

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under TestEAMoS (TIN2016-76956-C3-1-R) project and ERDF funds, and by the European Project ElasTest in the Horizon 2020 research and innovation program (GANo. 731535).

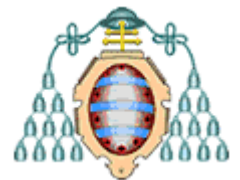
E2E Resource-Aware Test Orchestration Framework

Cristian Augusto, Jesús Morán, Antonia Bertolino, Claudio de la Riva and Javier Tuya
Software Engineering Research Group / Software Engineering & Dependable
Computing Laboratory

<http://giis.uniovi.es> / <http://labsedc.isti.cnr.it>



University of Oviedo
&
Istituto di Scienza e Tecnologie
dell'Informazione "A. Faedo"



Context

- Continuous integration practices → Incremental changes in the code →
New faults
- Usual Strategy → Automate and Re-execute all test per each change

Context

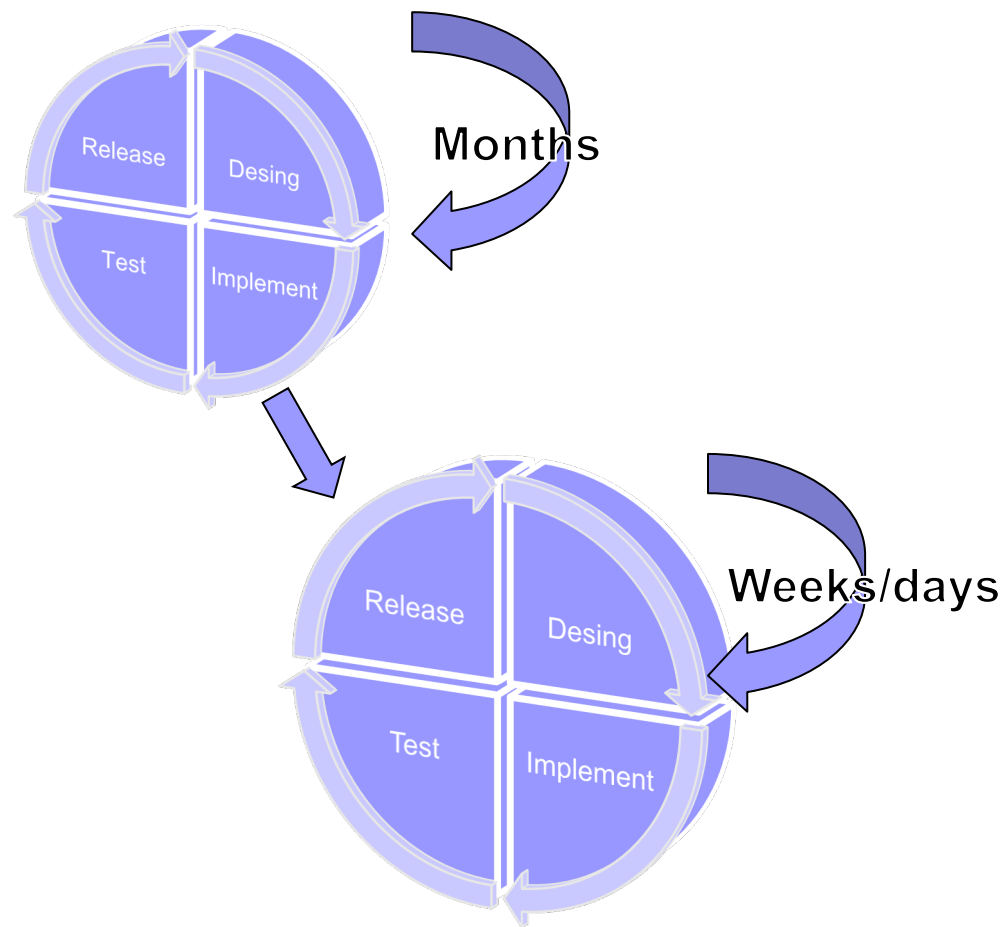
- Nowadays software development cycles are **shorter** than the past
- With this *increase of the release speed*, **tests suites** become **larger** and are executed more frequently



Microsoft have huge test suites with 60k of tests, that take the better part of a day to run



Google Test Automation Platform (TAP) executes daily more than 150 Million test runs



¹How we approach testing VSTS to enable continuous delivery | Brian Harry's Blog. (n.d.). Retrieved September 8, 2019, from <https://devblogs.microsoft.com/bharry/testing-in-a-cloud-delivery-cadence/>

²Memon, A., Zebao Gao, Bao Nguyen, Dhanda, S., Nickell, E., Siemborski, R., & Micco, J. (2017). Taming Google-scale continuous testing. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 233–242. <https://doi.org/10.1109/ICSE-SEIP.2017.16>

Motivation

- This test-suites improve software **quality** and **reliability**
- As a **number** of test **increases is not feasible re-execute them** each time:
 - ✓ **Minimization** and **Prioritization** techniques emerge as possible solution.
 - ✗ Not useful in all contexts

Motivation

- Certain testing levels, like End to End testing (E2E) requires **great amount** of resources
- Several approaches tries to flexible and optimize the resources used during testing:
 - Resource sharing
 - Containerized execution

Motivation



■ Elastest:

- Elastest (*European platform to ease E2E testing*):¹
 - Provides a containerized execution of the test via TJobs (Containerized tests with the SUT that they require)
- Addresses several problems (Centralized log analysis, easy deployment of the SUTs...), but there is a room of improvement in resource usage terms:
 - Problems with resources employed in E2E testing

¹ [Project available in :https://elastest.eu/](https://elastest.eu/)

E2E Orchestration Framework Proposal

- Objective → Optimize / minimize the resources used into E2E testing
- How → Through a orchestration method based on the resources needed to execute each test

E2E Orchestration Framework Proposal

- Resource: Physical, logical and/or computational entities required by the execution of one or more test cases. Examples:
 - One webcam required by the test
 - An Streaming server
 - The physical memory consumed
- Resource characteristics:
 - Elasticity → Is feasible instantiate several resources?
 - Sharing → Can be accessed concurrently?
 - Lifecycle → Set-up / Test-phase / Dispose

E2E Orchestration Framework Proposal

- Some types of access modes:
 - Read – Only → Streaming link
 - Read – Write → Database table
 - Write – Only → Log file
 - Dynamic

E2E Orchestration Framework Proposal

- Resource properties:
 - Allocated → Could where is deployed be located?
 - Measurable → Is there any kind of quantitative indicator?
 - Traceable → Could you know where it is in the cycle?
 - Elasticity cost → How much cost get another instance?
 - Test Environment → Environment on which they belong

E2E Orchestration Framework Proposal

INPUT

Test-Case 1
Test-Case 2
Test-Case 3
Test-Case 4
Test-Case 5
Test-Case 6
Test-Case 7
Test-Case 8
Test-Case 9
Test-Case 10
Test-Case 11
Test-Case 12
Test-Case 13
Test-Case 14
Test-Case 15
Test-Case 16
Test-Case 17
Test-Case 18

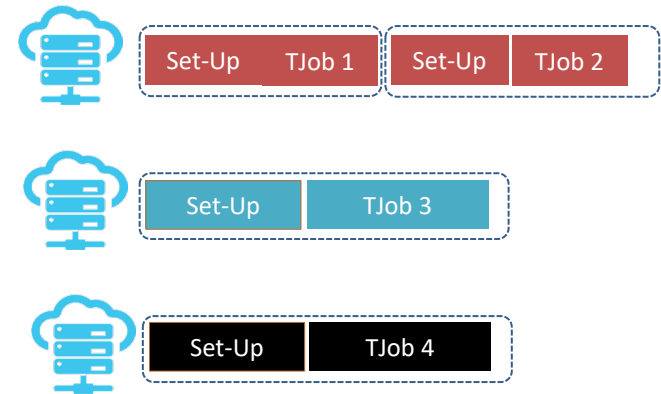


Test-Case 1
Test-Case 2
Test-Case 3
Test-Case 4
Test-Case 5
Test-Case 6
Test-Case 7
Test-Case 8
Test-Case 9
Test-Case 10
Test-Case 11
Test-Case 12
Test-Case 13
Test-Case 14
Test-Case 15
Test-Case 16
Test-Case 17
Test-Case 18



Orchestration:
Grouping
and Scheduling

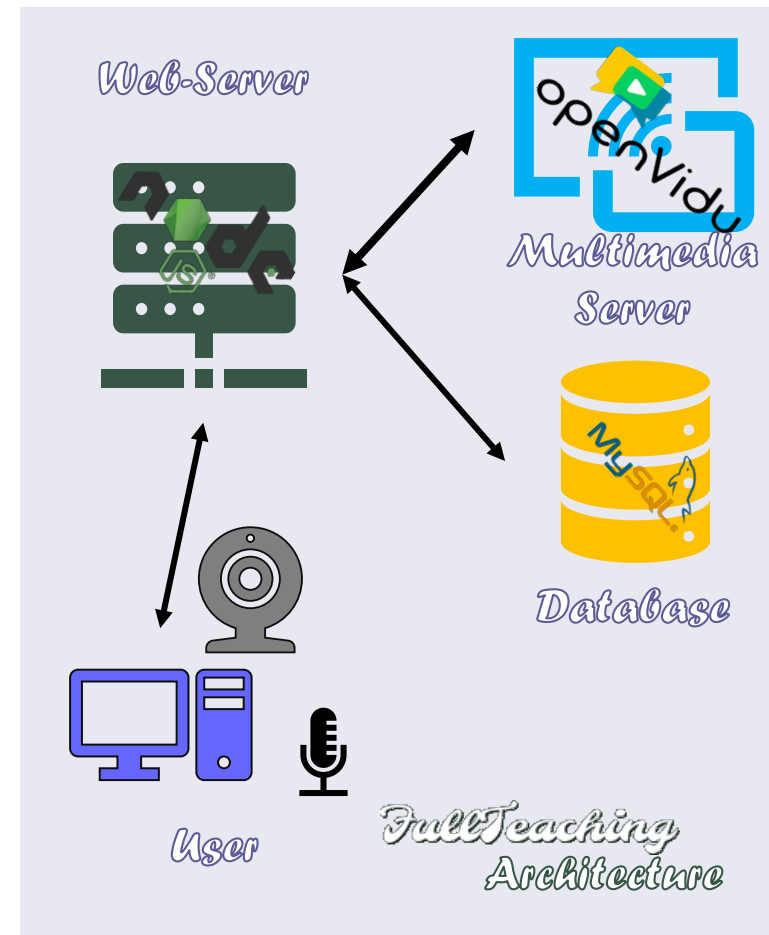
OUTPUT



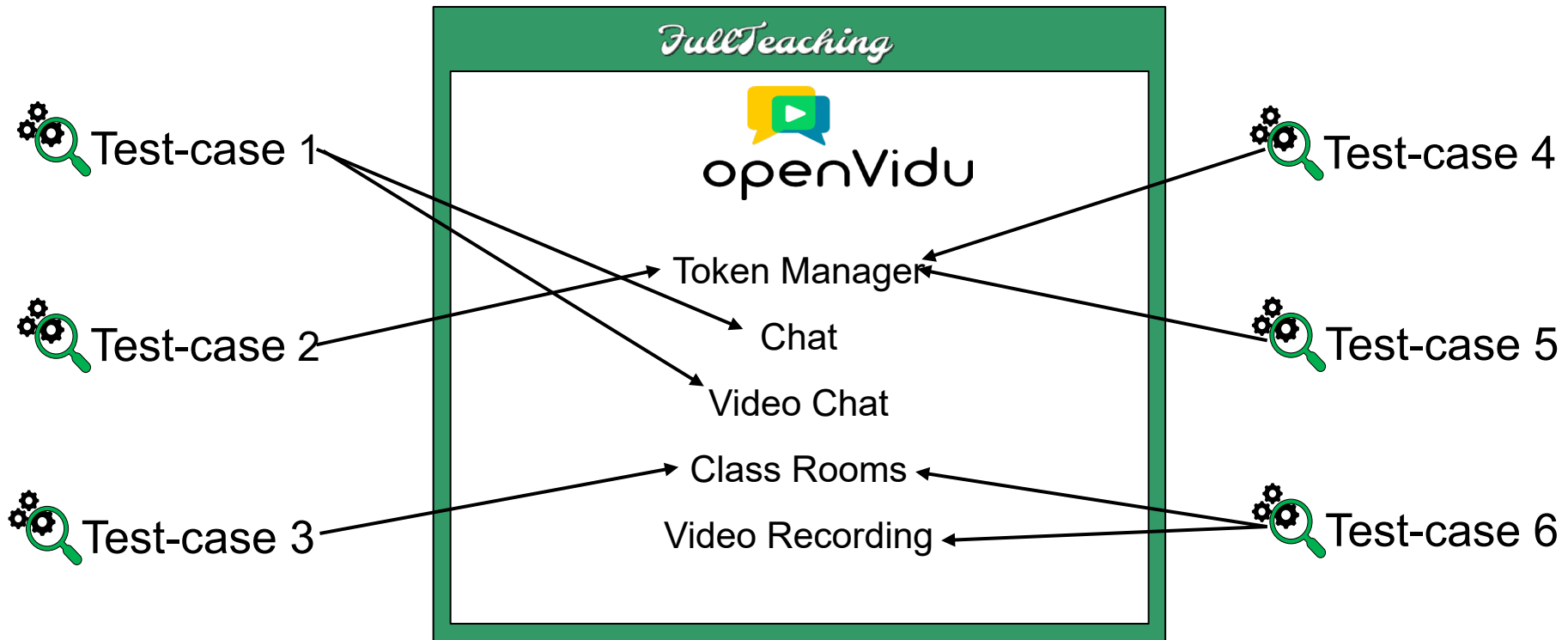
Time →

Example : FullTeaching

- FullTeaching is an educational web platform, available as demo example in Elastest. Is composed by resources as:
 - Databases
 - Multimedia Server
 - Web Servers.

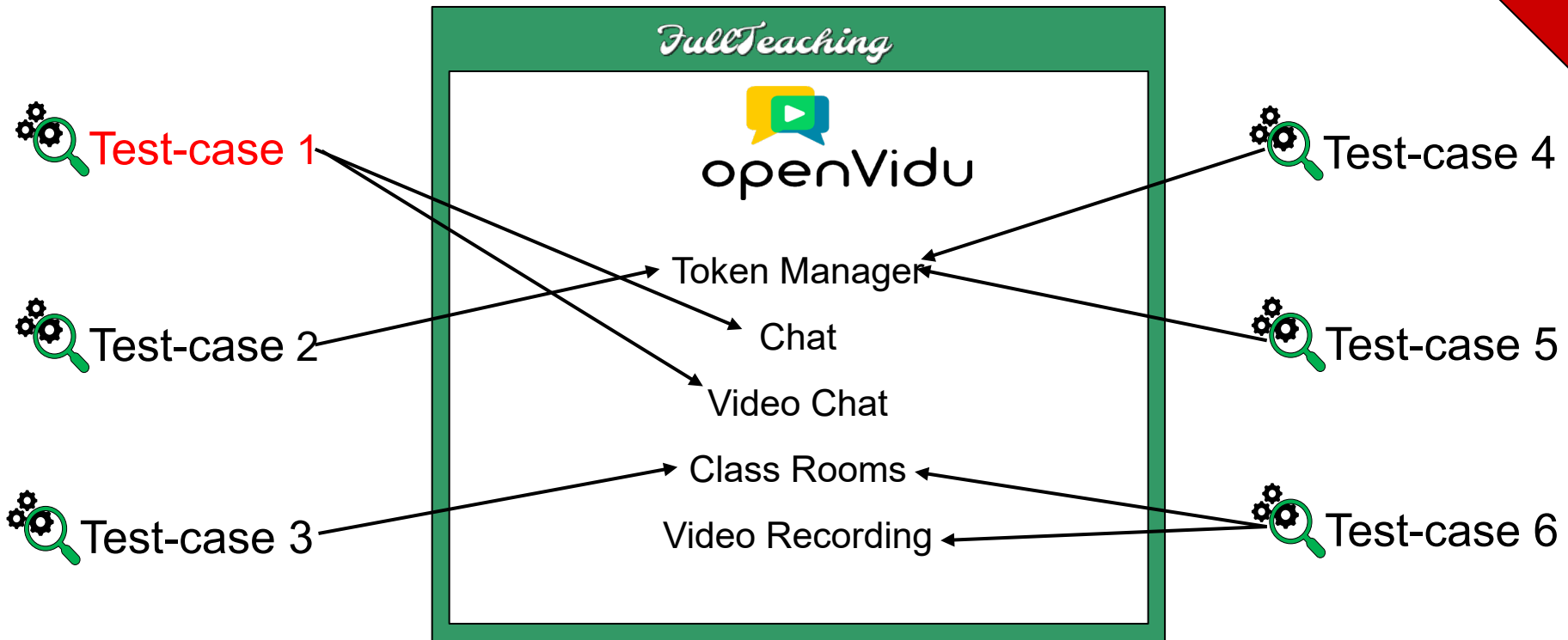


Example: Fullteaching Resource Identification



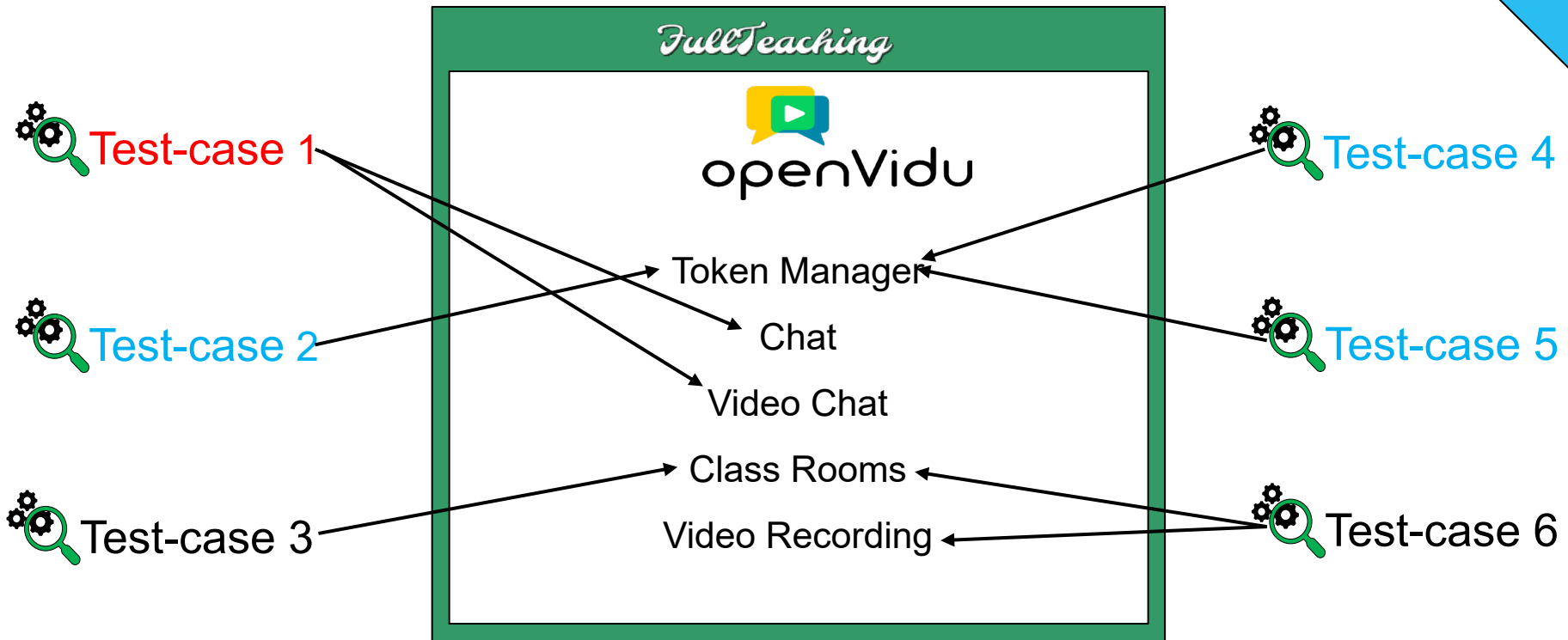
Example: Fullteaching Resource Identification

Medium Openvidu Identification



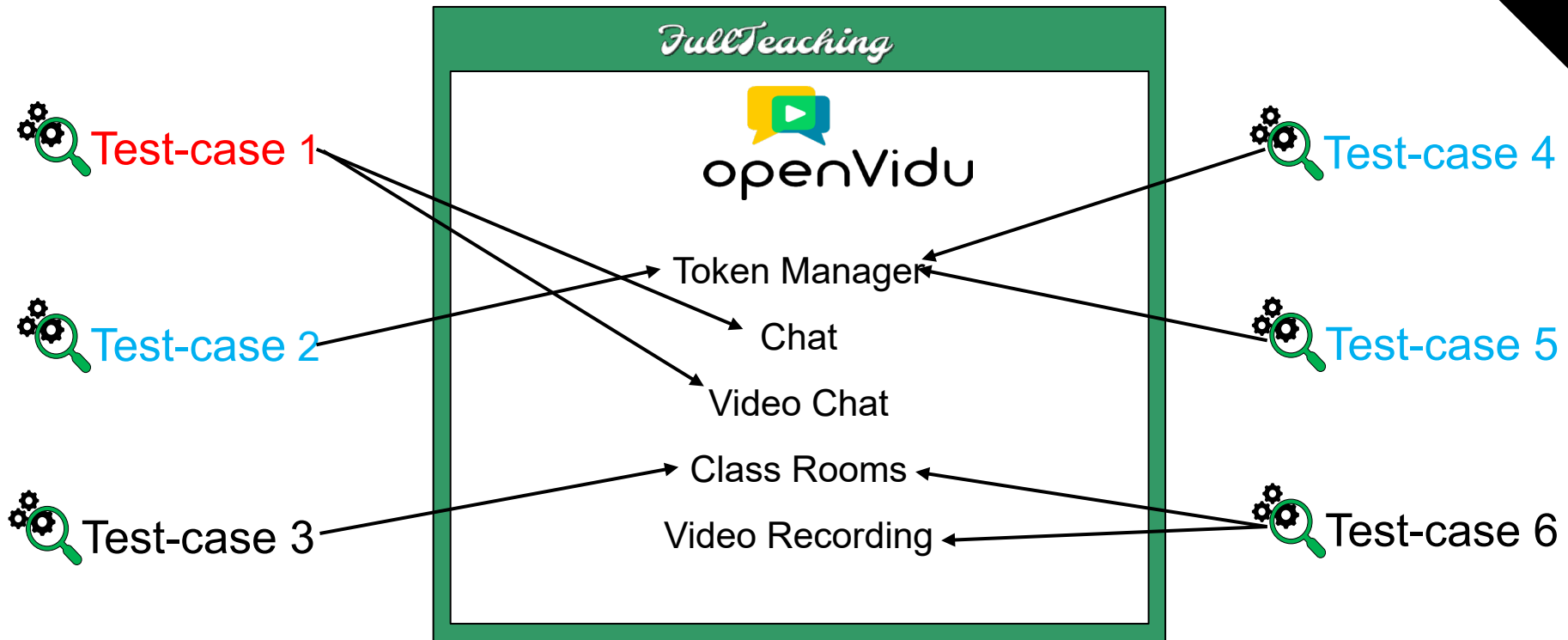
Example: Fullteaching Resource Identification

Light Openvidu Identification

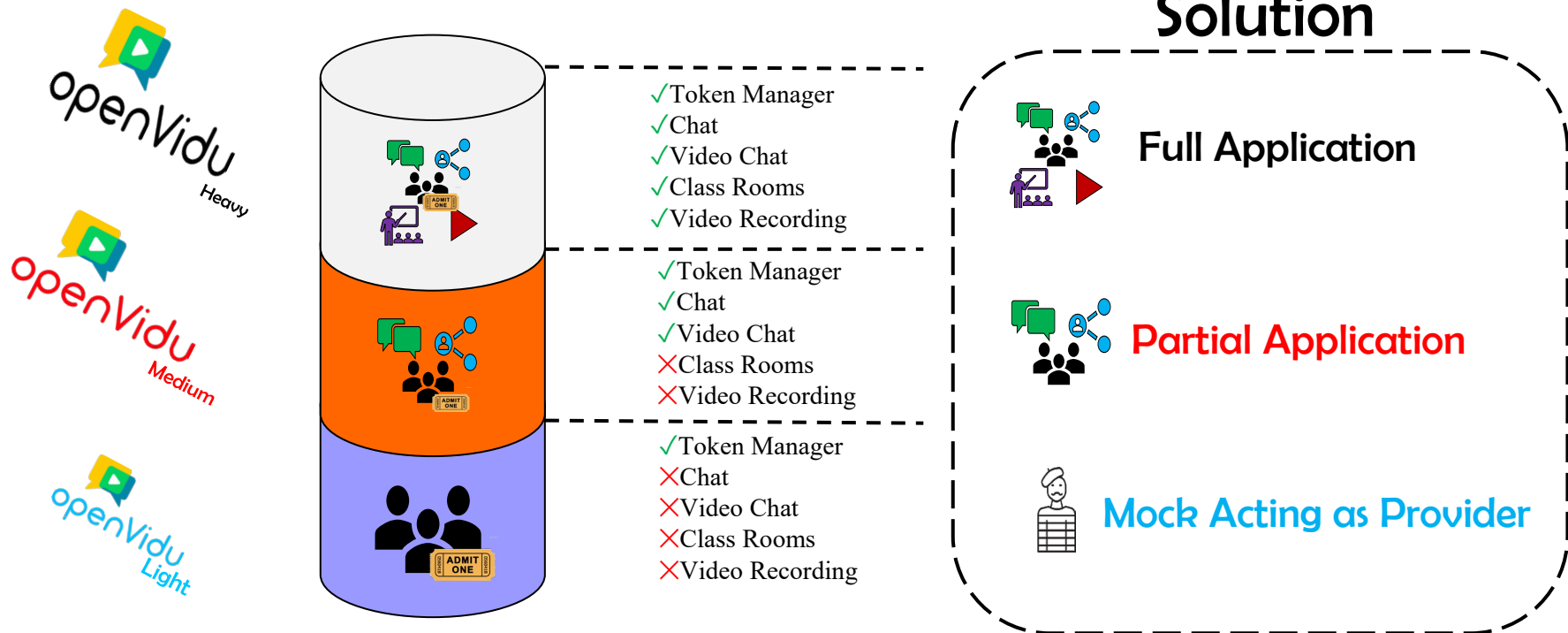


Example: Fullteaching Resource Identification












Heavy Openvidu Identification



Example: Fullteaching Resource Identification



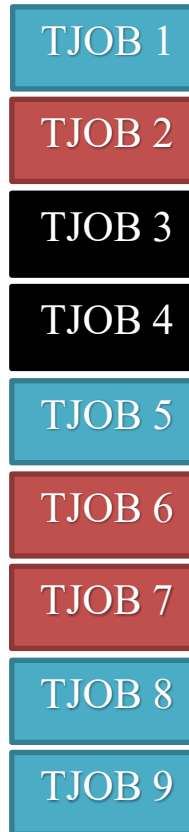
Example: Fullteaching Resource Identification

Type of Server	 openVidu Light	 openVidu Medium	 openVidu Heavy
Elasticity cost	 Not limited	 Not limited	 Only one
Resources Required			
Type of access	 Concurrent	 Sequential	 Sequential

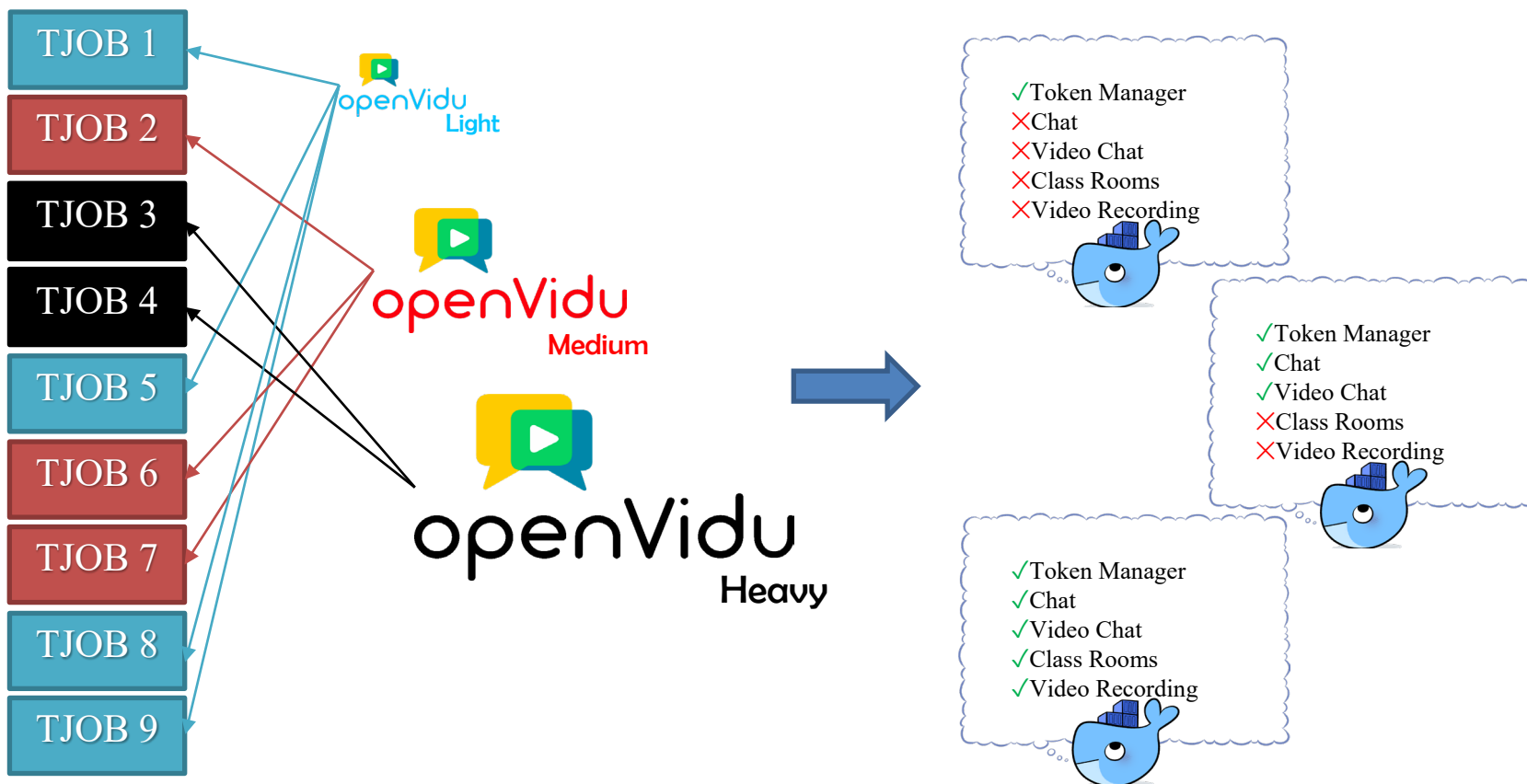
Example : FullTeaching Grouping

INPUT

Test-Case 1
 Test-Case 2
Test-Case 3
 Test-Case 4
Test-Case 5
 Test-Case 6
Test-Case 7
 Test-Case 8
Test-Case 9
 Test-Case 10
Test-Case 11
 Test-Case 12
Test-Case 13
 Test-Case 14
Test-Case 15
 Test-Case 16
Test-Case 17
 Test-Case 18



Example : FullTeaching Grouping



Example : FullTeaching

Several feasible schedules

a

L

1	5	8	9
---	---	---	---

M

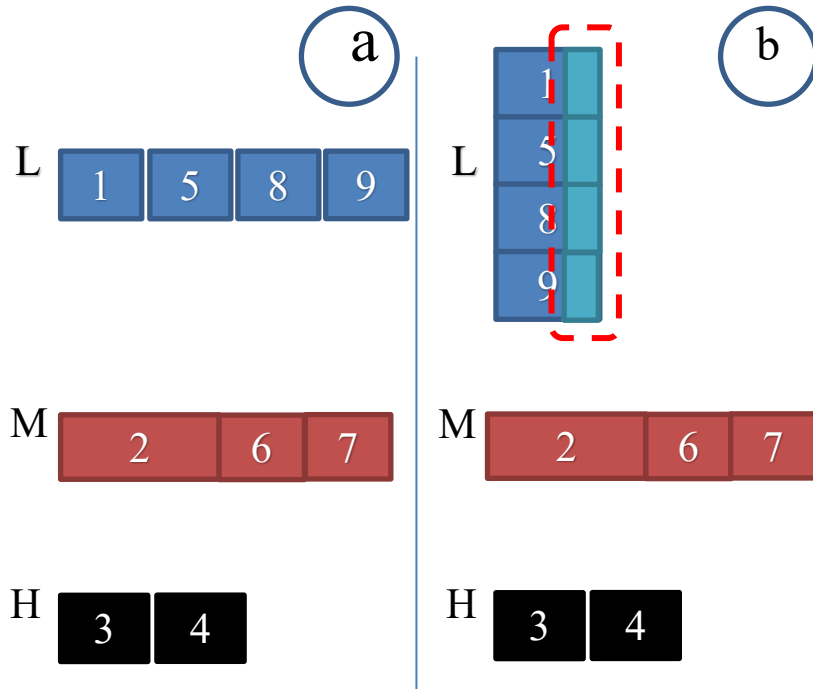
2	6	7
---	---	---

H

3	4
---	---

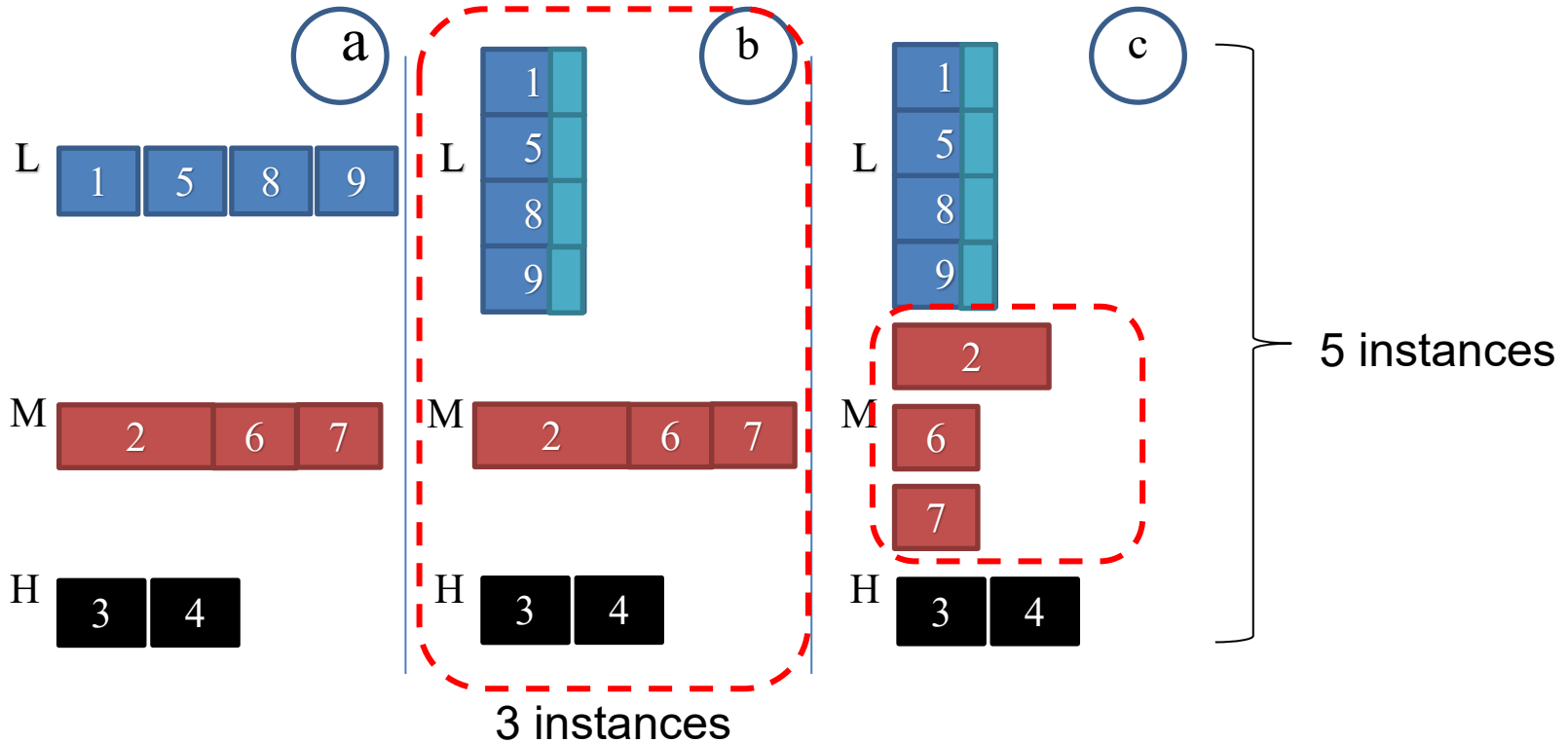
Example : FullTeaching

Several feasible schedules



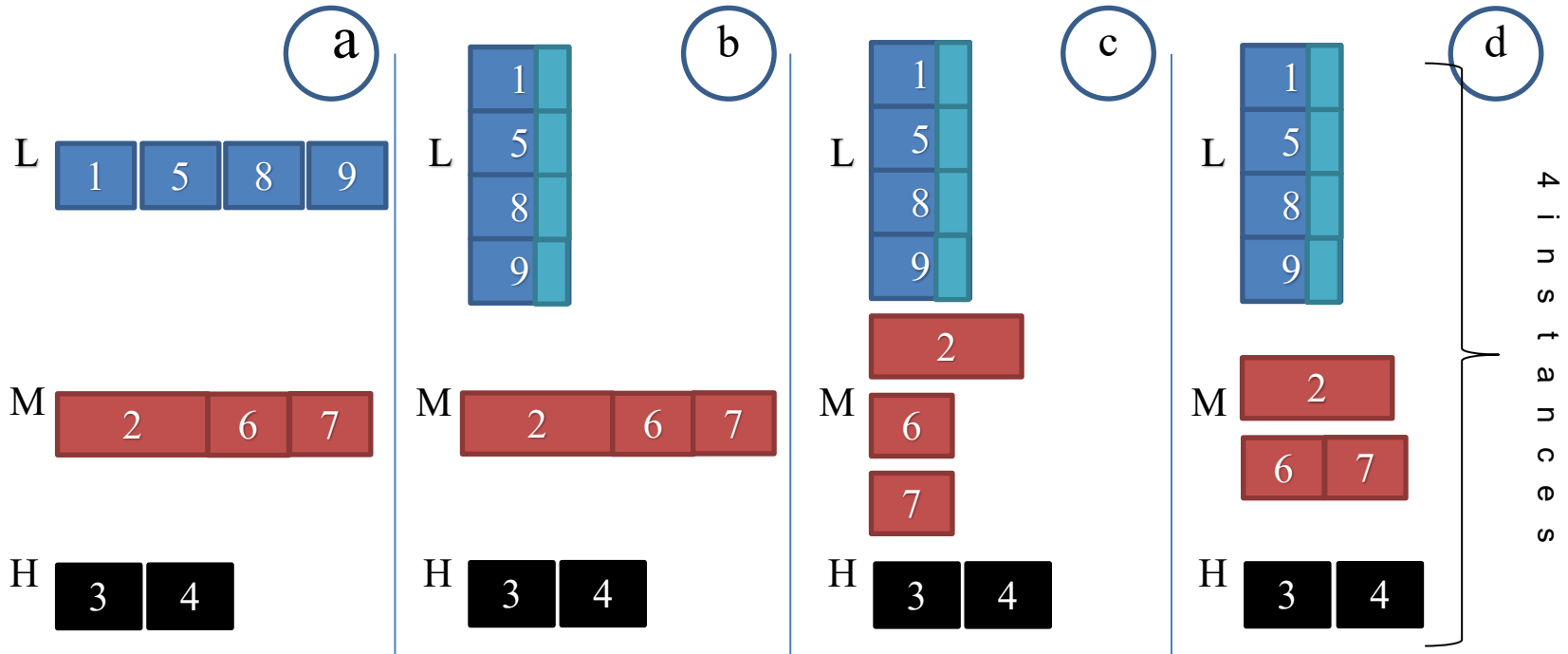
Example : FullTeaching

Several feasible schedules



Example : FullTeaching

Several feasible schedules



Summary

- Through the resource identification, grouping and scheduling, in order to minimize the resources and deploy them in parallel, could optimize the resource usage in several ways.

Future work

■ Future work

- Automation of the resource identification process, detecting the dependencies between tests.
- Develop an orchestration method based on Scheduling and Grouping to optimize the resources employed on E2E testing

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under TestEAMoS (TIN2016-76956-C3-1-R) project and ERDF funds, and by the European Project ElasTest in the Horizon 2020 research and innovation program (GANo. 731535).

Any Question?

Cristian Augusto, Jesús Morán, Antonia Bertolino, Claudio de la Riva and Javier Tuya
Software Engineering Research Group / Software Engineering & Dependable
Computing Laboratory

<http://giis.uniovi.es> / <http://labsedc.isti.cnr.it>



University of Oviedo
&
Istituto di Scienza e Tecnologie
dell'Informazione "A. Faedo"

