

LECTURE 05 - CSCC01 SUMMER 2021
WEEK 5 - AGILE PLANNING. COHESION AND COUPLING
METRICS.

Ilir Dema

University of Toronto

Jun 8, 2021

SCRUM TEAM

- ScrumMaster:
 - Educates the team about scrum
 - Ensures practices are followed
 - Facilitates problem solving and runs interference for the team
- Product Owner:
 - Communicates the vision of the product
 - Maximizes the return on investment (ROI) by prioritizing desirable features
- Developers and Domain Experts
 - Everyone from every discipline needed to complete the sprint goals

SCRUM MASTER

- Not a traditional lead or management role
- Improves the use of Scrum through coaching, facilitation and rapid elimination of any distractions to the team
- Ensure principles behind scrum remain intact even as the company's own implementation evolves
- Sometimes scrum may be difficult to follow - the team may want to cut corners to deliver a sprint, the scrum master needs to remind them about what they must do
- Nurture sense of ownership

PRODUCT OWNER

RESPONSIBILITIES:

- Managing ROI (Return On Investment)
- Establishes a shared vision for the product among the customers / developers
- Knowing what to build and in what order
- Owns the product backlog
- Creating release plans and establishing delivery dates
- Supporting spring planning and reviews
- Representing the customers

THE OTHER MEMBERS OF THE TEAM

Members of the team are primarily developers and testers, however in many projects, specialists from other disciplines may also join the team.

DETERMINING SPRINT LENGTH

TYPICALLY TWO TO FOUR WEEKS

- Factors:
 - Frequency of customer feedback
 - How long can the stakeholders go without seeing progress / giving input
 - Team's level of experience
 - Sequential (new teams - using "small waterfall" approach) vs. Parallel (experienced teams)
 - Time overhead for planning and reviews
 - Review and planning require a good chunk of the day regardless of sprint length
 - Ability to plan the entire sprint
 - If there is uncertainty about how to achieve the sprint goal a shorter duration is advisable
 - Intensity of the team over the sprint
 - Avoid mini-crunch periods

AGILE PLANNING

- Traditional software projects place the bulk of project planning at the start of the project
- Agile spreads project planning out over the entire duration of the project
 - Agile teams may actually spend more time overall in planning, but it is not as concentrated

GOAL OF THE PRODUCT BACKLOG

- Prioritize stories so teams are always working on the most important features
- Supports continual planning as the game emerges so the plan matches reality (instead of a stale design document)
- Improves forecasts so stakeholders can make better decisions about the project

PRIORITIZING THE PRODUCT BACKLOG

- Prior to each sprint, the product owner is allowed to change the priorities in the product backlog
- Decisions can be based on the following criteria:
 - Value: what value does the story add to the player buying the game, helps maximize ROI
 - Cost: some features may prove too costly to implement and affect ROI
 - Risk: uncertainty about value/cost. In particular, COD.
 - Knowledge: if the product owner doesn't have enough information about feature to do a proper estimate they can introduce a spike to explore it and get more info

VELOCITY

- Velocity is measured by calculating the size of the user stories completed each sprint
- Changes in velocity are an effective way to see how changes / improvements to the agile process affect the team

WHEN DO WE CREATE THE ESTIMATES?

- During story workshops or sprint planning meetings
- Estimates should be a relatively quick process involving the following:
 - Expert opinion: domain experts can help inform the group about issues and effort required
 - Analogy: stories are compared to each other to help determine size. Triangulation can be used to help provide better results (compare it to a larger and a smaller story)
 - Disaggregation: larger stories may be difficult to estimate, so they can be broken down into smaller ones to help make the estimates more accurate
- Having to come up with a physical number usually removes any fuzziness about the requirements of the story

CONVERTING STORY POINTS TO HOURS

- Suppose for some reason you track how long every one-story-point story takes a given team to develop. If you did, you'd likely notice that while the elapsed time to complete each story varied, the amount of time spent on one-point stories takes on the shape of the familiar normal distribution.
- A tool used to convert story points to hours, is ideal days.
 - Another unit of measure - can be used as a transition for teams that are new to agile
 - Represents an ideal day of work - with no interruptions (phone calls, questions, broken builds, etc.)
 - Associated with something really so easier to grasp initially
 - Doesn't mean an actual day of work to finish, useful for relative measurements when compared to other stories
- That said, converting story points to hours, is something to be done carefully.

BREAKING DOWN USER STORIES INTO TASKS

- Tasks are estimated in hours
 - Estimation is an ideal time (without interruptions / problems)
 - Smaller tasks estimates are more accurate than large
- After all tasks have been estimated the hours are totaled up and compared against the remaining hours in the sprint backlog
 - If there is room, the PBI is added and the team commits to completing the PBI
 - If the new PBI overflows the spring backlog, the team does not commit
 - PBI can be returned to the product backlog and a smaller PBI chosen instead
 - Break original PBI into smaller chunks
 - Drop an item already in the backlog to make room
 - Product owner can help decide best course of action

EXERCISE - SPRINT PLANNING

Suppose you have the following user stories on the table:

story ID	estimated cost	value
1	1	2
2	2.5	4
3	1	3
4	1.2	4
5	3	1

From the previous iterations, you estimate the project velocity to be 5. Which user stories, and in which order, should be implemented in the next iteration (ignoring risks)? Why?

EXERCISE - BURNDOWN CHART

Suppose the costs of the user stories for a sprint that lasts 10 days are:

story ID	actual cost
1	2
2	1.5
3	2.5
4	1.7
5	2

Produce a burndown chart of the iteration. Assume each story was completed in two days in the same order as shown on the table.

EXERCISE - RELEASE BURNDOWN CHART

Initial velocity is 60, strategy=3 sprints avg.

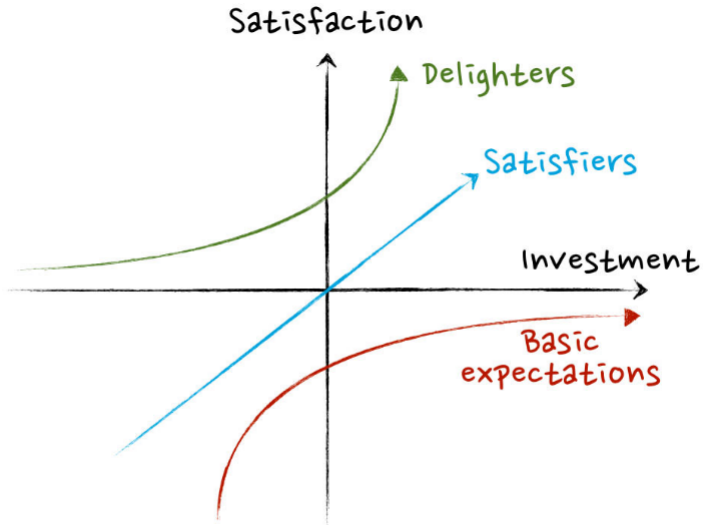


Does the team need to change the velocity in they do the planning for sprint 2? How about sprint 4?

HOW TO PRIORITIZE?

- Prior to each sprint, the product owner is allowed to change the priorities in the product backlog
- Decisions can be based on the following criteria:
 - Value: what value does the story add to the player buying the game, helps maximize ROI
 - Cost: some features may prove too costly to implement and affect ROI
 - Risk: uncertainty about value/cost
 - Knowledge: if the product owner doesn't have enough information about feature to do a proper estimate they can introduce a spike to explore it and get more info

KANO MODEL



KANO MODEL

- Developed by Noriaka Kano in the 1970s and 1980s while studying quality control and customer satisfaction.
- Basic features that a user expects to be there and work will never score highly on satisfaction, but can take inordinate amounts of effort to build and maintain.
- At the opposite end of the spectrum are features that delight the user. These score very highly on satisfaction and in many cases may not take as much investment. Small incremental improvements here have an outsized impact on customer satisfaction.
- Satisfiers: requirements that customers are not expecting, but the more of them that are included, the happier the customer.

PRIORITIZING DESIRABILITY

SELECTING FEATURES

- *If you have a choice of two things and cant decide, take both.*
 - Gregory Corso
- *When you come to a fork in the road, take it.*
 - Yogi Berra

ASSESSING THEMES ON THE KANO MODEL

- Kano proposed determining the category of a feature by asking two questions:
 - how the user would feel if the feature were present in the product and
 - how the user would feel if it were absent.
- The answers to these questions are on the same five-point scale:
 - 1 I like it that way
 - 2 I expect it to be that way
 - 3 I am neutral
 - 4 I can live with it that way
 - 5 I dislike it that way

EXAMPLE

THE SWIMSTATS WEBSITE

- SwimStats is a hypothetical website for swimmers and swim coaches.
- SwimStats will be sold as a service to competitive age-group, school, and college swim teams.
- **Coaches** will use it to keep track of their roster of swimmers, organize workouts, and prepare for meets
- **Swimmers** will use the site to see meet results, check their personal records, and track improvements over time.
- **Officials** at swim meets will enter results into the system.

THE SWIMSTATS WEBSITE

The screenshot shows a web browser window titled "BVSSL Swimmer Profile". The address bar shows the file path: `file:///Users/mikecohn/Documents/Webs/...`. The browser has a menu bar with "Cross-Platf...", "Conversion", "Apple", ".Mac", "Amazon", "eBay", and "Yahoo!". Below the menu bar is a navigation bar with "Boulder Valley Summer Swim League" (highlighted in yellow), "Search", "League", "Dual Meets", "Community", and "News".

The main content area shows the profile for **Willy VanDehy**, **Lafayette Seals**, **Gender: M**.

Below the profile is a section titled "Leagues (Email us.)" with a table:

League	Team(s)	Season start date	Events Swam	First	Second	Third
2003 BVSSL	Lafayette Seals	05/15/2003	18	12	4	2

Click on Team Name, Swimmer's Name or Meet Date to view more information. Meets are sorted with most recent meet on top.

A yellow box indicates Winner.

07/19/2003 Lafayette Seals at Rally Sport

Event Number	Age Group	Event	Time	Place	Points
23	Boys 9-10	50 Free	32.47	1	6
33	Boys 9-10	50 Fly	40.51	1	6
43	Boys 9-10	50 Back	39.02	1	6

07/12/2003 Lafayette Seals host Ranch Country Club

Event Number	Age Group	Event	Time	Place	Points
33	Boys 9-10	50 Fly	46.42	2	5
43	Boys 9-10	50 Back	46.15	1	6
73	Boys 9-10	100 IM	1:45.12	1	6

Source: Cohn, Agile Estimating and Planning.

KANO MODEL FOR SWIMSTATS

- Suppose we are contemplating three new features:
 - The ability to see a graph of a swimmer's times in an event over the past season
 - The ability for swimmers to post autobiographical profiles
 - The ability for any registered site member to upload photos
- To determine which type of feature this is, we would survey prospective users asking them:
 - If you can graph a swimmer's times in an event over the past season, how do you feel?
 - If you cannot graph a swimmer's times in an event over the past season, how do you feel?
 - If swimmers can post autobiographical profiles, how do you feel?
 - If swimmers cannot post autobiographical profiles, how do you feel?
 - If you can upload photos, how do you feel?
 - If you cannot upload photos, how do you feel?

SAMPLE ANSWERS

Functional form of question	→	If you can graph a swimmer's times in an event over the past season, how do you feel?	I like it that way	
			I expect it to be that way	X
			I am neutral	
			I can live with it that way	
			I dislike it that way	
Dysfunctional form of question	→	If you cannot graph a swimmer's times in an event over the past season, how do you feel?	I like it that way	
			I expect it to be that way	
			I am neutral	
			I can live with it that way	
			I dislike it that way	X

Source: Cohn, Agile Estimating and Planning.

MANAGING CONFLICTING ANSWERS

Customer Requirements		Dysfunctional Question				
		Like	Expect	Neutral	Live with	Dislike
Functional Question	Like	Q	E	E	E	L
	Expect	R	I	I	I	M
	Neutral	R	I	I	I	M
	Live with	R	I	I	I	M
	Dislike	R	R	R	R	Q

M Must-have

L Linear

E Exciter

R Reverse

Q Questionable

I Indifferent



Source: Cohn, Agile Estimating and Planning.

DISTRIBUTION OF RESULTS FROM SURVEYING USERS.

Theme	E	L	M	I	R	Q	Category
Graph event times	18.4	43.8	22.8	12.8	1.7	0.5	Linear
Can upload photos	8.3	30.9	54.3	4.2	1.4	0.9	Must-have
Post autobiographical profile	39.1	14.8	36.6	8.2	0.2	1.1	Exciter Must-have

Source: Cohn, Agile Estimating and Planning.

VALUE/COST: A PROXY FOR ROI

- A working definition for ROI is Value/Cost.
- Need to have a relative measure of value and cost, w.r.t other features.
- Why?
 - Easy to understand and calculate
 - It makes apparent economical sense
- We have a proxy for effort - story points.
- What is a proxy for value?

WHAT ELSE DO WE NEED TO ACCOUNT FOR?

- First, we need to quantify value, and numbers can be read from the Kano graph.
- Next, need to consider the cost of delay. How much relative value do we lose if we deliver x-feature after y-feature?
- Last, but not least, need to consider how much do we reduce risk and uncertainty.
 - If two user stories have same ROI, the one with highest CoD must be prioritized first.

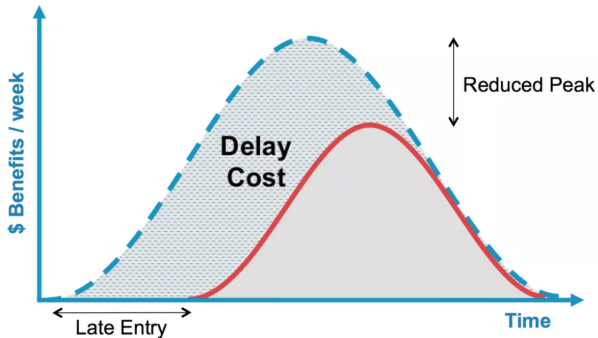
COST OF DELAY

DEFINE COST OF DELAY

- What is cost of delay?
 - The impact of time on value (or rephrased: Impact of time on the outcomes we hope to achieve)
- It combines urgency and value.
 - $\text{Cost of delay} = \text{value} \times \text{urgency}$
- Urgency: Describes the development of value over a given timeframe

URGENCY: SHORT BENEFIT HORIZON.

Short benefit horizon

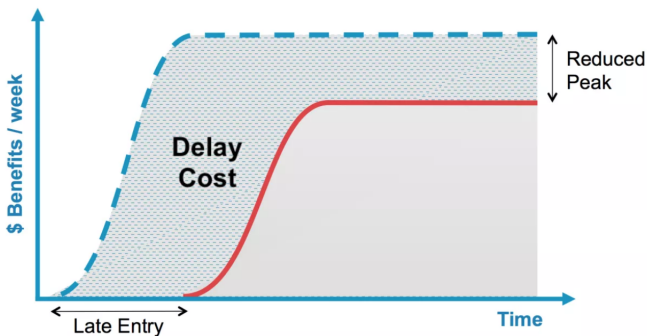


Short benefits horizon, and reduced peak due to late delivery

Source: <http://www.ontheagilepath.net/>

URGENCY: LONG LIFE BUT SMALLER PEAK DUE TO LATE ENTRY.

Long life but smaller peak due to late entry

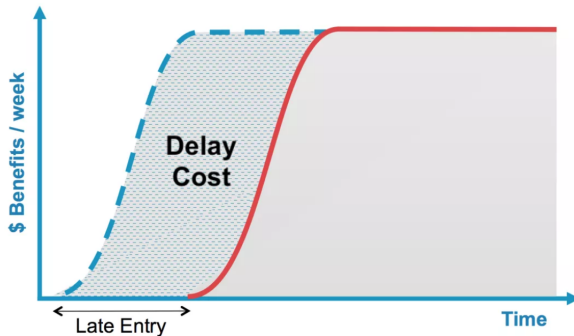


For ideas with a very long-life, with reduced peak due to later delivery

Source: <http://www.ontheagilepath.net/>

URGENCY: LONG LIFE , WITH PEAK NOT AFFECTED BY DELAY.

Long life , with peak not affected by delay

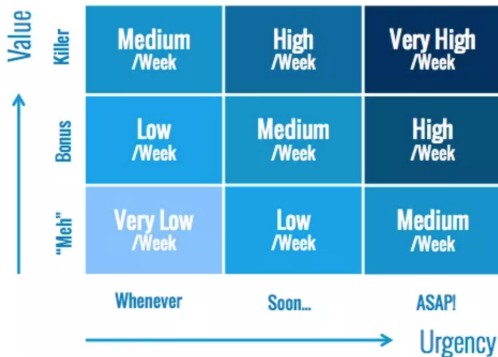


For ideas with a very long-life, with peak unaffected by delay

Source: <http://www.ontheagilepath.net/>

COMPUTING COST OF DELAY.

Qualitative Cost of Delay



Value ↑	Killer	Medium /Week	High /Week	Very High /Week
	Bonus	Low /Week	Medium /Week	High /Week
	"Meh"	Very Low /Week	Low /Week	Medium /Week
		Whenever	Soon...	ASAP!
		→ Urgency		



Source: <http://www.ontheagilepath.net/>

EXAMPLE



Feature	Effort	Value	ROI	CoD	Priority
1	1	3	3	3	1
2	3	6	2	4	2
3	10	20	2	3	3

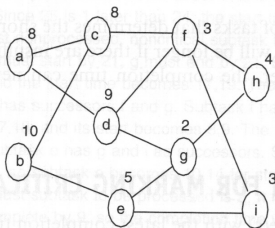
WHAT IS CRITICAL PATH?

- Critical path is the sequential activities from start to the end of a user story. Although many user stories have only one critical path, some may have more than one critical paths depending on the flow logic used in the user story implementation.
- If there is a delay in any of the activities under the critical path, there will be a delay of the user story delivery.

KEY STEPS IN CRITICAL PATH METHOD

- Step 1: Activity specification
 - Break down a user story in a list of activities.
- Step 2: Activity sequence establishment
 - Need to ask three questions for each task of your list.
 - Which tasks should take place before this task happens.
 - Which tasks should be completed at the same time as this task.
 - Which tasks should happen immediately after this task.
- Step 3: Network diagram
 - Once the activity sequence is correctly identified, the network diagram can be drawn.
- Step 4: Identify critical path
 - The critical path is the longest path of the network diagram. If an activity of this path is delayed, the user story will be delayed.

EXAMPLE



TaskID	Time	Dep.	CP
a	8		
b	10		*
c	8	a,b	*
d	9	a	
e	5	b	
f	3	c	*
g	2	d	
h	4	f,g	*
i	3	e,f	

SLACK TIME

Tasks on the critical path have to started as early as possible or else the whole project will be delayed. However tasks not on the critical path have some flexibility on when they are started. This flexibility is called the slack time.

TaskID	Start Time	Comp.Time	CP
a	0,1	8,9	*
b	0	10	
c	10	18	
d	8,9	17,18	*
e	10,14	15,19	
f	18	21	
g	17,19	19,21	*
h	21	25	
i	21,22	24,25	

WHAT IS PERT?

- PERT (Program Evaluation and Review Technique) is one of the successful and proven methods used in task scheduling.
- PERT was initially created by the US Navy in the late 1950s. The pilot project was for developing Ballistic Missiles and there have been thousands of contractors involved.
- After PERT methodology was employed for this project, it actually ended two years ahead of its initial schedule.

THE PERT BASICS

- At the core, PERT is all about management probabilities. Therefore, PERT involves in many simple statistical methods as well.
- Sometimes, people categorize and put PERT and CPM together. Although CPM (Critical Path Method) shares some characteristics with PERT, PERT has a different focus.
- Same as most of other estimation techniques, PERT also breaks down the tasks into detailed activities.

THE PERT BASICS

- In PERT, we assume that it is not possible to have precise time estimate for each activity and instead, probabilistic estimates of time alone are possible. A multiple time estimate approach is followed here. In probabilistic time estimate, the following 3 types of estimate are possible:
 - Optimistic time estimate t_o
 - Most likely time estimate t_m
 - Pessimistic time estimate t_p
 - $t_o < t_m < t_p$
- Time estimate is given by $t_e = \frac{t_o + 4t_m + t_p}{6}$
- The standard deviation for t_e is given by $\sigma = \frac{t_p - t_o}{6}$ and the variance is given by $\sigma^2 = (\frac{t_p - t_o}{6})^2$

EXERCISE - BEST TIME ESTIMATE

Two experts A and B examined an activity and arrived at the following time estimates.

Expert	Time Estimate		
	t_o	t_m	t_p
A	4	6	8
B	4	7	10



Determine which expert is more certain about his estimates of time:

WORKED EXAMPLE

Critical Activity	Optimistic time estimate (to)	Most likely time estimate (tm)	Pessimistic time estimate (tp)	Range (tp - to)	Standard deviation = $\sigma = \frac{t_p - t_o}{6}$	Variance $\sigma^2 = \left(\frac{t_p - t_o}{6} \right)^2$
A: 1 2	3	6	9	6	1	1
C: 2 3	6	12	18	12	2	4
F: 3 5	8	11	14	6	1	1
I: 5 8	2	4	6	4	2/3	4/9

Variance of project time (Also called Variance of project length) =
 Sum of the variances for the critical activities = $1 + 4 + 1 + 4/9 = 58/9$ Weeks.

Standard deviation of project time = $\sqrt{\text{Variance}} = \sqrt{58/9} = 2.54$ weeks.

MEASURING COHESION

PROGRAM SLICES

```
# x, y: positive integers
# z is the product of x and y
z = 0
while x > 0:
    z = z + y
    x = x - 1
return z
```

- *Output Slice*: statements that affect the value of the output (initialization does not count)
- *Input Slice*: statements that are affected by the value of the input
- Either can be used to define cohesion.
- We will use *output slices* following Bieman and Ott.
- The code above contains one output slice, computing z.

MEASURING COHESION

TOKENS

```
# x, y: positive integers
# q, x: quotient and remainder of
# x divided by y
q = 0
while x >= y:
    q = q + 1
    x = x - y
```

- How many output slices are in this piece of code?
- *Token*: A variable or a constant (x , y , q , 0 , 1)
- *Glue token*: token present in more than one slice (x , y)
- *Superglue token*: token present in all slices (none)



FUNCTIONAL COHESION MEASURES

- *Weak functional cohesion*: The ratio of glue tokens versus total tokens
- *Strong functional cohesion*: The ratio of superglue tokens versus total tokens
- *Adhesiveness of a token*: Percentage of output slices containing that token
- *Code adhesiveness*: The average adhesiveness of all tokens

MEASURING COHESION EXAMPLE

TOKENS

```
# x, y: positive integers
# q, x: quotient and remainder of
# x divided by y
q = 0 #initialization, does not count
while x >= y:
    q = q + 1
    x = x - y
```

- Output slices: 2
- Adhesiveness of tokens:
 - $A(x) = A(y) = A(q) = A(1) = 1/2 = 50\%$
 - $A = (4 \times 50\%) / 4 = 50\%$



MEASURING COUPLING

DEFINING COUPLING

- Coupling may be defined as the degree of interdependence that exists between software modules and how closely they are connected to each other.
- There are several dimensions of coupling:
 - Content coupling: this is a type of coupling in which a particular module can access or modify the content of any other module.
 - Common coupling: this is a type of coupling in which you have multiple modules having access to a shared global data
 - Control coupling: this is a type of coupling in which one module can change the flow of execution of another module
 - Data coupling: in this type of coupling, two modules interact by exchanging or passing data as a parameter

MEASURING COUPLING

MODULE COUPLING

- Data and control flow coupling
 - Let **di** (resp. **do**) be the number of data input (resp. output) parameters
 - Let **ci** (resp. **co**) be the number of control input (resp. output) parameters
- Global coupling
 - Let **gd** (resp. **gc**) be the number of data global (resp. control) variables
- Environmental coupling
 - Let **w** number of modules called (fan-out)
 - Let **r** number of modules calling the module under consideration (fan-in)

$$\text{Coupling}(C) = 1 - \frac{1}{di+2ci+do+2co+gd+2gc+w+r}$$