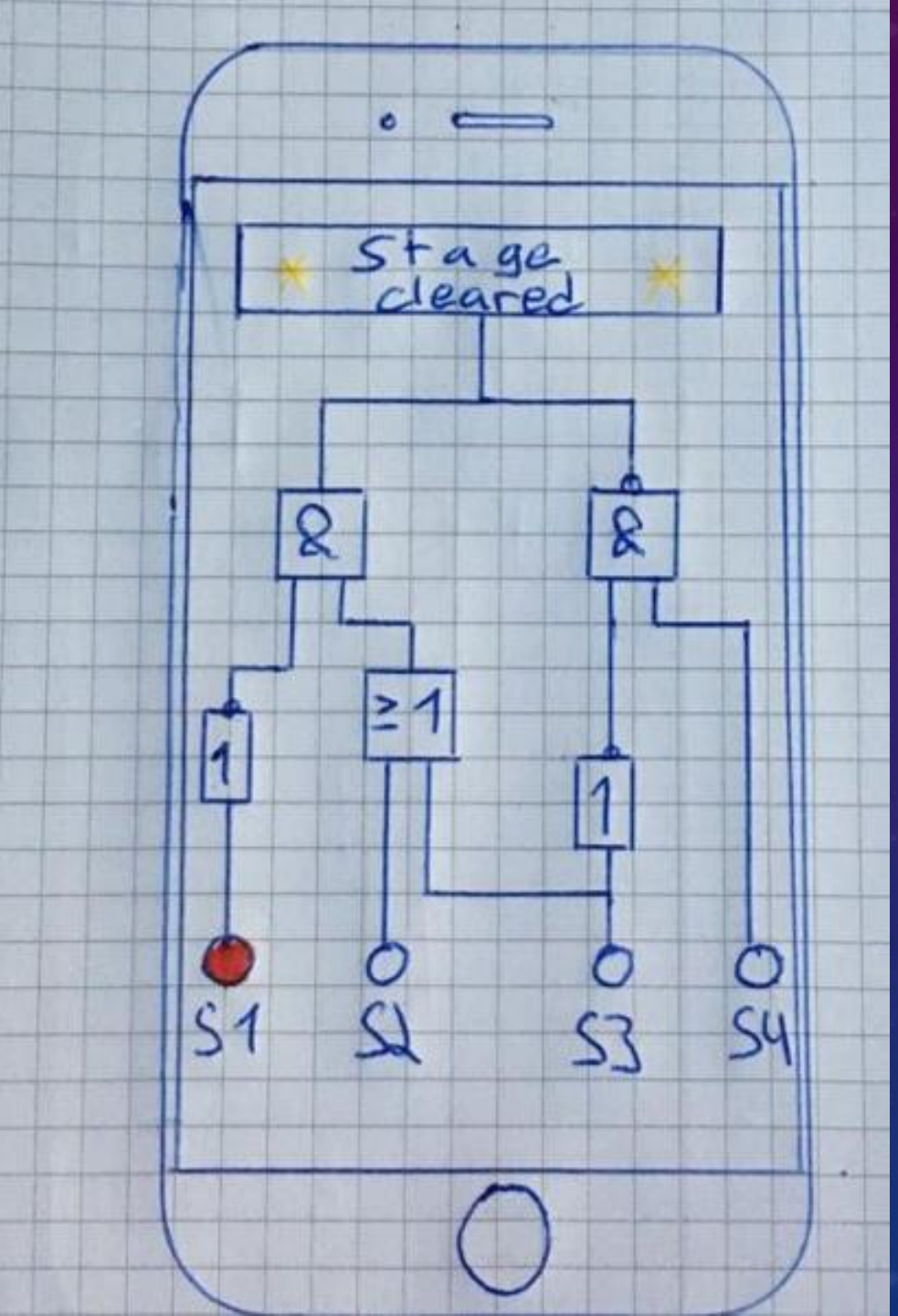
LOGICSGAME

BY TIM FARAHANI, MAX MAIER & MARCEL SOMMER

AGENDA

- BUSINESS NEEDS
  - VISION
  - USE CASE(S)
  - SOFTWARE REQUIREMENT SPECIFICATION
  - SCOPE
  - BLOG
  - PROJECT MANAGEMENT
- TECHNICAL ABILITIES
  - DEMO
  - CLASS DIAGRAM
- QUALITY
  - ARCHITECTURE
  - CONFIGURATION
  - RISK MANAGEMENT
  - TESTING
  - PATTERNS
  - METRICS

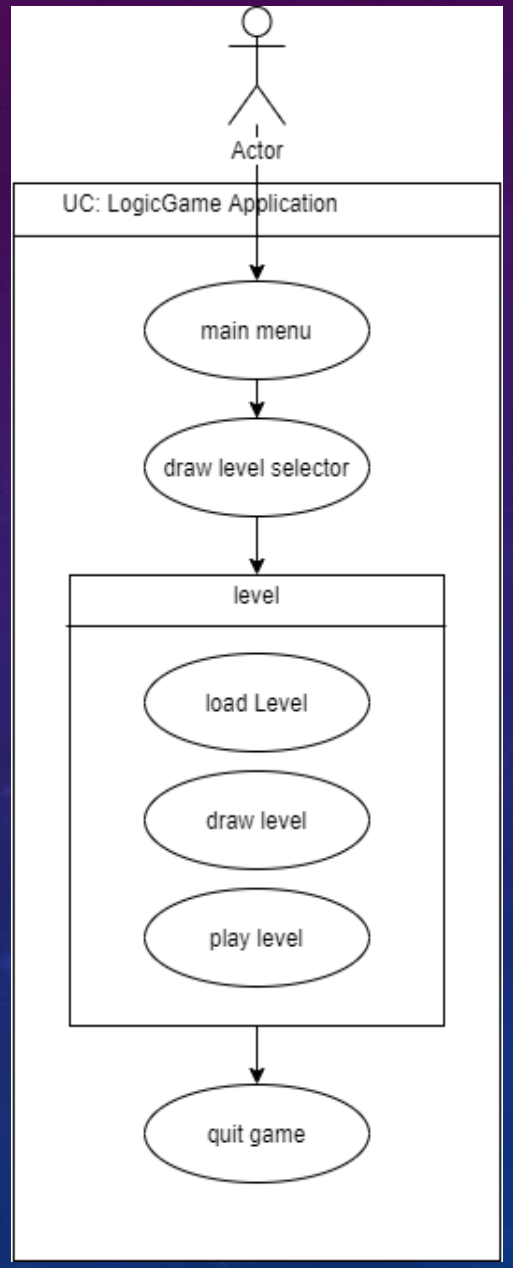
BUSINESS NEEDS



VISION

- Smartphone game
- Educational gaming
- Easy to understand
- Help players to understand logical gates
- Deploy in schools
- Lure pupils into technology

OVERALL USE CASE



USE CASES

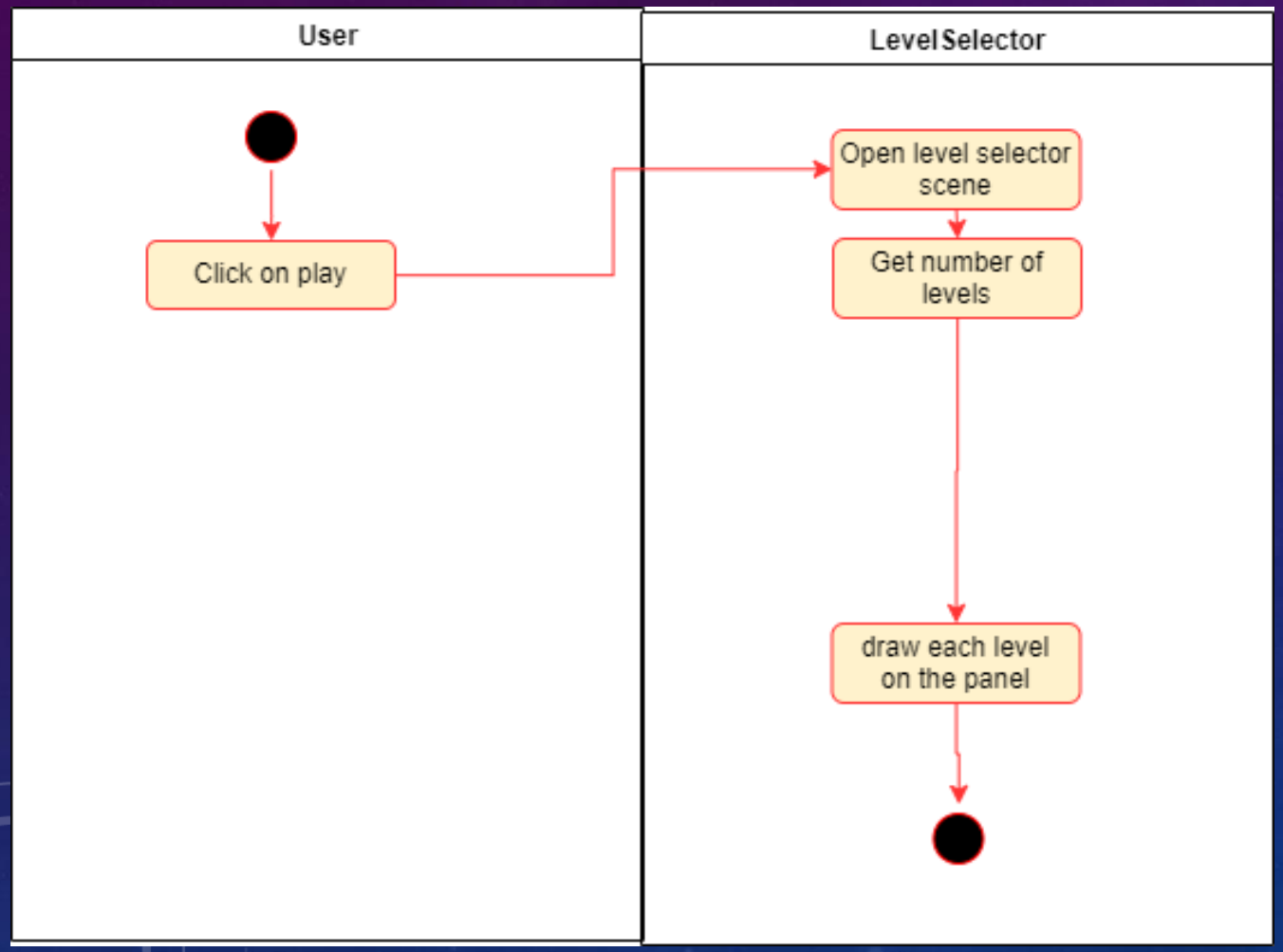
6 Use Cases:

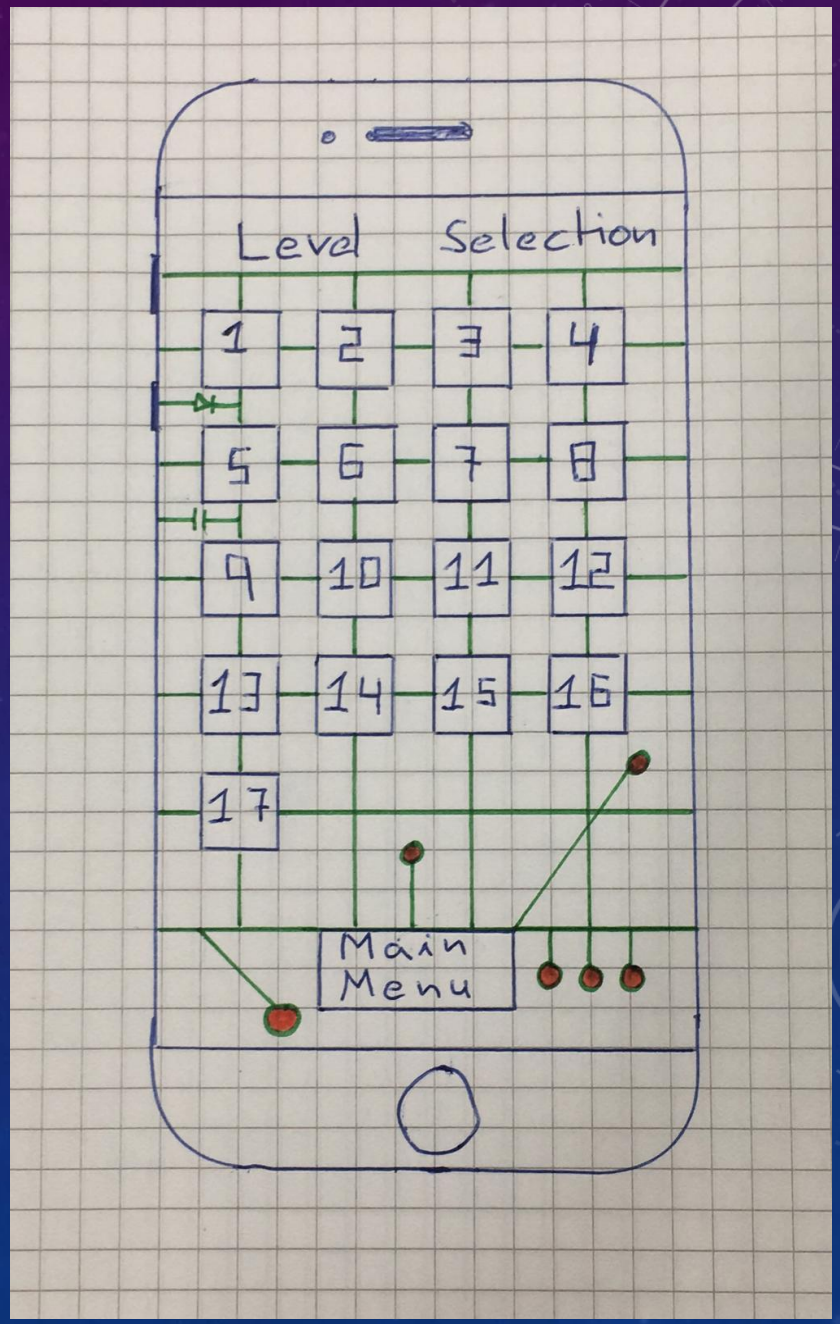
- Main menu
- Draw level selector
- Load level
- Draw level
- Play level
- Quit game

Use case specification document for each one of them:

- Description
- Activity Diagram
- Mock up
- Function Point calculation

USE CASE EXAMPLE





SOFTWARE REQUIREMENT SPECIFICATION (SRS)

- Document describes software requirements such as:
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
  - Design Constraints
  - On-line User Documentation and Help System Requirements
  - Purchased Components
  - Interfaces
  - Licensing Requirements
  - Legal, Copyright, and Other Notices
  - Applicable Standards

SCOPE

KISS – keep it simple (and) stupid

game will be an android app only

only newer android phones will be supported

user interface where the user can choose the level that he wants to play

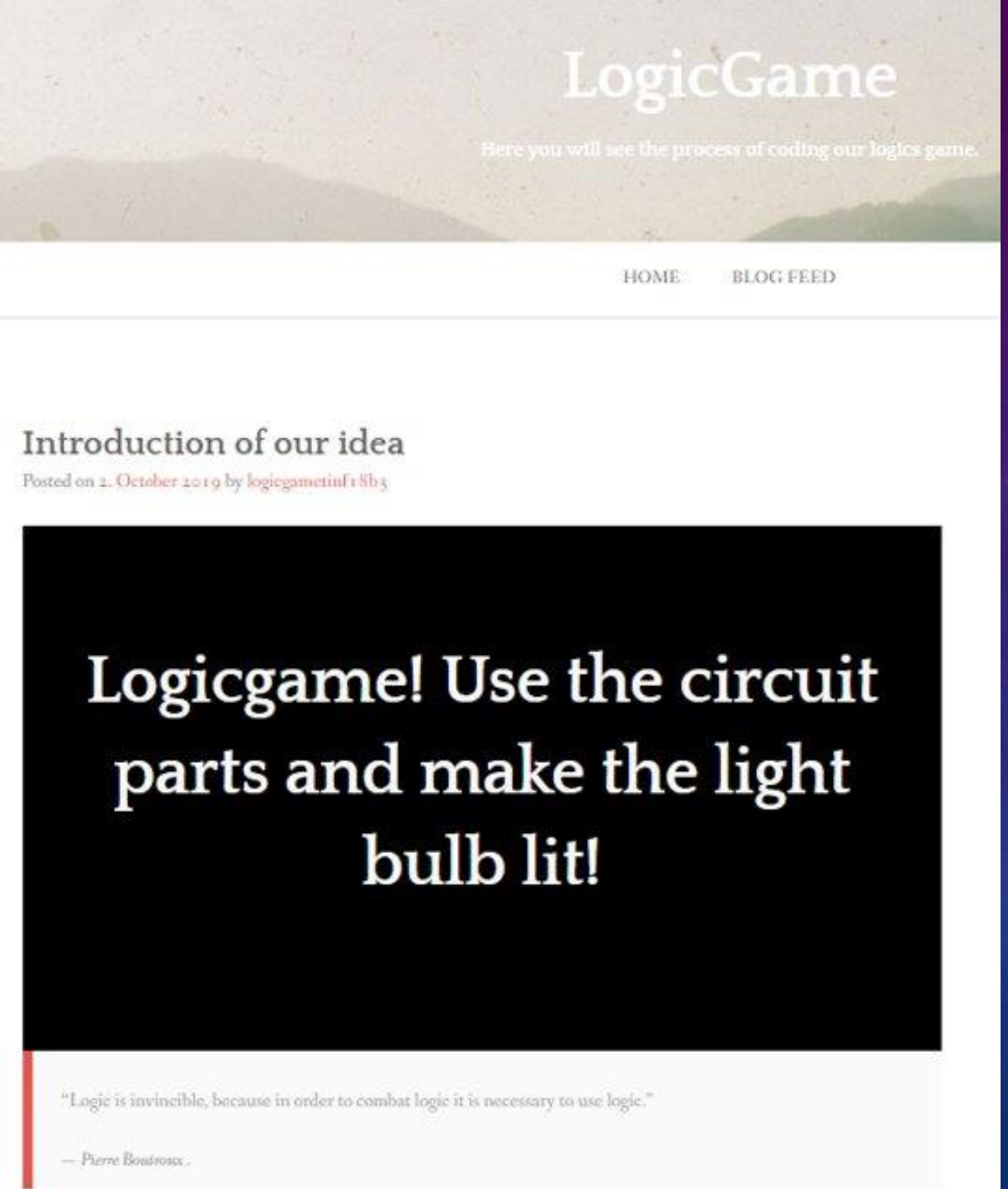
interface where the user can play the game

level is solved when the logic gates are correctly connected and the light bulb is lit

overall goal is to have a playable game, where more features and levels can be easily added later, after the project is done.

Out of scope: deploy game in Playstore and eventually in schools

BLOG


LogicGame
Here you will see the process of coding our logics game.
HOME
BLOG FEED
Introduction of our idea
Posted on 2. October 2019 by logicgametinf18b3
Logicgame! Use the circuit parts and make the light bulb lit!
"Logic is invincible, because in order to combat logic it is necessary to use logic."
— Pierre Boutroux .

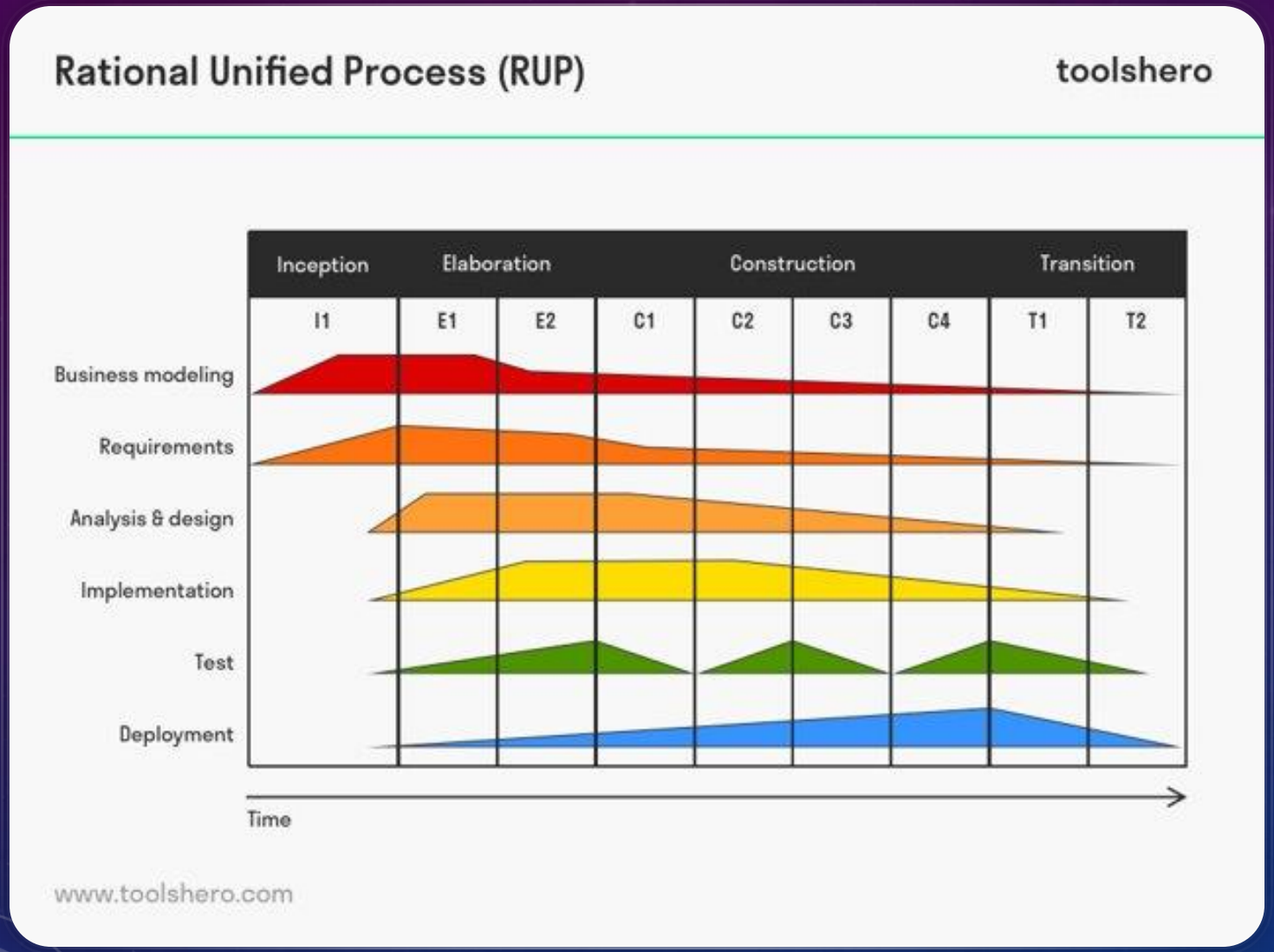
PROJECT MANAGEMENT WITH YOUTRACK

- Free for students of DHBW
- Issue list
- Tracks Time spent
- Integration with github
- Different kind of reports
- Coordinate tasks
- Allows sharing content
- Easy to use
- Recommended

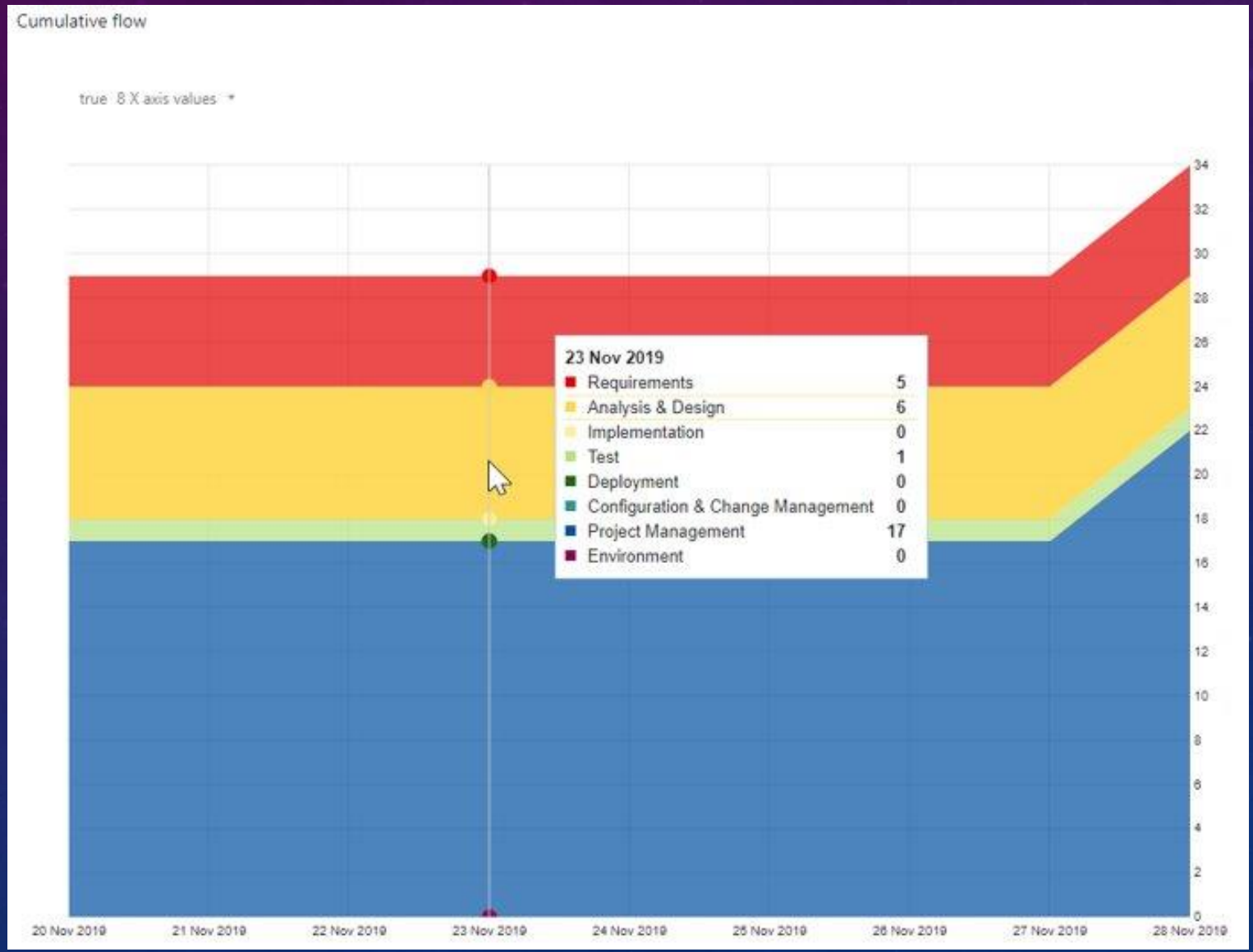
PROJECT METHODOLOGY: SCRUM

- Agile project management
- Iterative process
- Way more flexible than waterfall
- Adjustable to costumer’s specification changes
- Weekly Sprints --> check and communicate with team members
- Agile board as an overview
- Showing partial result to inform costumer about the progress

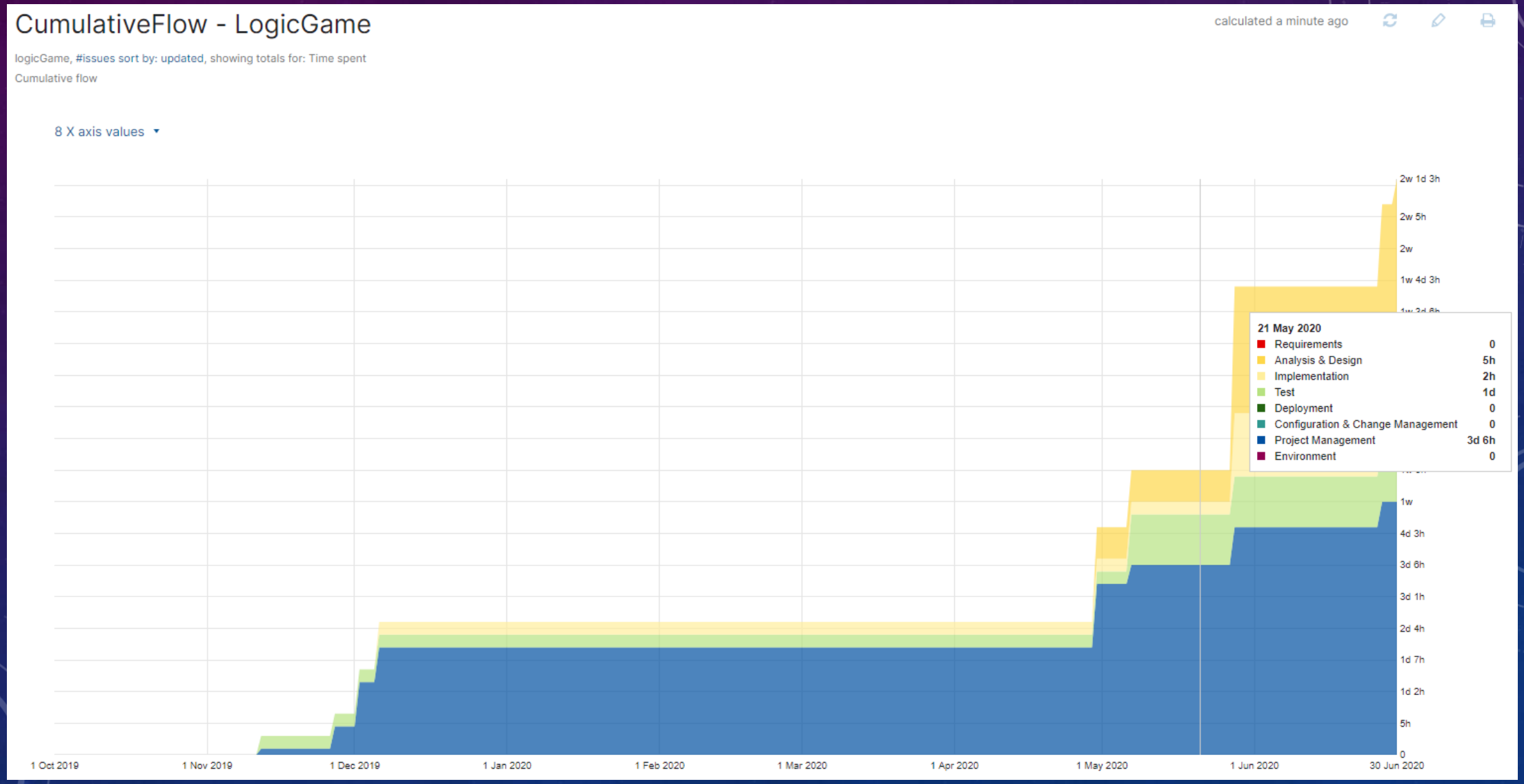
RUP


Rational Unified Process (RUP)
toolshero
Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business modeling
Requirements
Analysis & design
Implementation
Test
Deployment
Time
www.toolshero.com

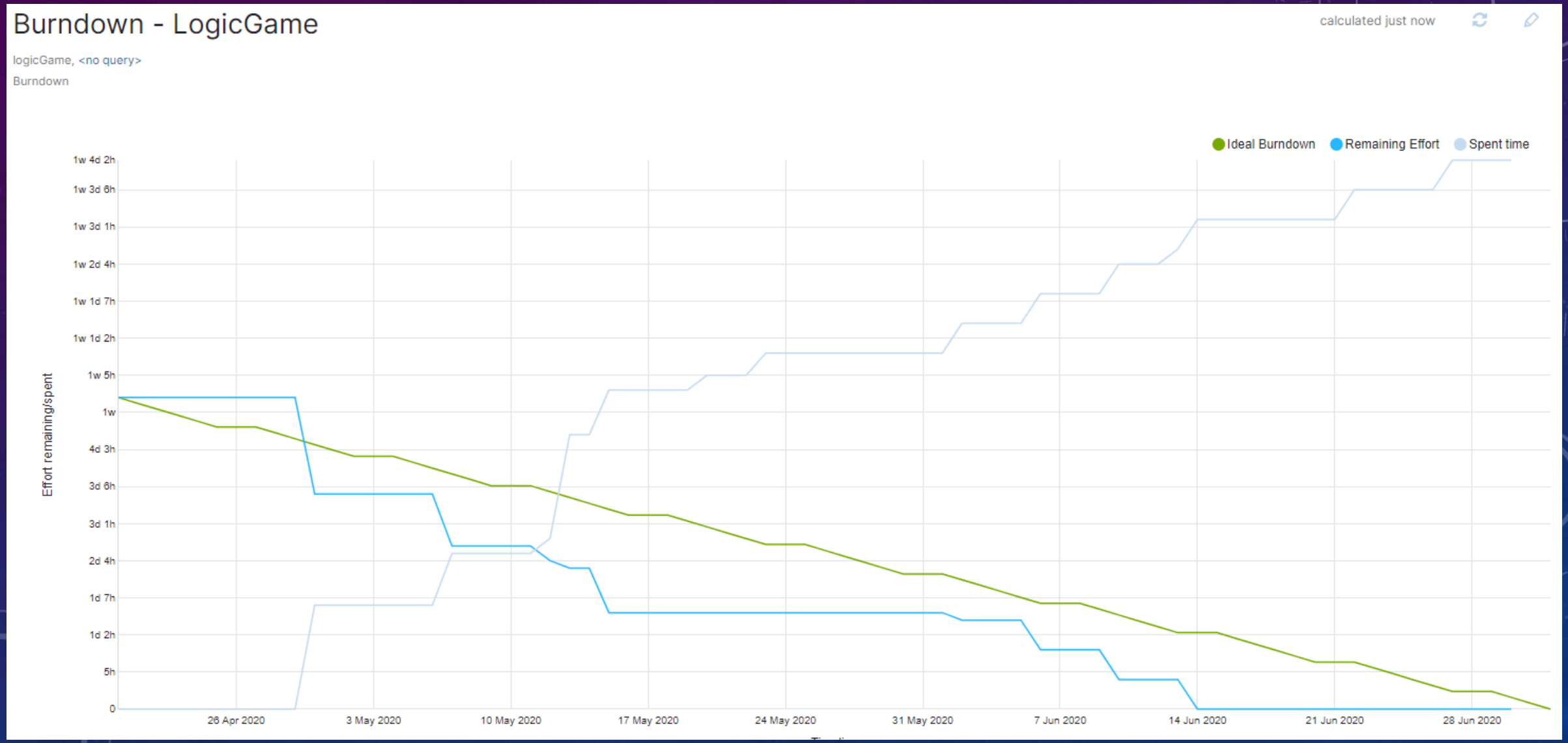
CUMULATIVE FLOW PROJECT BEGIN



COMULATIVE FLOW PROJECT PROGRESS

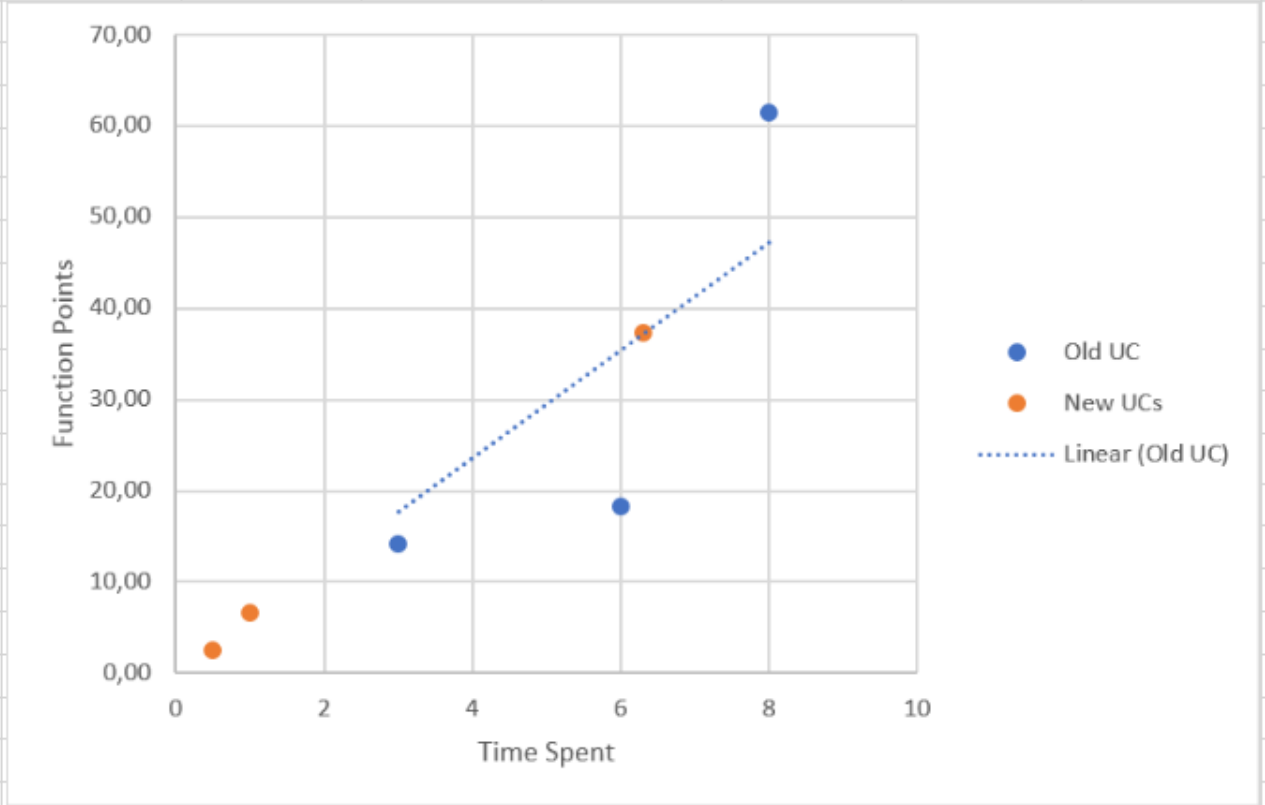


BURNDOWN


Burndown - LogicGame
calculated just now
logicGame, <no query>
Burndown
Ideal Burndown
Remaining Effort
Spent time
Effort remaining/spent
1w 4d 2h
1w 3d 6h
1w 3d 1h
1w 2d 4h
1w 1d 7h
1w 1d 2h
1w 5h
1w
4d 3h
3d 6h
3d 1h
2d 4h
1d 7h
1d 2h
5h
0
26 Apr 2020
3 May 2020
10 May 2020
17 May 2020
24 May 2020
31 May 2020
7 Jun 2020
14 Jun 2020
21 Jun 2020
28 Jun 2020

COST ESTIMATION (TIME IS MONEY)

	A	B	C
1	UseCases	FunctionPoints	TimeSpent
2			
3			
4			
5			
6	Load Level	14,11	3
7	Draw level	18,26	6
8	Draw Level Selector	61,42	8
9			
10			
11			
12	Main Menu	6,64	1
13	Exit Game	2,49	0,5
14	Play Level	37,35	6,30
15			
16			
17			
18			
19			
20			
21			



TIME REPORT

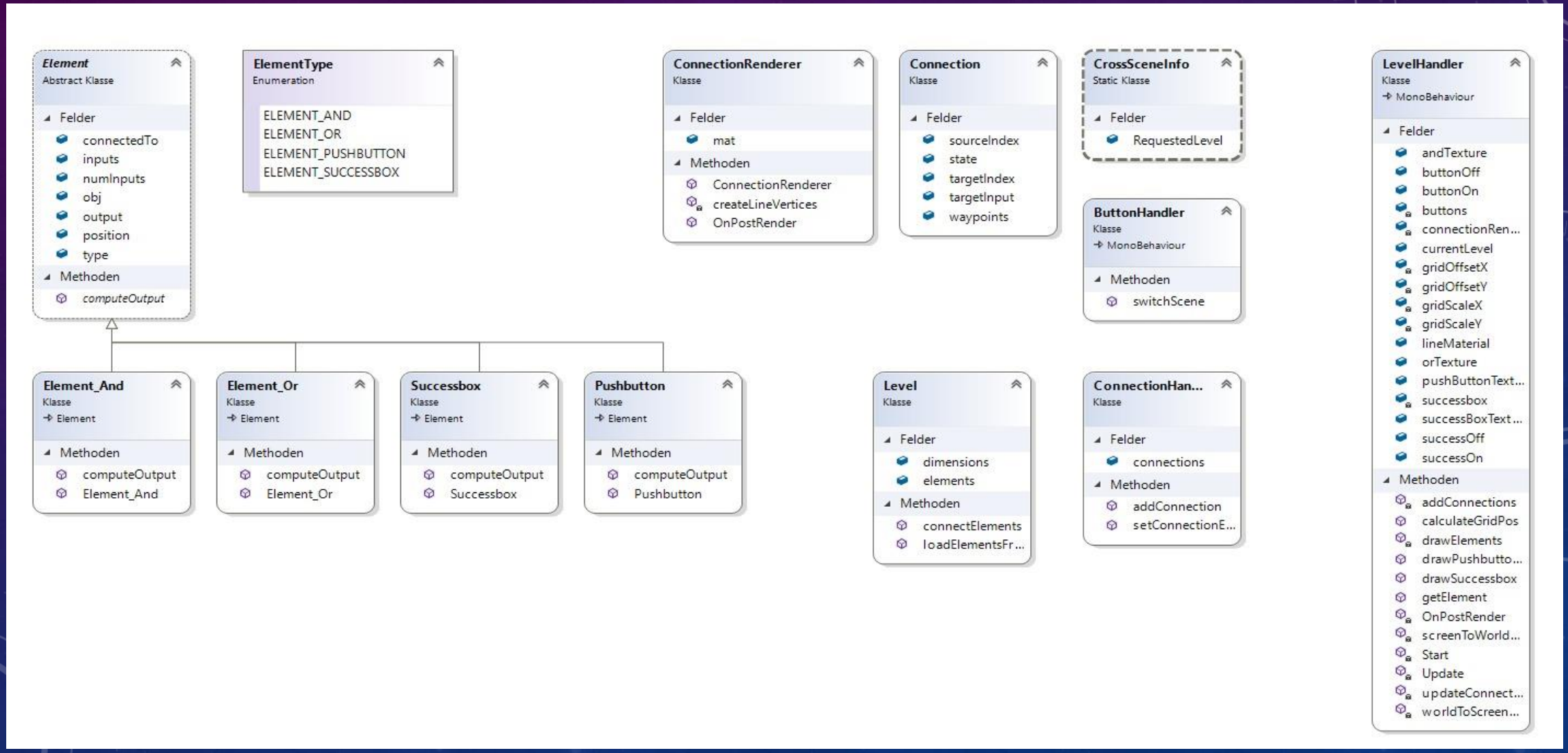
per user | per issue | per project | per work item | Show work types | S

Issues		Time estimated	Time spent
Total time		**73h 00m**	**74h 00m**
LG-1	Create Main Menu	4h 00m	4h 00m
LG-49	Final Presentation	3h 00m	4h 00m
LG-50	Blog entry Week 1 & 2	1h 00m	1h 00m
LG-51	Blog entry Week 3	1h 00m	1h 00m
LG-52	Create Overall UCS	3h 00m	3h 00m
LG-53	Create Sub-UCs for Overall UC	2h 00m	2h 00m
LG-54	Creating Riskscope	1h 00m	1h 00m
LG-55	Calculate Function Points	3h 00m	3h 00m
LG-56	Creating Peer Reviews	1h 00m	1h 00m

TECHNICAL ABILITIES

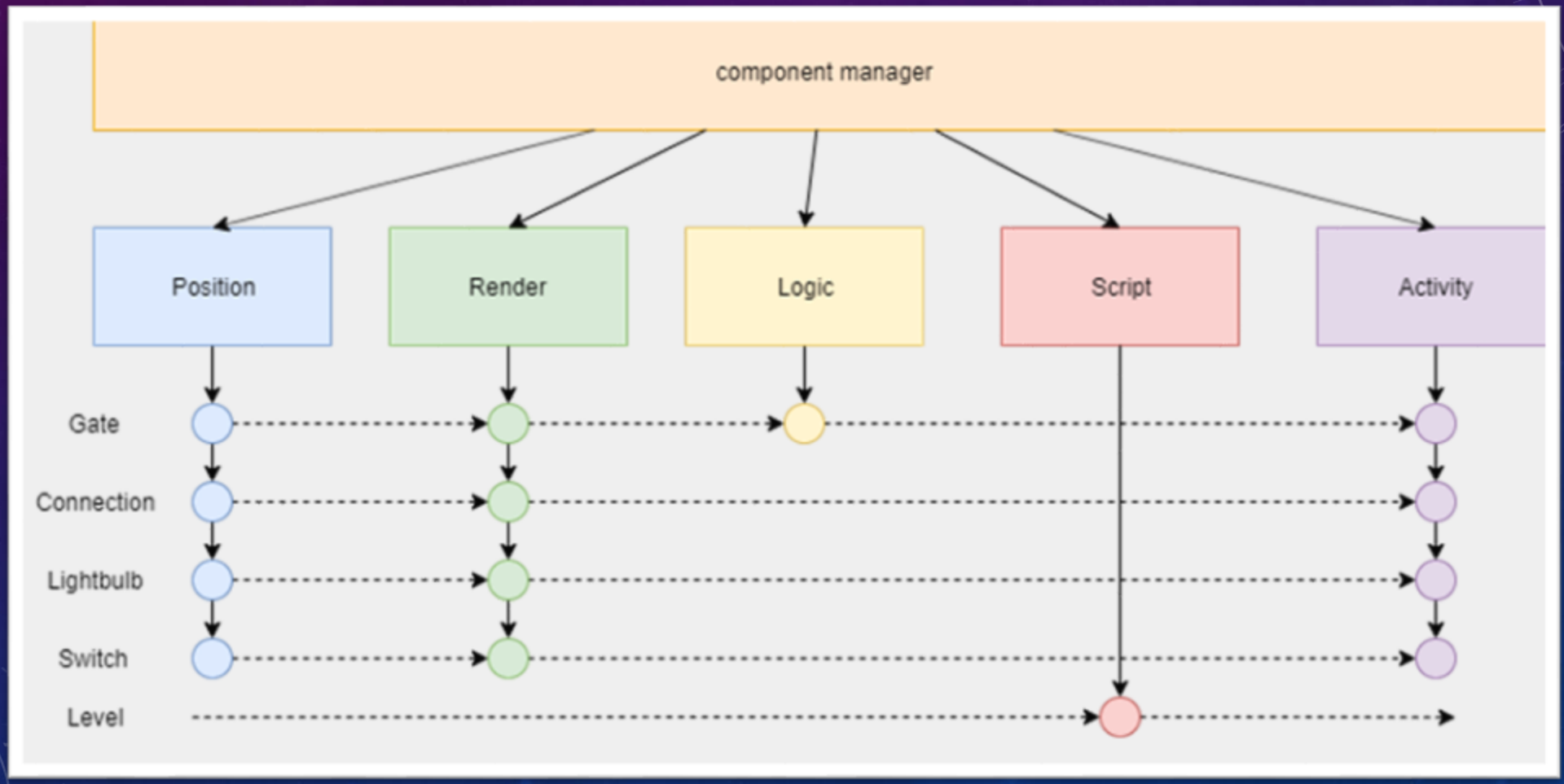
DEMO

CLASS DIAGRAM



QUALITY

COMPONENT ARCHITECTURE



SOFTWARE ARCHITECTURE DOCUMENT (SAD)

- Document describes Software Architecture such as:
  - **Architectural Representation**
  - **Architectural Goals and Constraints**
  - **Logical View**
  - **Design Patterns**
  - **Metrics**

CONFIGURATION

UNITY

- Game-Engine & Framework
- Component-based programming
- Easy & free to use
- Huge community
- Cross-Platform

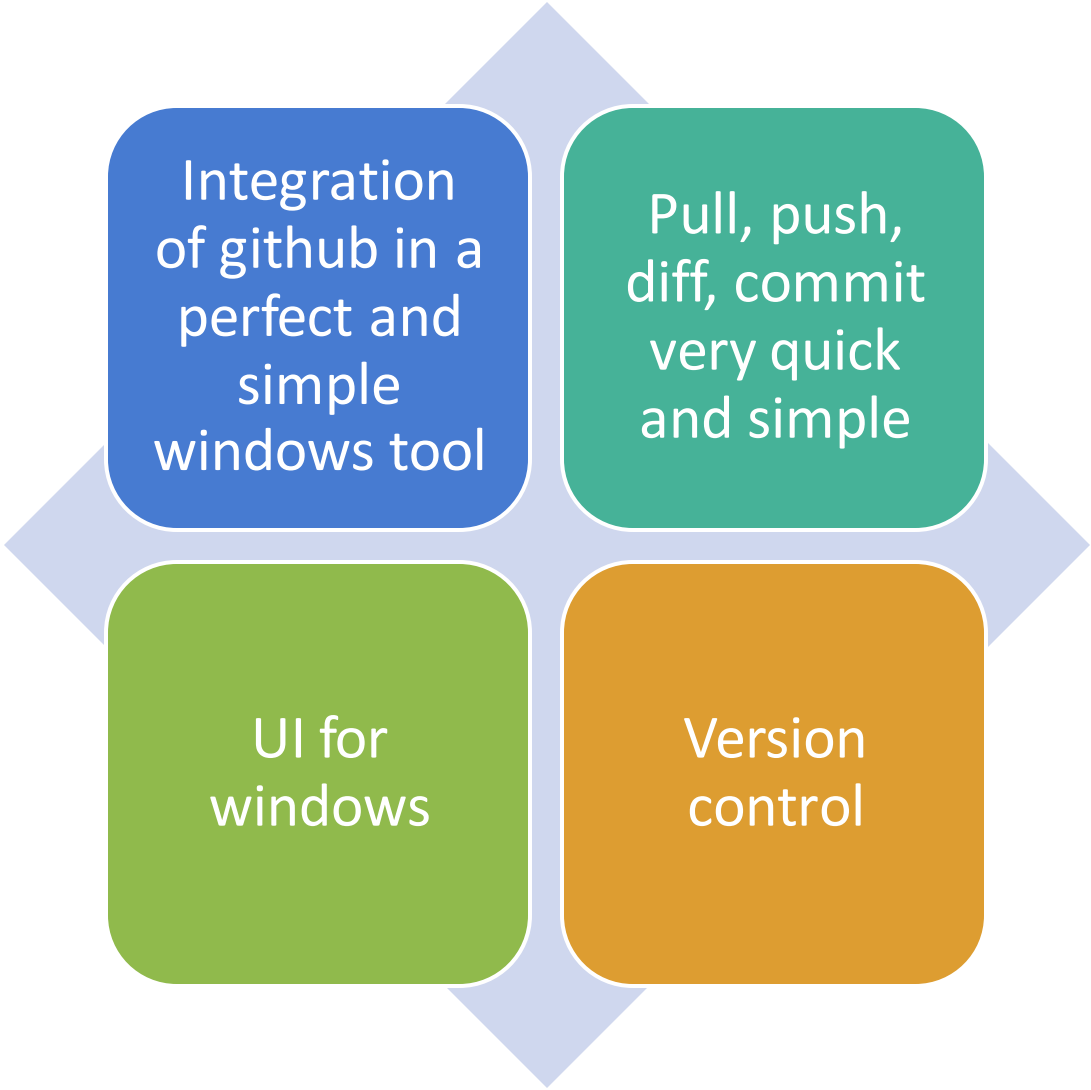
VISUAL STUDIO COMMUNITY 2019

- IDE
- Free for students
- Syntax highlighting
- Autocomplete
- Plugins for Unity
- Goes hand in hand with unity
- C# compiler
- Visual Studio Liveshare

GITHUB

- Development platform
- Host and share code
- Review options
- Automatically distributed versions
- Documentation
- Accessible from everywhere & everyone

TORTOISE GIT



AUTOMATION & LIFECYCLE MANAGEMENT

- A Push into github triggers Codacy and Codeclimate --> Metrics are updated
- Automatic version control due to github commits
- Documentation updates files automatically due to relative links to git repository
- New levels can be created and added easily
- Future: automatic deployment of new releases due to Playstore integration

RISK MANAGEMENT

- excel sheet that shows a list of risks
- kept up to date on a weekly basis
- Example:

Risk Name	Risk Description	Propability of	Risk Impact	Risk Factor	Risk Mitigation	Person in Charge of Tracking
Level Crashes	Crashes when loading a nerw level due to missing files or unknown bugs	0,20	9,00	1,80	Unittesting and beta tests with random classmates	Tim

TESTING

- Document describes test plan:
  - Introduction
  - **Evaluation Mission and Test Motivation**
  - **Outline of Planned Tests**
  - **Test Approach**
  - **Entry and Exit Criteria**
  - **Deliverables**
  - **Testing Workflow**
  - **Environmental Needs**
  - **Responsibilities, Staffing, and Training Needs**
  - **Iteration Milestones**
  - **Metrics**

PATTERNS

Singleton: software **design pattern** that restricts the instantiation of a **class** to one “single” instance

useful when exactly one object is needed to coordinate actions across the system

Created a branch on github --> not in our actual code --> unity project

Doesn't make sense to use in class “elements” --> can't have multiple elements with different properties united in one instance

Makes sense to use in "cross scene info" class --> exchange variables between the scenes --> makes sense to only have one instance of it

METRICS: CODACY AND CODECLIMATE

- Extra branch in github --> not implemented in our actual code
- Parts of the metrics don't make sense to implement
- Example:
  - Codacy suggested to remove our start() method, because it thought it was unused. The Method is never called manually, Unity calls it automatically.
  - Codacy suggested to make a public variable private and use getters and setters for it. Those variables are set in the unity editor and therefor cannot be accessed with getter and setter methods.

FUTURE POSSIBILITIES

- More levels --> automatic level generator
- Making money through advertisement
- Pay for pro version to get it ad free
- Registration required
- Options menu

THANK YOU FOR YOUR ATTENTION