Estimations

# Practical task

1. Discover 3 estimation technique by yourself apart of techniques mentioned in lection. Compare all of them with each other and highlight PROs and CONs of different methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Estimation name | Description | Pros | Cons | Distinctive characteristics |
| By development | Method depends on count of testers and developers.  Formulas:  Testing working days = (Dev working days)/3  Testing engineers = (Dev engineers)/2 | -Simple  -Quick  -Minimum information required  -Minimum thinking required | This method doesn’t consider that:  -Not linear dependences between testers and developers  -Regression  -Different testing types (e.g. cannot compare GUI testing and DB testing)  -Count of bugs  -Tester’s productivity  -Developer’s productivity | -Suits for project that has known count of testers and developers  -Suits for project without detailed requirements/no info about functionality  -Suits if we have the same experience on last project  -Suits for long-term planning  -Suits for project where developer’s time is sold  -Overall, this method cannot be main estimation method on project, can be like additional |
| By UCP | Short description: 1)Identify, classify and weight “actors”  2) Identify, classify and weight use cases  3)Identify and Weight Technical Factors  4)Identify and Weight Environmental Factors  5)Calculate Adjusted Use Case Points  6)Converting Points into Time  Please see detailed step’s description below | -Widely recognized  -Suits for testing and development estimations  -Doesn’t require detailed specification  -Works for new projects  -This UCP Method gives us concept about testing time for the project you estimate | -Will not work for maintenace project (only for new projects)  -Little real experience in EPAM  -Different formula versions exist  -No information about formula’s coefficient  -Hard to argue the UCP mark to Customer  -Hard to predict how complex some parts of functionality would be  -Hard to predict how many roles and how complex they would be  -We couldn't know real technical and human factors for the future project | -The main method’s problem is related to correct determination of values for various factors. For right determination we have to have good experience  -3 variants of how to convert UCP into time (Karner approach, EPAM approach, based on the realities of your project, when one UCP will be measured not in hours, but in minutes) |
| Dot-voting | 1)All user stories are put on a wall, virtual or real by the Product Owner  2) Every team member is given 4-5 votes; these can be small round sticky notes  3) Every team member is asked to give their votes on the stories they think are bigger  4) Team member puts / pastes the round red sticky notes on the stories they think are big in size  5) Every team member performs this process until their all 4-5 votes are exhausted / used. At the end of this process, the story with higher votes is termed as biggest and that with low number of votes are smallest  6) Product owner then orders the story from higher votes to lower votes  7) One person can vote more than once for one story | -Simple  -Quick  -Look like Planning Poker | -Requires all team members participation to come up with consensus-based  -Doesn’t provide an answer “how much time this story would take?” | -Suits for small count of User Stories (8-10)  -Same technique is used to decide priority as well. More votes means higher priority item  -Alternative to Planning Poker |

**Detailed UCP steps description:**  
**1) *Determine the number of “actors” in the System***

**UAW =Unadjusted Actor Weights**

“Actors” is a different inside users/factors, that can interact with system.

**End users** are simple “actors”.

**Average actors** interact with the system through some protocols etc. or they could be Data stores.

**Complex users** are separate systems that interact with the SUT through an API.

Each actor has individual weight, use the next table for actors’ weight determination:  
Text

Description automatically generated with medium confidence

After determination actors’ weight, sum actors’ weight.

Total actor weight = **UAW**

**2) *Identify, classify and weight Use Cases***

Determine Use Cases, if you don’t have any Use Cases, you have to imagine Use Cases.

Use Cases is a description how smt/smb interact with system.

Each Use Cases has individual weight, use the next table for Use Cases’ weight determination:  
Graphical user interface, text, application, email

Description automatically generated

After determination Use Cases’ weight, sum Use Cases’ weight.  
Total Use Case weight = **UCP**

**UUCP=Unadjusted Use Case Points**

**UUCP=UAW+UCP**

**3) *Identify and Weight Technical Factors***

At this step we estimate technical factor. Also, each of technical factor has individual weight (value from 0 to 5). For the higher (5) value, the more time will be spent on testing.

Each Technical Factor has individual weight, use the next table for Technical Factor’s weight determination:

Table

Description automatically generated

After determination Technical Factor’s weight, sum Technical Factor’s weight.

**TFactor = Sum of Weight \* Value column**

**Technical Complexity Factor = TCF**

**TCF = 0.6 + (0.01 \* TFactor)**

*4) Identify and Weight Environmental Factors*

Each Environmental Factor has individual weight, use the example of table for Environmental Factors weight determination:  
Table

Description automatically generated

After determination Environmental Factor’s weight, sum Environmental Factor’s weight.

**Note:** The better we have something, the higher we will set the value, which means that at the output we will have less time to add for testing.

**EF-Factor = Sum of (Weight \* Value) column**

**Environmental Complexity Factor = ECF**

**ECF = 1.4 + (-0.03 \* EF-Factor)**

***5) Calculate Adjusted Use Case Points***

**UCP = UUCP \* TCF \* ECF**

***6) Converting Points into Time***

Karner:

* 1 UCP= 20-28 hours
* 1 week=35 hours
* Add 25% for risks

N. Miranovich (Epam reality):

* 1 UCP=20 hours
* 1 week= 40 hours

1. Estimate your tasks using all relevant to your project methods from lecture as well as your discovery result and compare the accuracy in the end of iteration.

Example:

User Story: Customer Authentication - Biometric login permissions - First login

**As a**: Trade Customer

**I want**: to be able to login to the Howdens Mobile App using biometric data (on iOS / Android devices)

**So that**: I can further use the Mobile App

**Important Notes:**

Ability for customers to login with biometrics

Fully test and sign off that this functionality works as expected

*Original testing estimate = 18h  
Original dev estimate = 18h*

1. By Development estimation

Testing working days = (Dev working days)/3 = 18h/3 = 6h ≈ 1d

Testing engineers = (Dev engineers)/2 = 1/2= 0.5 testing engineers ≈ 1 testing engineer

1. By UCP estimation

Determine actors:

|  |  |  |
| --- | --- | --- |
| Actors | Type definition | Factor |
| End users | GUI | 1 |
| Native biometric phone features | Phone low-level interactions | 3 |

UAW = 1\*1+1\*3 = 1 + 3 = 4

Determine Use Cases:

|  |  |  |
| --- | --- | --- |
| Use Case type | Definition | Factor |
| Log in without enabled biometric | 1 transaction | 1 |
| Log in with enabled biometric> Pass native biometric checks | 2 transactions | 1 |
| Log in with enabled biometric> Disable in app biometric | 2 transactions | 1 |
| Log in with enabled biometric> Disable on device biometric | 2 transactions | 1 |

UCP = 1\*1 + (2\*1) \* 3 = 1 + 6 = 7

UUCP = UAW + UCP = 4 + 7 = 11

Determine Technical factors:

|  |  |  |  |
| --- | --- | --- | --- |
| Technical Factor  Description | Weight | Value | Weight \* Value |
| Test Tools | 5 | 2 | 10 |
| Documented inputs | 5 | 3 | 15 |
| Development Environment | 2 | 3 | 6 |
| Test Environment | 3 | 1 | 3 |

TFactor = Sum of Weight \* Value column = 10 + 15 + 6 + 3 = 34

TCF = 0.6 + (0.01 \* TFactor) = 0.6 + (0.01 \* 34) = 0.94

Determine Environmental Factors:

|  |  |  |  |
| --- | --- | --- | --- |
| Environmental Factor  Description | Weight | Value | Weight \* Value |
| Familiar with RUP | 2 | 5 | 10 |
| Application experience | 3 | 5 | 15 |
| Object-oriented experience | 1 | 5 | 5 |
| Lead analyst capability | 0.5 | 5 | 2.5 |
| Motivation | 1 | 5 | 5 |
| Stable requirements | 2 | 4 | 8 |

EF-Factor = Sum of (Weight \* Value) column = 45,5

ECF = 1.4 + (-0.03 \* EF-Factor) = 1.4 + (-0.03 \* 45.5) = 0.035

UCP = UUCP \* TCF \* ECF = 11 \* 0.94 \* 0.035 = 0,362

Karner converting: 0,362 \* 28 = 10h

25% on risks

**Estimation by Karner = 13h**

EPAM converting: 0.362 \* 20 = 7,24h

**EPAM estimation = 7,5h**

1. By Test Cases

Areas:

|  |  |
| --- | --- |
|  | Count of tests |
| Face ID | 4 |
| Touch ID | 3 |
| Android biometric | 3 |
| Biometric is not enabled | 1 |
| Total | 11 |

Test Cases and Passes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | TC to create | TC to pass | Number of passes | Total to pass |
| Biometric | 5 | 13 | 2 | 39 |

Overall, I can create TC in 12 minutes

Biometric functionality is too complex, so 1 TC for biometric in 30 minutes

Test Case Creation time = 30 \* 5 = 150 minutes on TC creation

35% on risks, 150min + 35% = 203min = 3,5h on TC creation

Possible risks:

-Troubleshooting

-Emails

-Meetings

-Risks ( illness, build failed, some tests not counted, development delays, etc)

Overall, I can pass complex TC in 12-15 minutes

Test Case Passing time = 15 \* 39 = 585 minutes on TC passing

35% on risks, 585min + 35% = 790min = 13h on TC passing

Estimation by Test Cases = 3,5h on TC creation + 13h on TC passing = 16,6h on testing

1. By 3 point estimation

E = (O + 4M + P) / 6

E: Estimate (mean value)

O: An optimistic, best case scenario

P: A pessimistic worst-case scenario

M: Most likely scenario

O – take from by development estimation = 6 h

M – take from by UCP estimation = 13 h

P – take from by Test Cases estimation + 25 % (risks see below in 3 point)

E = (6h + 4\* 13h + 21,5h)/6 = 14h

**Conclusion:** the most real estimations was performed by UCP (Karner) and by Test Cases. The most suitable estimation, to my mind, is estimation by Test Cases because can take less time than by UCP, simple than UCP, no need any complex formulars and estimation by Test Cases we can explain Customers. Also, would like to notice that 3 point estimation close to real estimation, but make to think about all risks that can be on project.

1. Note tasks, risks and additional activities that were not included in estimations

Not included risks:  
- 1h on Test Case review  
- bug verification and investigation  
- any reports  
- explain functionality new-comers  
- extra sick leaves

- environment is available always when needed

- development\testing activities going exactly as planned

- investigation results are accurate

- developed architectural/technical solutions do not evolve

- all related tasks are completed just as planed, no delays, no blockers (if any)

- stakeholders communication is prompt and efficient

- scope does not evolve

- no illnesses or unexpected vacations

1. For Scrum teams  
   If you are not practicing Planning Poker estimation, propose your team mates to conduct several session during 2-3 sprints and see if it works for you  
     
   If you already practicing Planning Poker estimation, analyze current process and try to introduce valuable changes.

Only ones have this experience. All was good except 1 thing: too much time for Planning Poker due to team don’t use useful tools for PP.  
PP tools: PlanITPoker – desktop app, ShiwaForce – Jira plugin, Scrummer (mobile app)

1. Based on work done and data received make conclusion about all techniques used and choose the most effective one for you daily activities.

All technics have pros and cons and all technics are suitable, but for different type of project. To my mind the simplest technic is by development but this estimation type like an additional type of estimation because of this type doesn’t notice risk , and the mark by this estimation is superficial estimation for task.

The most complex estimation is by UCP. Has lot of disadvantages, not applied on projects, has complicated formulars with weird coefficients and if you would like to give estimation with this technic, book big time stop for this activity.

To my mind the most suitable estimation to me and current project is by 3 point and by Test Cases estimations. Both types simple, clear and understandable. Notice big part of risks and with this types easy to describe what and we estimate this or than task/User Story.