2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

PyRDV: a Python-based solution to the requirements traceability problem

Fernando Gabriel Orge

Allegro microsystems, Buenos Aires, Argentina

ALLEGRO microsystems

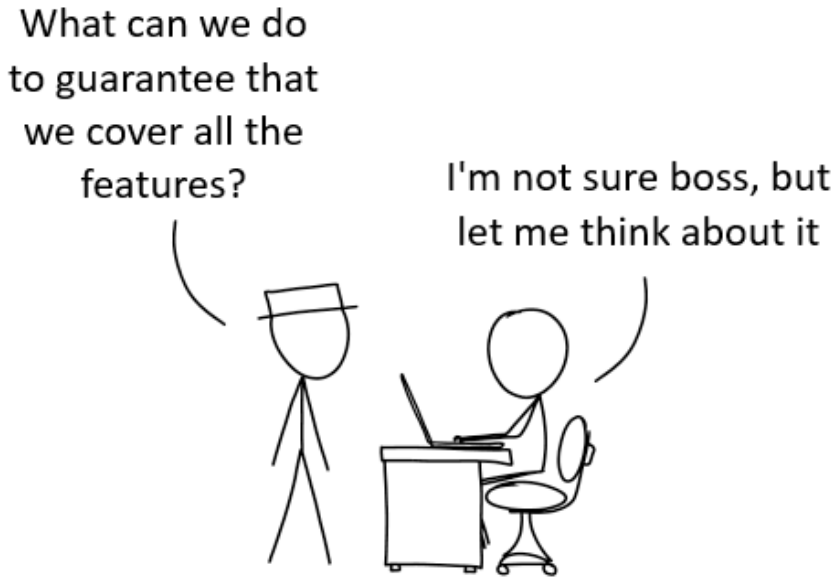
accellera SYSTEMS INITIATIVE

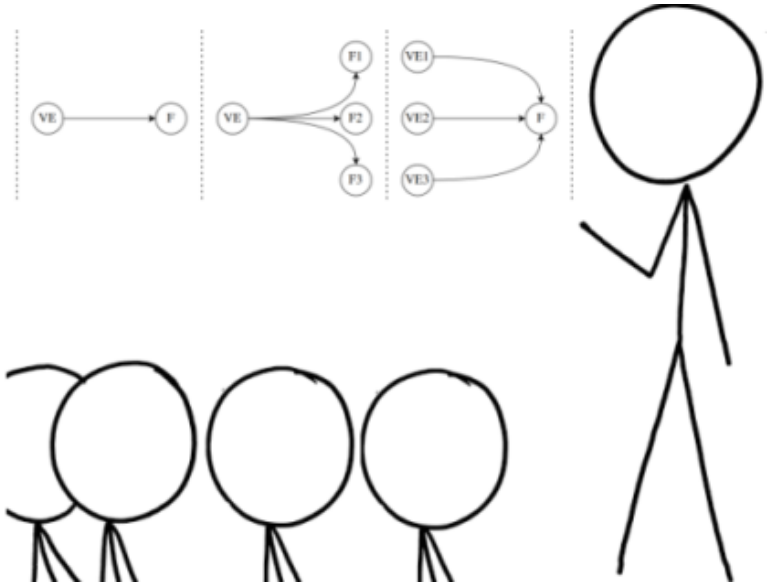
Agenda

- Motivation
- What's PyRDV?
- Sample Case
- Conclusions

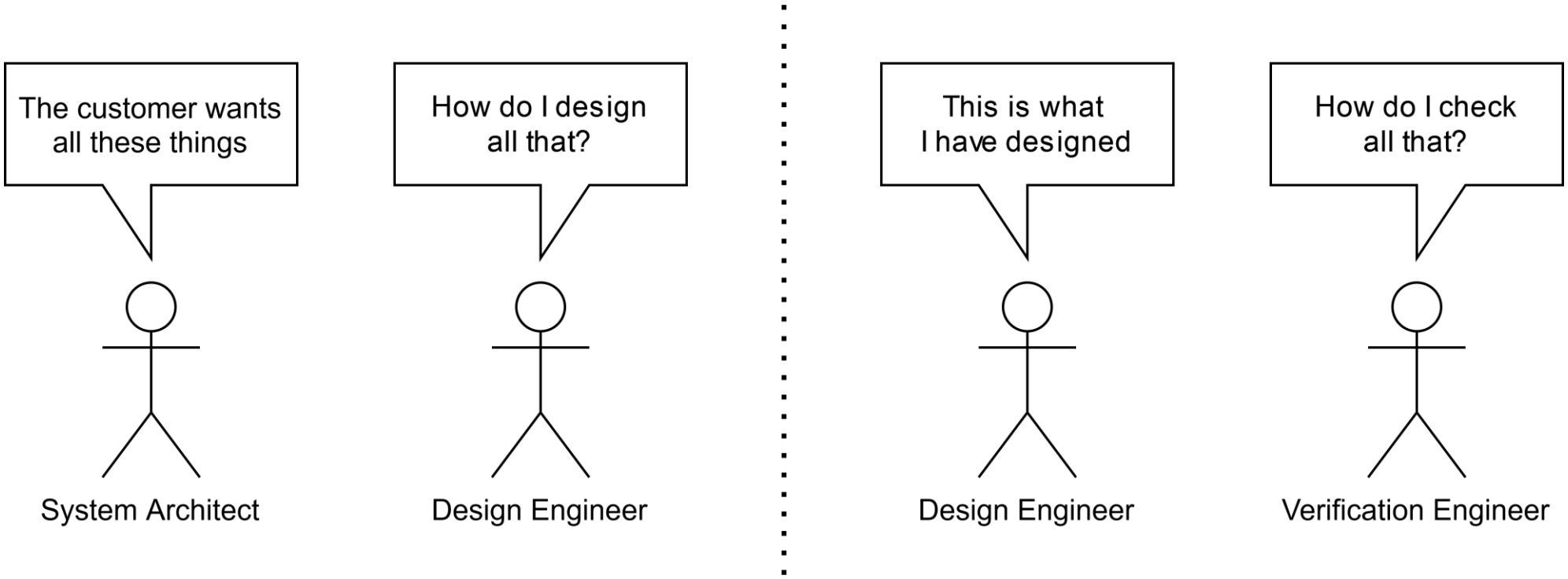
Motivation


Hi boss, I'm going to do a master!
Let me know when you've learned something new...


What can we do to guarantee that we cover all the features?
I'm not sure boss, but let me think about it


VE
F
VE
F1
F2
F3
VE1
VE2
VE3
F

Motivation



Two consumer-producer problems

Motivation

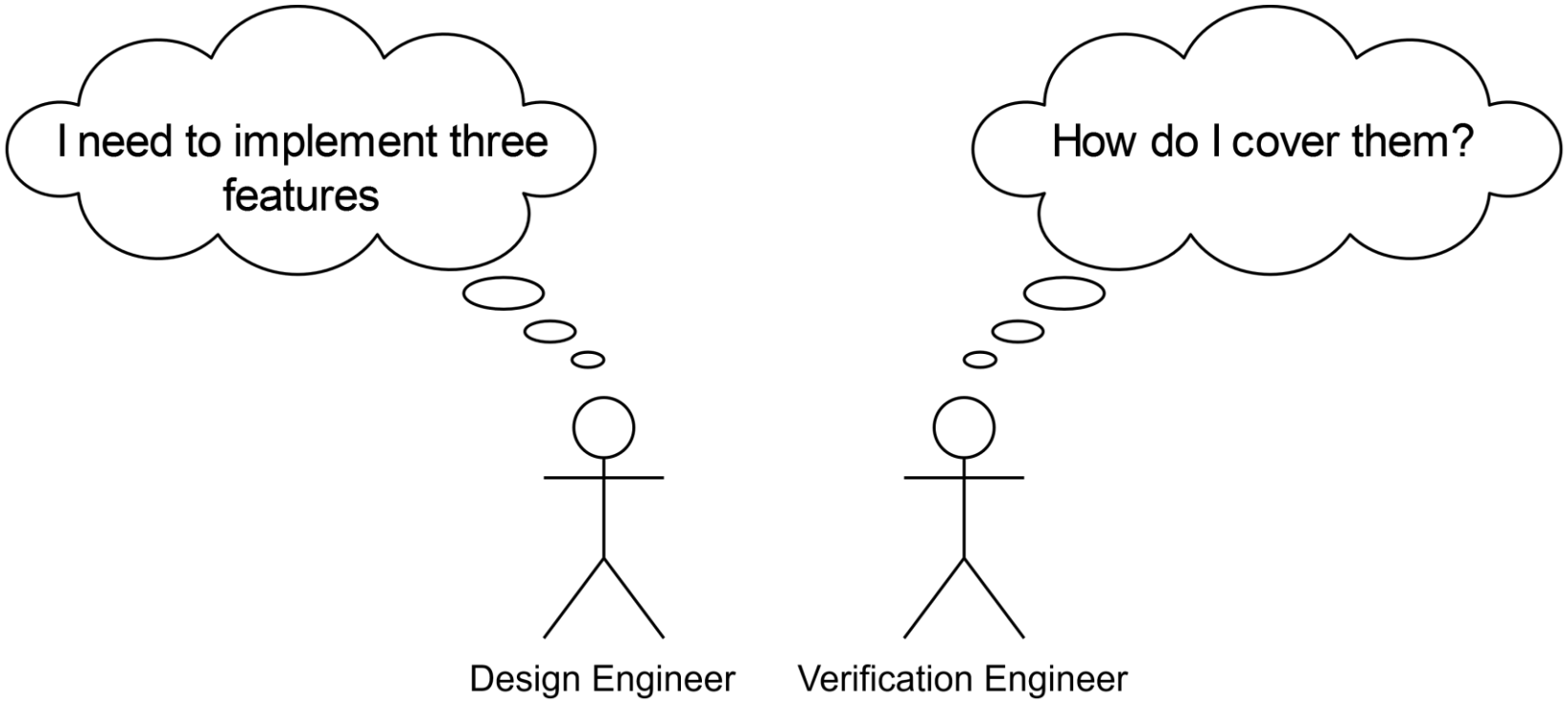
- A successful design will meet the following conditions:
  - Designers must implement all the requirements
  - Verification engineers must verify all design specifications

- We need to solve
  - Requirements To Features Mapping Problem
  - Features To Verification Elements Mapping Problem

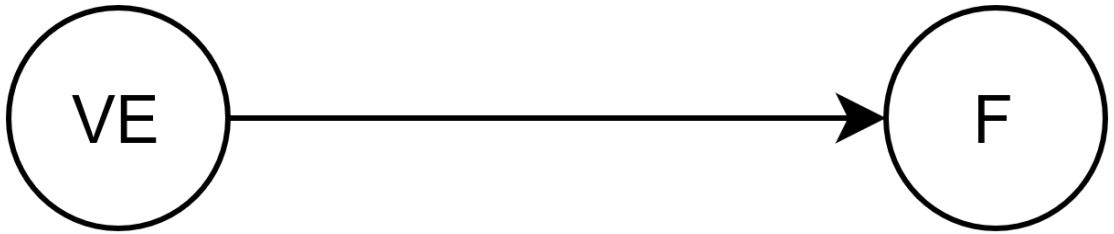
What's PyRDV?

- A theoretical framework to prove the solution to the problem
- A detailed workflow for IC Developers
- A CI / CD service to periodically check for sign-off metrics
- A Python-based software solution

Theoretical Framework

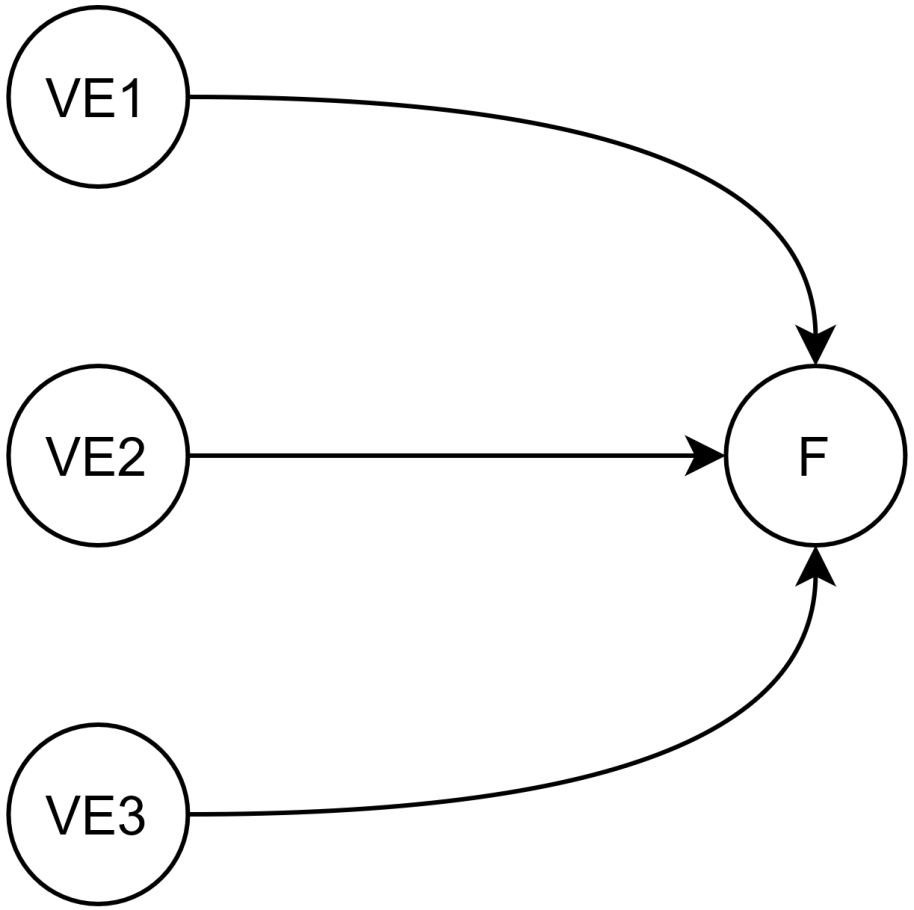


Theoretical Framework



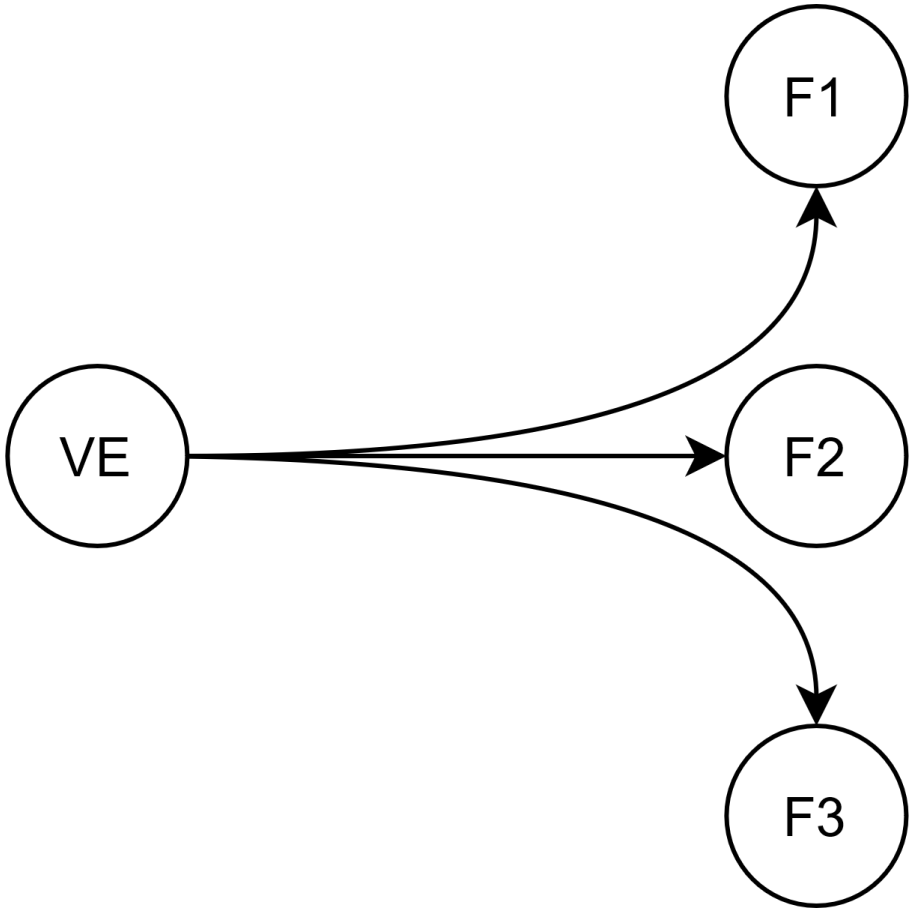
- One-To-One Relation
- Preferred case
- **Strongly covered** feature
- **Strongly linked** verification element

Theoretical Framework



- Many-To-One Relation
- **Weakly covered** feature
- Not atomic feature
- Might require feature slicing

Theoretical Framework



- One-To-Many Relation
- **Strongly covered** features
- Integration/top-level test
- Simple features

Theoretical Framework

- Unrelated elements
- **Uncovered** feature
  - Coverage hole
- **Useless** verification element
  - Over-engineering

Theoretical Framework

$$g : F \times V \rightarrow \{0,1\} \; such \; that \; g(i,j) = \begin{cases} 1 & if \; feature_i \; is \; verified \; by \; verification \; element_j \\ 0 & if \; not \end{cases}$$

$$\begin{pmatrix} g(1,1) & g(1,2) & ... & g(1,N) \\ g(2,1) & g(2,2) & ... & g(2,N) \\ ... & ... & ... & ... \\ g(M,1) & g(M,2) & ... & g(M,N) \end{pmatrix} = \begin{pmatrix} 0 & 1 & ... & 0 \\ 1 & 0 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 1 \end{pmatrix}$$

Theoretical Framework

$$\begin{pmatrix} g(1,1) & g(1,2) & \dots & g(1,N) \\ g(2,1) & g(2,2) & \dots & g(2,N) \\ \dots & \dots & \dots & \dots \\ g(M,1) & g(M,2) & \dots & g(M,N) \end{pmatrix} = \begin{pmatrix} 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

Theoretical Framework

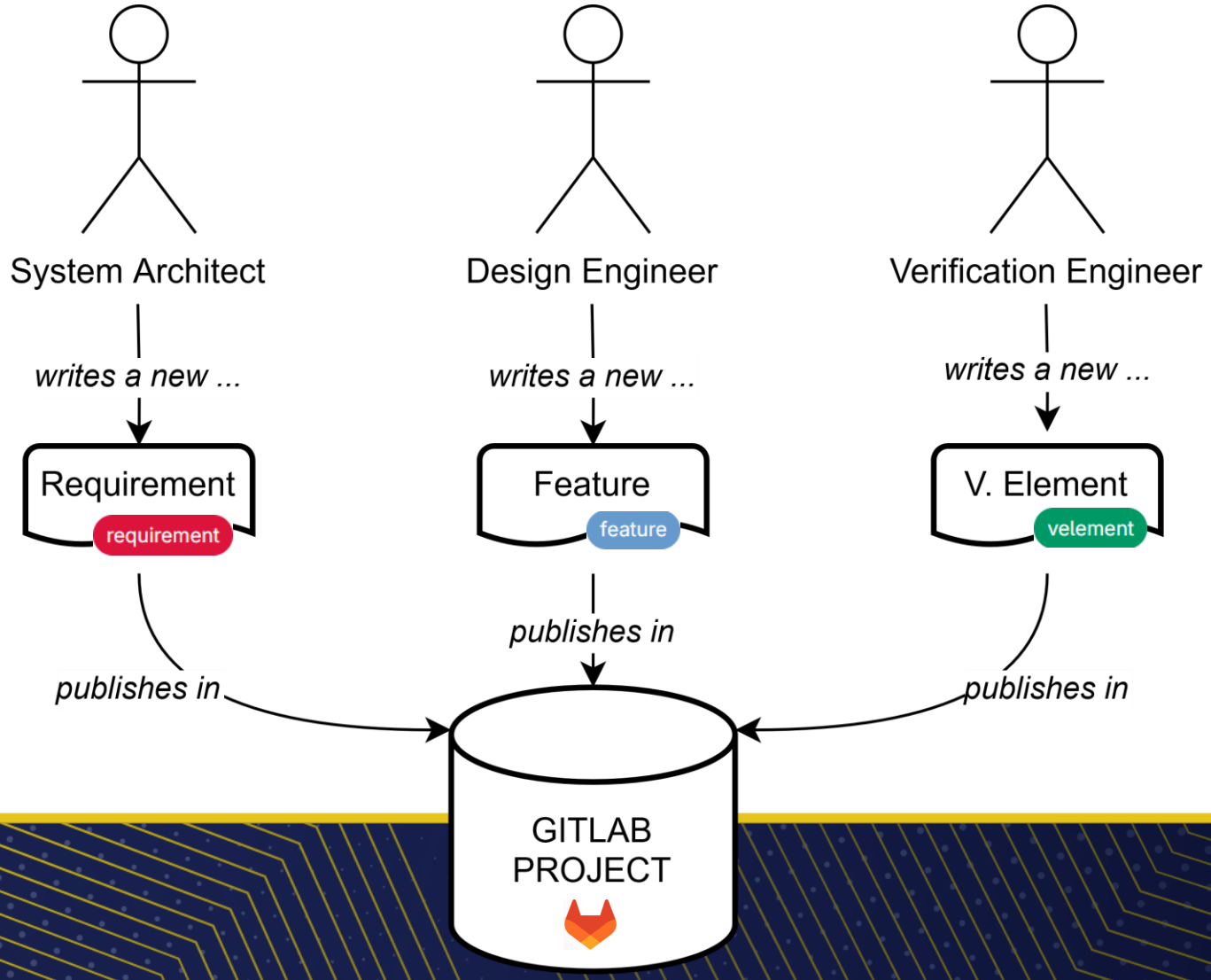
$$\begin{pmatrix} g(1,1) & g(1,2) & \dots & g(1,N) \\ g(2,1) & g(2,2) & \dots & g(2,N) \\ \dots & \dots & \dots & \dots \\ g(M,1) & g(M,2) & \dots & g(M,N) \end{pmatrix} = \begin{pmatrix} 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

Theoretical Framework: Job is done when…

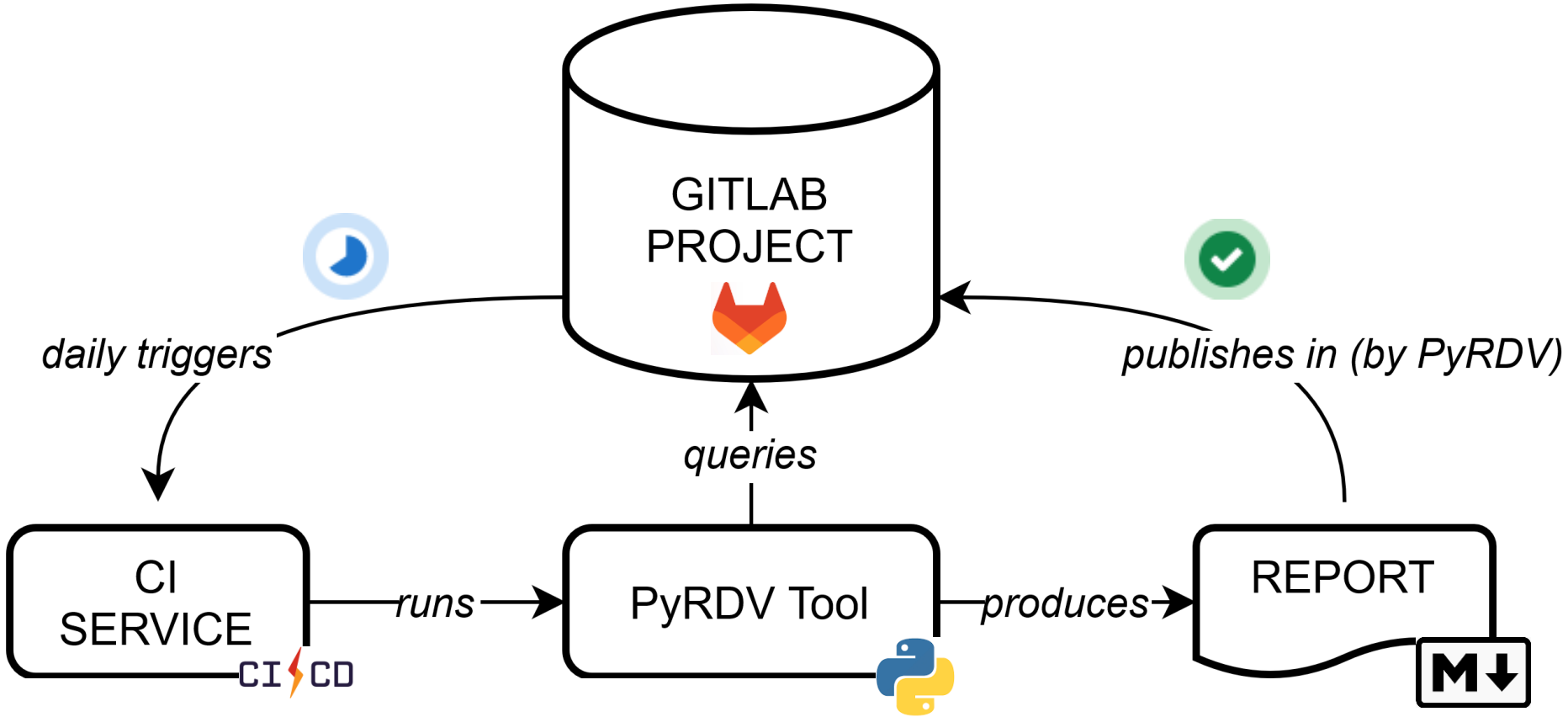
$$\sum_{j \in V} g(i,j) \geq 1, \ \forall i \in F \quad and \quad \sum_{i \in F} g(i,j) \geq 1, \ \forall j \in V$$

- All features are **strongly** or **weakly covered**
- All verification elements are **linked**
- This is the **completeness condition**
- Can be extrapolated to also solve
  - Requirements To Features Mapping Problem
  - Requirements To Verification Elements Mapping Problem

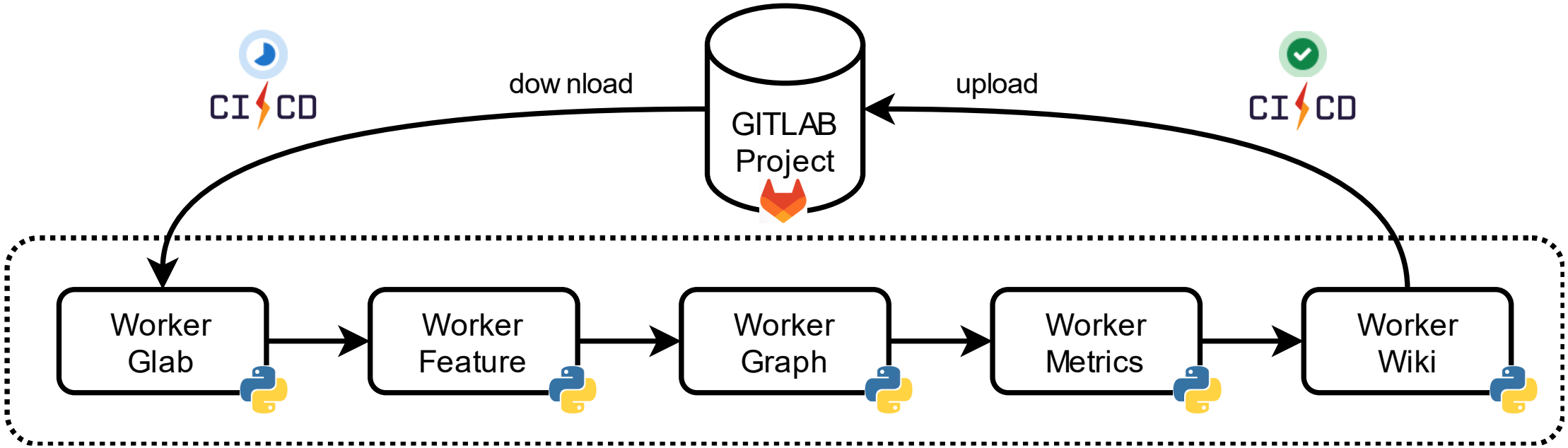
Workflow of IC Developers



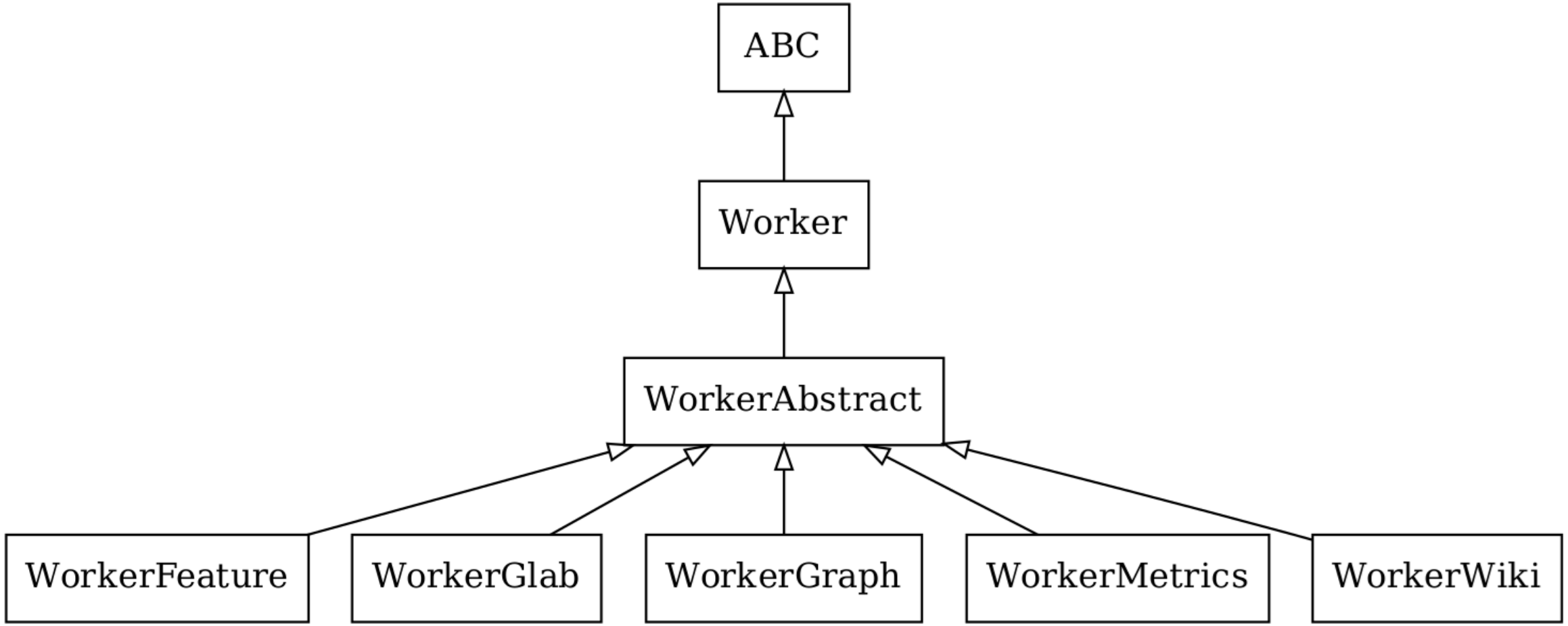
CI/CD Service



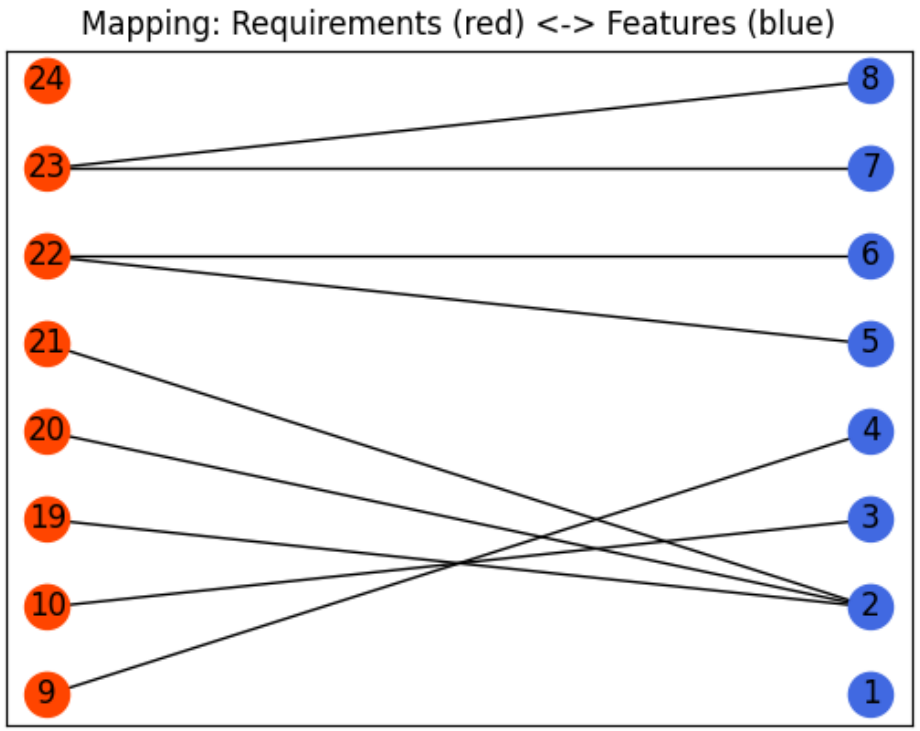
Python-based software solution

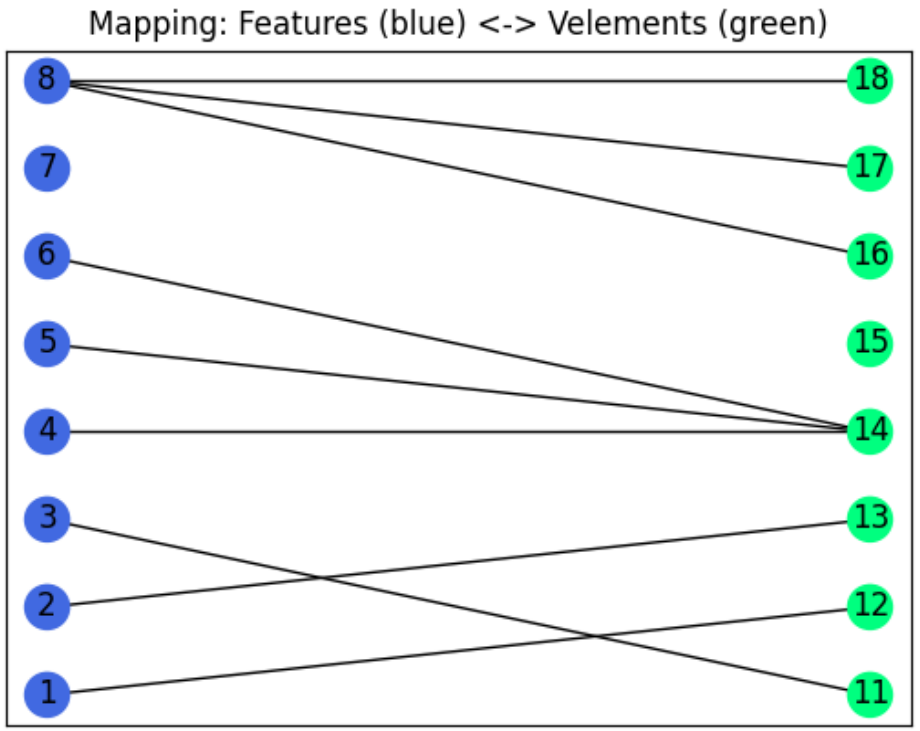


Python-based software solution

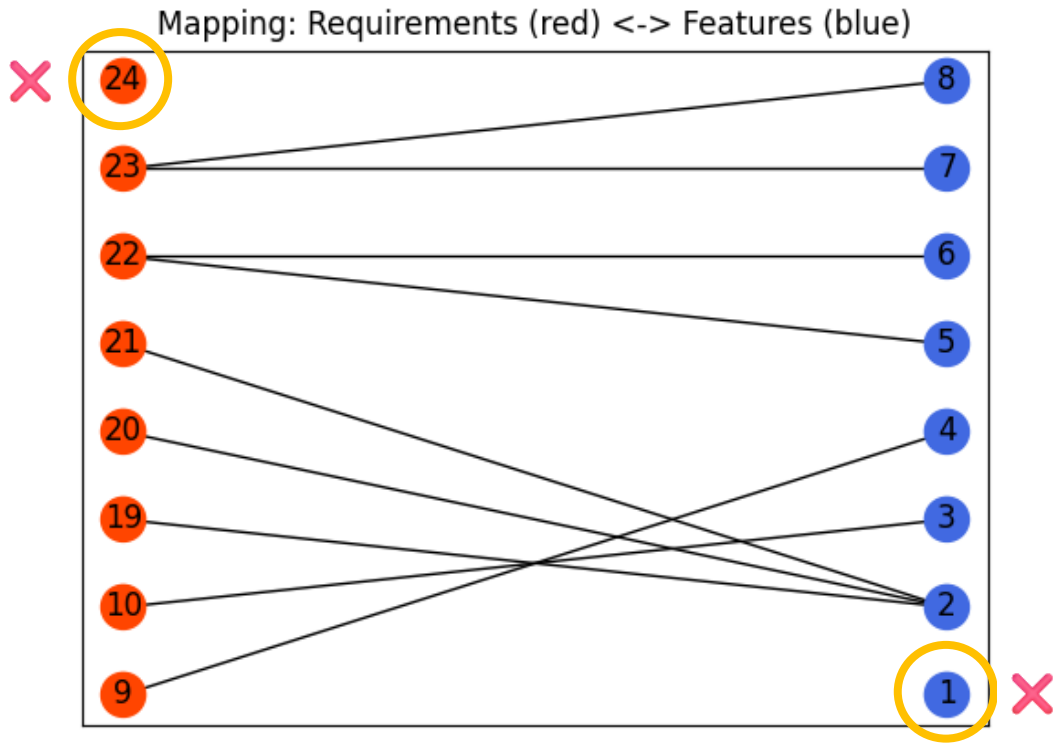


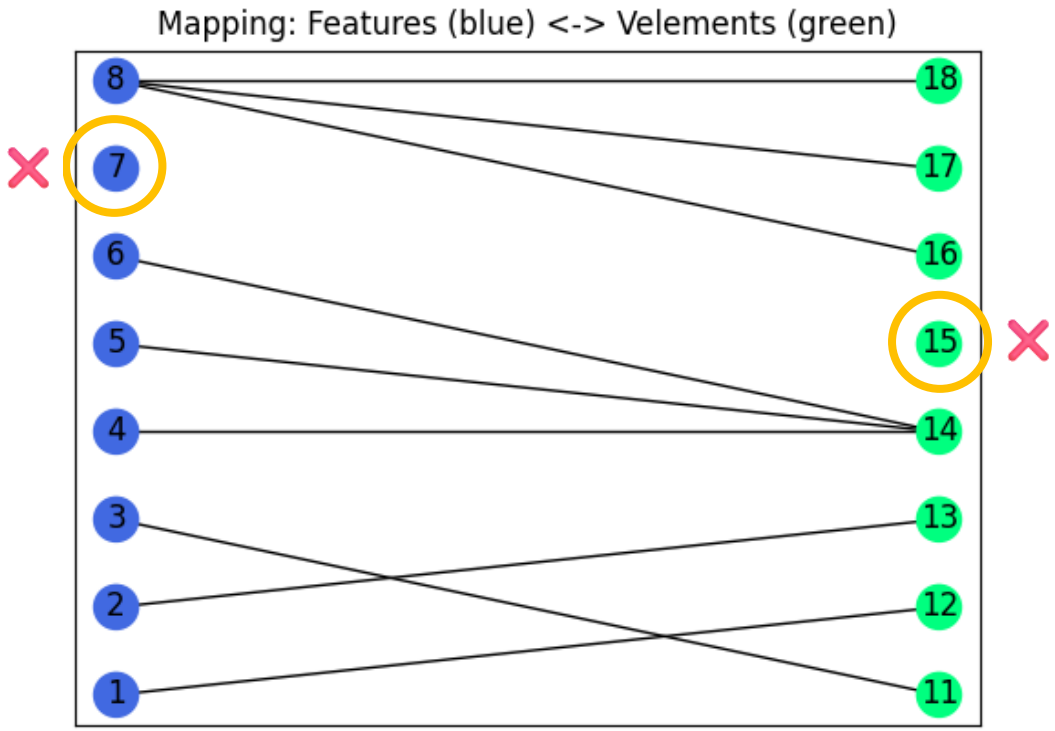
Sample Case


Mapping: Requirements (red) <-> Features (blue)
24
23
22
21
20
19
10
9
8
7
6
5
4
3
2
1

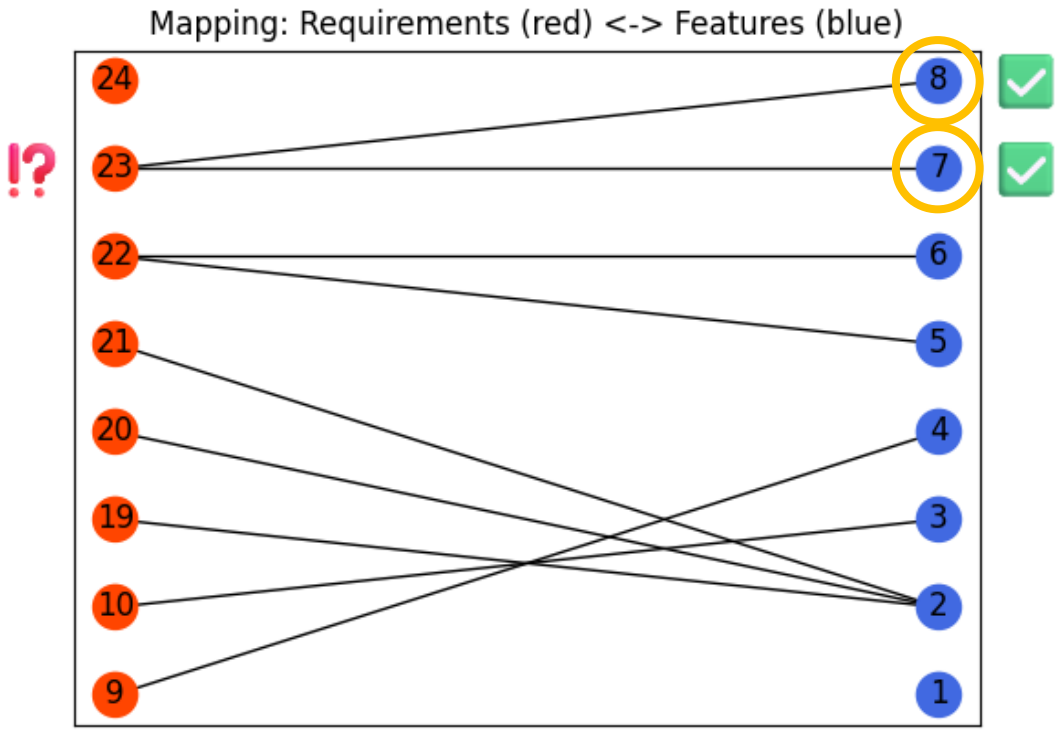

Mapping: Features (blue) <-> Velements (green)
8
7
6
5
4
3
2
1
18
17
16
15
14
13
12
11

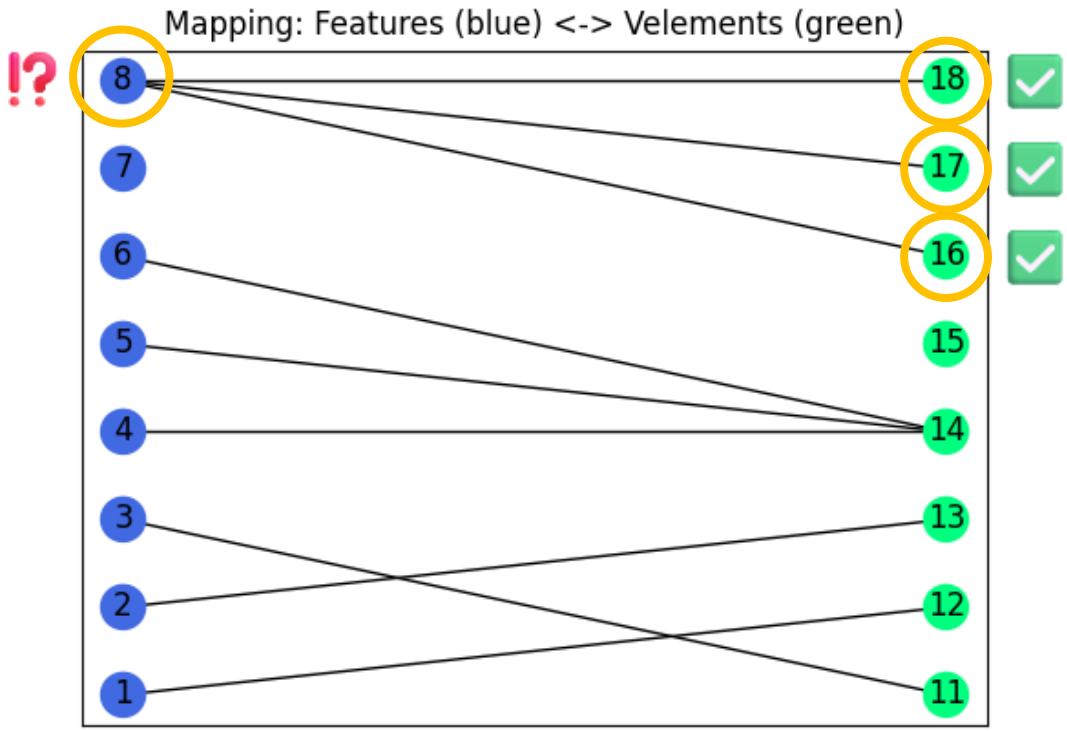
Sample Case


Mapping: Requirements (red) <-> Features (blue)
24
23
22
21
20
19
10
9
8
7
6
5
4
3
2
1

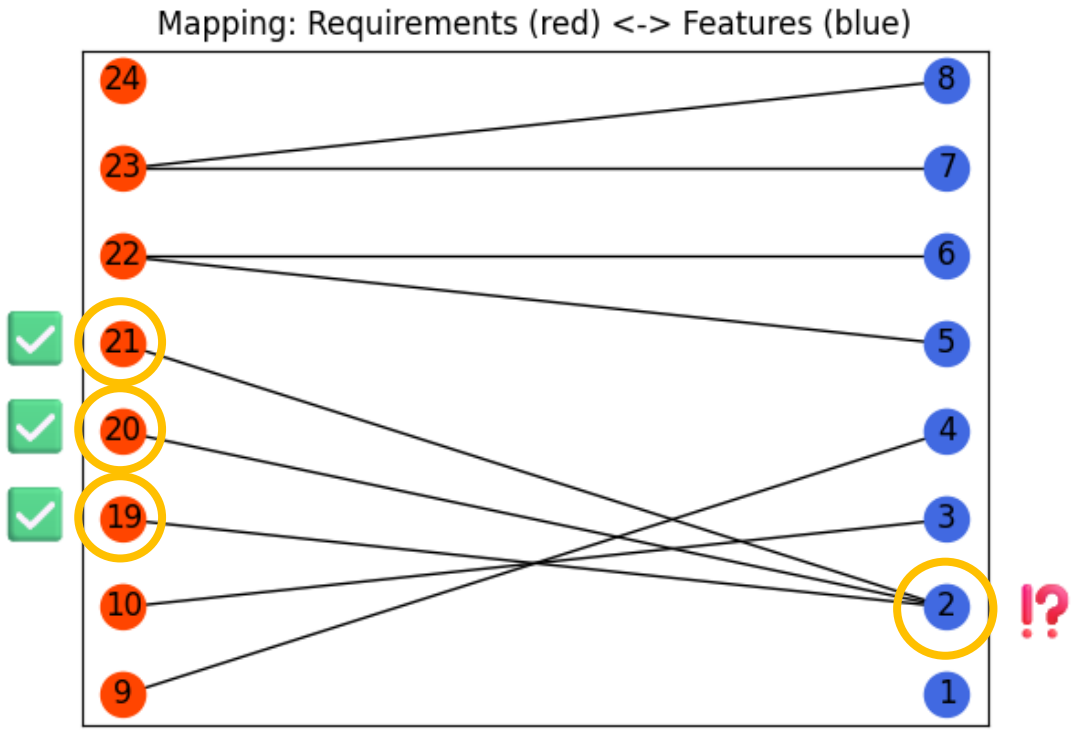

Mapping: Features (blue) <-> Velements (green)
8
7
6
5
4
3
2
1
18
17
16
15
14
13
12
11

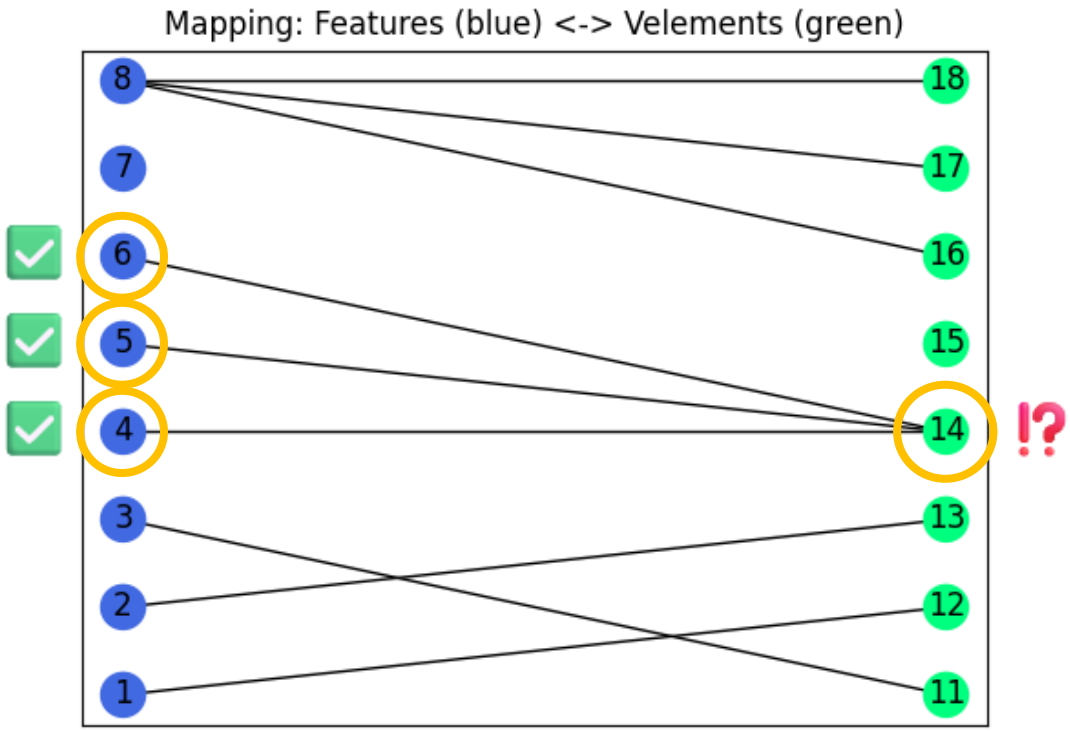
Sample Case


Mapping: Requirements (red) <-> Features (blue)
!?
24
23
22
21
20
19
10
9
8
7
6
5
4
3
2
1

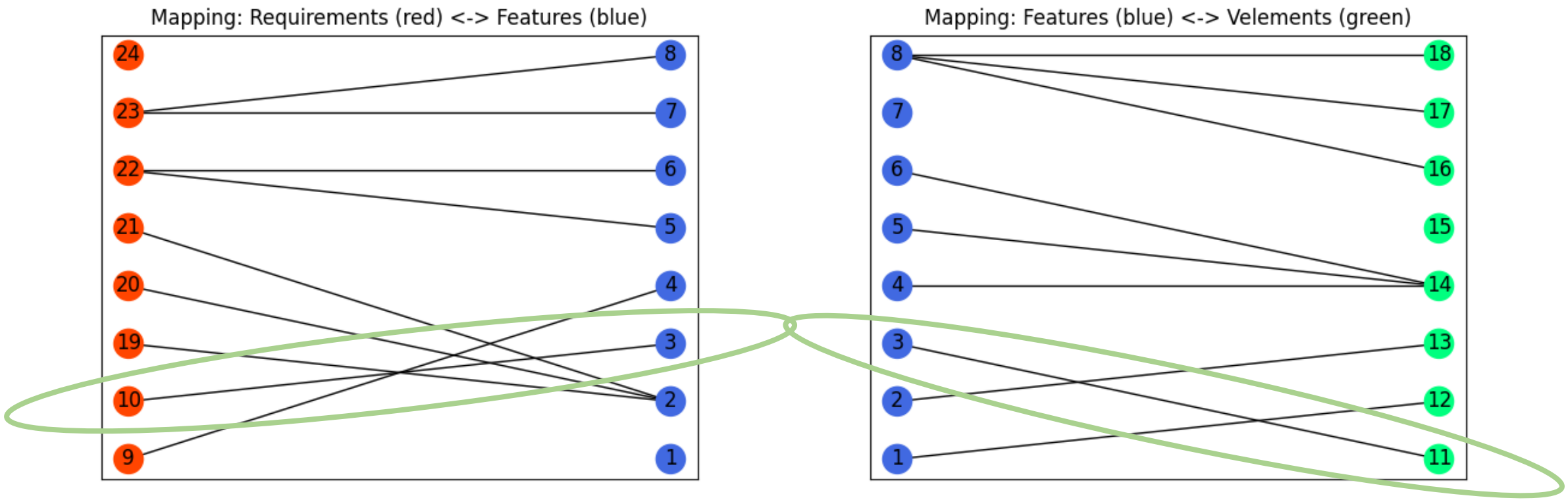

Mapping: Features (blue) <-> Velements (green)
!?
8
7
6
5
4
3
2
1
18
17
16
15
14
13
12
11

Sample Case

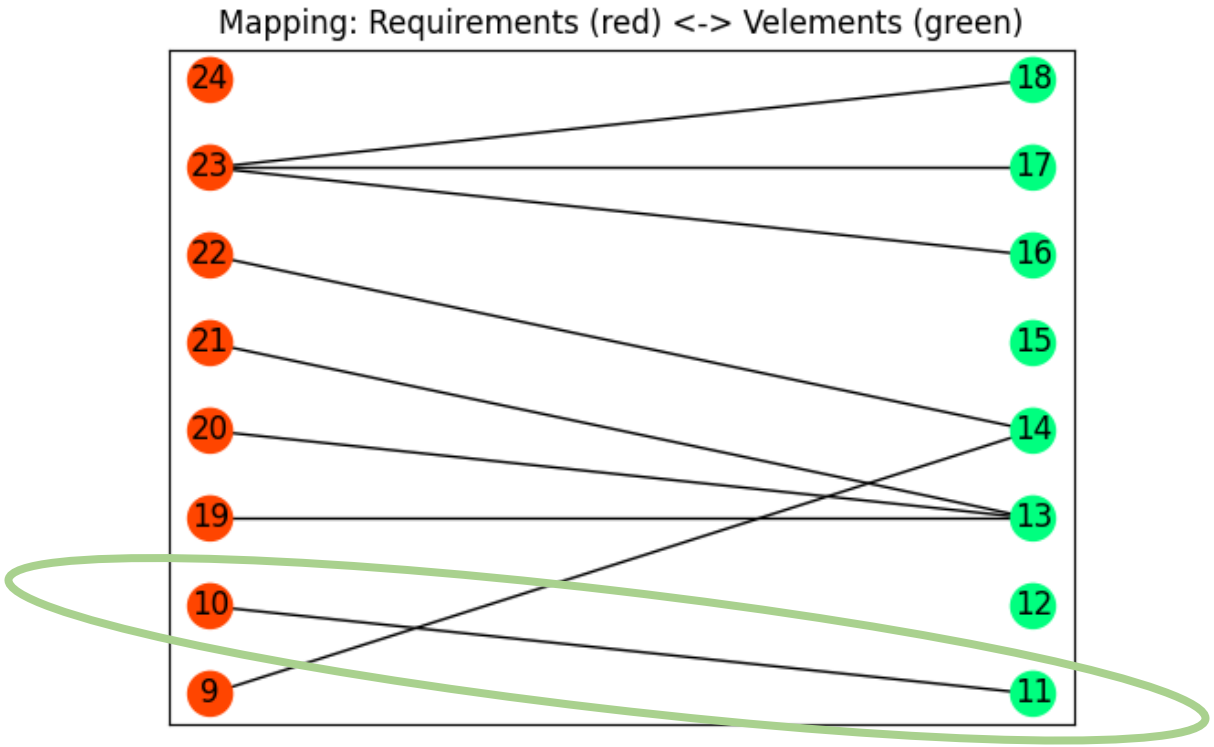

Mapping: Requirements (red) <-> Features (blue)
24
23
22
21
20
19
10
9
8
7
6
5
4
3
2
1
!?


Mapping: Features (blue) <-> Velements (green)
8
7
6
5
4
3
2
1
18
17
16
15
14
13
12
11
!?

Sample Case


Mapping: Requirements (red) <-> Features (blue)
24
23
22
21
20
19
10
9
8
7
6
5
4
3
2
1
Mapping: Features (blue) <-> Velements (green)
8
7
6
5
4
3
2
1
18
17
16
15
14
13
12
11

Sample Case


Mapping: Requirements (red) <-> Velements (green)
24
23
22
21
20
19
10
9
18
17
16
15
14
13
12
11

Reports

Requirements

Requirement	Related Features	GRADE	STATUS
#9	#4	STRONGLY COVERED	✅
#10	#3	STRONGLY COVERED	✅
#19	#2	STRONGLY COVERED	✅
#20	#2	STRONGLY COVERED	✅
#21	#2	STRONGLY COVERED	✅
#22	#5 #6	WEAKLY COVERED	⁉
#23	#7 #8	WEAKLY COVERED	⁉
#24		NOT COVERED	❌

Features

Feature	Related Vplan Elements	GRADE	STATUS
#1	#12	STRONGLY COVERED	✅
#2	#13	STRONGLY COVERED	✅
#3	#11	STRONGLY COVERED	✅
#4	#14	STRONGLY COVERED	✅
#5	#14	STRONGLY COVERED	✅
#6	#14	STRONGLY COVERED	✅
#7		NOT COVERED	❌
#8	#16 #17 #18	WEAKLY COVERED	⁉

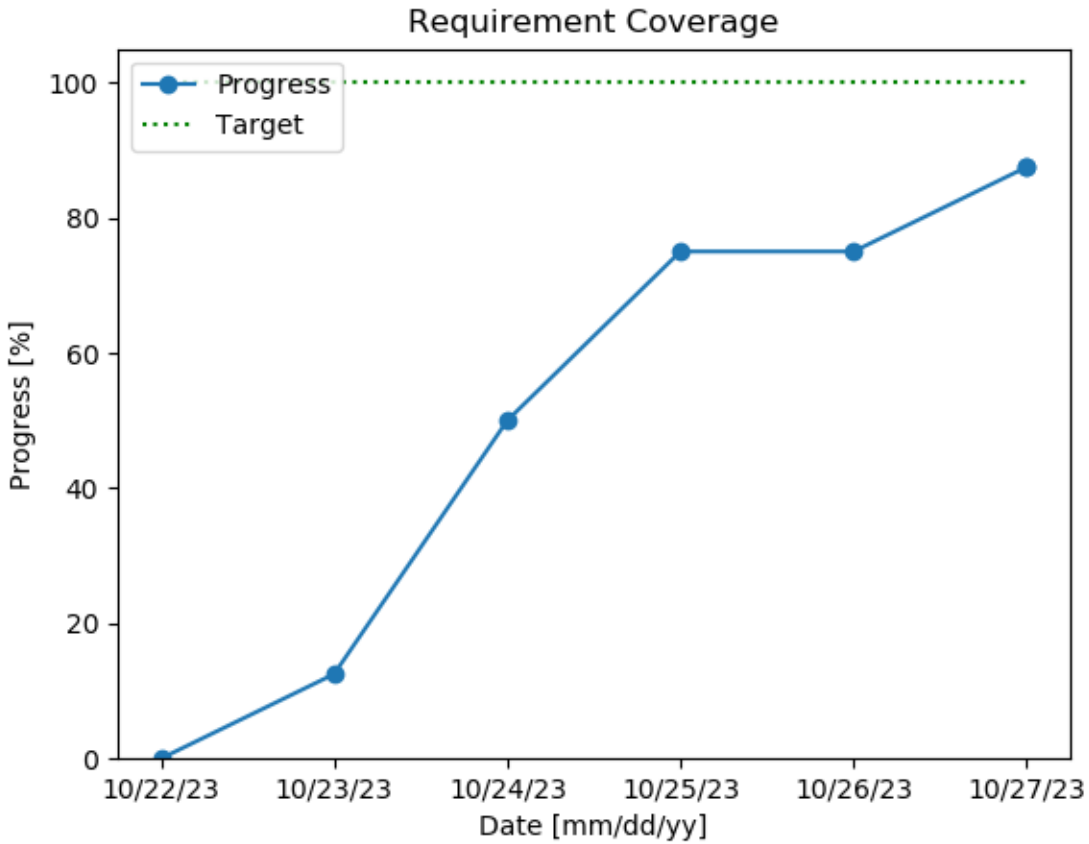
Features

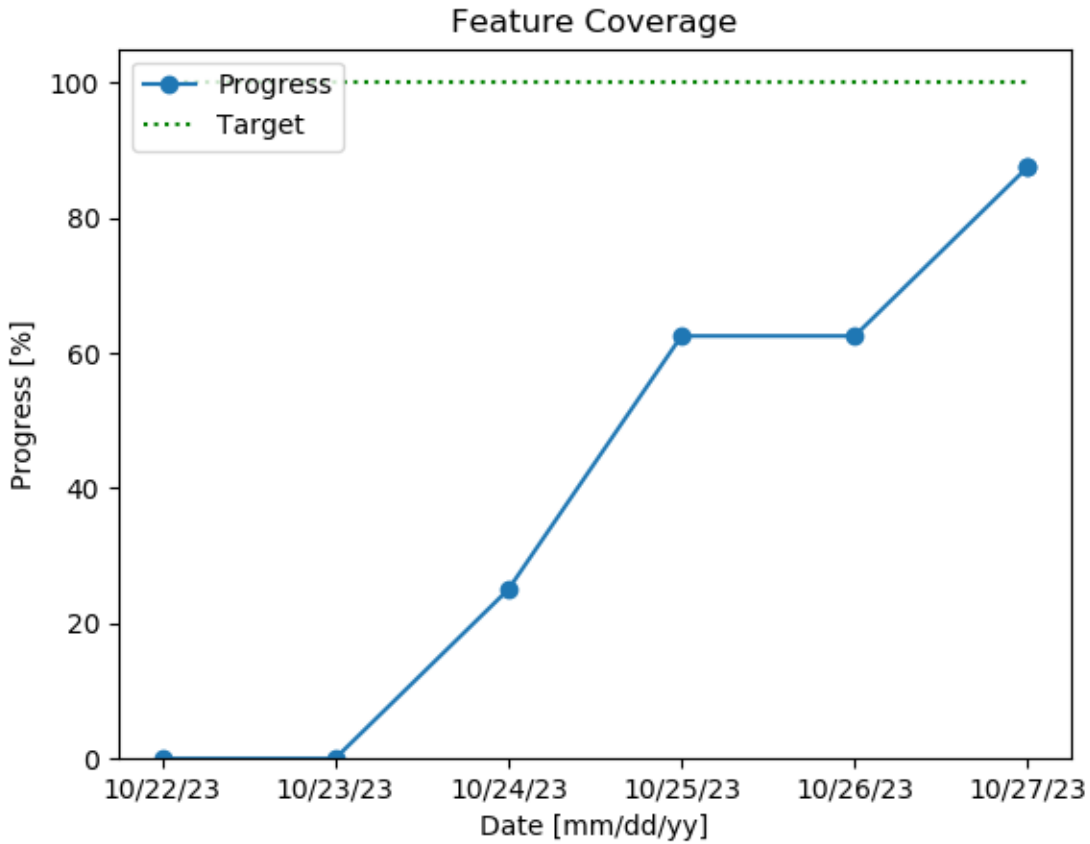
Feature	Related Requirements	GRADE	STATUS
#1		NOT LINKED	❌
#2	#21 #20 #19	WEAKLY LINKED	⁉
#3	#10	STRONGLY LINKED	✅
#4	#9	STRONGLY LINKED	✅
#5	#22	STRONGLY LINKED	✅
#6	#22	STRONGLY LINKED	✅
#7	#23	STRONGLY LINKED	✅
#8	#23	STRONGLY LINKED	✅

Vplan Elements

Vplan Element	Related Features	GRADE	STATUS
#11	#3	STRONGLY LINKED	✅
#12	#1	STRONGLY LINKED	✅
#13	#2	STRONGLY LINKED	✅
#14	#4 #5 #6	WEAKLY LINKED	⁉
#15		NOT LINKED	❌
#16	#8	STRONGLY LINKED	✅
#17	#8	STRONGLY LINKED	✅
#18	#8	STRONGLY LINKED	✅

Reports


Requirement Coverage
Progress
Target
100
80
60
40
20
0
Progress [%]
10/22/23
10/23/23
10/24/23
10/25/23
10/26/23
10/27/23
Date [mm/dd/yy]


Feature Coverage
Progress
Target
100
80
60
40
20
0
Progress [%]
10/22/23
10/23/23
10/24/23
10/25/23
10/26/23
10/27/23
Date [mm/dd/yy]

Conclusions

- We used GitLab to centralize and manage all the information.
  - Including requirements, features or design specification and verification plan.
- We eliminated the need for developers to adapt to a new tool.
  - GitLab was already a tool in use by the company.
- We used CI/CD services to improve design and verification workflow.
  - It allows us to find potential coverage holes faster.
- The framework is not limited to GitLab.
  - Can be applied to any platform that offers an issue-tracking capability.

Conclusions

- We developed a Python-base solution.
  - The algorithms and the design pattern can be applied to any other language.
- We can improve the effectiveness and efficiency of the design process.
  - We can always check that all requirements are implemented.
  - We can always check that all specifications are verified.
- We are planning to use AI techniques to predict the execution time of future projects based on the information collected by the PyRDV tool.

Thanks for attending
Questions?