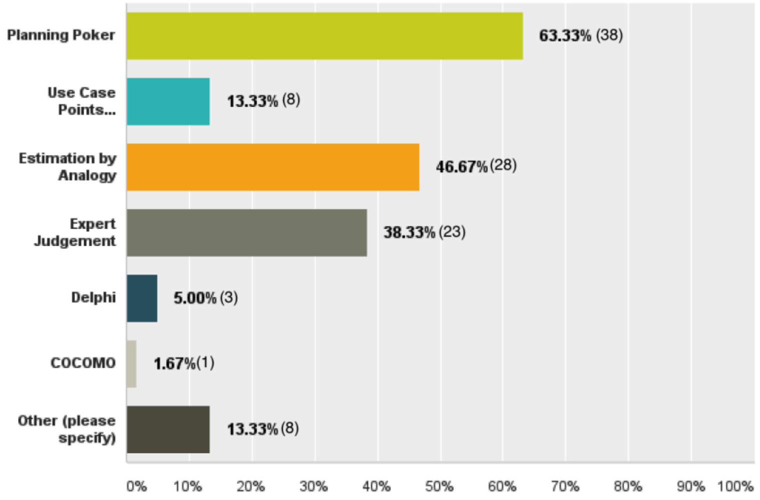
* http://softwarecost.org/tools/COCOMO/

Effort estimation in practice

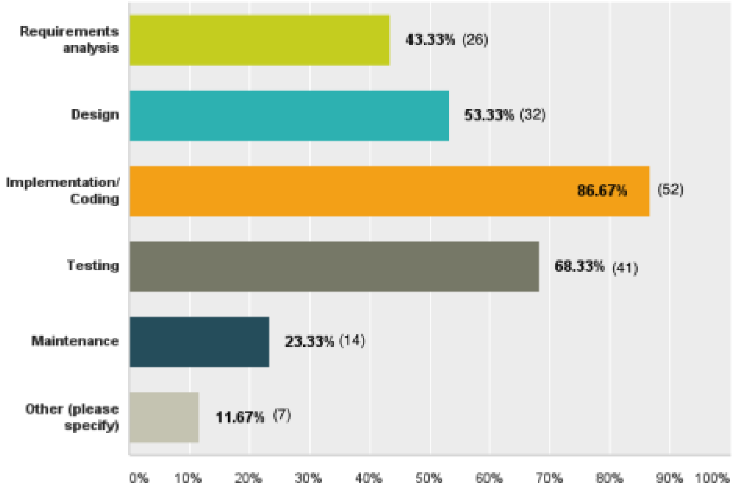
- N = 60

How





What and how



When

M. Usman et al.: *Effort Estimation in Agile Software Development: A Survey on the State of the Practice.* EASE (2015)

Effort estimation challenges

- People
  - Lack of skilled personnel to properly manage project portfolios
  - External contractors having lead role

- Processes
  - Poor knowledge management for understanding processes
  - Standard estimation methodologies for project types and phases

- Tools
  - Tools that vary among different projects
  - No integrated automated tools to collect, manage, analyze estimates

D. Basten, T. Hoerstrub: *Organizational Effort Estimation.* IEEE Computer 47(8): 76-79 (2014)

What are we actually estimating?

Interpretation (as claimed in hindsight)
Ideal effort
Most likely effort
Median effort
Risk-averse effort
Don’t know/gut feeling/other

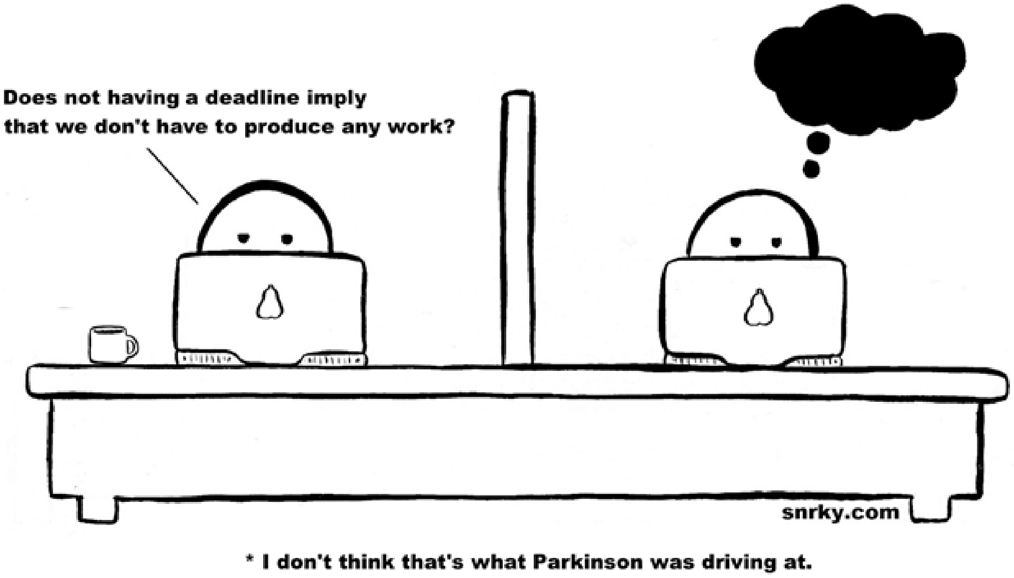
Interpretation (as claimed in hindsight)	Frequency of interpretation
Ideal effort	37%
Most likely effort	27%
Median effort	5%
Risk-averse effort	9%
Don’t know/gut feeling/other	22%

T. Halkjelsvik, M. Jorgensen: *Predictions and the Uncertainty of the Future* in Time Predictions (2018)

Parkinson's law

- "work expands so as to fill the time available for its completion"

- Example
  - Software has to be delivered in 12 months and 5 people are available
  - Effort required is estimated to be 60 person-months
  - Risk that effort is determined by resources, not objective assessment


Does not having a deadline imply
that we don't have to produce any work?
snrky.com
* I don't think that's what Parkinson was driving at.

Agenda

1. Requirements analysis – overview
2. Estimation
3. Prioritization

Prioritizing requirements

- Why?
  - Time and resource limitations: Goal of RE is to add business value
  - Select subset of requirements to produce system that still meets needs

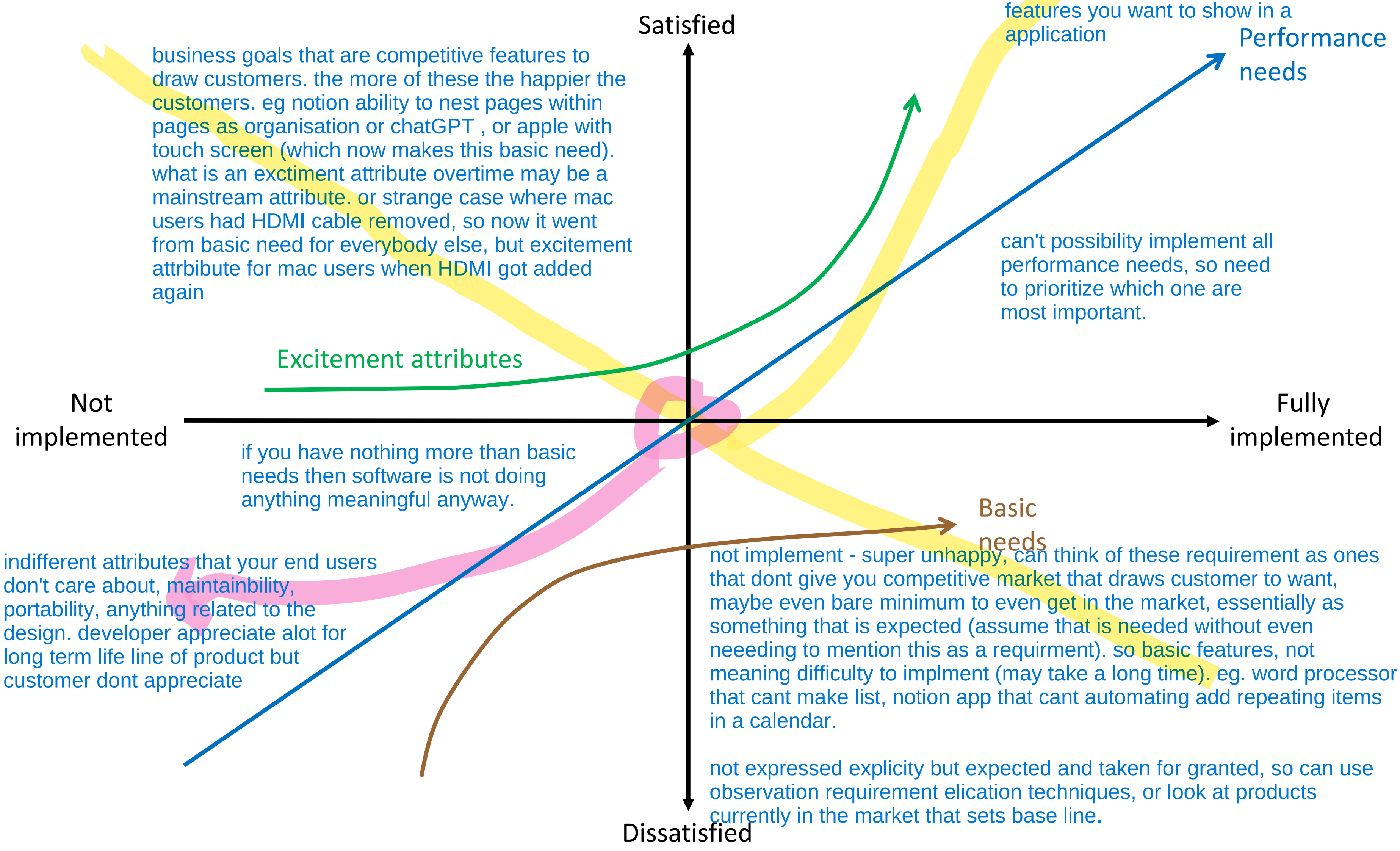
- Who?
  - Prioritization is done by stakeholders
  - Prioritized requirements should be at same level of abstraction

- Based on what?
  - Requirements may be prioritized based on various criteria, e.g.,
    - Necessity, value
    - Cost or time to implement
    - Risk, volatility

Kano model for customer satisfaction

look at requirements from POV of customers, as in end of day want to make them happy

oo much features that you don't need, overloaded noise, notifications, and ads, eg photoprocessor that is overloading with features you dont need because you are a novice. more is not actually better in some case. UX may be impacted by how much features you want to show in a application



business goals that are competitive features to draw customers. the more of these the happier the customers. eg notion ability to nest pages within pages as organisation or chatGPT , or apple with touch screen (which now makes this basic need). what is an exctiment attribute overtime may be a mainstream attribute. or strange case where mac users had HDMI cable removed, so now it went from basic need for everybody else, but excitement attrbibute for mac users when HDMI got added again

can't possibility implement all performance needs, so need to prioritize which one are most important.

if you have nothing more than basic needs then software is not doing anything meaningful anyway.

indifferent attributes that your end users don't care about, maintainbility, portability, anything related to the design. developer appreciate alot for long term life line of product but customer dont appreciate

not implement - super unhappy, can think of these requirement as ones that dont give you competitive market that draws customer to want, maybe even bare minimum to even get in the market, essentially as something that is expected (assume that is needed without even neeeding to mention this as a requirment). so basic features, not meaning difficulty to implment (may take a long time). eg. word processor that cant make list, notion app that cant automating add repeating items in a calendar.

not expressed explicity but expected and taken for granted, so can use observation requirement elication techniques, or look at products currently in the market that sets base line.

Basic needs

- Dissatisfiers, minimum/basic requirements
  - Usually “must haves”, price for entry into a market
  - No opportunity for product differentiation
  - Not usually mentioned by customers, taken for granted
  - If implemented: customers are neutral
  - If absent: extreme dissatisfaction

- Recap – elicitation
  - Typical source: legacy systems, system(s) to be replaced
  - Typical techniques: observational techniques, document analysis

Performance needs

- Critical, key requirements
  - Typical functional requirements from stakeholders
  - The more implemented, the higher satisfaction (and willingness to pay)*
  - Expressed requirements from customers
  - Basis for competition

- Recap – elicitation
  - Typical source: stakeholders
  - Typical techniques: question-based techniques

* Assuming that requirements are used/valuable

Excitement attributes

- “Wow requirements”, “delighters”
  - Often not expressed explicitly, not normally expected
  - Unexpectedly delight customers, thus often unforeseen and unspoken
  - Differentiate us from competition
  - Satisfaction when achieved, do not cause dissatisfaction when not fulfilled

- Recap – elicitation
  - Typical source: stakeholders, market
  - Typical techniques: creativity-based techniques, observational techniques

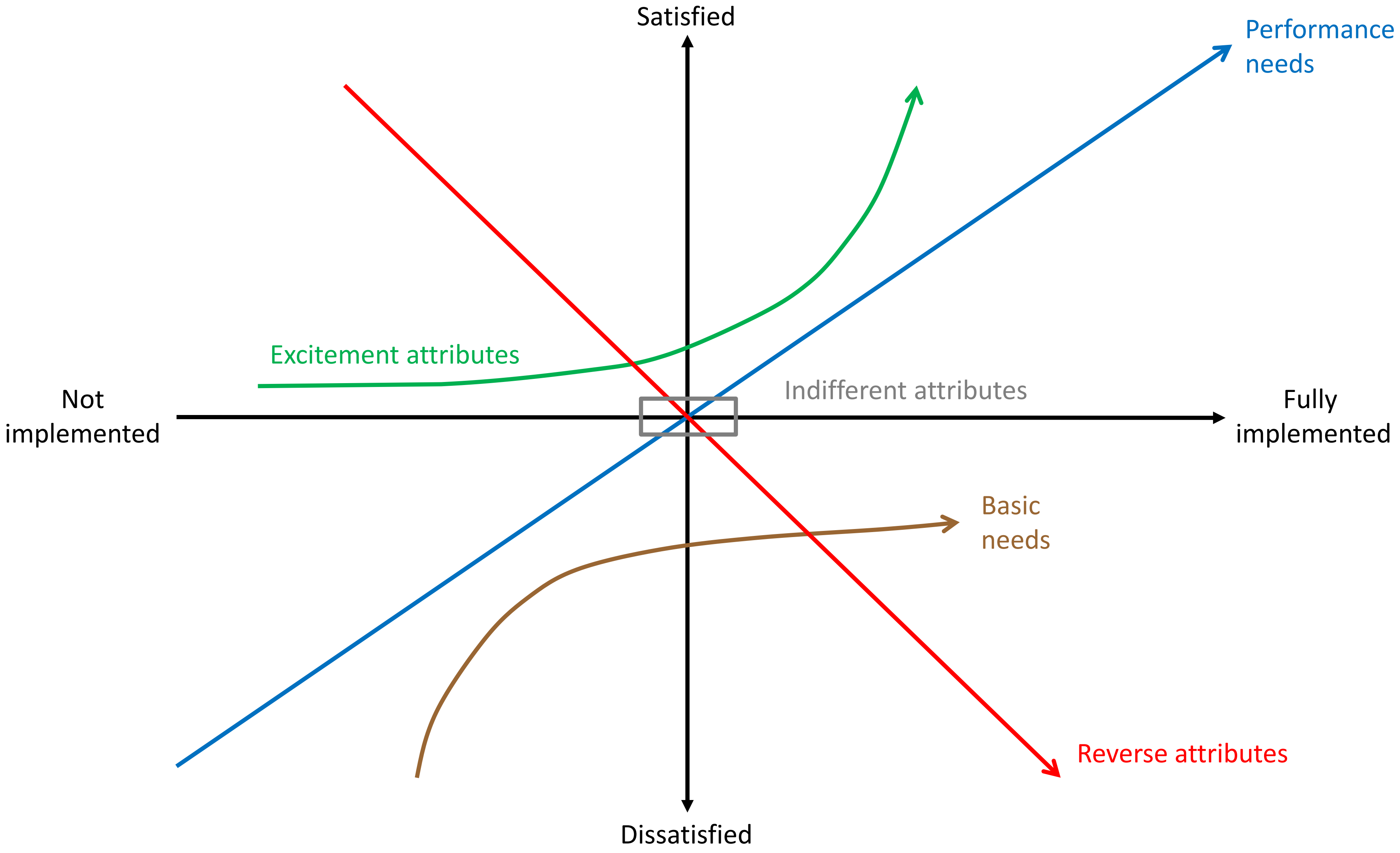
Other categories

security requirement, as sometimes don't appreciate them being there. may not even know they are there or not unless there was a security breach. it a quality attribute so has complexity to prioritize.

- Indifferent attributes
  - Neither good nor bad, do not result in satisfaction/dissatisfaction
  - Often related to design time quality attributes (users are not aware)
  - Try to avoid if about functionality

- Reverse attributes
  - High degree of achievement results in dissatisfaction
  - Why: not all customers are alike
  - Example: expert versus basic versions

Kano model for customer satisfaction



Caution

- Excitement factors become performance factors

  if take too long to ship excitement feature, then can lose its competitive edge.

- Performance factors become basic needs

- Example
  - Mobile phones with touchscreens (required suitable OS) – basic today

Application of Kano’s model

- Rate satisfaction if product has requirement
  - Possible answers: satisfied, neutral, dissatisfied, don’t care

- Example
  - Basic attributes
    - Satisfaction if product has attribute/requirement: neutral
    - Satisfaction if product did not have attribute/requirement: dissatisfied
    - Do not exclude
  - Performance attributes
    - Customers rate most functionality as important and might be dissatisfied
    - Trade-off: how much extra would user be willing to pay?

Simple prioritization (by necessity)

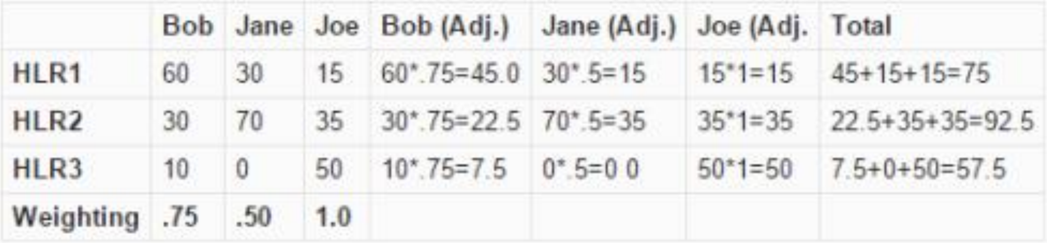
- Importance for success of system (coarse-grained priorities)
  - Critical (essential or mandatory)
    - System will not be accepted if such requirement is not met
  - Major (conditional, desirable, important, optional)
    - System should meet these requirements; not meeting them is no show stopper
  - Minor (nice-to-have or optional)
    - Implementing these requirements is nice, but not needed

Single-criterion prioritization

- Simple ranking
  - Rank according to a given criterion (e.g., urgency, value, complexity)
  - Evaluation based on a bipolar scale (Likert scale)
    - Extremes: “very important“, “very unimportant“
    - Intermediate intervals
- Pro
  - Easy and fast to carry out
  - Limited options
- Con
  - Gives impression that requirements are evaluated independently
  - Little differentiation is possible
  - Careful with deriving mean values, etc. on Likert scales

Cumulative voting

- Cumulative prioritization (e.g., 100 dollar test) and weighting
  - Assigning points: total of *n* points distributed among *m* requirements



	Bob	Jane	Joe	Bob (Adj.)	Jane (Adj.)	Joe (Adj.	Total
HLR1	60	30	15	60*.75=45.0	30*.5=15	15*1=15	45+15+15=75
HLR2	30	70	35	30*.75=22.5	70*.5=35	35*1=35	22.5+35+35=92.5
HLR3	10	0	50	10*.75=7.5	0*.5=0 0	50*1=50	7.5+0+50=57.5
Weighting	**.75**	**.50**	**1.0**				

- Pro
  - Easy and fast to carry out
  - More differentiated evaluation than with the Likert scale method
  - The value "0" can be assigned

- Con
  - High number of requirements; no consideration of dependencies

Multi-criteria prioritization

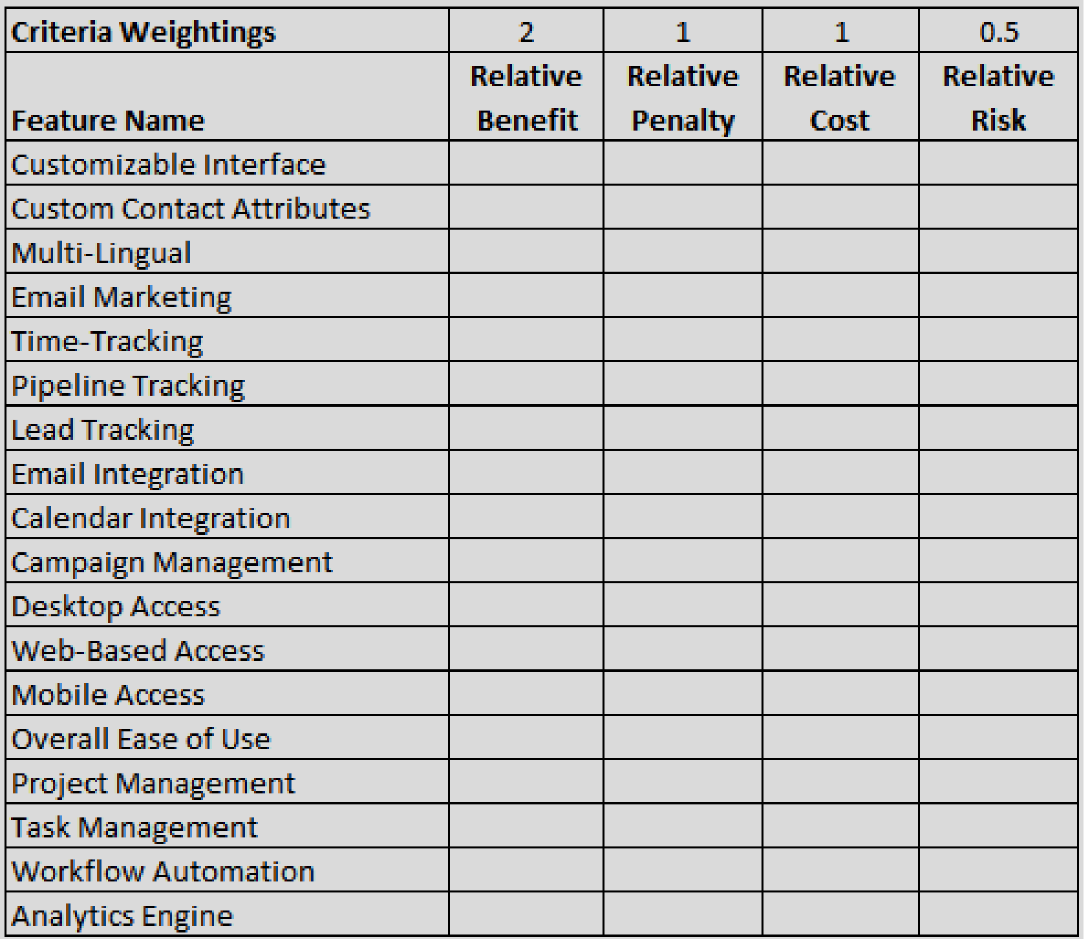
- Ranking based on combining ranking of different criteria
  - Criteria may have different weights

- Examples
  - Wiegers' matrix (Wiegers 1999)
    - Estimates relative benefit, detriment, cost, and risk for each requirement
    - Uses these values to calculate a weighted priority
      - (W Benefit % + W Penalty %) / (W Cost % + W Risk %)
    - Ranks according to calculated priority values
  - AHP – Analytic Hierarchy Process (Saaty 1980)
    - Algorithmic multi-criterion decision making process
    - Relies on pair-wise comparison

expand on peer to peer, most cited, but not very scable so ppl usually extend this.

Wieger’s matrix (example)

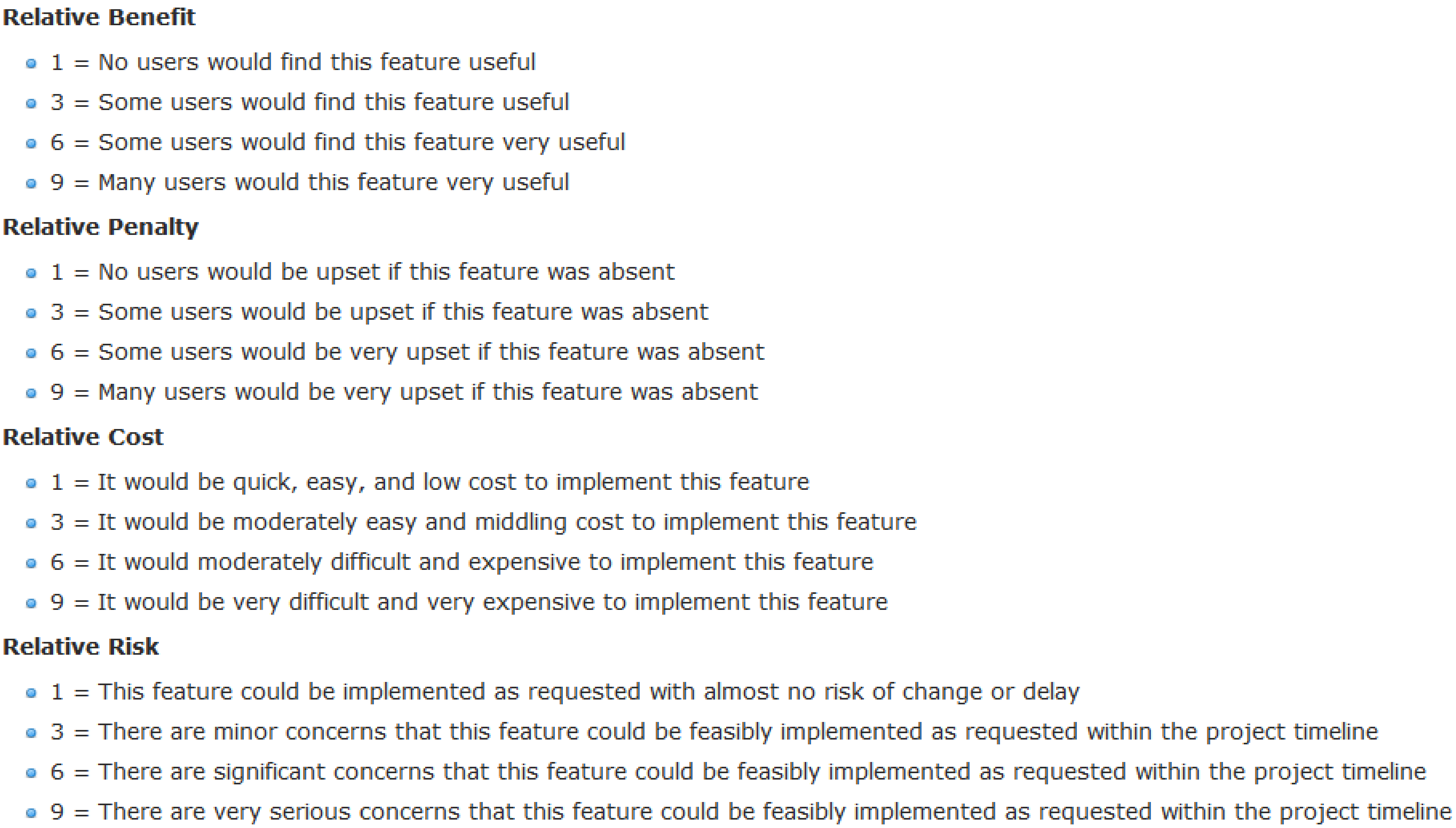
(W Benefit % + W Penalty %) / (W Cost % + W Risk %)



Criteria Weightings	2	1	1	0.5
Feature Name	Relative Benefit	Relative Penalty	Relative Cost	Relative Risk
Customizable Interface				
Custom Contact Attributes				
Multi-Lingual				
Email Marketing				
Time-Tracking				
Pipeline Tracking				
Lead Tracking				
Email Integration				
Calendar Integration				
Campaign Management				
Desktop Access				
Web-Based Access				
Mobile Access				
Overall Ease of Use				
Project Management				
Task Management				
Workflow Automation				
Analytics Engine				

http://www.bawiki.com/wiki/techniques/matrix-prioritization/

Wieger’s matrix (meaning of values)



Relative Benefit

- 1 = No users would find this feature useful
- 3 = Some users would find this feature useful
- 6 = Some users would find this feature very useful
- 9 = Many users would this feature very useful

Relative Penalty

- 1 = No users would be upset if this feature was absent
- 3 = Some users would be upset if this feature was absent
- 6 = Some users would be very upset if this feature was absent
- 9 = Many users would be very upset if this feature was absent

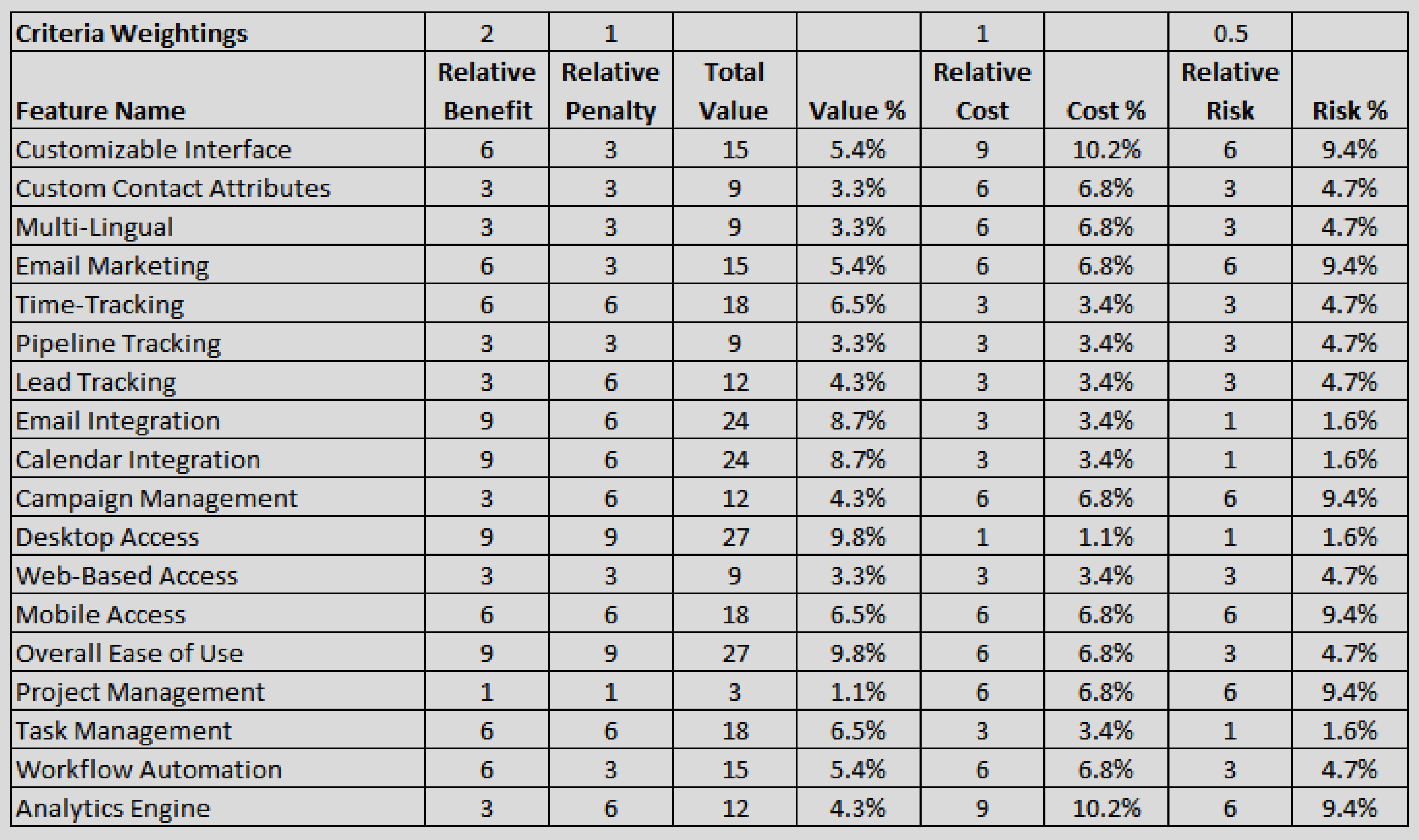
Relative Cost

- 1 = It would be quick, easy, and low cost to implement this feature
- 3 = It would be moderately easy and middling cost to implement this feature
- 6 = It would moderately difficult and expensive to implement this feature
- 9 = It would be very difficult and very expensive to implement this feature

Relative Risk

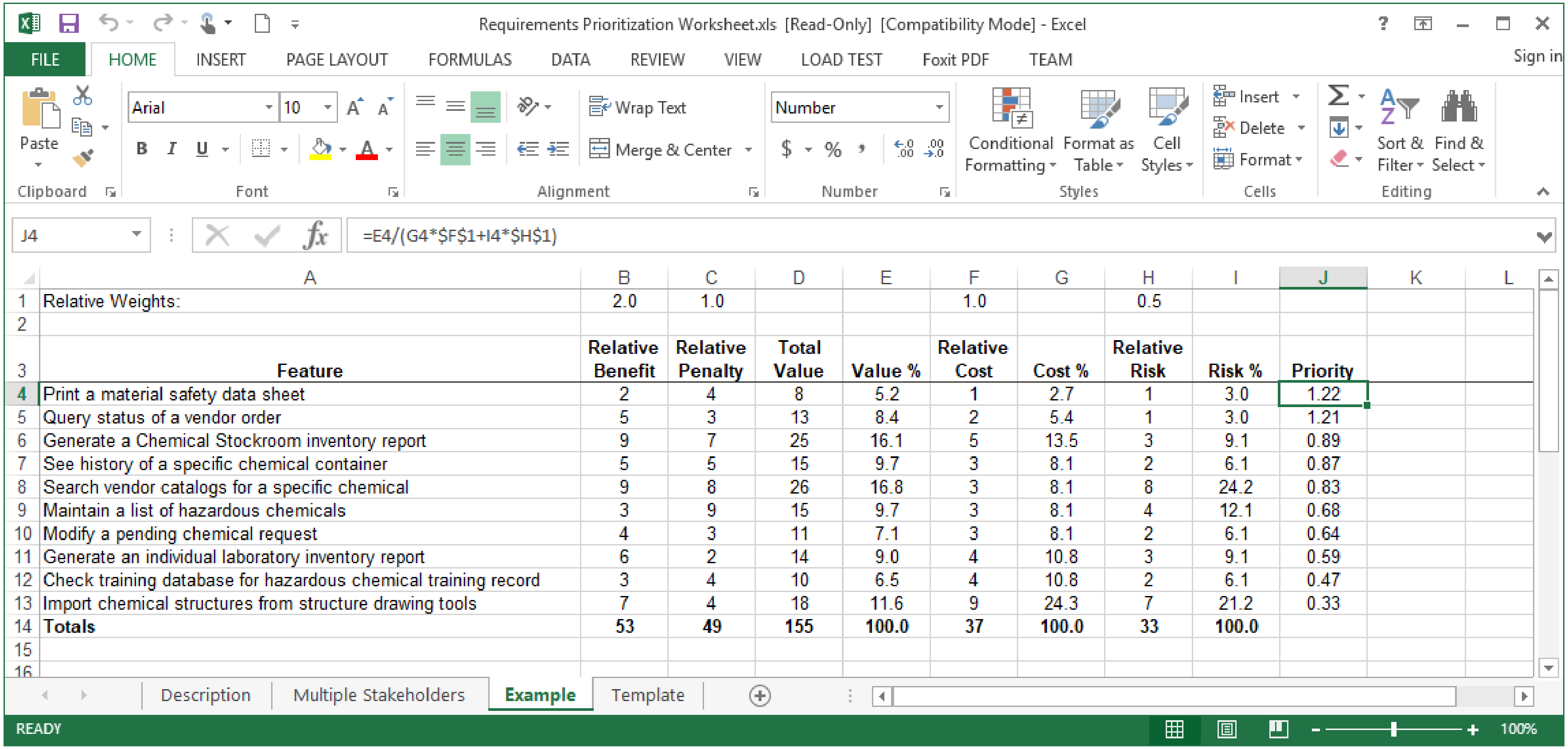
- 1 = This feature could be implemented as requested with almost no risk of change or delay
- 3 = There are minor concerns that this feature could be feasibly implemented as requested within the project timeline
- 6 = There are significant concerns that this feature could be feasibly implemented as requested within the project timeline
- 9 = There are very serious concerns that this feature could be feasibly implemented as requested within the project timeline

Wieger’s matrix: filled



Criteria Weightings	2	1			1		0.5	
Feature Name	Relative Benefit	Relative Penalty	Total Value	Value %	Relative Cost	Cost %	Relative Risk	Risk %
Customizable Interface	6	3	15	5.4%	9	10.2%	6	9.4%
Custom Contact Attributes	3	3	9	3.3%	6	6.8%	3	4.7%
Multi-Lingual	3	3	9	3.3%	6	6.8%	3	4.7%
Email Marketing	6	3	15	5.4%	6	6.8%	6	9.4%
Time-Tracking	6	6	18	6.5%	3	3.4%	3	4.7%
Pipeline Tracking	3	3	9	3.3%	3	3.4%	3	4.7%
Lead Tracking	3	6	12	4.3%	3	3.4%	3	4.7%
Email Integration	9	6	24	8.7%	3	3.4%	1	1.6%
Calendar Integration	9	6	24	8.7%	3	3.4%	1	1.6%
Campaign Management	3	6	12	4.3%	6	6.8%	6	9.4%
Desktop Access	9	9	27	9.8%	1	1.1%	1	1.6%
Web-Based Access	3	3	9	3.3%	3	3.4%	3	4.7%
Mobile Access	6	6	18	6.5%	6	6.8%	6	9.4%
Overall Ease of Use	9	9	27	9.8%	6	6.8%	3	4.7%
Project Management	1	1	3	1.1%	6	6.8%	6	9.4%
Task Management	6	6	18	6.5%	3	3.4%	1	1.6%
Workflow Automation	6	3	15	5.4%	6	6.8%	3	4.7%
Analytics Engine	3	6	12	4.3%	9	10.2%	6	9.4%

Example tool*



J4 =E4/(G4*F1+I4*H1)

	A	B	C	D	E	F	G	H	I	J
1	Relative Weights:	2.0	1.0			1.0		0.5		
2										
3	**Feature**	**Relative Benefit**	**Relative Penalty**	**Total Value**	**Value %**	**Relative Cost**	**Cost %**	**Relative Risk**	**Risk %**	**Priority**
4	Print a material safety data sheet	2	4	8	5.2	1	2.7	1	3.0	1.22
5	Query status of a vendor order	5	3	13	8.4	2	5.4	1	3.0	1.21
6	Generate a Chemical Stockroom inventory report	9	7	25	16.1	5	13.5	3	9.1	0.89
7	See history of a specific chemical container	5	5	15	9.7	3	8.1	2	6.1	0.87
8	Search vendor catalogs for a specific chemical	9	8	26	16.8	3	8.1	8	24.2	0.83
9	Maintain a list of hazardous chemicals	3	9	15	9.7	3	8.1	4	12.1	0.68
10	Modify a pending chemical request	4	3	11	7.1	3	8.1	2	6.1	0.64
11	Generate an individual laboratory inventory report	6	2	14	9.0	4	10.8	3	9.1	0.59
12	Check training database for hazardous chemical training record	3	4	10	6.5	4	10.8	2	6.1	0.47
13	Import chemical structures from structure drawing tools	7	4	18	11.6	9	24.3	7	21.2	0.33
14	**Totals**	**53**	**49**	**155**	**100.0**	**37**	**100.0**	**33**	**100.0**	
15										
16										

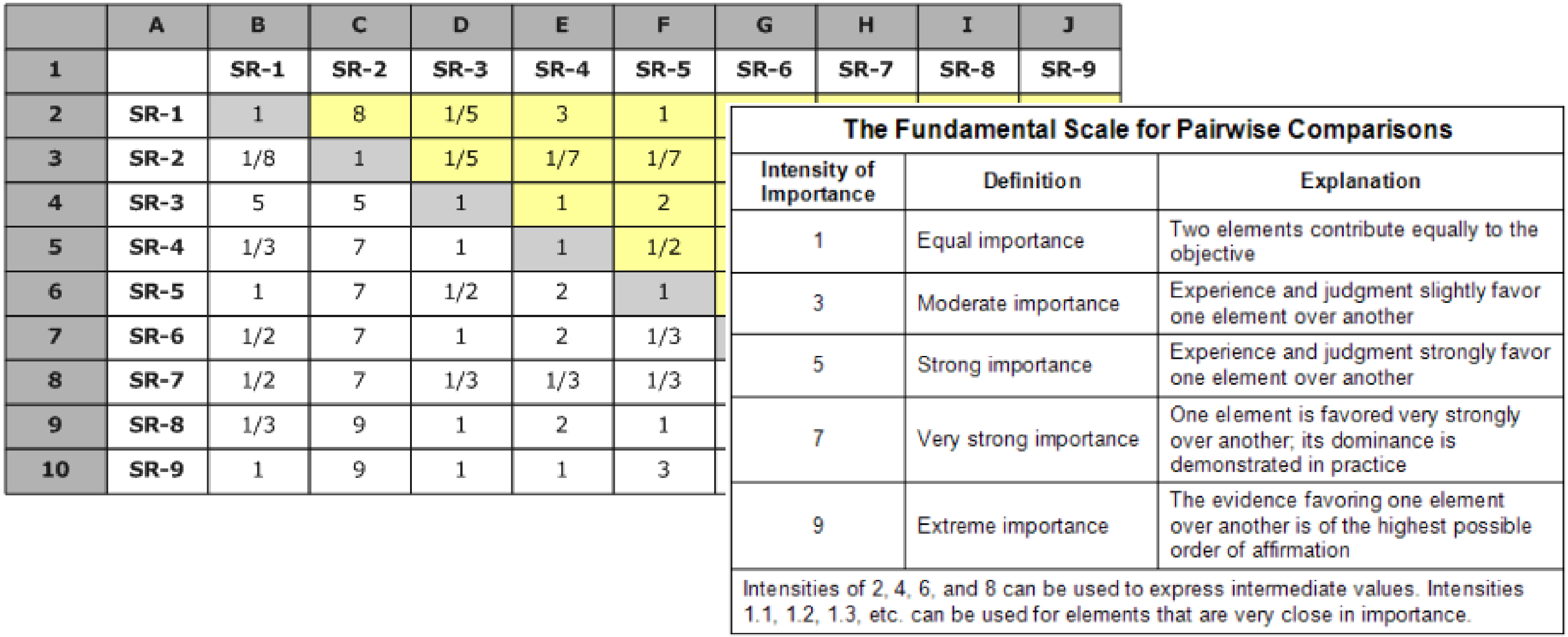
* https://cs.anu.edu.au/courses/comp3120/public_docs/Requirements Prioritization Worksheet.xls

Wieger’s matrix

- Pro
  - Easy to carry out (Likert scale) and for stakeholder to undesrtand, and more guided for stakeholders
  - Several factors are part of the calculation

- Con
  - Evaluation of the factors
  - Weighting factors

AHP (example)



	A	B	C	D	E	F	G	H	I	J
1		SR-1	SR-2	SR-3	SR-4	SR-5	SR-6	SR-7	SR-8	SR-9
2	SR-1	1	8	1/5	3	1				
3	SR-2	1/8	1	1/5	1/7	1/7				
4	SR-3	5	5	1	1	2				
5	SR-4	1/3	7	1	1	1/2				
6	SR-5	1	7	1/2	2	1				
7	SR-6	1/2	7	1	2	1/3				
8	SR-7	1/2	7	1/3	1/3	1/3				
9	SR-8	1/3	9	1	2	1				
10	SR-9	1	9	1	1	3				

The Fundamental Scale for Pairwise Comparisons

Intensity of Importance	Definition	Explanation
1	Equal importance	Two elements contribute equally to the objective
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Strong importance	Experience and judgment strongly favor one element over another
7	Very strong importance	One element is favored very strongly over another; its dominance is demonstrated in practice
9	Extreme importance	The evidence favoring one element over another is of the highest possible order of affirmation

Intensities of 2, 4, 6, and 8 can be used to express intermediate values. Intensities 1.1, 1.2, 1.3, etc. can be used for elements that are very close in importance.

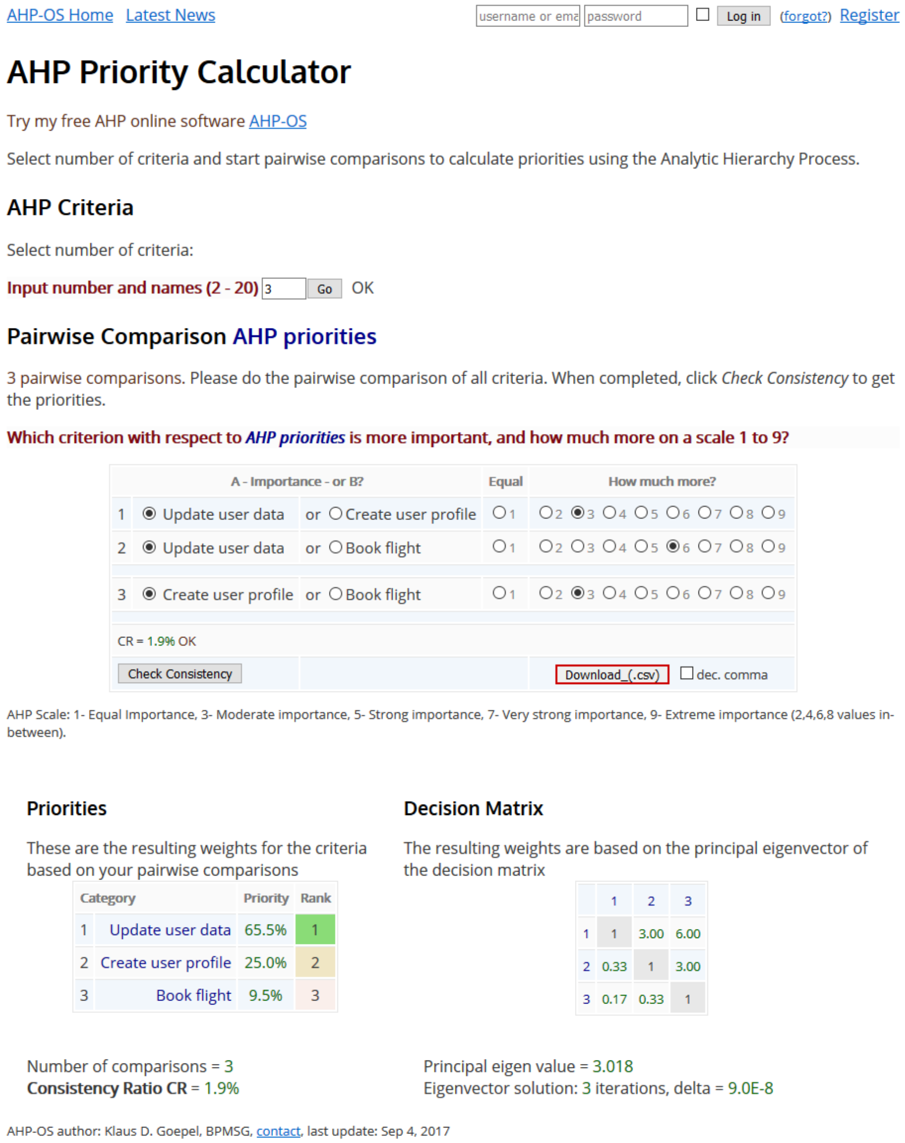
- Results in priority vector
  - Ranked list of requirements

if i compare this to that can make a more accurate decision on whats prioritized, instead of all knowing that they are important

pairwise compare when theres lots can have

- Provides consistency ratio (aim for < 10%)

Example tool*



AHP-OS Home Latest News username or ema password □ Log in (forgot?) Register

AHP Priority Calculator

Try my free AHP online software AHP-OS

Select number of criteria and start pairwise comparisons to calculate priorities using the Analytic Hierarchy Process.

AHP Criteria

Select number of criteria:

Input number and names (2 - 20) 3 Go OK

Pairwise Comparison AHP priorities

3 pairwise comparisons. Please do the pairwise comparison of all criteria. When completed, click *Check Consistency* to get the priorities.

Which criterion with respect to *AHP priorities* is more important, and how much more on a scale 1 to 9?

	A - Importance - or B?			Equal	How much more?
1	● Update user data	or	○ Create user profile	○ 1	○ 2 ● 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9
2	● Update user data	or	○ Book flight	○ 1	○ 2 ○ 3 ○ 4 ○ 5 ● 6 ○ 7 ○ 8 ○ 9
3	● Create user profile	or	○ Book flight	○ 1	○ 2 ● 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9

CR = 1.9% OK

Check Consistency Download (.csv) □ dec. comma

AHP Scale: 1- Equal Importance, 3- Moderate importance, 5- Strong importance, 7- Very strong importance, 9- Extreme importance (2,4,6,8 values in-between).

Priorities

These are the resulting weights for the criteria based on your pairwise comparisons

	Category	Priority	Rank
1	Update user data	65.5%	1
2	Create user profile	25.0%	2
3	Book flight	9.5%	3

Number of comparisons = 3
Consistency Ratio CR = 1.9%

Decision Matrix

The resulting weights are based on the principal eigenvector of the decision matrix

	1	2	3
1	1	3.00	6.00
2	0.33	1	3.00
3	0.17	0.33	1

Principal eigen value = 3.018
Eigenvector solution: 3 iterations, delta = 9.0E-8

AHP-OS author: Klaus D. Goepel, BPMSG, contact, last update: Sep 4, 2017

* https://bpmsg.com/ahp/ahp-calc.php

AHP – why pair-wise comparisons

- Prioritization based on relative value rather than absolute
  - Easy, faster to tell that something is larger than something else
    - E.g., “A has value of 5” versus “A has higher value than B”
  - Might be more accurate, trustworthy

min compare is n*(N/2) so not scalable

- Partially verifiable

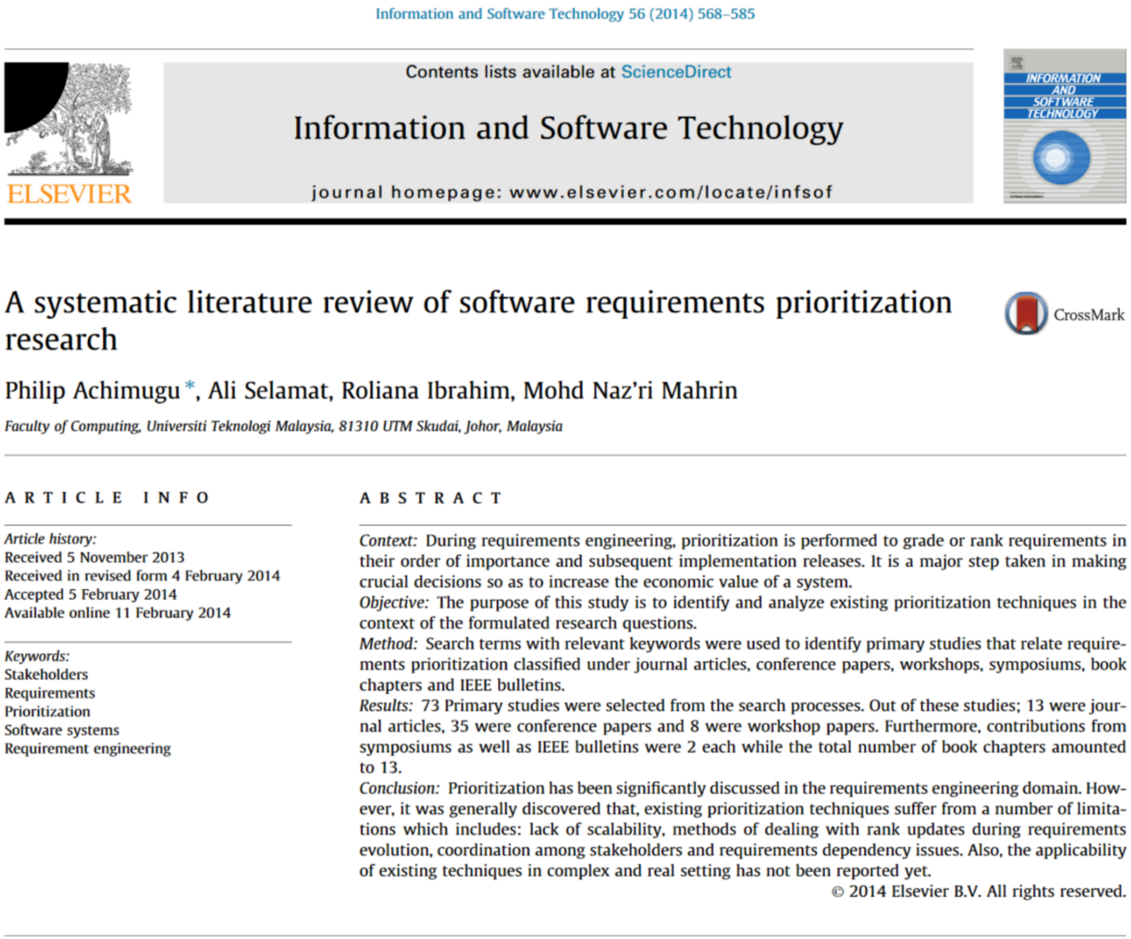
- Tool support

- Industry-proven

AHP – known issues

- Consistency
  - Use measures such as consistency ratio
- Interdependency of requirements
  - E.g., high-cost and low-value requirement should be implemented first
  - Bundle requirements
- Number of pairwise comparisons is $O(n^2)$
  - Tool support (calculation, rationale documentation)
  - Prioritize only subset of all requirements

Group reading



Information and Software Technology 56 (2014) 568–585

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A systematic literature review of software requirements prioritization research

Philip Achimugu *, Ali Selamat, Roliana Ibrahim, Mohd Naz'ri Mahrin

Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia

ARTICLE INFO

Article history:
Received 5 November 2013
Received in revised form 4 February 2014
Accepted 5 February 2014
Available online 11 February 2014

Keywords:
Stakeholders
Requirements
Prioritization
Software systems
Requirement engineering

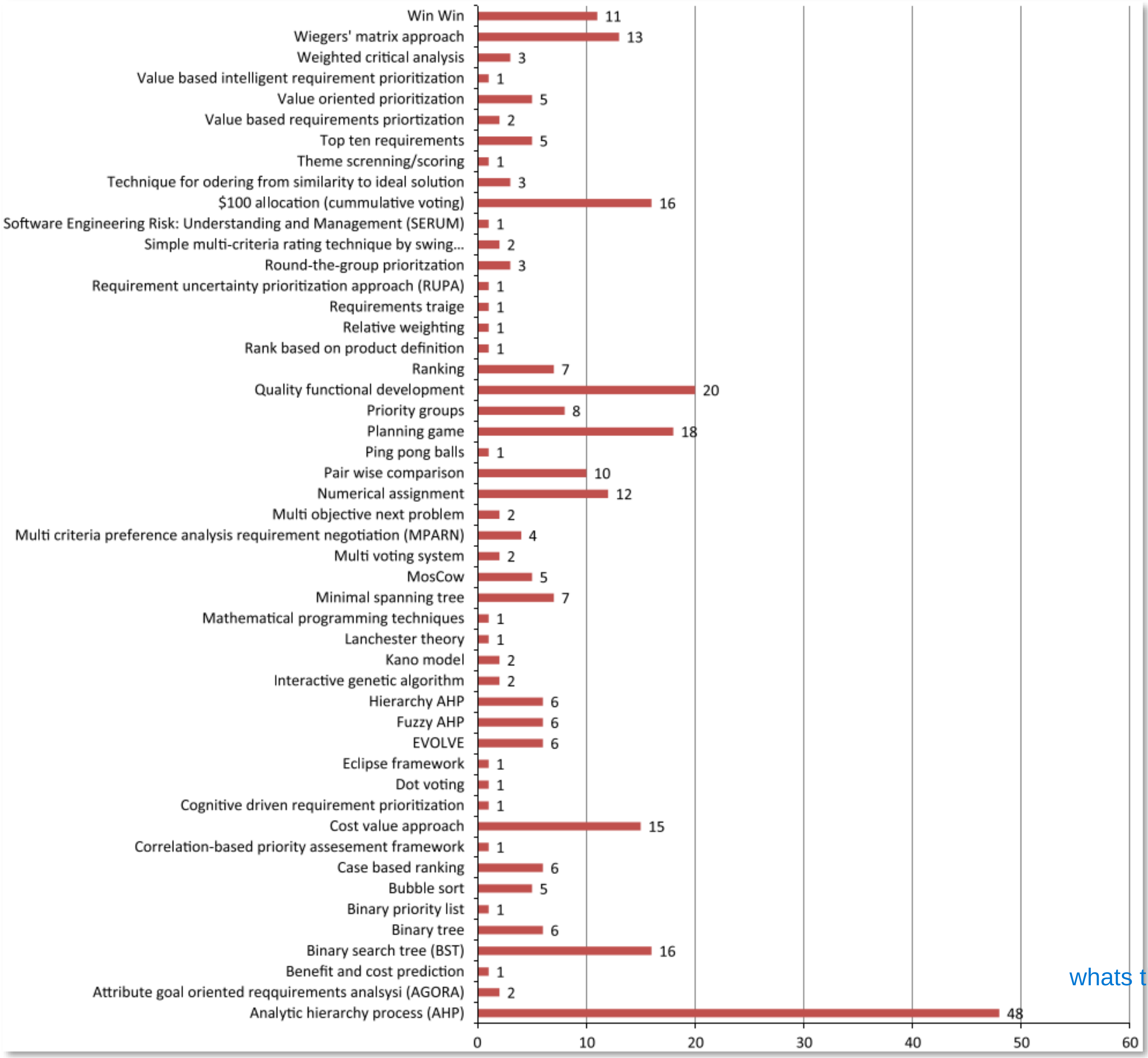
ABSTRACT

Context: During requirements engineering, prioritization is performed to grade or rank requirements in their order of importance and subsequent implementation releases. It is a major step taken in making crucial decisions so as to increase the economic value of a system.
Objective: The purpose of this study is to identify and analyze existing prioritization techniques in the context of the formulated research questions.
Method: Search terms with relevant keywords were used to identify primary studies that relate requirements prioritization classified under journal articles, conference papers, workshops, symposiums, book chapters and IEEE bulletins.
Results: 73 Primary studies were selected from the search processes. Out of these studies; 13 were journal articles, 35 were conference papers and 8 were workshop papers. Furthermore, contributions from symposiums as well as IEEE bulletins were 2 each while the total number of book chapters amounted to 13.
Conclusion: Prioritization has been significantly discussed in the requirements engineering domain. However, it was generally discovered that, existing prioritization techniques suffer from a number of limitations which includes: lack of scalability, methods of dealing with rank updates during requirements evolution, coordination among stakeholders and requirements dependency issues. Also, the applicability of existing techniques in complex and real setting has not been reported yet.

Study overview

- Study question
  - What are the existing techniques used for prioritizing requirements?
  - What are the descriptions and limitations of existing techniques?
  - What taxonomy of prioritization scales does each technique exhibit?
  - What are the processes involved in software requirements prioritization
- Method
  - Literature review
- Your task (in pairs)
  - Read paper, focus on one approach
  - Prepare summary, including example (should fit on one slide)

Summary of findings



whats the practical application

Summary

1. Requirements analysis – overview ✓

2. Estimation ✓

3. Prioritization ✓