



ERICSSON

# TESTING IN A LARGE SCALE AGILE DEVELOPMENT ENVIRONMENT

# CONTENT/AGENDA



1. Introduction
2. Agile on a large scale
3. Testing in a large scale agile environment
4. Conclusion

# INTRODUCTION



- › Ericsson:
  - The leading vendor for mobile network infrastructure
- › Tough competition on the telecommunication market:
  - Number of competitors
  - Merging of traditional Circuit Switched Technology and Packet Switched Technology towards all IP Network
- › Motivation:
  - Increase the feedback loops towards design
  - Delivery of high quality products to the customer
  - Being flexible to adapt faster to the market's demand
  - Early and Frequent Delivery



# AGILE ON A LARGE SCALE

# AGILE



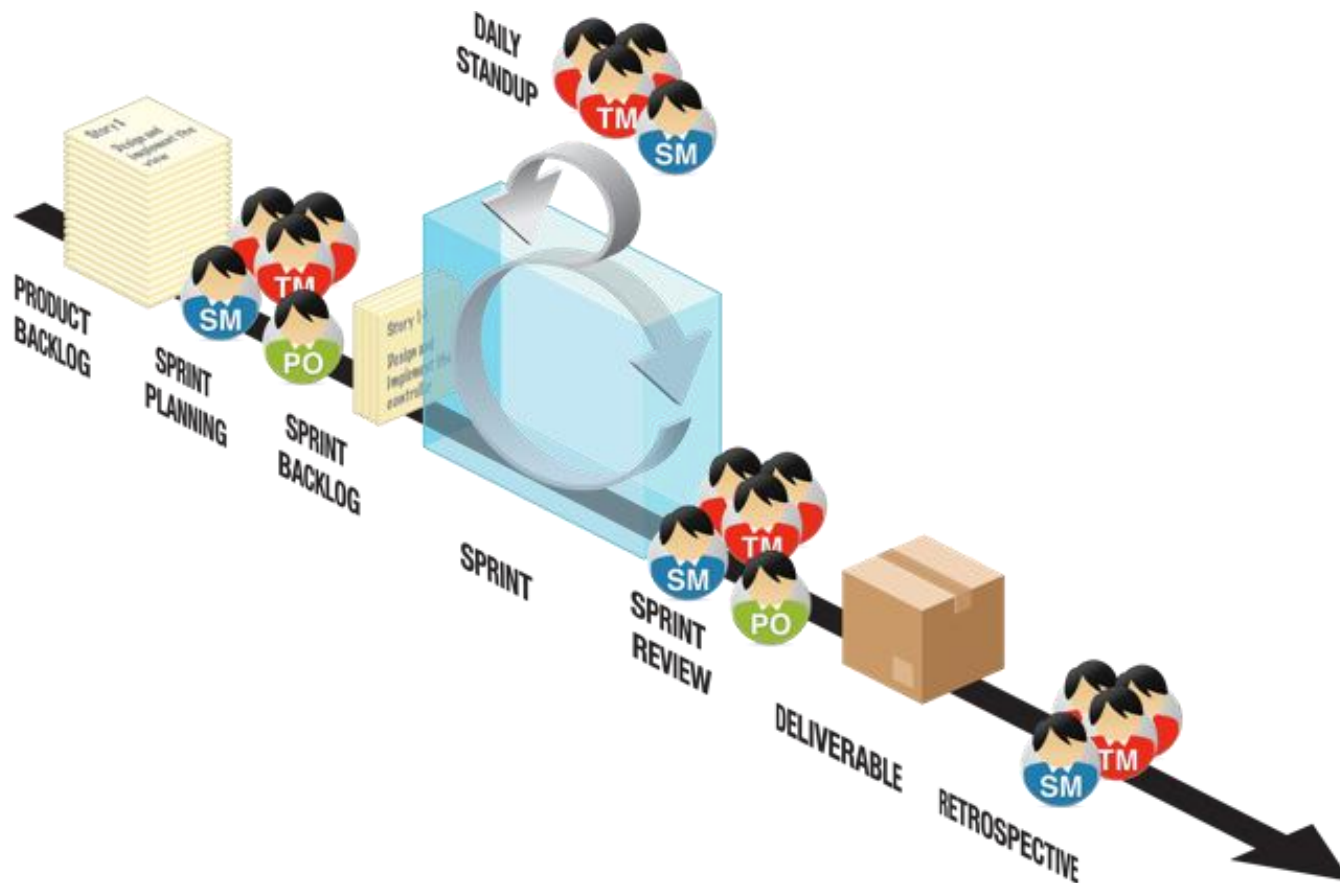
## › **Manifesto for Agile Software Development**

- › We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
  
- › That is, while there is value in the items on the right, we value the items on the left more.

# SCRUM IN A NUTSHELL



- › One team collaborating to one feature



# LARGE SCALE



- Chief Product Owner: 1
- Number of Product owners: 5
- Number of Scrum teams: 8
- Size of the teams: 6-8
- Number of Scrum Masters: 8
- Number of people: around 70
- Number of locations: 4
- Number of Releases: 2
- Number of manhours: ~100000
- Number of User Stories: ~700



# LARGE SCALE



- › Our legacy: The Ericsson Mobile Soft Switch
- › 8085042 PLEX statements
- › 20212606 ASA instructions corresponding to approximately 95000000 Intel instructions
- › 2196 SW blocks





# TESTING IN A LARGE SCALE AGILE ENVIRONMENT

# WAYS OF WORKING

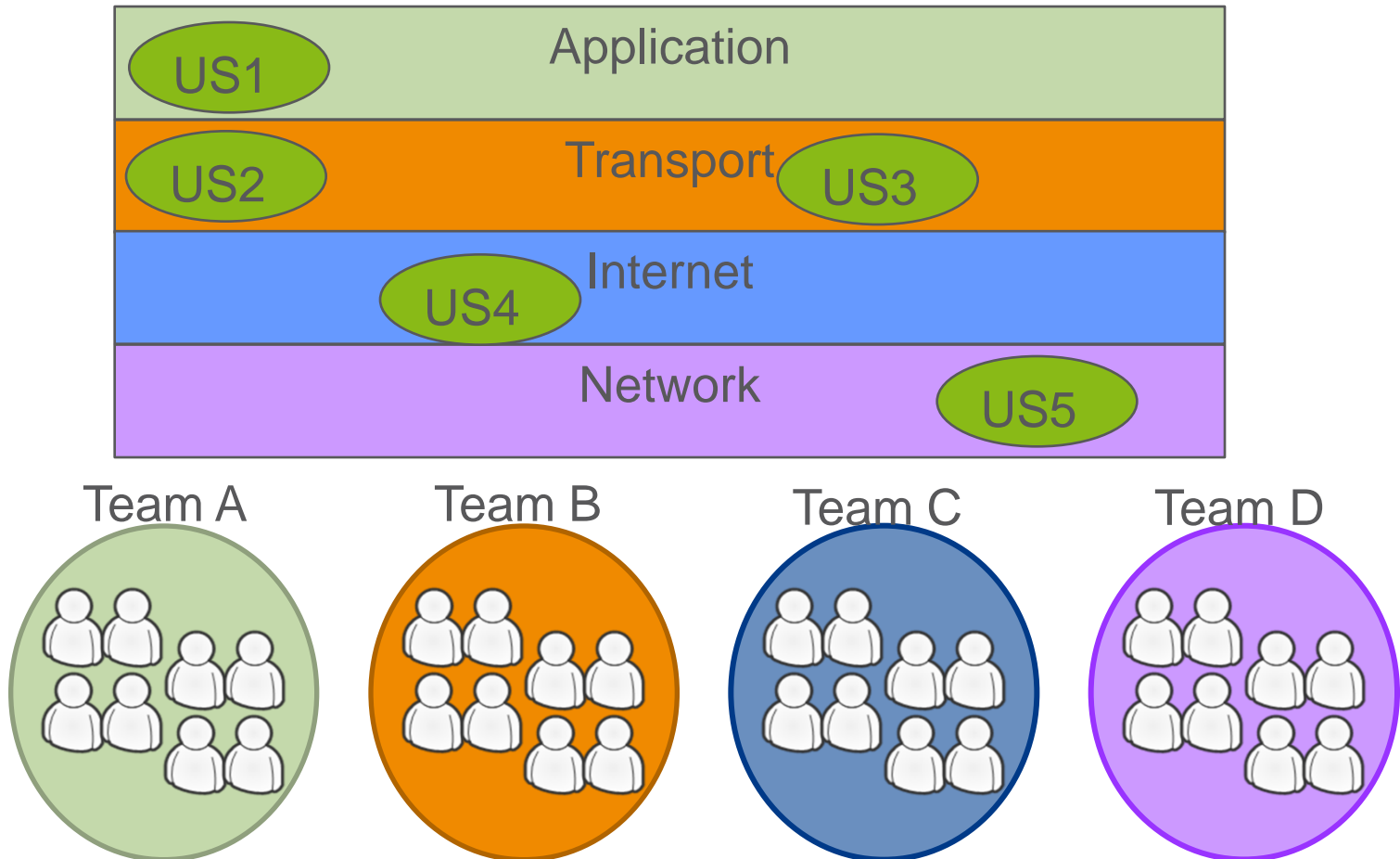


- › Cross functional team of 7 +/- 1 SW developers:
- › Virtual teams
- › Synchronized sprints inside a feature
- › Common Backlog
- › Common ceremonies to keep everybody informed

# WAYS OF WORKING



Backlog Setup strictly based on functional areas

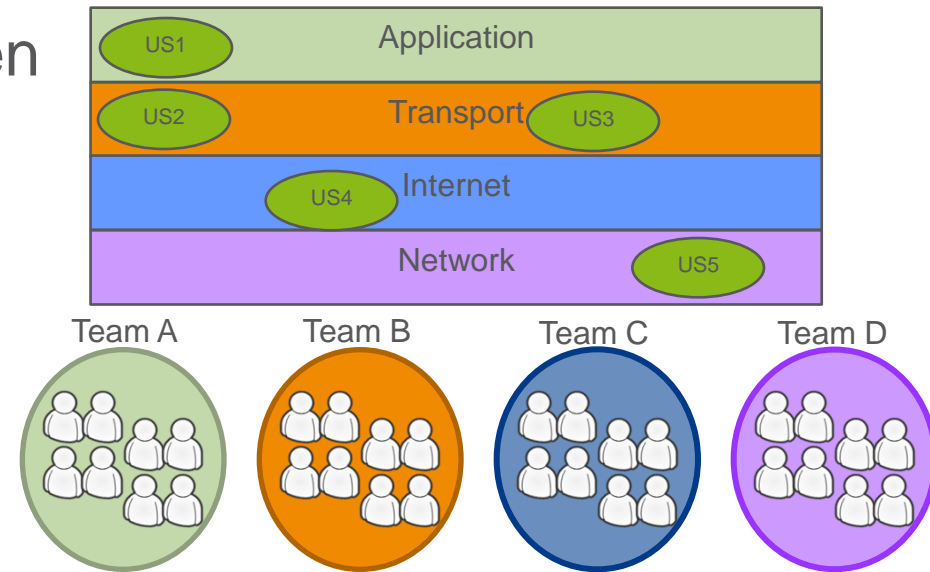


# WAYS OF WORKING



Backlog Setup strictly based on functional areas

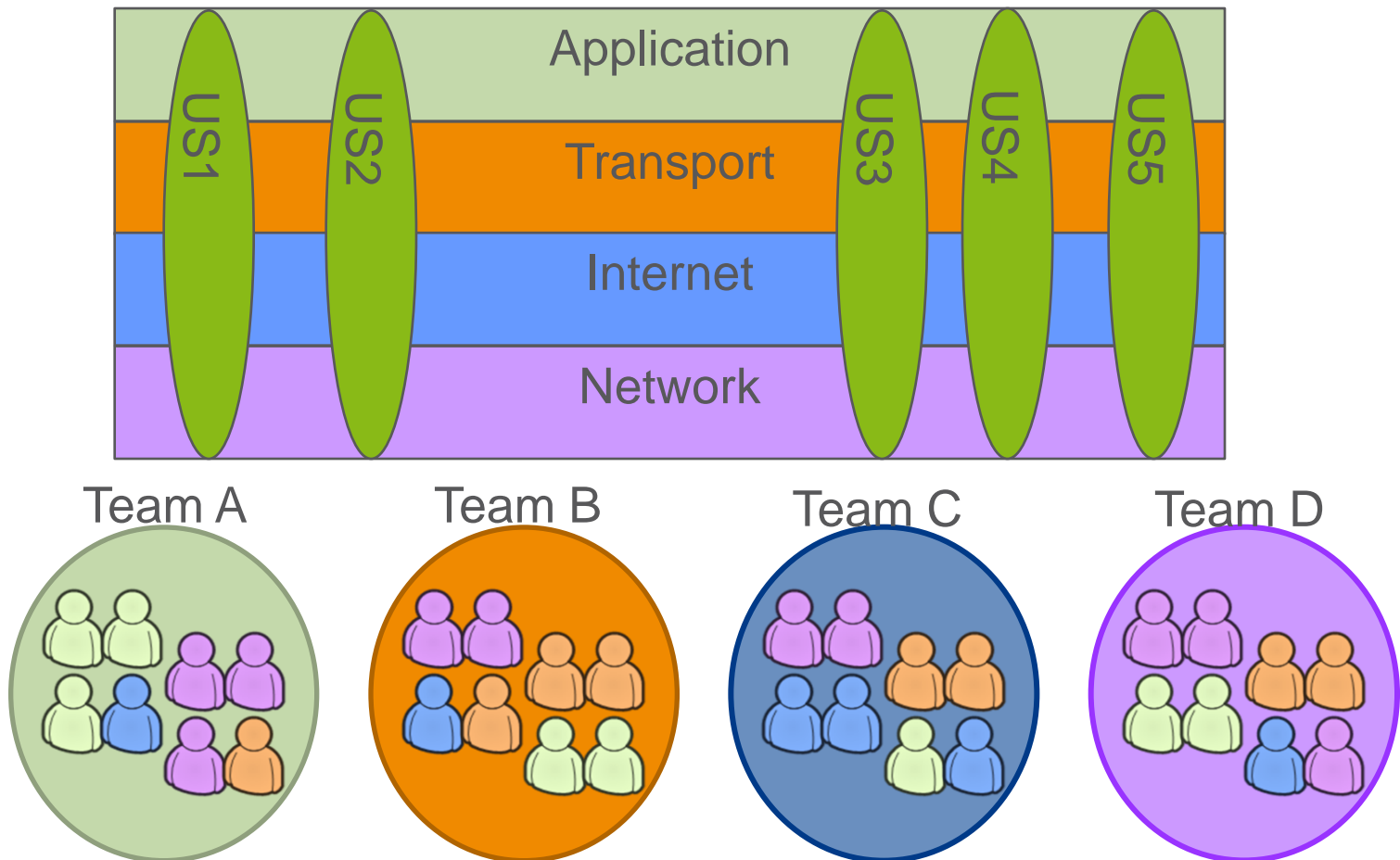
- › Risk of dependencies between User Stories
- › Difficulties to test from an “end to end” perspective
- › More planning needed
- › Prone to generate delays



# WAYS OF WORKING



Backlog Setup with end to end focus and functional areas

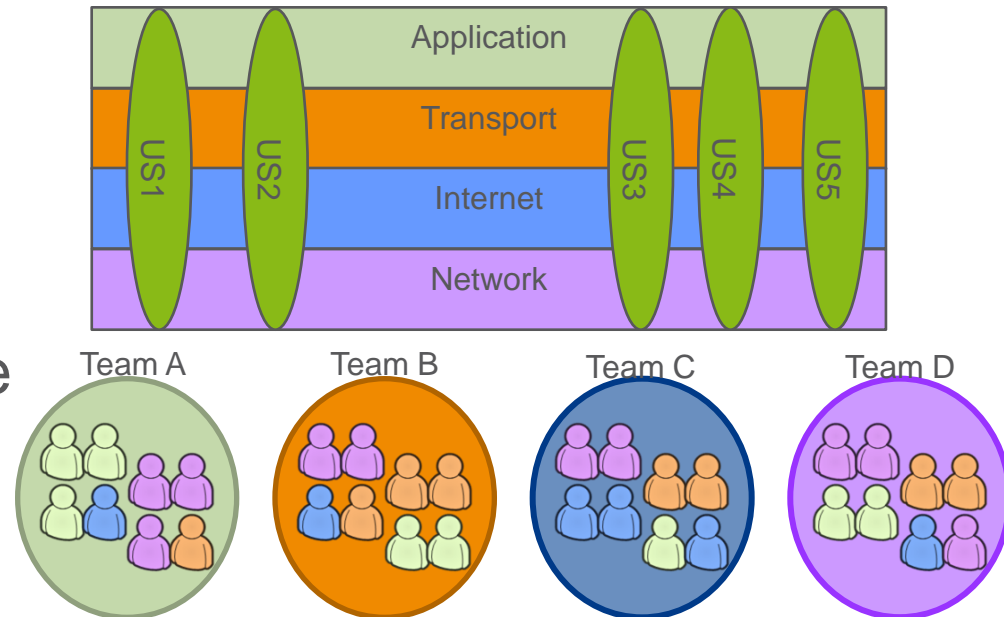


# WAYS OF WORKING



## Backlog Setup with end to end focus and functional areas

- › Testing possible from “end to end” perspective as the User Story is concluded
- › Less planning needed
- › Reduce dependencies due to functional areas
- › Coordination might increase
- › Competence build through training on the job and mix of competences

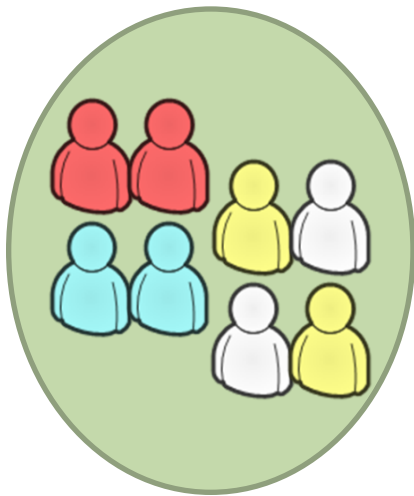


# WAYS OF WORKING



## Setup of virtual teams

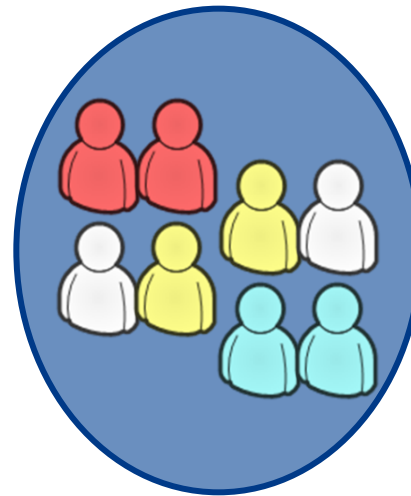
Team A



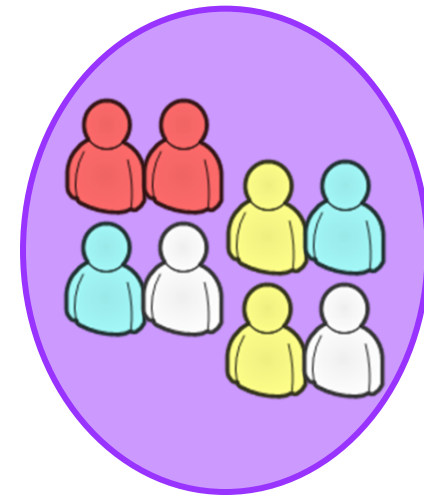
Team B



Team C



Team D



Virtual Test Team



LSV team



System Architecture Team

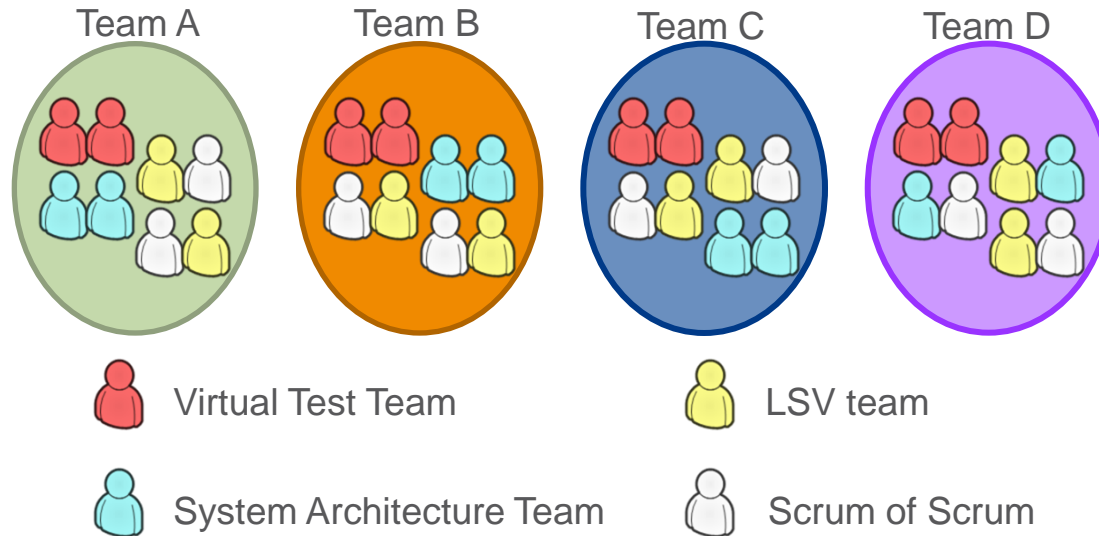


Scrum of Scrum

# WAYS OF WORKING



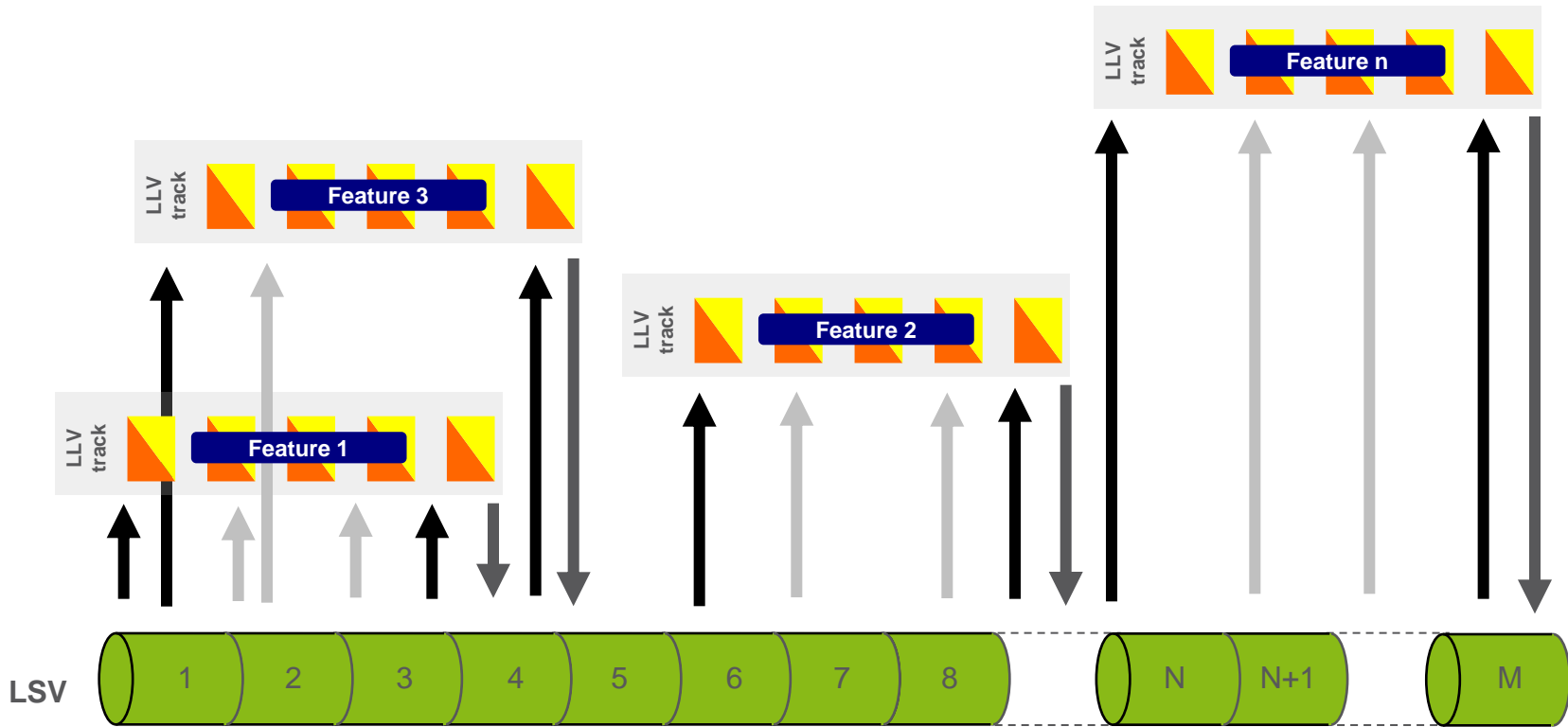
## Setup of virtual teams



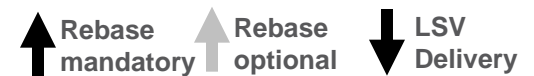
- › Overview over the activities with higher priority
  - Testing, Architecture, Software Delivery, Organizational topics



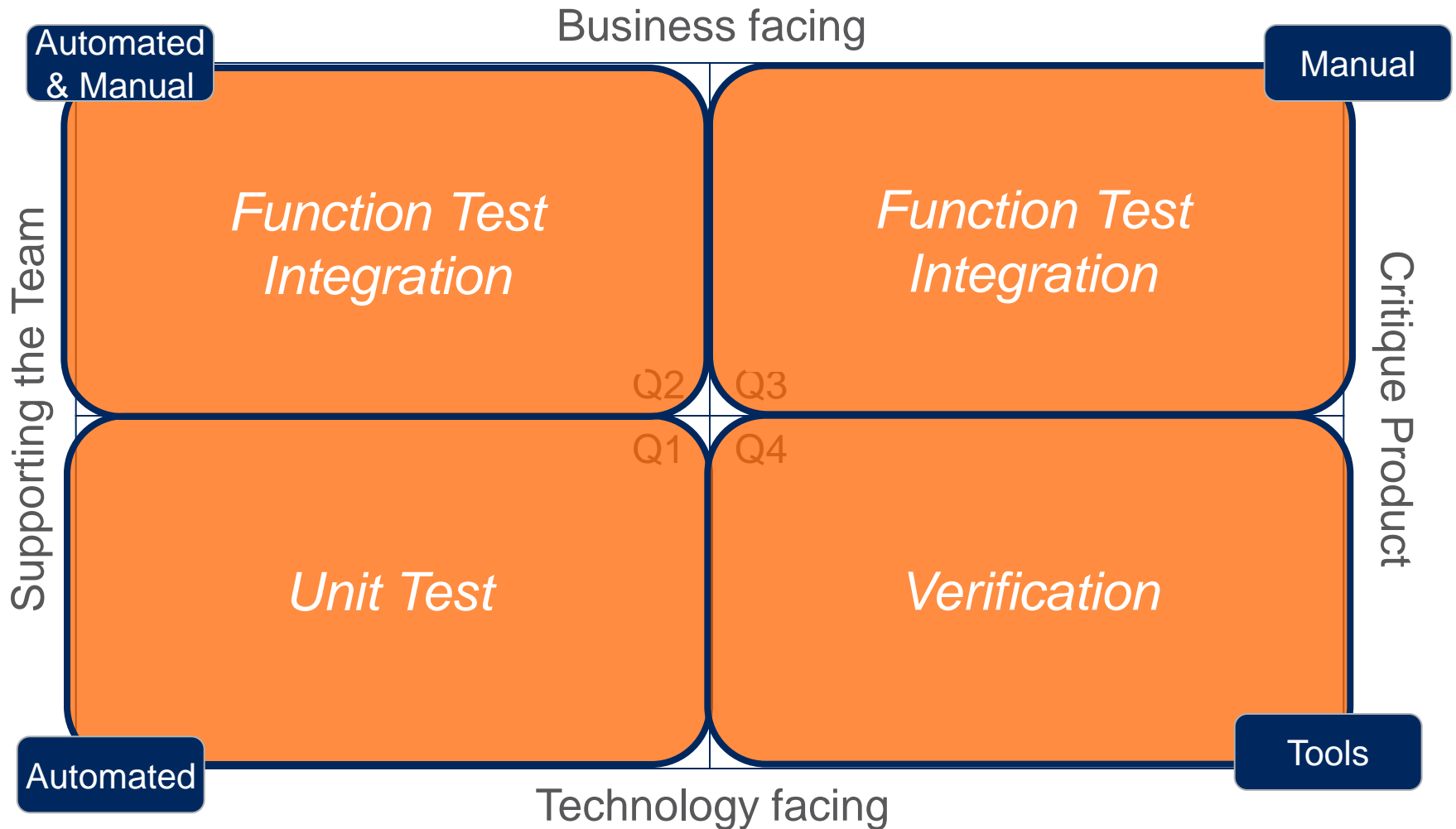
# TEST OVERVIEW



Note: Simplified view, details on following slides



# TESTING ACTIVITIES



1: taken from Crispin/Gregory: Agile Testing, Addison Wesley, 2009

# TESTING ACTIVITIES



- › Q1 Kind of tests are
  - Unit tests – testing small pieces of code
  - System Module Tests – test that units work together correctly
- › Q2 Kind of tests are
  - Functional & Integration tests – positive and deterministic negative tests, if possible to automate
  - User Story Acceptance Tests

# TESTING ACTIVITIES



## › Q3 Kind of tests

- Exploratory Testing – feedback into stories
- Feature Test and Regression Test with Background Traffic
- (early) Characteristics

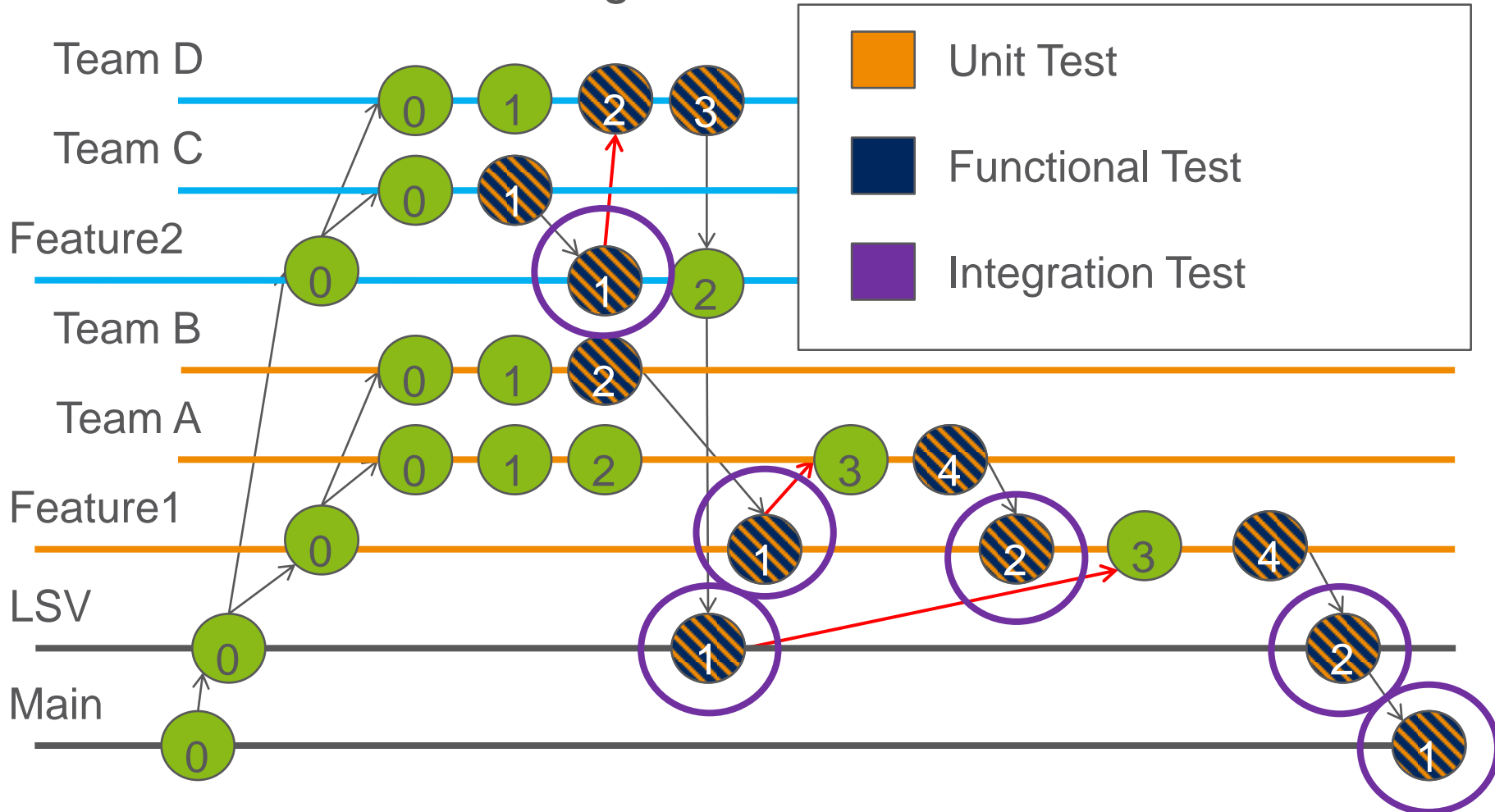
## › Q4 Kind of tests

- Dimensioned Load on LLV/LSV
- High Load/Overload on LSV (most likely)
- Parafunctional tests (Performance, stress)
- Interoperability, Security

# TEST DETAILED VIEW



## › Detailed view on Testing Timeframe

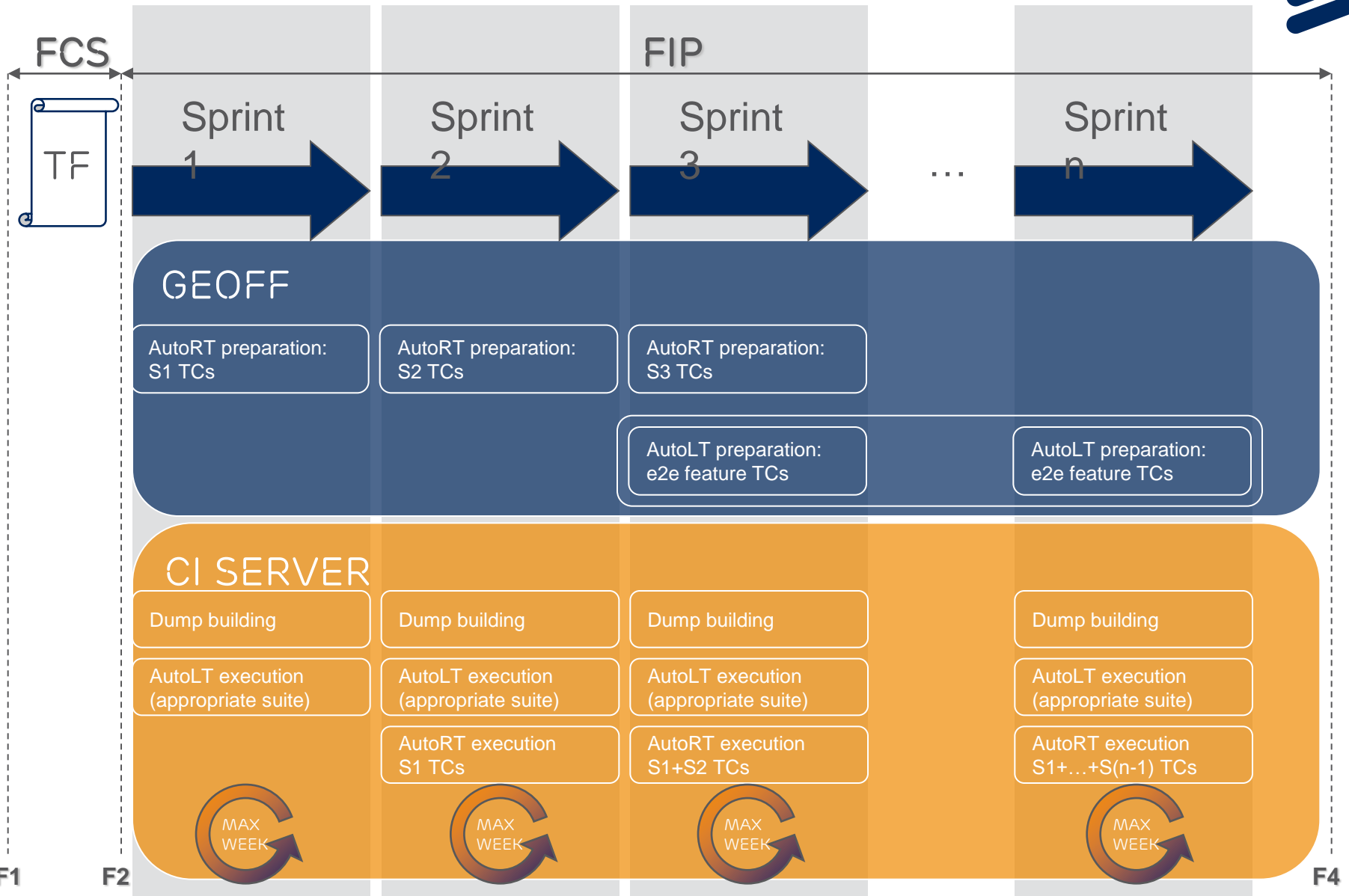


# CONTINUOUS INTEGRATION

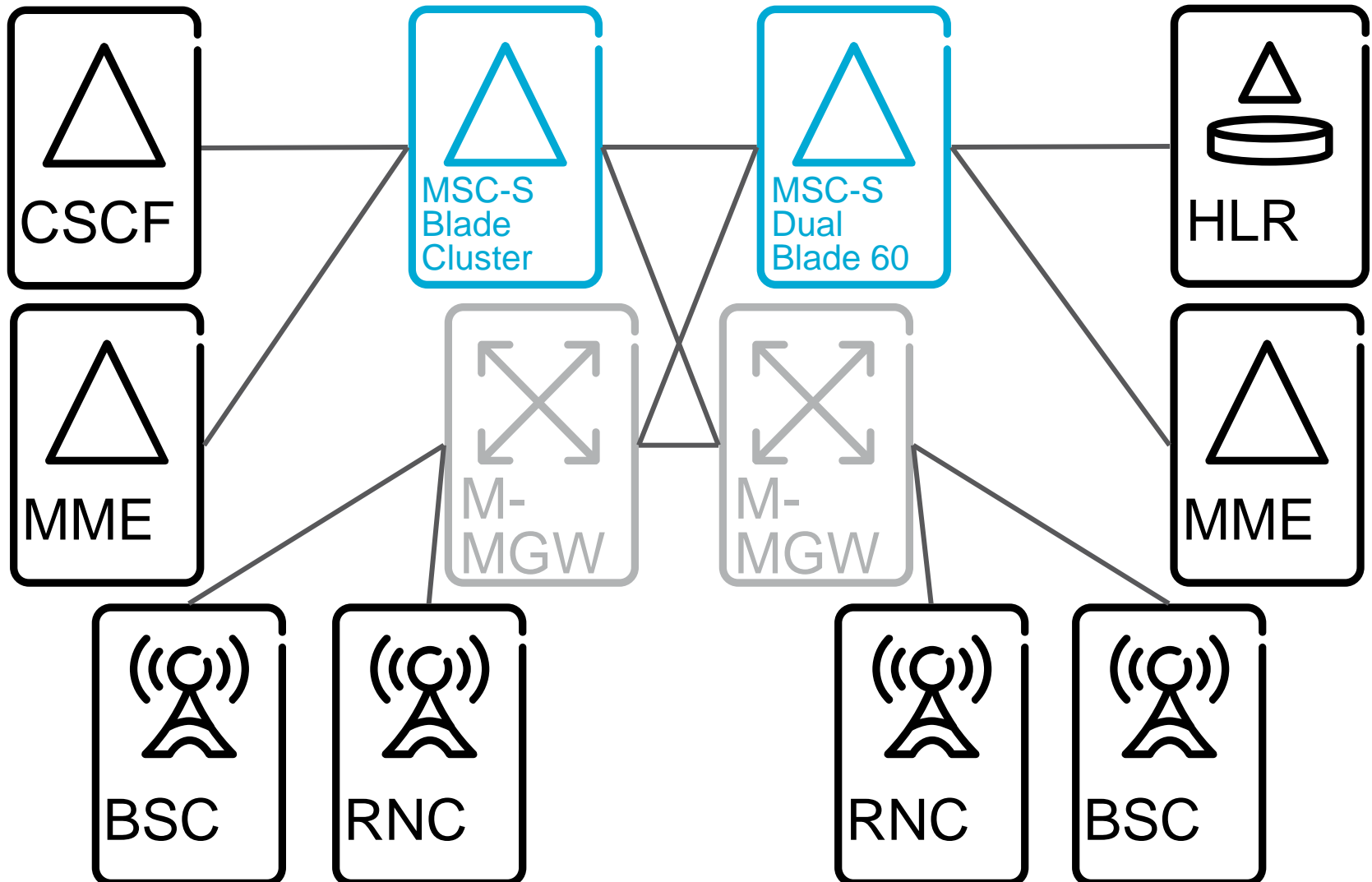
*“...a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily – leading to multiple integrations per day. Each integration is verified by an automated build (incl. test) to detect its errors as quickly as possible...”*

*Martin Fowler*

# CI WORKFLOW



# TEST TOOLS & ENVIRONMENT





# TEST TOOLS & ENVIRONMENT



## › Unit Test:

- Ericsson proprietary tool testing the interfaces

## › Function Test:

- Manual: using a simulated environment simulating all nodes but with the possibility to load the real MSC software
- Automated: using a proprietary tool with an open source framework based on TTCN-3

## › Regression Test:

- Based on the automated test cases from the Function Test
- Running on a Continuous Integration Server (Jenkins)

## › Feature load (moderate/high/overload):

- Proprietary tool based on TTCN-3 generating load on the real environment

# CONCLUSION/WRAP-UP



## Challenges

- › Coordination
- › Maintenance of a high reliability
- › Working high quality Legacy
- › Documentation
- › Test tool & environment

# CONCLUSION



## Benefits

- › Eliminate waste
- › Early indication of Software Quality
- › Early corrective actions possible
- › Reduce test time enabling earlier release of product
- › Enabling of Continuous Integration as first step towards a possible Continuous Delivery
- › Early customer involvement

# CONCLUSION



## › **Manifesto for Agile Software Development**

- › We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

- › **That is, while there is value in the items on the right, we value the items on the left more.**



**ERICSSON**