



# EGS-CC Software Development Environment Final Presentation



08/12/2015



# Agenda

- Context
- Task 1: SDE Definition
- Task 2: SDE Development and Integration



# Context

# 1



- ➔ The EGS-CC Initiative aims at developing a common European Monitoring & Control infrastructure
  
- ➔ Service oriented and component based architecture providing fundamental functions of a Monitoring and Control system
  
- ➔ Supports the whole lifecycle of the space system
  
- ➔ Can be used at all levels of the space system architecture:
  - › Instrument Development System
  - › SCOE controller
  - › Functional Verification
  - › AIT
  - › Operations



# Context

## EGS-CC Development



- ➔ Large number of stakeholders, integrators and development teams
- ➔ Short schedule (3 years)
- ➔ Intermediate releases in parallel to the development



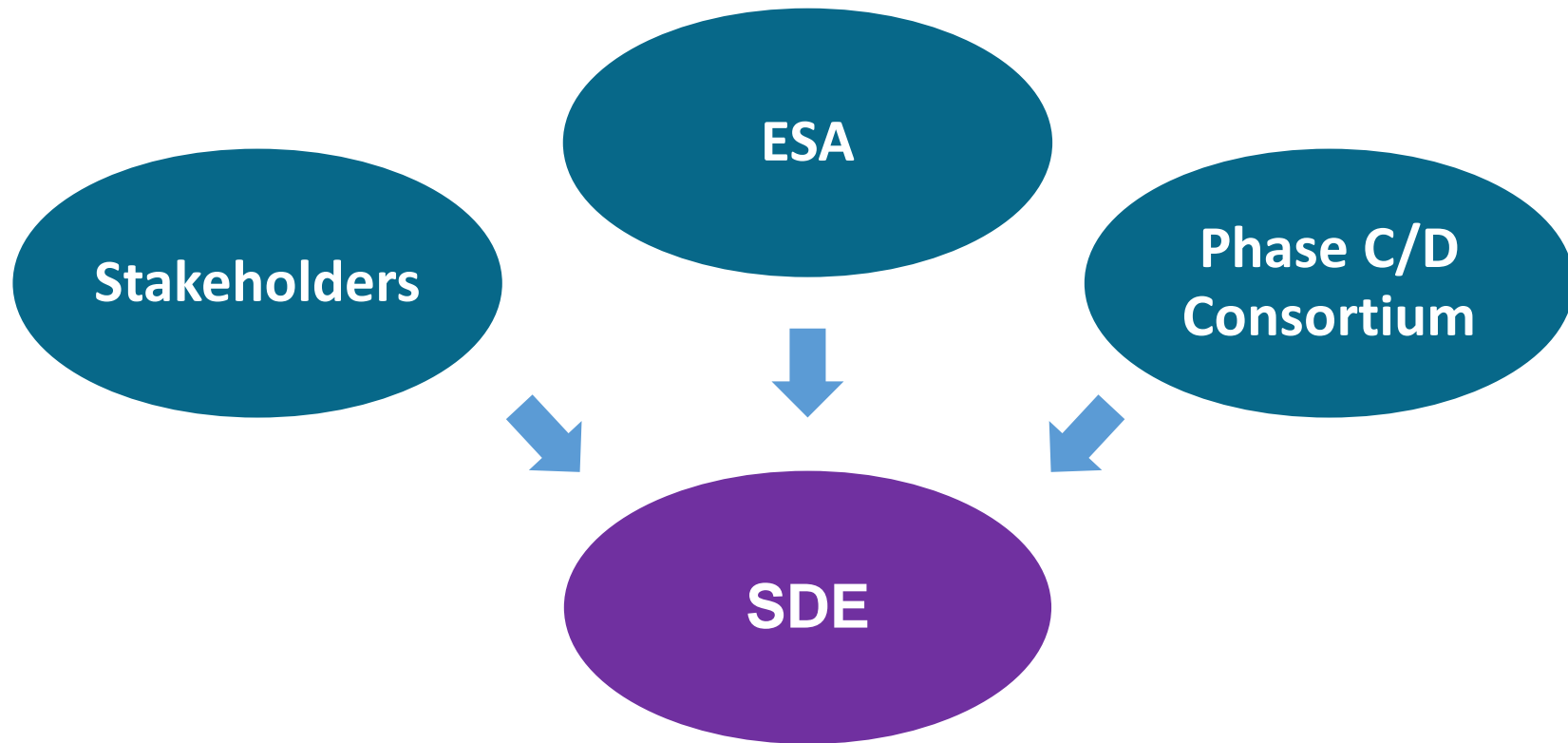
## EGS-CC Development Process

- ➔ Highly incremental development process
  - › Model driven development
  - › Continuous integration of releases by the Integrators
  - › Feedback provided by the Integrators during the integration activities
  - › Close collaboration: Integrators are required to be pro-active
- ➔ Specifically adapted Software Development Environment
  - › Collaboration platform (CSDE)
  - › Integration SDE
  - › Developer SDE
  - › Automation of activities and generation of activities



## Collaborative Environment

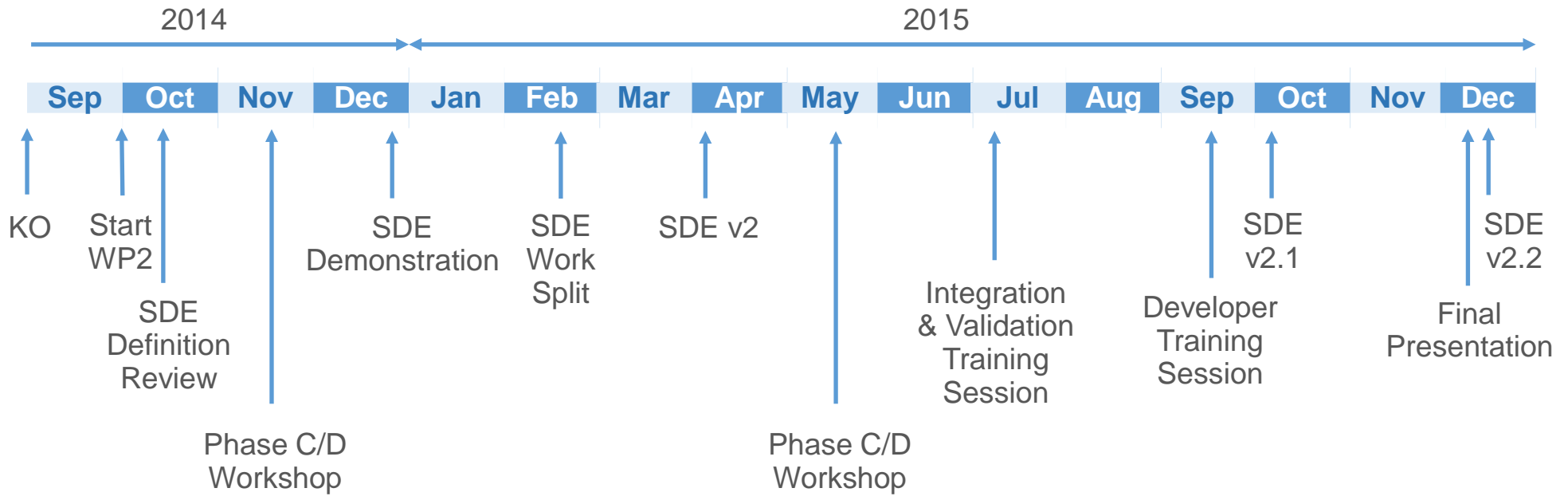
➔ Inputs to the SDE





# Context

## Calendar







# SDE Definition

# 2



# SDE Definition

## → Scope

- › Analyse the required functions and preliminary list of proposed tools
- › Produce a complete list of tools to be integrated in the SDE

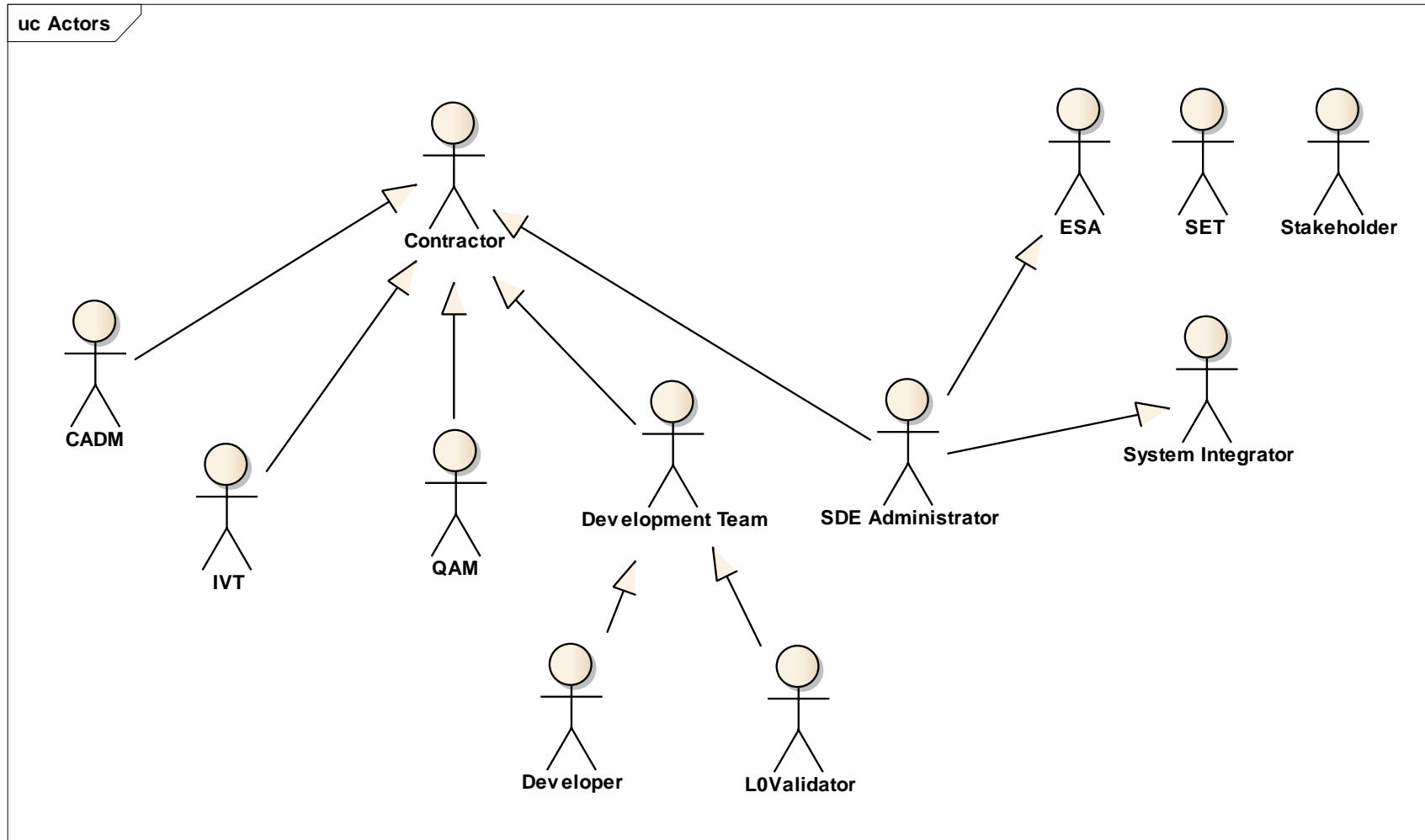
## → Outputs

- › SDE Definition TN:  
EGSCC-SDE-TN-1001-SDEDefinitionDocument



# SDE Definition

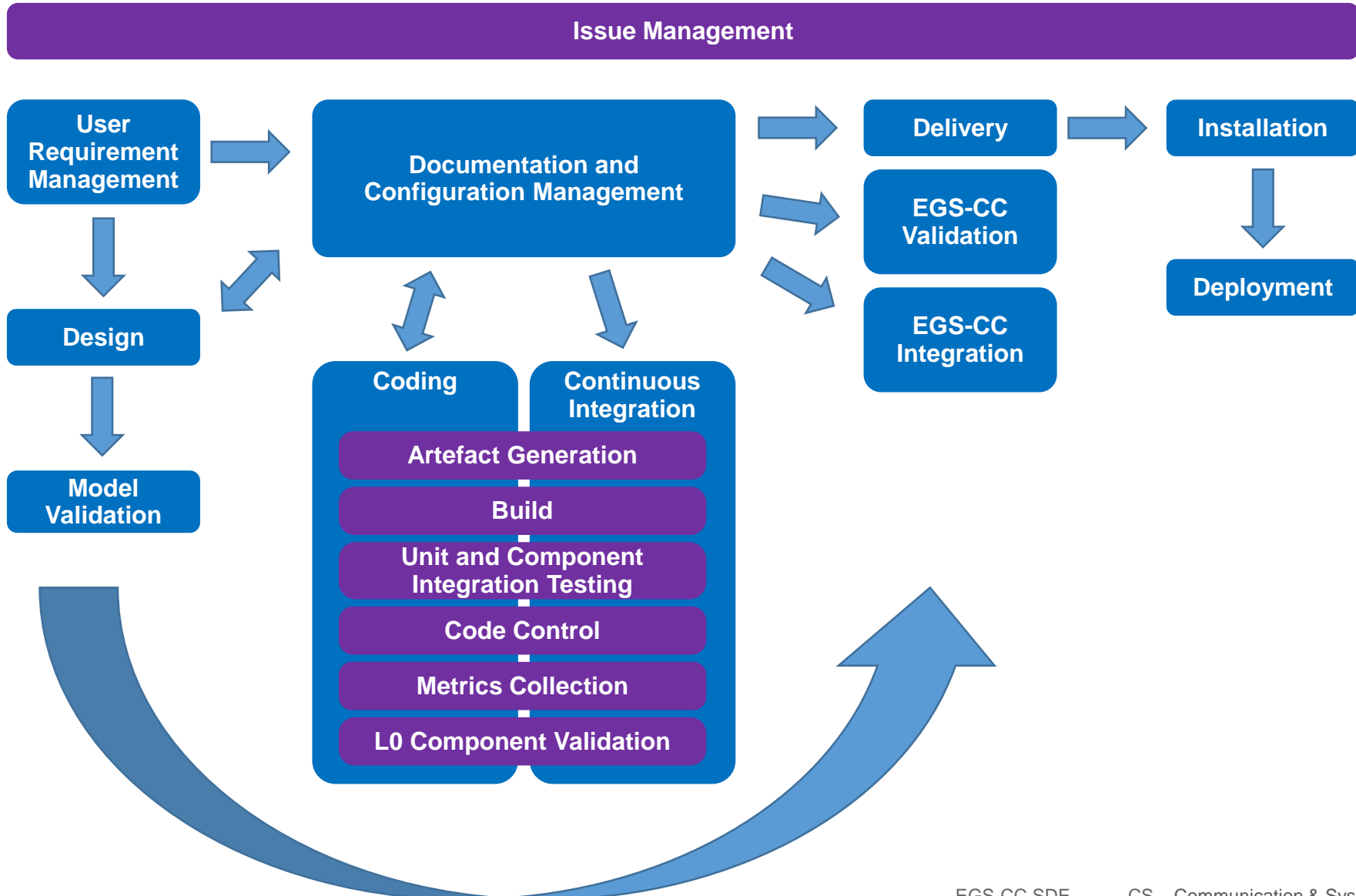
## Actors





# SDE Definition

## Main Use Cases





# SDE Definition

## Final Tools Selection

Function	Tools
User Requirement Management	DOORS
Model Management	MagicDraw
Build	Maven
Coding	Eclipse with plugins
Unit and Component Integration Testing	JUnit, UTF and CTF
Code Control	Checkstyle, JaCoCo
Continuous Integration	Jenkins with plugins, Nexus
Configuration and Document Management	Confluence, FishEye, Git
Validation	Ad-hoc Test Management System, CTF and RTF
Metrics Collection	SonarQube with plugins
Releasing / Delivery	Git
Issue Management	JIRA



## Design Modelling Tool

- ➔ The Phase B Consortium experienced problems with Enterprise Architect
- ➔ The Phase C/D Consortium proposed to migrate to MagicDraw
  
- ➔ Thorough case study performed:
  - › Definition of actors, design workflows, a reference model and deployment strategies
  - › Outputs collected in an annex to the SDE Definition
- ➔ Conclusion:
  - › Enterprise Architect supports poorly the studied use case
  - › The design model is migrated to MagicDraw



# SDE Definition

## Test Management System

- TestLink proposed by the SDE Definition
- No consensus for TestLink
- TestLink was demonstrated in the EGS-CC context
- No clear alternative solutions identified
- SDE delivered with TestLink



# SDE Definition

## Defined deployments

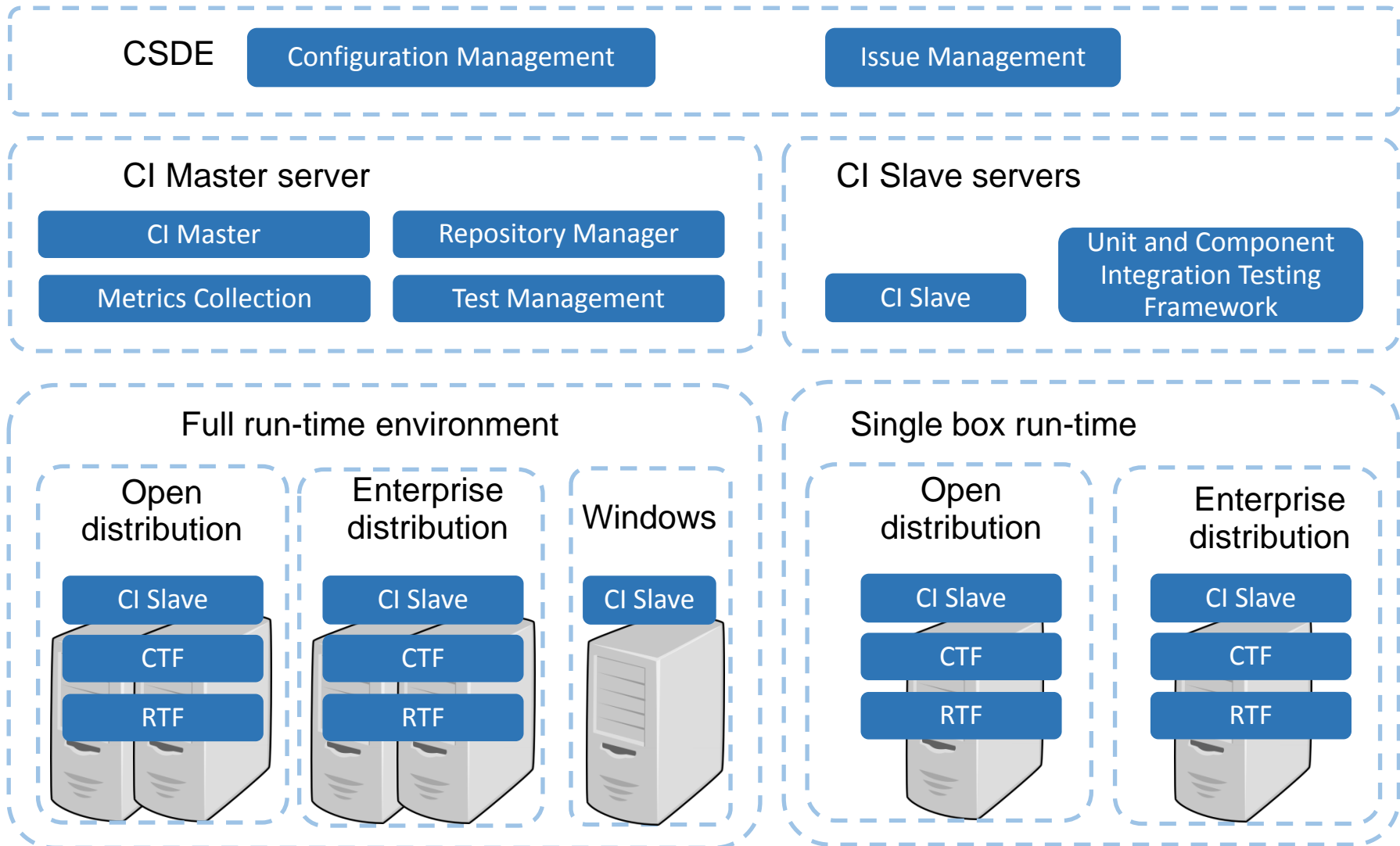
CSDE	Collaboration Tool	SCM Tools	User Requirement Management Tool	Issue Management Tool
EGS-CC Integration and Validation SDE	Continuous Integration Tool	Build Tool	Unit and Component Integration Testing Framework	Delivery Tool
	Design Modelling Tool	Code Control Tool	Test Management System	Installation Tool
				Deployment Tool
L0 Component SDE	Continuous Integration Tool	Build Tool	Unit and Component Integration Testing Framework	
	Design Modelling Tool	Code Control Tool	Test Management System	





# SDE Definition

## Integration and Validation set-up





# SDE Development and Integration

3



# SDE Development and Integration

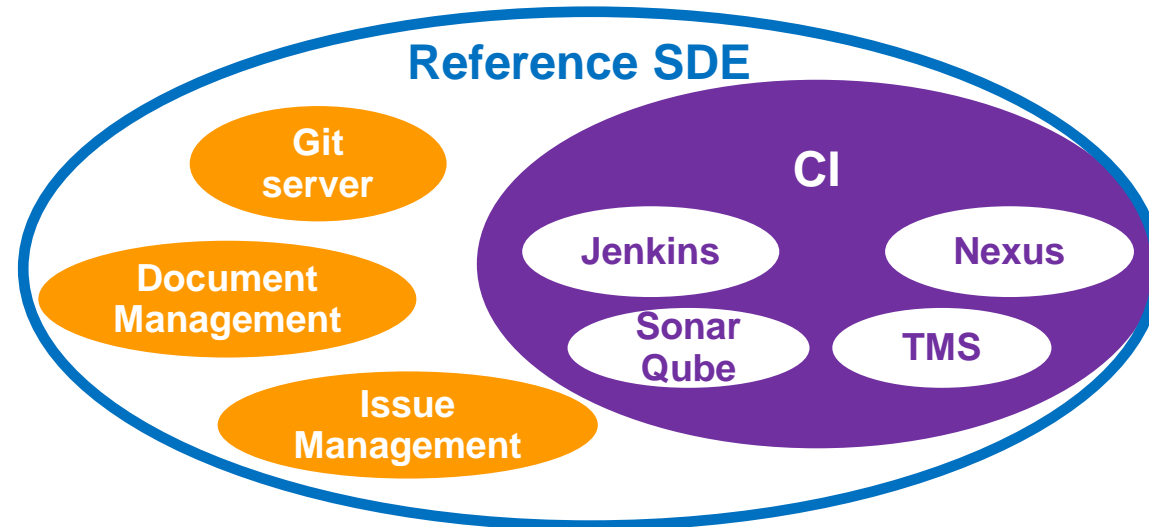
## Task 2: Work Packages

- ➔ WP1: SDE Development and Integration
- ➔ WP2: SDE Training
- ➔ WP3: Phase C/D SDE Support

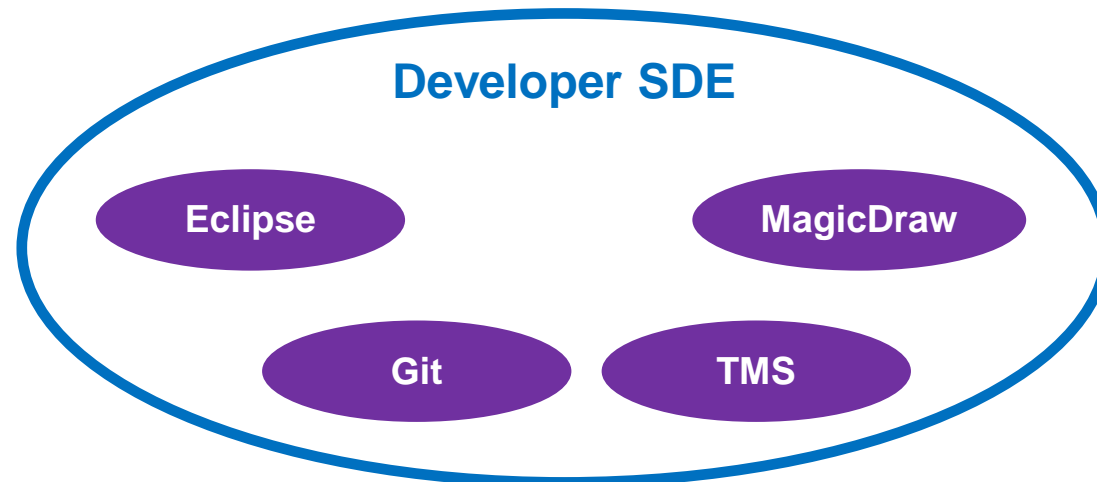


## Deployments

➔ Server side



➔ Developer

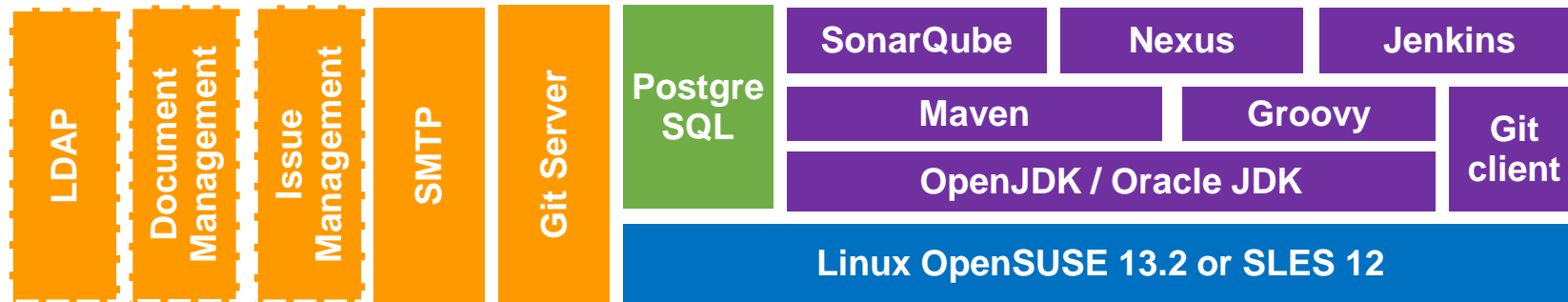




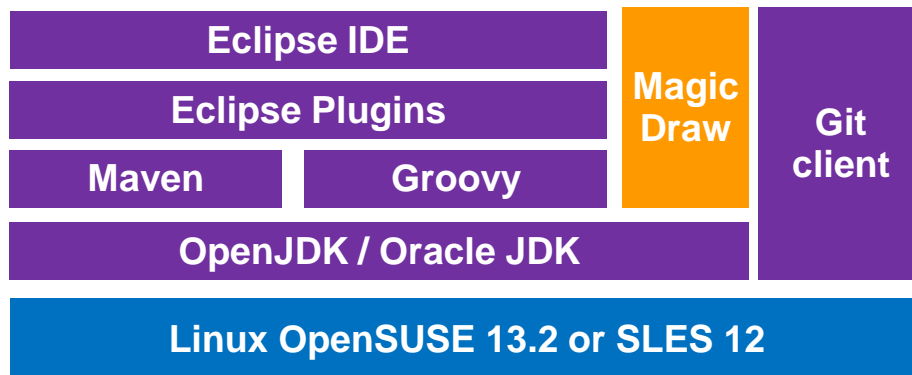
# WP1: SDE Development and Integration

## Technology stacks

### ➔ Server side



### ➔ Developer



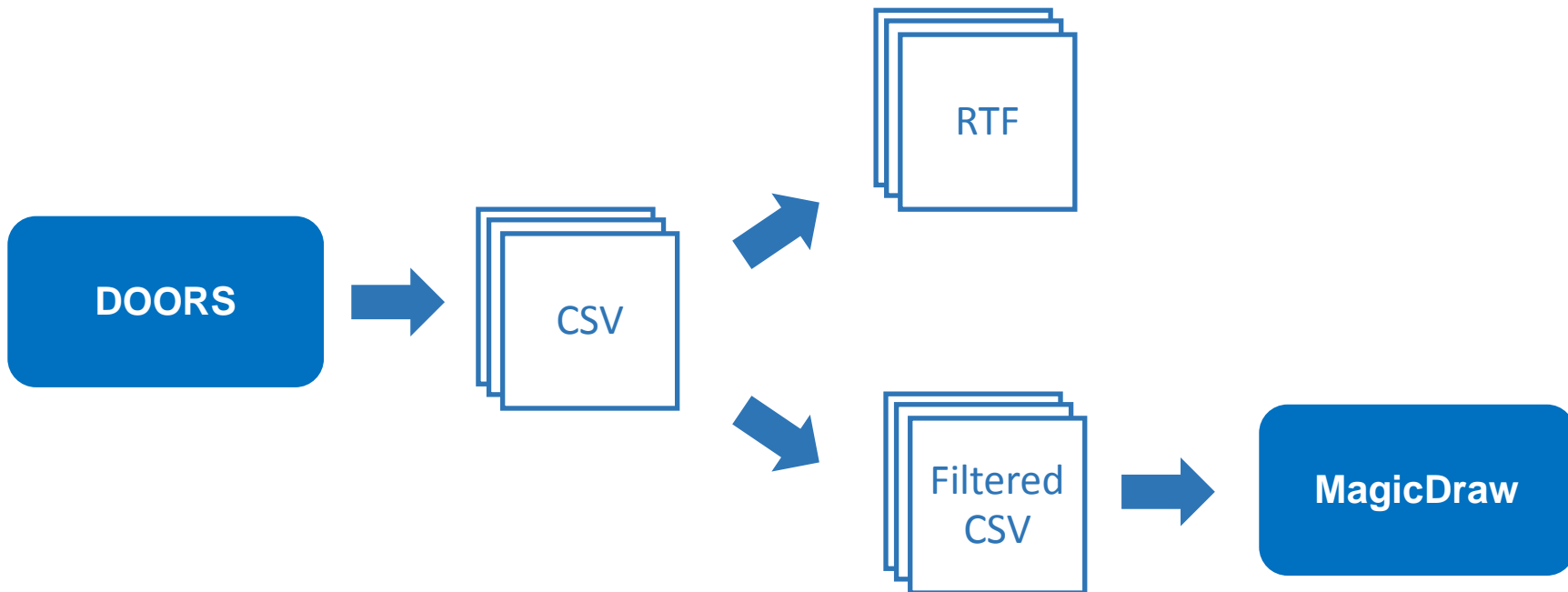
#### Legend





## User Requirements Management

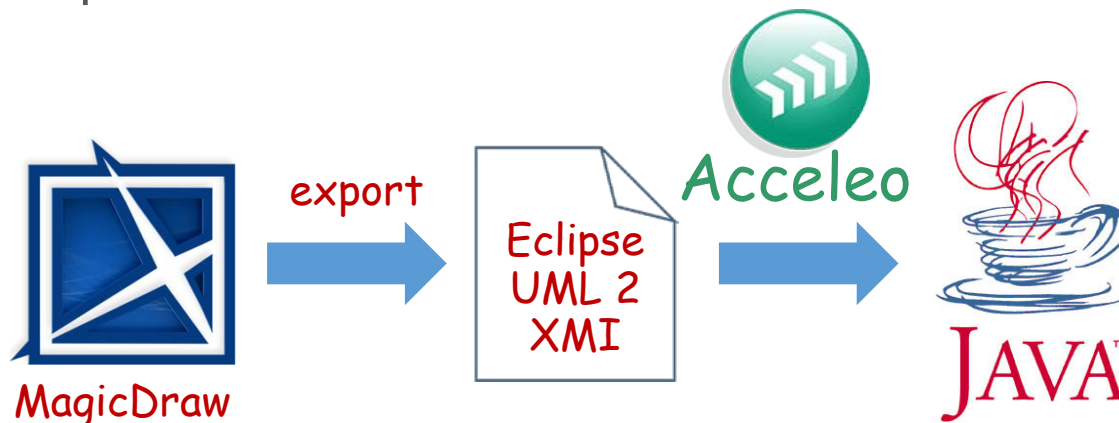
- ➔ User Requirements Export (based on DOORS feature)
- ➔ User Requirements Document Generator





## Design / Model Management

- ➔ User Requirements Import
- ➔ Document Generation (SSDD, ICD, SVR ...)
- ➔ Source Code Generation of the APIs
  - › MagicDraw export to EMF XMI
  - › Generation with Eclipse Acceleo
    - Java Source Code
    - pom.xml files





## Build

### ➔ Maven build process

- › Integrates the source code generator (APIs)
- › Builds source code
- › Runs unit tests
- › Computes code coverage by the tests
- › Enforces the coding standards
- › Generates test report (SUTP)
- › Generates Javadoc
- › Generates OSGi bundles
- › Runs component integration tests
- › Runs component validation tests
- › Deploys the built artifacts

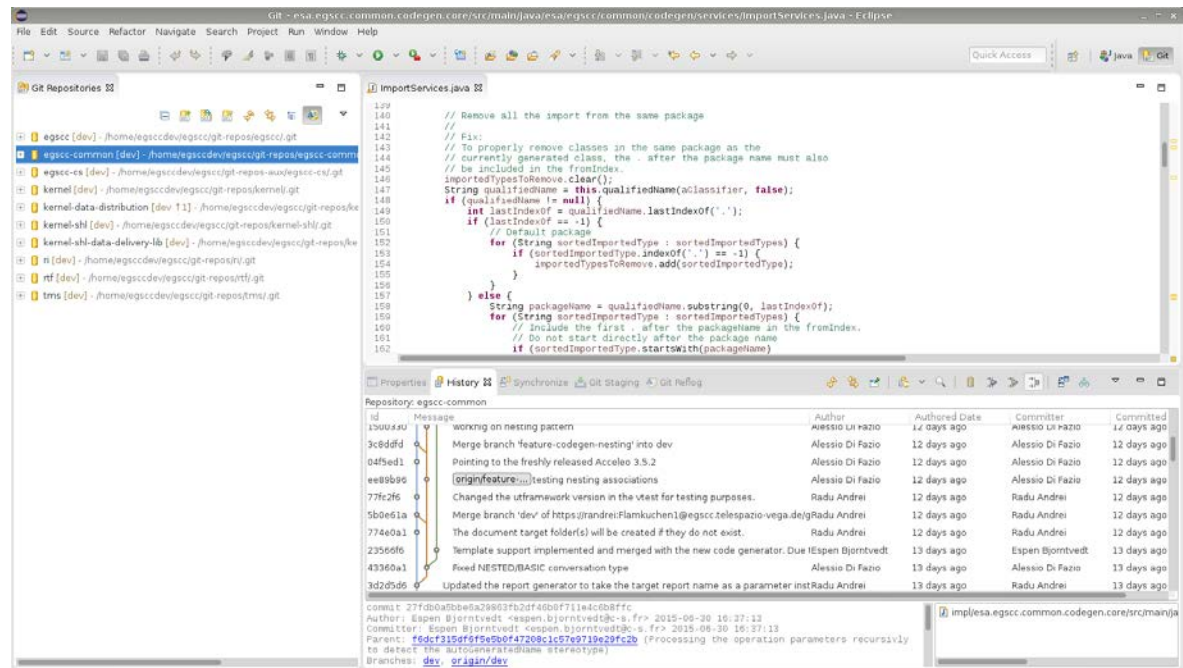




# WP1: SDE Development and Integration

## Coding

- ➔ IDE based on Eclipse with plugins
- ➔ All common activities are available
  - › API and Documentation access
  - › Coding
  - › Build
  - › Code Control
  - › Code Coverage
  - › Testing and Validation
  - › SCM





## Unit and Component Integration Testing

### → Unit Testing

- › Based on standard Maven conventions
- › Based on JUnit and Mockito
- › Examples provided

### → Component Integration Testing

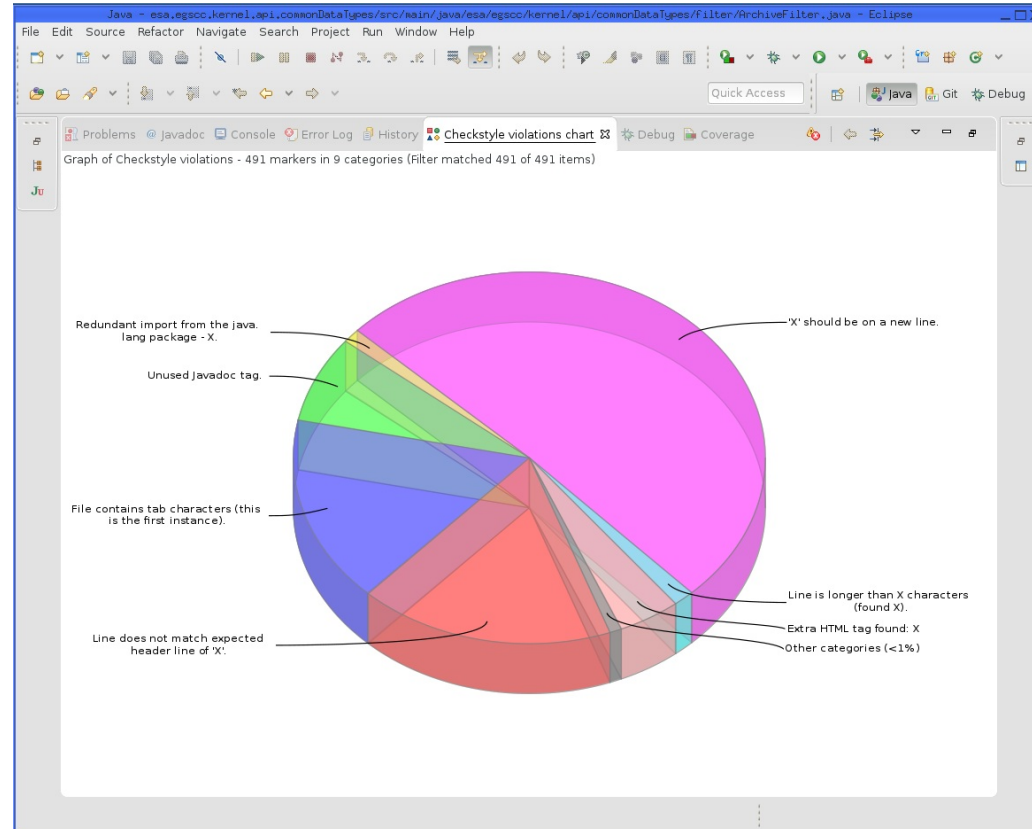
- › Based on standard Maven conventions
- › Based on JUnit, Pax Exam, Mockito and doubles
- › Examples provided



# WP1: SDE Development and Integration

## Code Control

- ➔ Defined set of coding standards and rules
- ➔ Integrated in the IDE
- ➔ Violations highlighted and collected



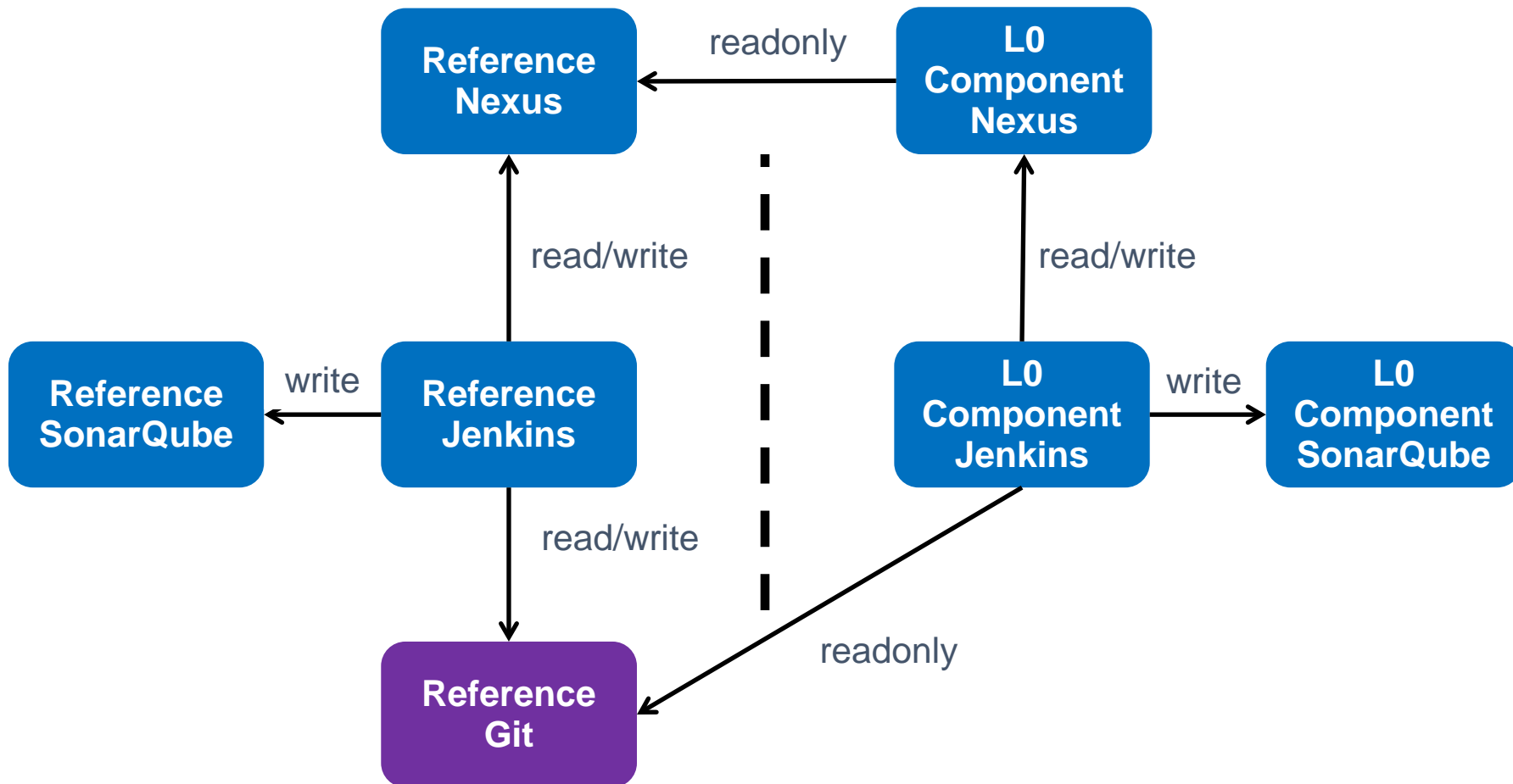


# WP1: SDE Development and Integration

## Continuous Integration – Deployment

Reference SDE

L0 Component SDE





## Continuous Integration

- ➔ Jenkins integrated with plugins (Git, Groovy, JaCoCo, SonarQube ...)
- ➔ Definition of workflows
  - › Build, Test, Code Coverage, Code Control, Validation
  - › Several maturity stages are covered
    - Development (dev branch)
    - Releases (rc-\* and rel-\* branches)
- ➔ Result distribution
  - › Maven artifacts
  - › Reports

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">dev-kernel-data-distribution_build</a>	3 hr 5 min - <a href="#">#86</a>	N/A	4 min 0 sec
		<a href="#">dev-kernel-data-distribution_build_all</a>	1 hr 14 min - <a href="#">#65</a>	N/A	5 min 13 sec
		<a href="#">kernel-data-distribution_syncWithGit</a>	14 hr - <a href="#">#104</a>	N/A	20 sec
		<a href="#">rc-kernel-data-distribution_build_rc-0.1.0</a>	N/A	6 min 56 sec - <a href="#">#4</a>	1 min 25 sec
		<a href="#">rc-kernel-data-distribution_build_rc-0.2.1</a>	6 min 43 sec - <a href="#">#3</a>	13 days - <a href="#">#2</a>	3 min 15 sec
		<a href="#">rc-kernel-data-distribution_build_template</a>	N/A	N/A	N/A



# WP1: SDE Development and Integration










## Configuration and Document Management

- ➔ Subversion initially proposed for the SDE
- ➔ Git preferred by the Phase C/D Consortium
  
- ➔ Git is selected as it is supported by the CSDE
- ➔ Different product structures are proposed by CS
- ➔ Final solution:  
Flat structure with independent repositories for each L0



## Configuration and Document Management

### → EGS-CC Product Organisation – L0 Component

Folder name	Description
 api	Generated interfaces
 conf	Configuration
 doc	System / subsystem / L0 documentation
 doubles	Test double bundles
 impl	Implementation bundles
 itest	Integration test bundles
 model	Architectural design model
 poms	High level pom.xml files
 vtest	Validation test bundles



## Configuration and Document Management

- ➔ Definition of release workflow
- ➔ Adapted to the Phase C/D Consortium's organisation
- ➔ Versioning scheme:
  - › [Major].[Sprint].[Hotfix]
  
- ➔ Confluence
  - › Document Access / Release
  - › Document Reference Numbers Assignment
  - › Document Templates





## Validation

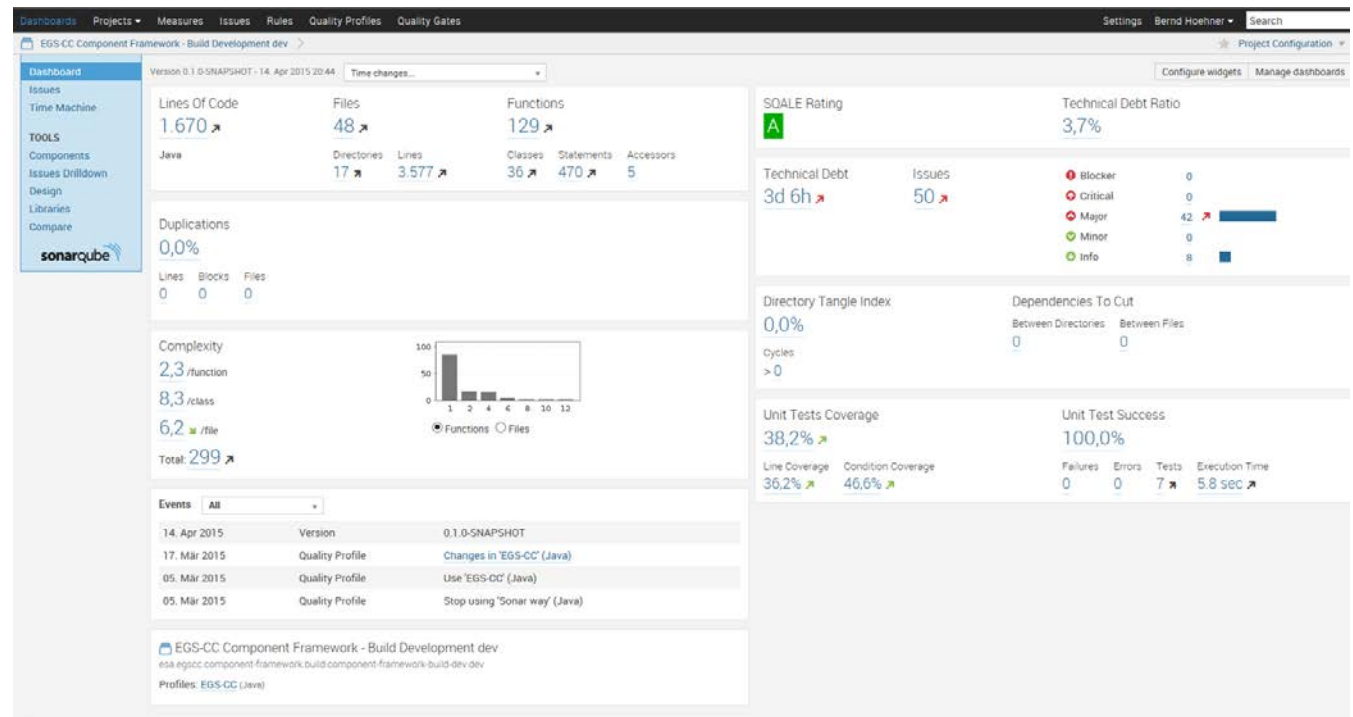
- L0 Component Validation
- EGS-CC Integration
- EGS-CC Validation
  
- TestLink proposed as Test Management System (TMS)
- Import procedures, workflows and document generators implemented and demonstrated
  
- Ad hoc development chosen by the Phase C/D Consortium



# WP1: SDE Development and Integration

## Metrics Collection

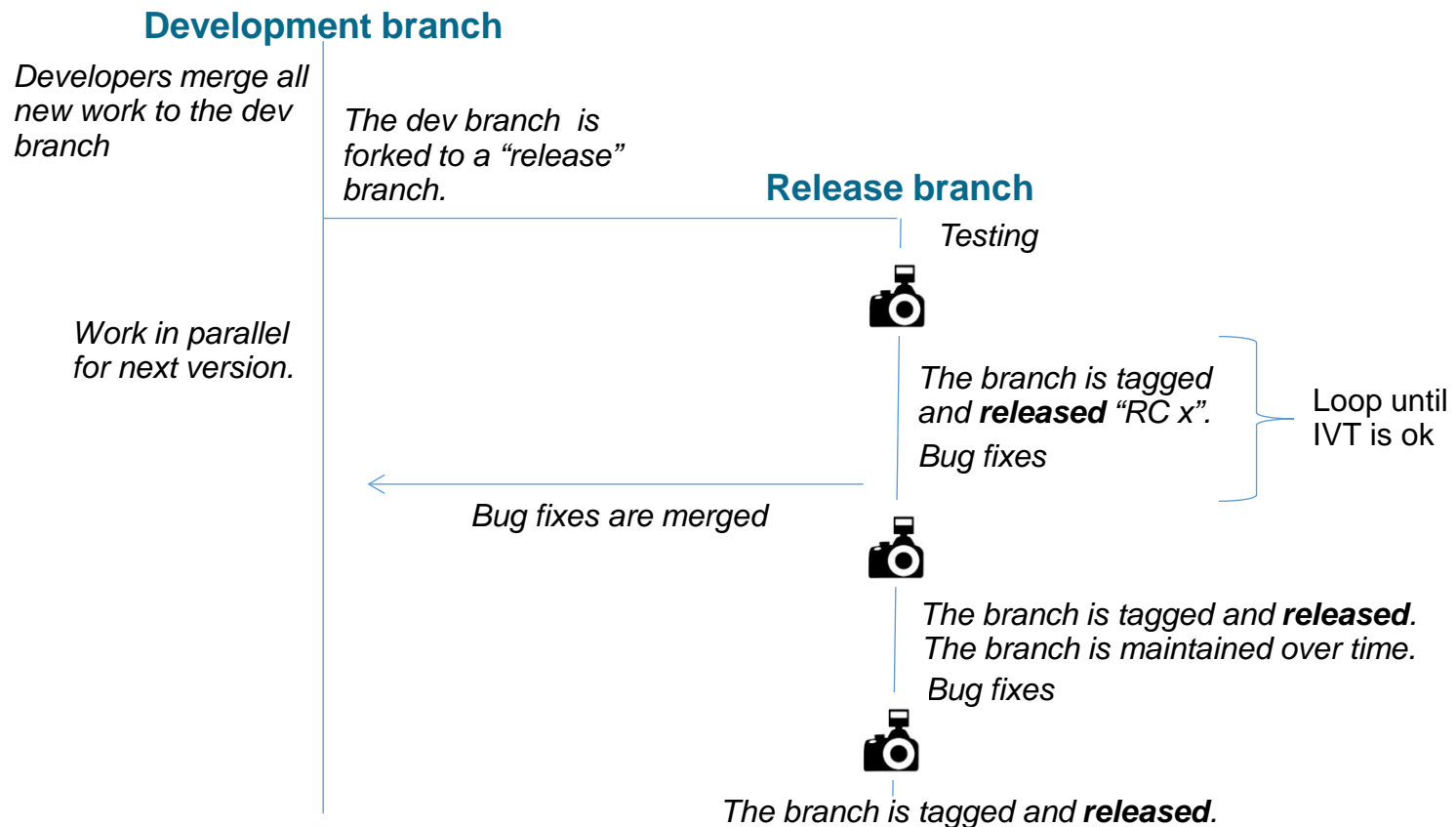
- ➔ SonarQube installed and integrated
- ➔ EGS-CC coding standards and thresholds implemented
- ➔ Part of the Continuous Integration process





## Releasing / Delivery

- ➔ Release process based on the Maven release plugin
- ➔ Delivery process based on Maven plugins and the Repository Manager





## Issue Management

### ➔ Multiple projects defined in CSDE JIRA

- › EGS-CC Actions (EA) and EGS-CC Actions – Contractual (EAC) for actions
- › EGS-CC Issues split in different projects for CRs (EGSCR), NCRs (EGSNCR) and SPRs (EGSSPR)
- › EGS-CC Contractor Issues (EC) for general issues that are not actions and that should not be made generally public, i.e. only visible to the Phase C/D Consortium and ESA

### Browse Projects

Recent Projects	EGS-CC		
EGS-CC			
No Category			
All Projects			
<b>EGS-CC</b>			
Project	Key	Project Lead	
EGS-CC Actions	EA	Juan María Carranza	
EGS-CC Actions - Contractual	EAC	Juan María Carranza	
EGS-CC Contractor Issues	EC	Juan María Carranza	
EGS-CC Issues - CR	EGSCR	Juan María Carranza	
EGS-CC Issues - NCR	EGSNCR	Juan María Carranza	
EGS-CC Issues - SPR	EGSSPR	Alexander Heim	
EGS-CC Sandbox	ES	Alexander Heim	

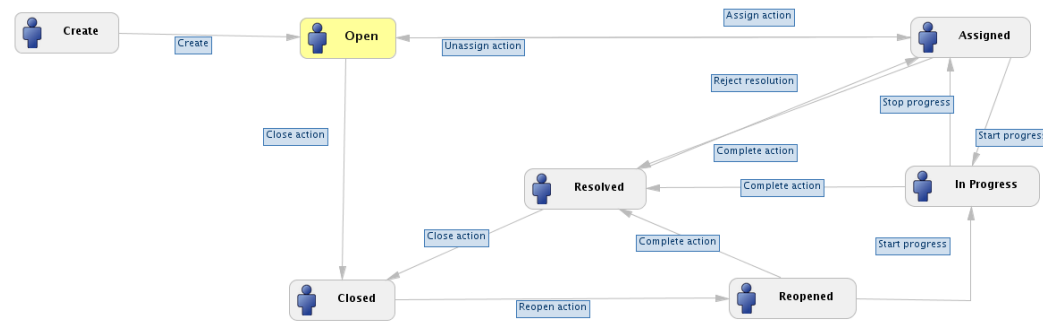


# WP1: SDE Development and Integration

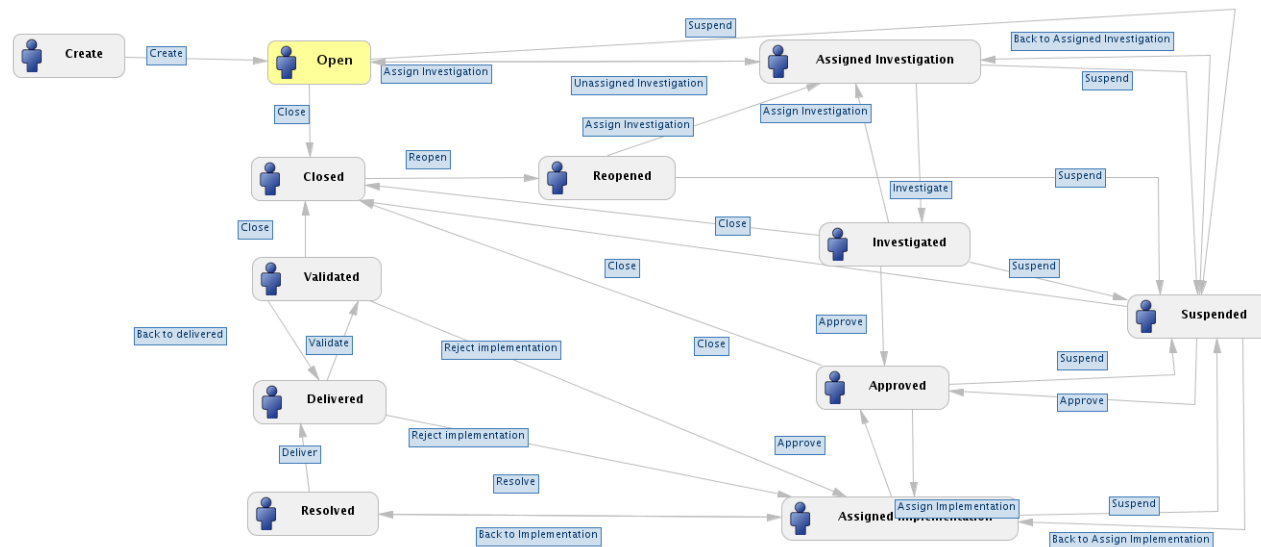
## Issue Management

➔ Dedicated workflows have been defined for the projects

➔ Action workflow



➔ SPR workflow





# WP1: SDE Development and Integration

## Lessons Learned

- ➔ Frequent technical meetings to present propositions and collect feedback
- ➔ Agile practices adopted to respond quickly to feedback



## WP2: SDE Training

### → Tasks

- › Prepare the training material
- › Give the training sessions

### → Two different training sessions:

- › SDE for integration and validation activities (8<sup>th</sup>/9<sup>th</sup> July in Toulouse)
- › SDE for development activities (14<sup>th</sup>/15<sup>th</sup> September in Darmstadt)



# Phase C/D SDE Support

- ➔ SDE workshops
- ➔ Code generation
  - › Adapting the code generator to the modified runtime environment and implementation of specific EGS-CC service stereotypes
- ➔ User Requirements Import
  - › Definition and integration of the User Requirements Import into the MagicDraw design model from DOORS





# Outputs

- ➔ SDE Definition Document
  - › EGSCC-SDE-TN-1001-SDEDefinitionDocument
- ➔ SDE User Manuals
  - › EGSCC-SDE-SUM-1002-CollaborativeSoftwareDevelopmentEnvironment
  - › EGSCC-SDE-SUM-1003-CSDEExtensions
- ➔ Phase C/D Consortium feedback TN
  - › EGSCC-SDE-TN-1002-SDEPhaseCDConsortiumFeedback
- ➔ SDE Usage Training Materials
- ➔ Final Report
  - › EGSCC-SDE-TN-1003-FinalReport
- ➔ Contract Closure Summary
  - › EGSCC-SDE-ContractClosureSummary
- ➔ Source code and configuration items



# Thank You

## People Involved

### ➔ SDE Development Team

- › Antoine Barthe
- › Christophe Pipo
- › Espen Bjorntvedt
- › Guilhem Bonnefille
- › Jeremy Costes
- › Juan F. Prieto
- › Laurent Cocault

### ➔ Different stakeholders

### ➔ ESA

- › Anthony Walsh
- › Jakob Livschitz
- › Juan María Carranza
- › Mauro Pecchioli

### ➔ Phase C/D Consortium

- › Alessio Di Fazio
- › Annabell Langs
- › Anthony Crowson
- › Bernd Höhner
- › Martin Götzelmann
- › Peter Ellsiepen



# Q&A

CS SI

5 Rue Brindejone des Moulinais  
31506 Toulouse Cedex 5  
France

Tel.: +33 (0)5 61 17 66 66

[uk.c-s.fr](http://uk.c-s.fr)