

Software Requirements and Architecture (SENG404)

Matthias Galster

Lecture 8 – Requirements management

March 29, 2023

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; Requirements analysis, Requirements validation
10	6	March 29	Requirements management
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 2

Student(s)	Project

Previous lecture

1. Requirements validation – overview
2. Requirements validation techniques

Reading for this session

- S. Jayatilleke and R. Lai. *A systematic review of requirements change management*. Information and Software Technology, 2018, pp. 163-185, doi: 10.1016/j.infsof.2017.09.004

agile something use excuse that things will change anyway to not specify requirements, however if we have specified at the start maybe would have stuck to it.

need a history of change to predict change. but can be risky to think future will be same as past.

would exec level change management take technical details into account when making decisions like this? should they?

executive changes, need to understand there is ripple effects (possible to down to technical level), so are you interested to understand this ripple effect

Questions and lessons



- https://jamboard.google.com/d/1Hn9QmwqDMsqWBr_4FI47UzOtA02smqwH6-Dy1A6Ju2g/edit?usp=sharing

Viewing for next session

- S. Brown. *Five things every developer should know about software architecture*. GoTo2020, <https://www.youtube.com/watch?v=9Az0q2XHtH8> (29 minutes, 44 seconds)

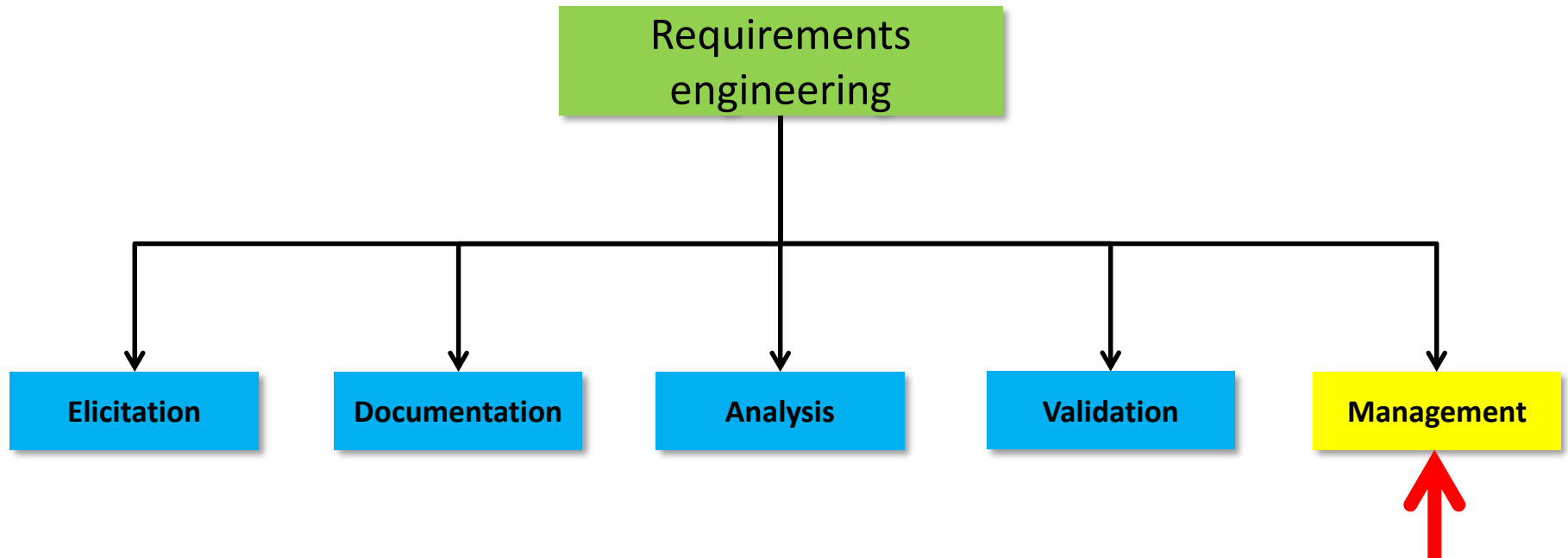
Agenda

1. Requirements management
2. Release management and planning

Agenda

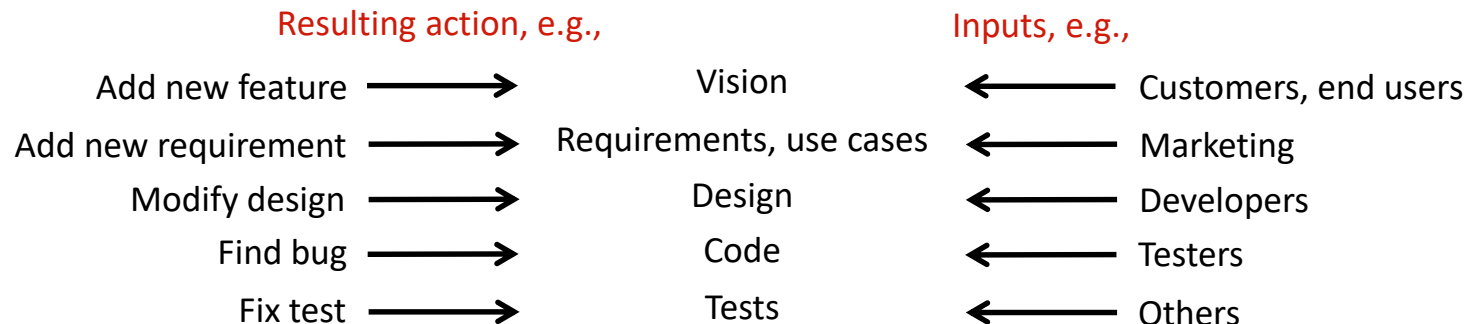
1. Requirements management
2. Release management and planning

Generic RE process activities



Requirements management

- Activities related to
 - identifying, understanding, and controlling
 - **changing (new, obsolete, updated) requirements**
 - **during system life time**
- Part of overall project and process management
 - Coordinate requirements-related activities
 - Link requirements-related activities to other activities



Enduring vs volatile requirements

Enduring



- **Stable**, typically related to
 - **core activities**
 - **domain** of system



Examples?

Volatile

higher chance of change



- **Change**
 - during development
 - when system is in use



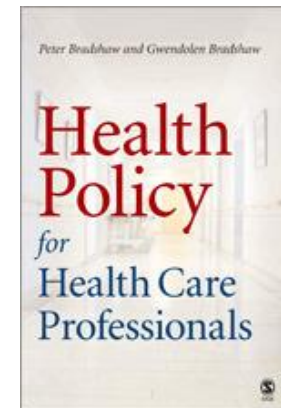
Examples?

ref model of customer satisfaction graph, more established on bottom of graph is more enduring
quality requirement - enduring? 11:47 am 29th march

Example: hospital



Requirements will always relate to doctors, nurses, etc.



Requirements derived from health care policy

Reasons for changing requirements

Environment/context

some can be essential, and some accidental too

“External” factors

Difficult for engineers
to control

Organization, project, etc.

“Internal” factors

Reasons for changing requirements – external

Environment/context



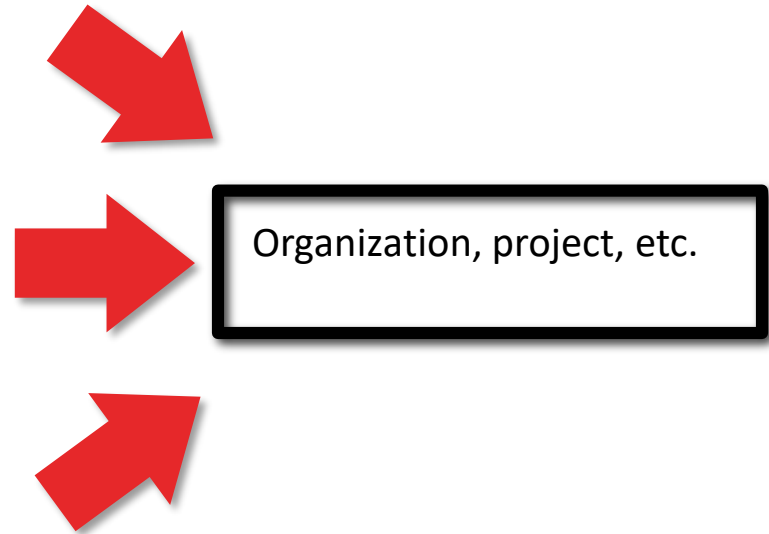
Users change their mind
(e.g., based on what they see)



Business environment change
(e.g., market, business strategy)



Technical environment change
(e.g., hardware, interfaces, software)



Example (hospital)

- Changes in funding of patient care
- Changes in treatment information

Reasons for changing requirements – internal

Environment/context

Organization, project, etc.



Incomplete, flawed elicitation and analysis



Learning during development



Introducing new system may change processes; new ways of working



Change in other software or business processes in organization

Reasons for changing requirements – examples

Environment/context

Hospital (patient management system)



Design process uncovers new requirements

- Missed data requirements
- New data format for storing treatment information



Introduction of patient management system

- New feature in hospital software
 - Electronic patient record
- (Consequential) new way of working/requirement
 - Electronic exchange of patient information

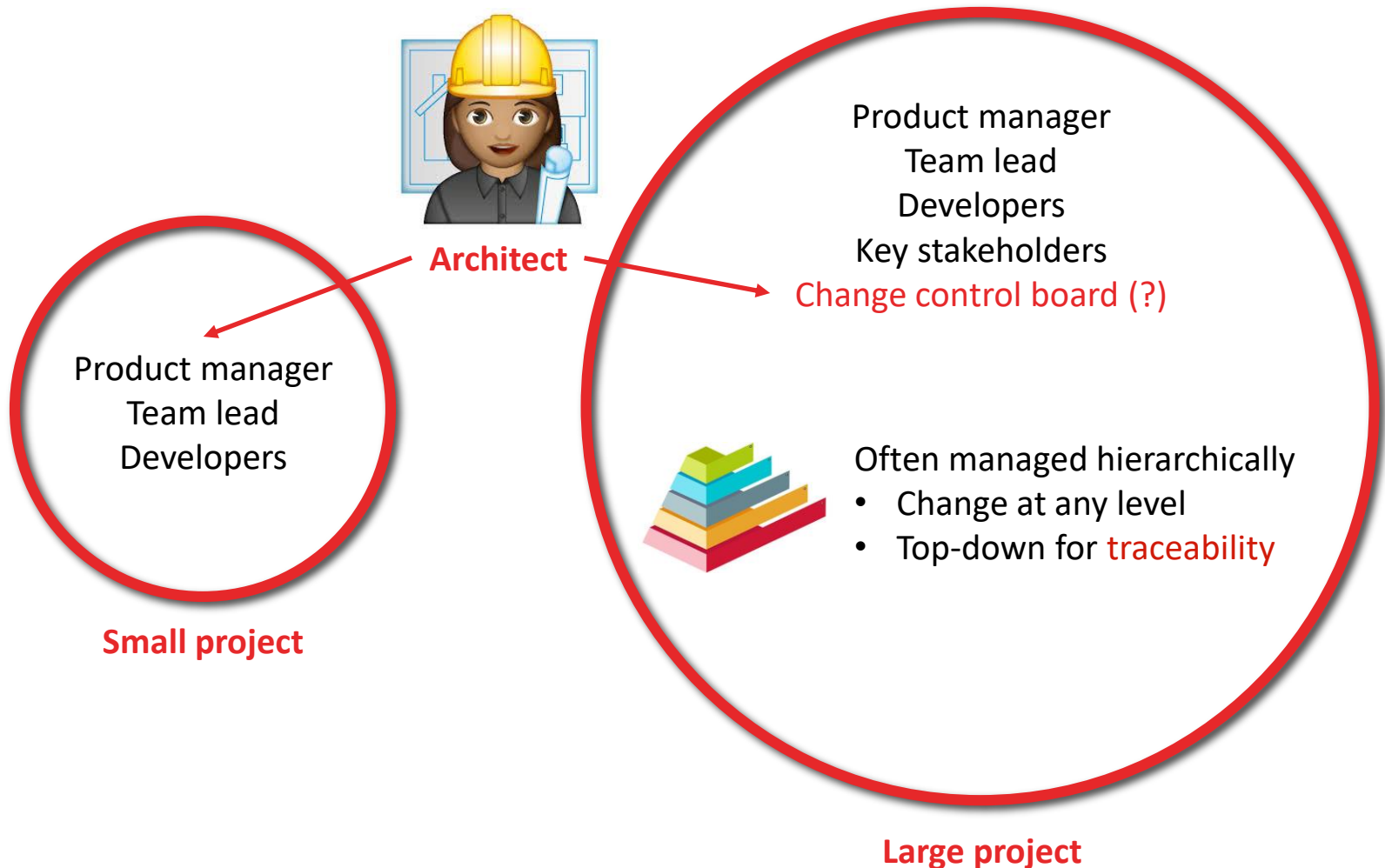


Change in other systems in hospital

- Existing system/business process
 - New organizational structure or ERP
- New compatibility
 - New communication protocol or format

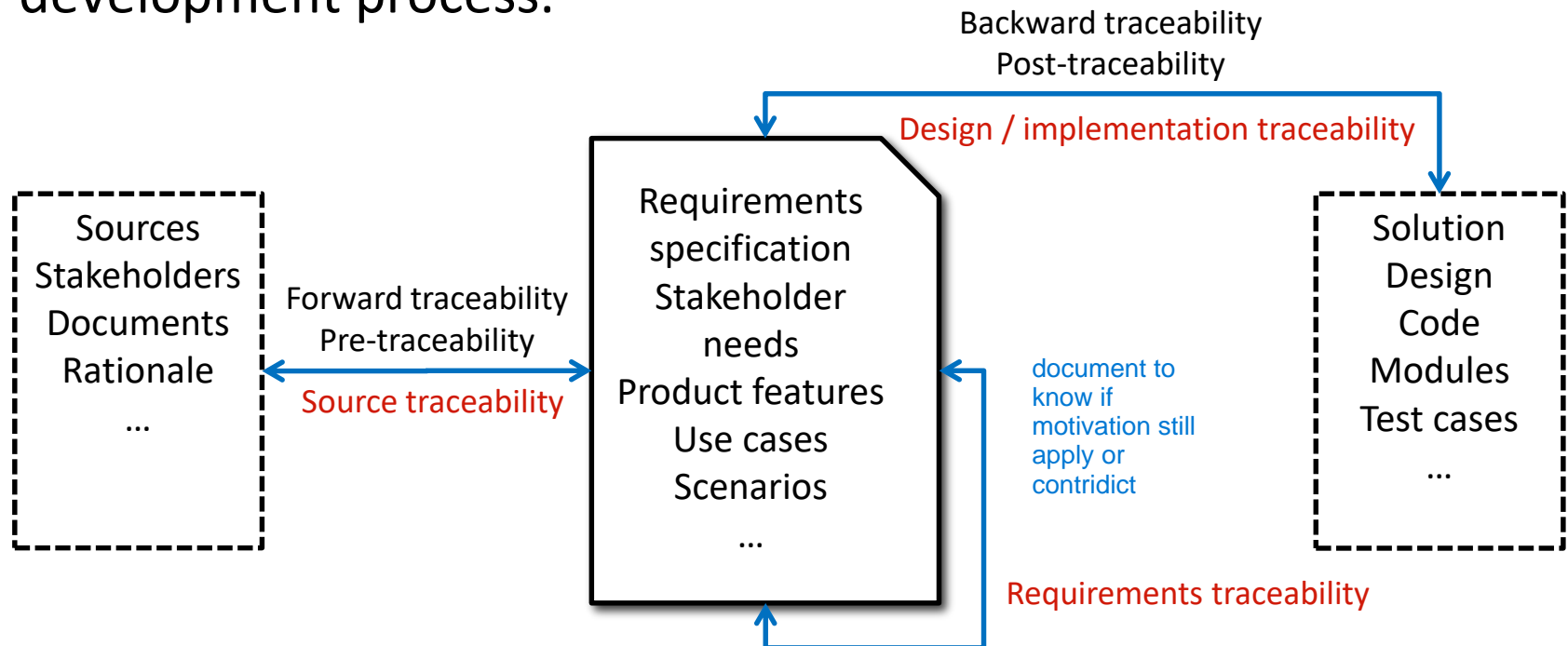
Single channel of change control

- Responsibilities to make “official” decision about change

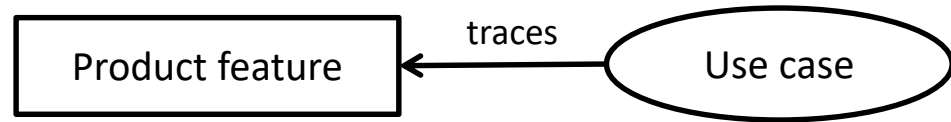


Traceability

- “[...] degree to which a relationship can be established between two or more products of the development process [...].”
- Traces:** “[...] **relationships** between two or more products of the development process.”



Examples



Use case 1 is use case without features

	Use case 1	Use case 2	Use case i	Use case k
Feature 1				x
Feature 2		x	x	
Feature n				
Feature m		x		

Feature n needs to be defined by some use case

	Feature 1	...	Feature i	Feature k
Goal 1	x			
Goal 2		x		x
Goal n		x	x	
Goal m				x

	NFR 1	...	NFR i	NFR k
Requirement 1	x			
Requirement 2		x		x
Requirement n		x	x	
Requirement m				x

Establishing and maintaining traces

- **Manually** create traces when creating artifacts
 - Requirements change requires updating traces, **high effort - trade-off?**
- **Automatically** create trace links between artifacts
 - May use tagging, information retrieval
- Traceability tools
 - For **small projects**, few requirements (1-10 person years, KLOC)
 - Spreadsheets, databases, basic traceability matrix, tagging, etc.
 - For **larger projects** basic traceability matrix expensive to maintain
 - Specialized and automated requirements management tools
 - Telelogic Doors, IBM RequisitePro, Jira, ReleasePlanner, RMTrak, etc.¹

¹<http://makingofsoftware.com/resources/list-of-rm-tools> (as of March 2020)

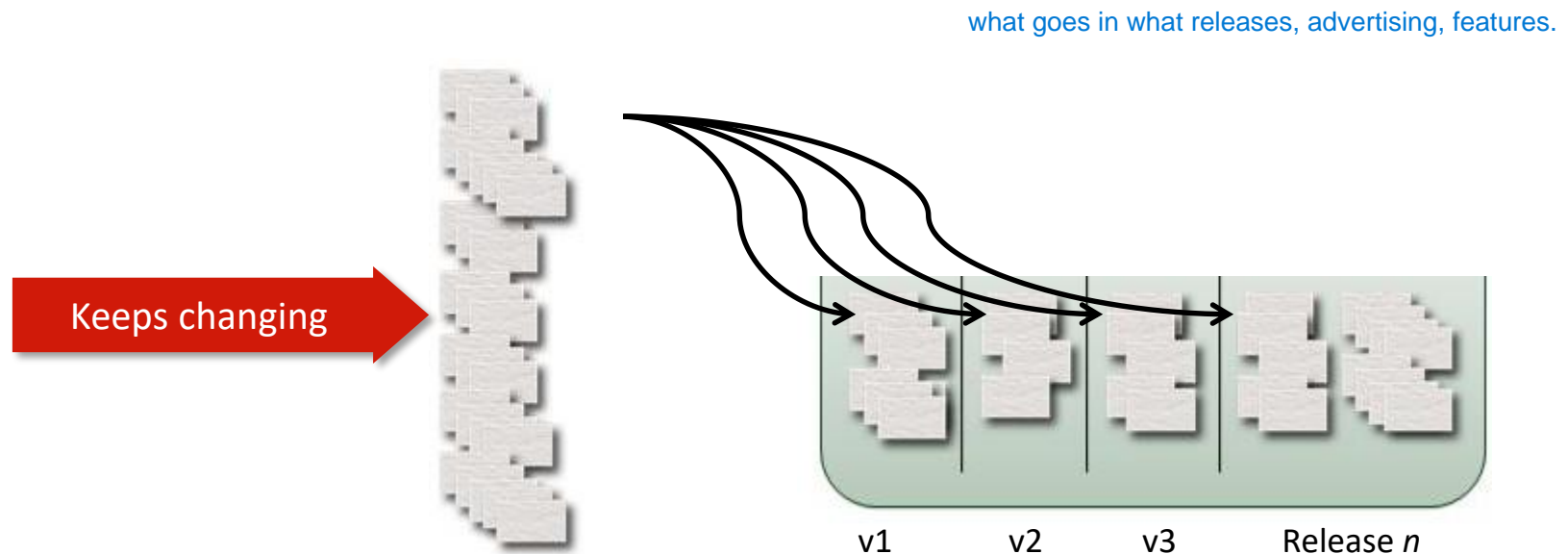
See also:
M. Panis: *Reflections on choosing a new requirements management tool*. IEEE Software 39(1): 7-10 (2022)
J. M. Carrillo de Gea et al. *Requirements engineering tools: An evaluation*. IEEE Software 38(3): 17-24 (2021)

Agenda

1. Requirements management
2. Release management and planning

What is it?

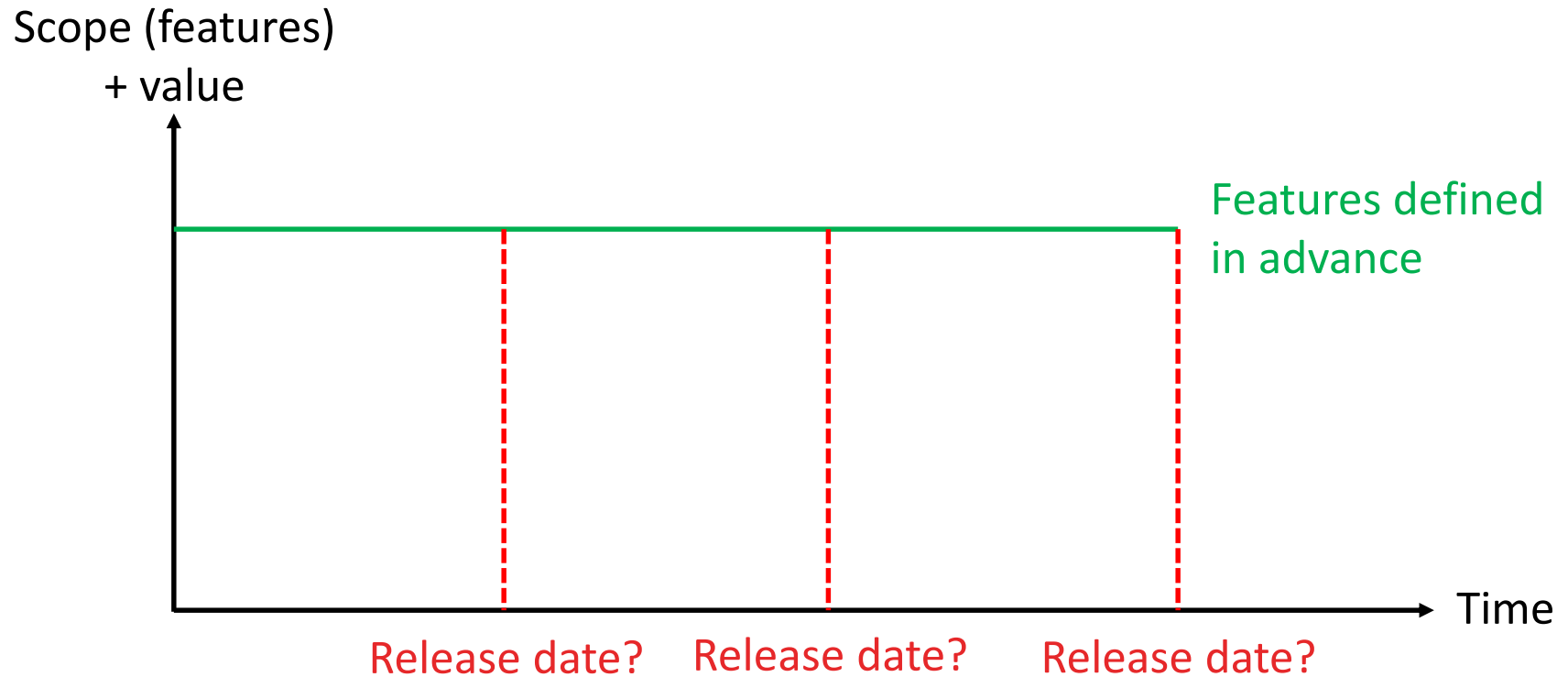
- Look forward a couple of releases (iterations, sprints)
- Make predictions of what could be delivered by then



Typical release decisions

- Which features should (should not) be offered in next release(s)?
- When is the best time for a product release?
hospital features in winter may be not great timing. computer games or new device models before christmas. tax related release not send out before elections, instead bug features instead
- Which previously planned features should be replaced?
eg headphone jack in iphones
- When and how often can or should we re-plan?
less specific planning for items far in future
- **Typical factors**
 - Competitors, market, process, resources, etc.
 - Plan at the last responsible moment (**reduce waste**)
 - The further away release, the fewer details in release plan

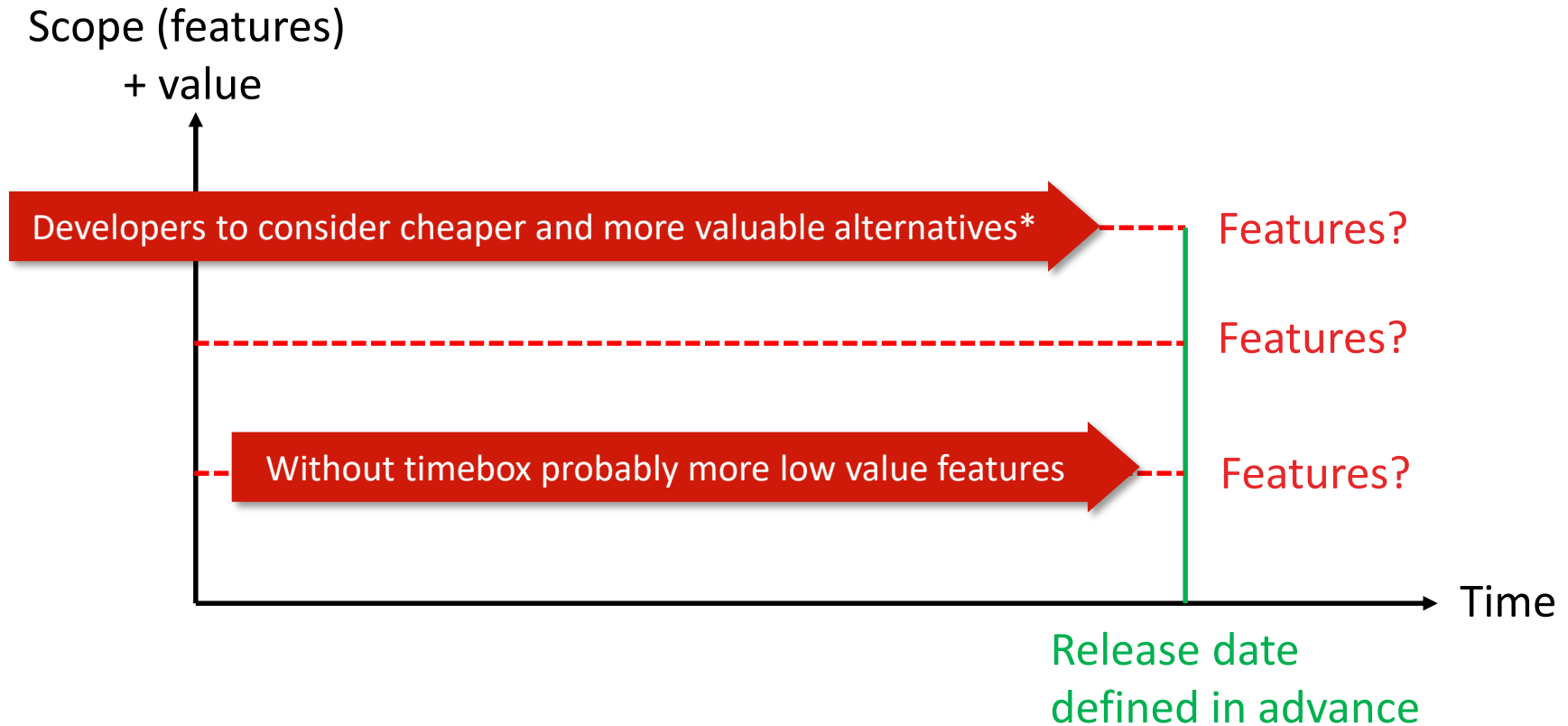
Scope-boxed release planning



features that are expensive to implement but doesn't provide the highest values, risk.

Time-boxed release planning

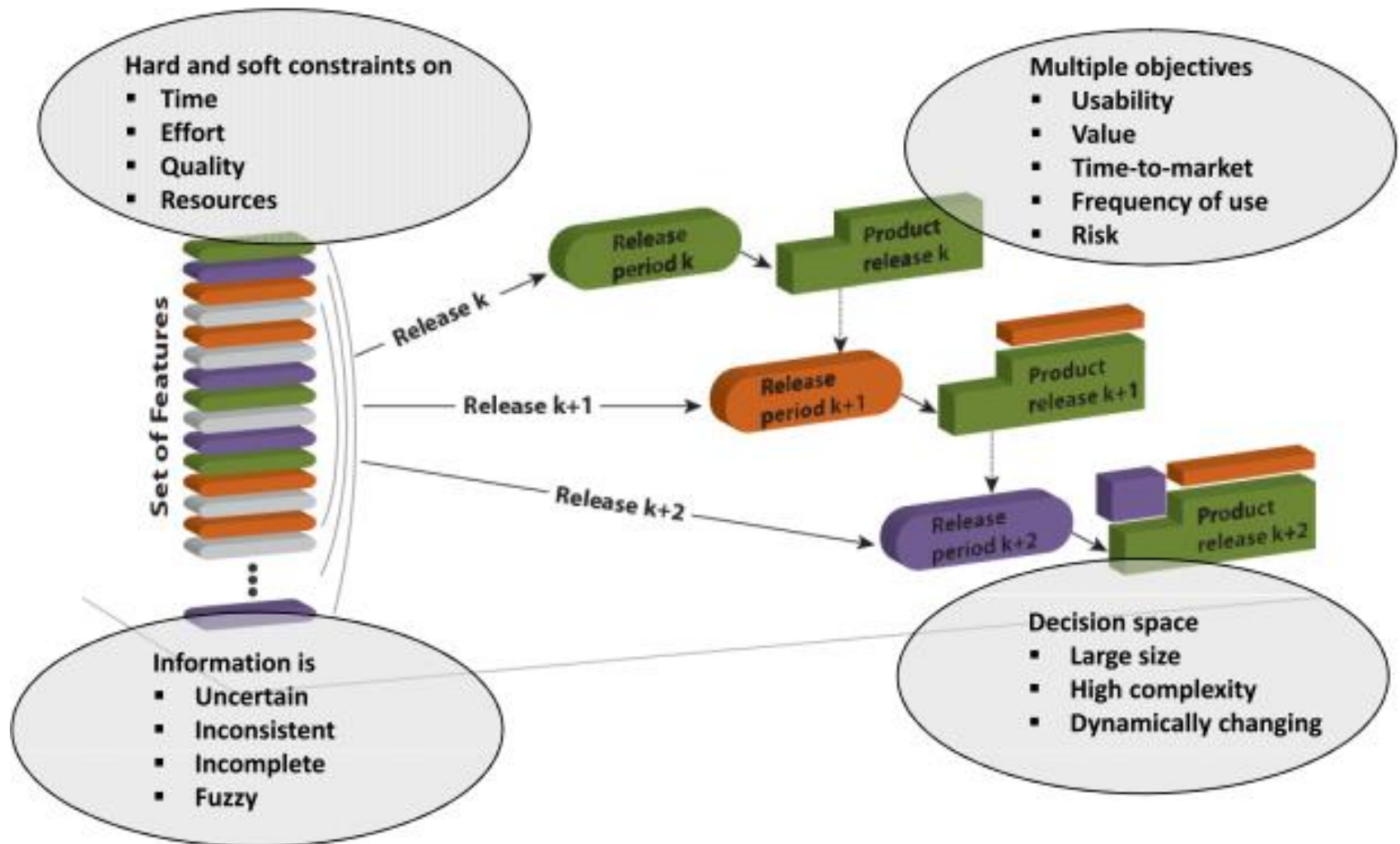
release can be thought of as outcome of a sprint



Could be optimization problem

- **Maximize** (objective function)
 - E.g., value
- **While balancing, e.g.,**
 - Individual stakeholder priorities
 - Urgency, importance
 - Implementation effort
- Under **constraints, e.g.,**
 - Risk, money, dependencies (technology, requirements)
 - Consumption of resources, either
 - Limited to exactly the release in which the feature is offered
 - To be distributed across different release periods

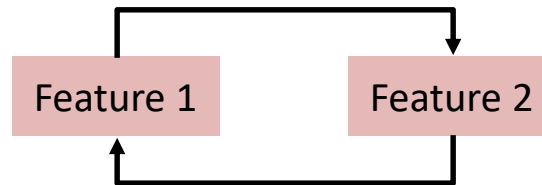
Efficient search for solution is needed



Dependencies (1)

- One **feature** requires the other to function
 - Consequence: Both in same release

eg. profile and login need to go into the same release as splitting them is not meaningful at all.



- One **feature** requires the other to function, **but not vice versa**
 - Consequence: One not in an earlier release than the other

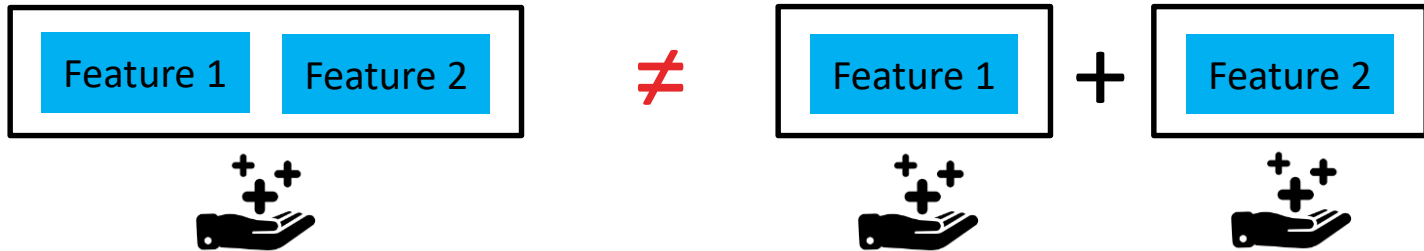
eg. profile and ordering products.



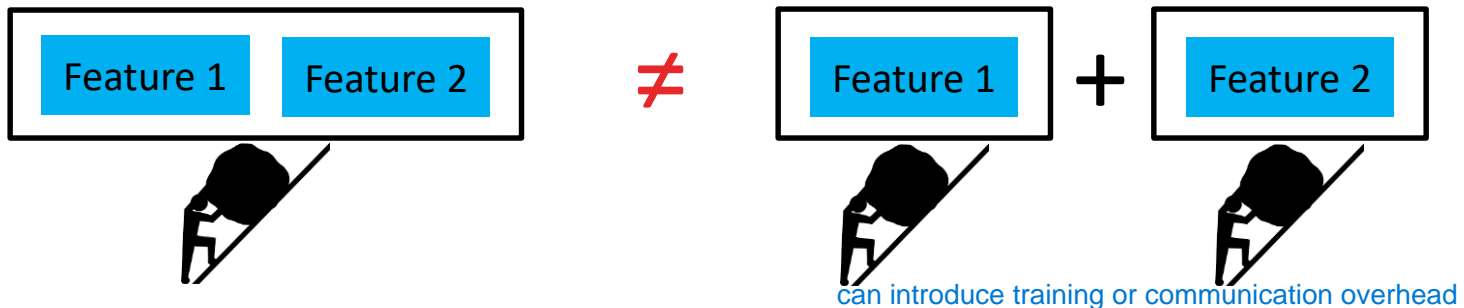
so this help create order, so order product can be in same or next release to profile

Dependencies (2)

- **Value** of features combined different to individual values



- **Effort (cost)** of features combined different from additive effort



Release planning – example

- Planning for the next two releases (or postpone)
 - Two stakeholders (total “vote” of 9 over three releases, e.g., 5, 4, 0)
 - Four resource types
 - Several coupling and precedence constraints $\{(7, 8), (9, 12), (13, 14)\} \in C$
 $\{(2, 1), (5, 6), (3, 11), (8, 9), (13, 15)\} \in P$

Features, resource consumption, and stakeholder feature evaluations								
Feature $r(i)$	Resources				Stakeholder $S(1)$		Stakeholder $S(2)$	
	Analyst & designers (hrs) $r(i,1)$	Developers (hrs) $r(i,2)$	QA (hrs) $r(i,3)$	Budget (US\$ in thousands) $r(i,4)$	Value value(1, i)	Urgency urgency(1, i)	Value value(2, i)	Urgency urgency(2, i)
1. Cost reduction of transceiver	150	120	20	1,000	6	(5, 4, 0)	2	(0, 3, 6)
2. Expand memory on BTS controller	75	10	8	200	7	(5, 0, 4)	5	(9, 0, 0)
3. FCC out-of-band emissions	400	100	20	200	9	(9, 0, 0)	3	(2, 7, 0)
4. Software quality initiative	450	100	40	0	5	(2, 7, 0)	7	(7, 2, 0)
5. USEast Inc., Feature 1	100	500	40	0	3	(7, 2, 0)	2	(9, 0, 0)
6. USEast Inc., Feature 2	200	400	25	25	9	(7, 2, 0)	3	(5, 4, 0)
7. China Feature 1	50	250	20	500	5	(9, 0, 0)	3	(2, 7, 0)
8. China Feature 2	60	120	19	200	7	(8, 1, 0)	1	(0, 0, 9)
9. 12-carrier BTS for China	280	150	40	1,500	6	(9, 0, 0)	5	(0, 8, 1)
10. Pole-mount packaging	200	300	40	500	2	(5, 4, 0)	1	(0, 0, 9)
11. Next-generation BTS	250	375	50	150	1	(8, 1, 0)	5	(0, 7, 2)
12. India BTS variant	100	300	25	50	3	(9, 0, 0)	7	(0, 6, 3)
13. Common feature 01	100	250	20	50	7	(9, 0, 0)	9	(9, 0, 0)
14. Common feature 02	0	100	15	0	8	(9, 0, 0)	3	(6, 3, 0)
15. Common feature 03	200	150	10	0	1	(0, 0, 9)	5	(3, 6, 0)
Total resource consumption	2,615	3,225	392	4,375				
Available capacity, Release 1	1,300	1,450	158	2,200				
Available capacity, Release 2	1,046	1,300	65	1,750				

Example – output

- Two valid release plan alternatives

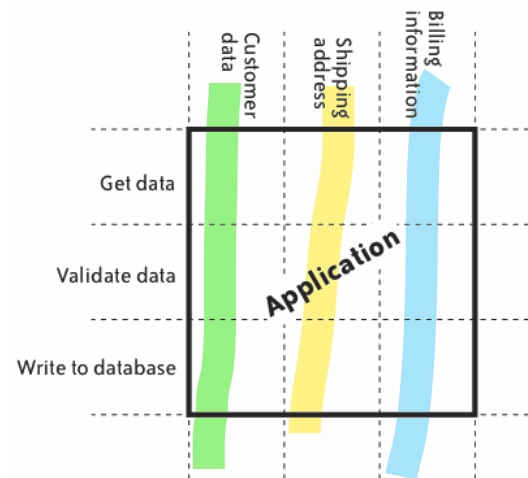
Two qualified release plan alternatives, listing the release to which each feature is assigned and each weighted average satisfaction

Feature $r(i)$	Release Plan x1		Release Plan x2	
	$x1(i)$	$WAS(i,k)$	$x2(i)$	$WAS(i,k)$
1. Cost reduction of transceiver	1	84.0	1	84.0
2. Expand memory on BTS controller	1	287.0	1	287.0
3. FCC out-of-band emissions	1	252.0	3	0.0
4. Software quality initiative	3	0.0	1	233.8
5. USEast, feature 1	1	134.4	3	0.0
6. USEast, feature 2	2	516.6	3	0.0
7. China feature 1	2	277.2	1	88.2
8. China feature 2	2	43.2	1	19.6
9. 12-carrier BTS for China	3	0.0	2	72.0
10. Pole-mount packaging	3	0.0	3	0.0
11. Next-generation BTS	3	0.0	3	0.0
12. India BTS variant	3	0.0	2	75.6
13. Common feature 01	1	37.8	1	516.6
14. Common feature 02	1	8.4	1	277.2
15. Common feature 03	2	54.0	2	54.0
Objective function value $F(x)$		1,694.6		1,708.0

Release often but include little?

notion

- MMF (Minimum Marketable Feature)
 - Provide competitive differentiation, revenue generation, cost savings
 - **Smallest item of functionality that provides value (see also Kano's model)**
- MUFS (Minimum Useful Feature Set)
 - MMF must be marketable, but whole release must be marketable as well
 - **Smallest amount of features that provide most new market value**



in left to right order to release

Example (1)

- New word processor
 - Market for word processors very mature
 - Might seem impossible to create a small first release
 - Much work to match competition, let alone to provide something new
 - Needs basic formatting, spellchecking, grammar checking, printing, etc.
 - Offering word processor may seem like a worthless effort
- Therefore, do not try to match competition
 - Focus on the features that make word processor unique
 - Release those features first – probably the features that have most value

Example (2)

- Competitive differentiation for word processor
 - E.g., powerful collaboration capabilities
 - E.g., web-based hosting
- First release
 - Basic formatting
 - Printing
 - Collaboration
 - Web-based hosting
- First release as technical preview to start generating buzz

Example (3)

- Later releases improve basic features, justify charging fee
 - E.g., tables
 - E.g., images
 - E.g., lists
- Next release
 - Grammar checking, etc.

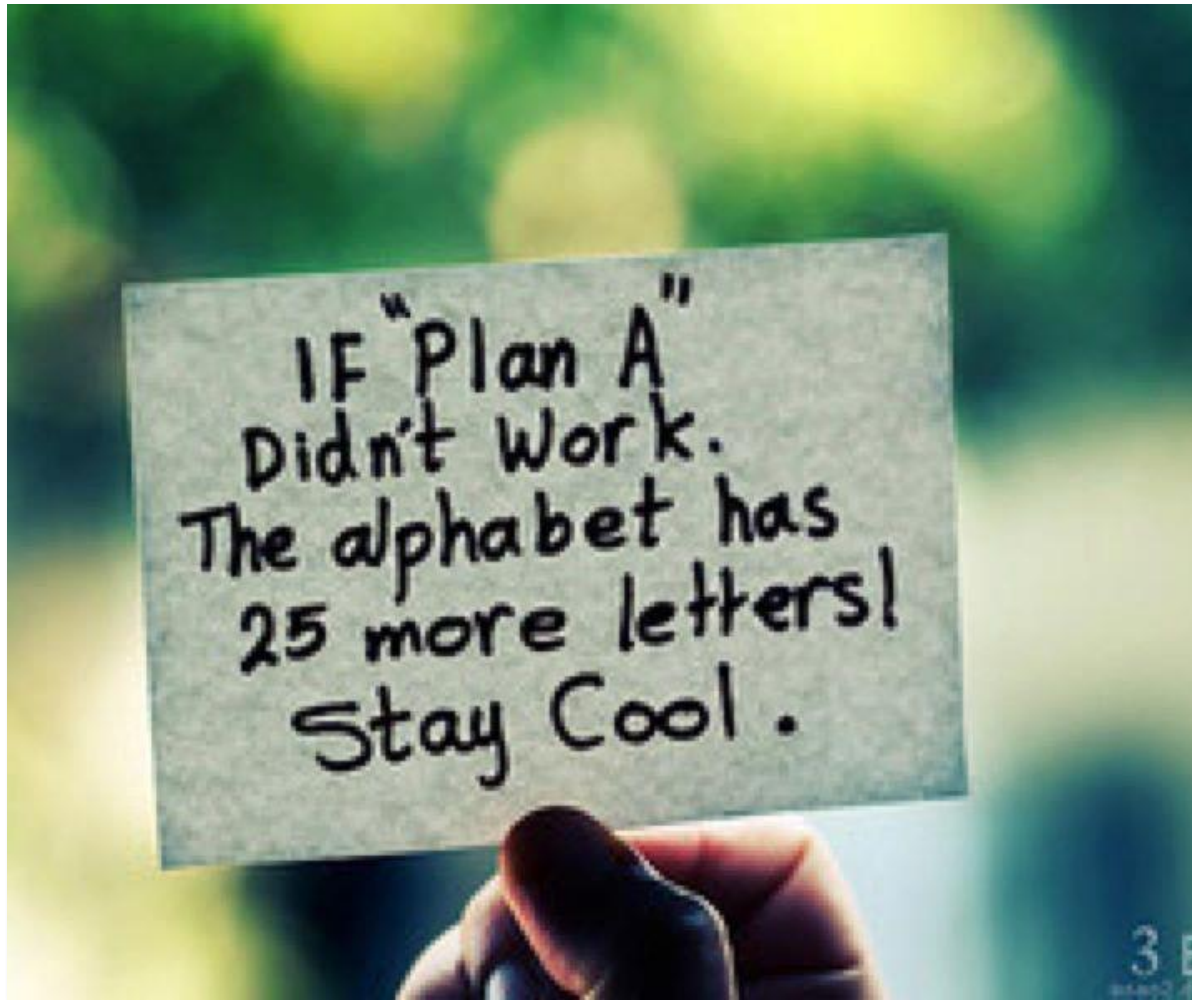
Writely

- Writely
 - Online word processing application
 - Does not have breadth of features of MS Word, Open Office, etc.
 - Focuses on what sets it apart
- First release two weeks after developers decided about Writely
- Ten months later, Google bought Writely
 - Even though Writely did not come close to other's feature sets
 - Writely became Google Docs (<http://www.writely.com/>)


Why would customers not like frequent releases?

- “Nobody likes changes”
 - **Therefore:** Install new versions automatically; hosted applications
- Rigorous testing and certification before installing new versions
 - **Therefore:** Automate
- Hidden costs of upgrading
 - Upgrades may require a difficult, costly installation process
 - **Therefore:** Allow release at any time, without user noticing
- Lack of commitment to involvement in requirements clarification
 - **Therefore:** Have dedicated role in development process (PO, etc.)

Release planning – summary



Summary

1. Requirements management 
2. Release management and planning 