Lifecycle Strategy in CTA Array Control System

WOSSL– 23.7.2020

I. OYA – CTAO gGmbH

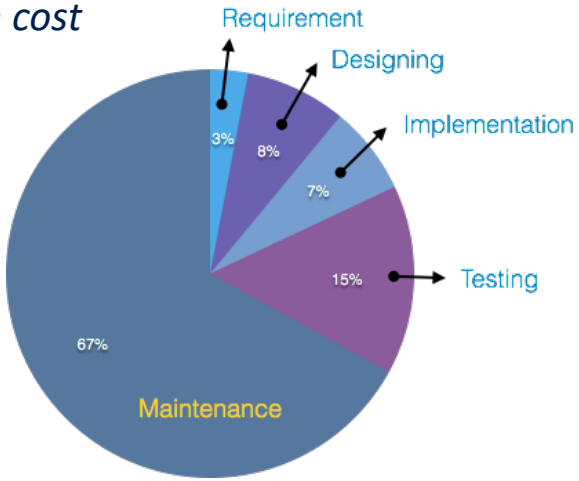
Based on the work of:
A. Zagar, U. Leben (COSYLAB) – Financed by DESY and Humbold Univesity
T. Murach, K. Mosshammer (DESY)
Igor Oya, M. Fuessling (CTAO gGmbH)

Software development lifecycle: why?

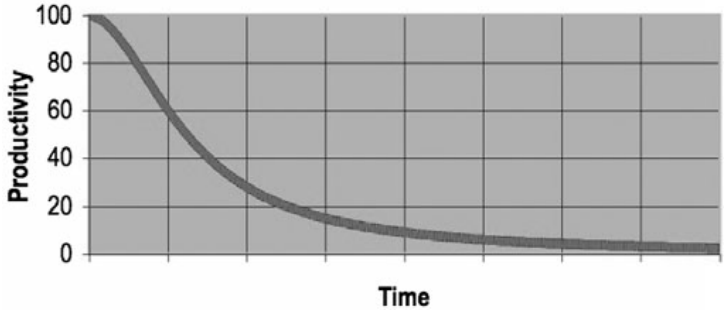
- Define the activities and roles in the implementation of a SW product.
- Establish processes that ensure good-quality SW that meets requirements and schedule
- Best use of existing resources and personnel
- Ensure that delivered SW is easy to maintain
- It is not an implementation schedule – but it is related to it

Code costs yearly ~10% of its production cost

60/60 rule

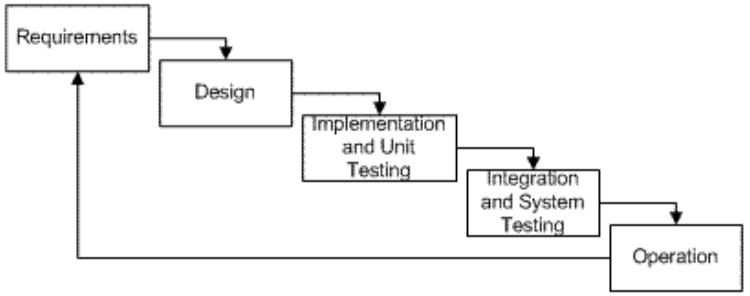


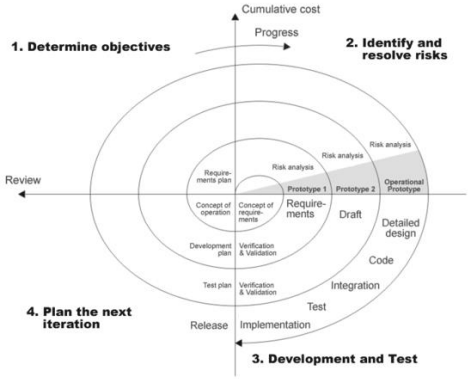
Productivity goes down as code rots

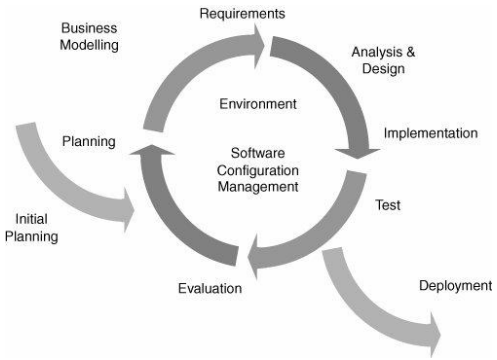


Existing Examples of SDLC approaches

- "Code and fix”
- Waterfall development
- Prototyping
- Incremental development
- Iterative development
- Spiral development
- Rapid development
- Agile development
- …

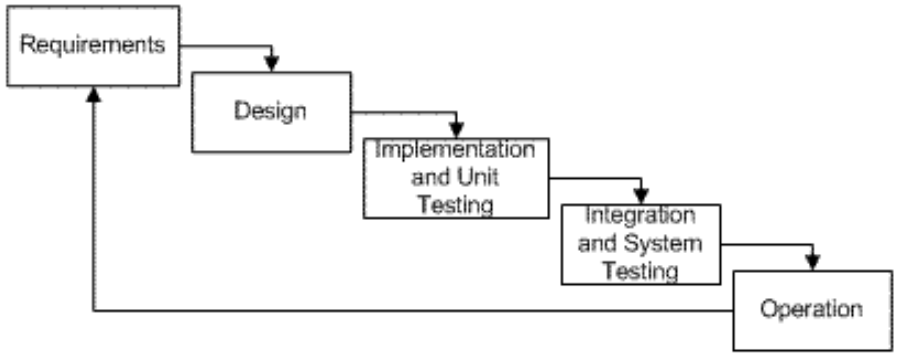






SDLC Phases with and example: Waterfall

SDLC Phases (e.g. Waterfall development):

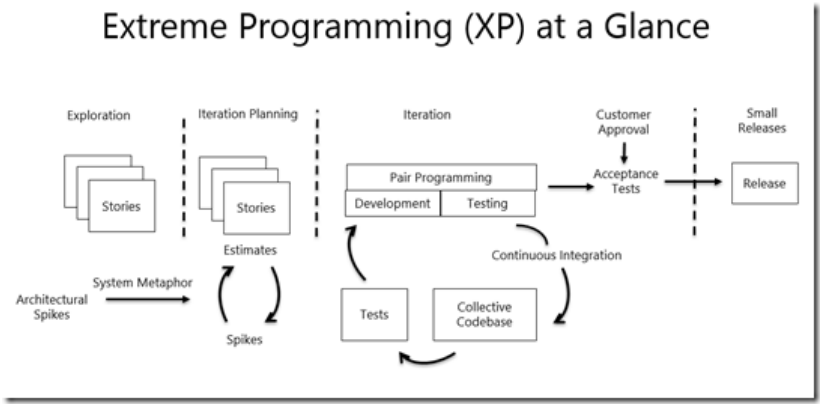


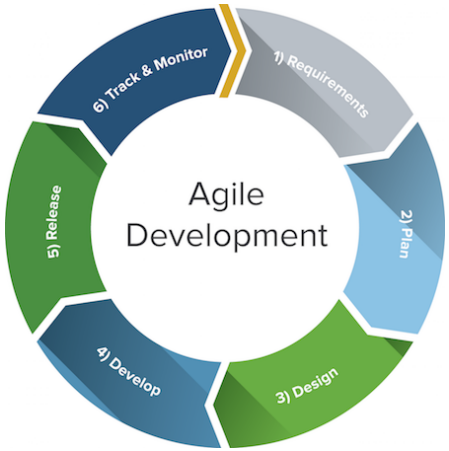
1. Requirements Analysis
2. Software Design
3. Implementation and Unit Testing
4. Integration and System Testing
5. Qualification Testing
6. Deployment/Installation/Commissioning
7. Maintenance

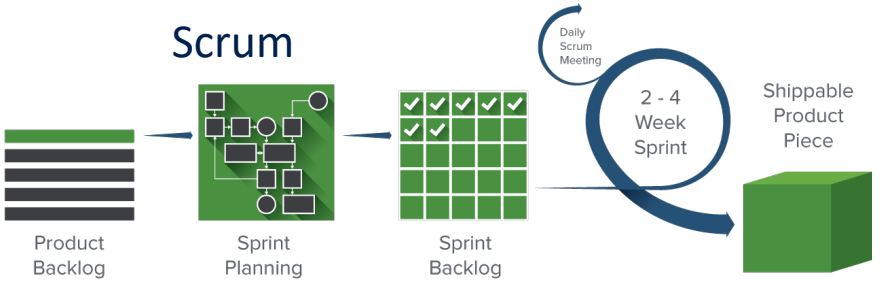
Any SDLC will incorporate more or less these phases, the key being how they are organized

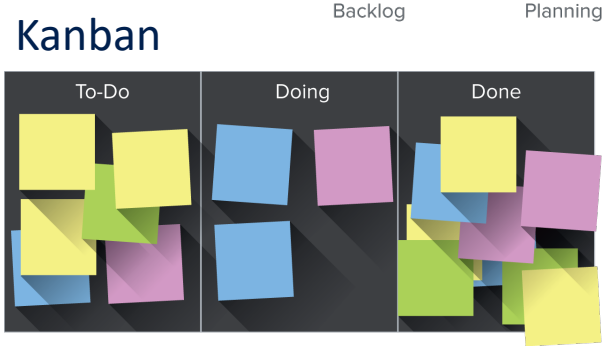
Example: Agile

- **Agile development**
  - Extreme programming
  - Scrum
  - Kanban
  - …









How to pick the appropriate SDLC?

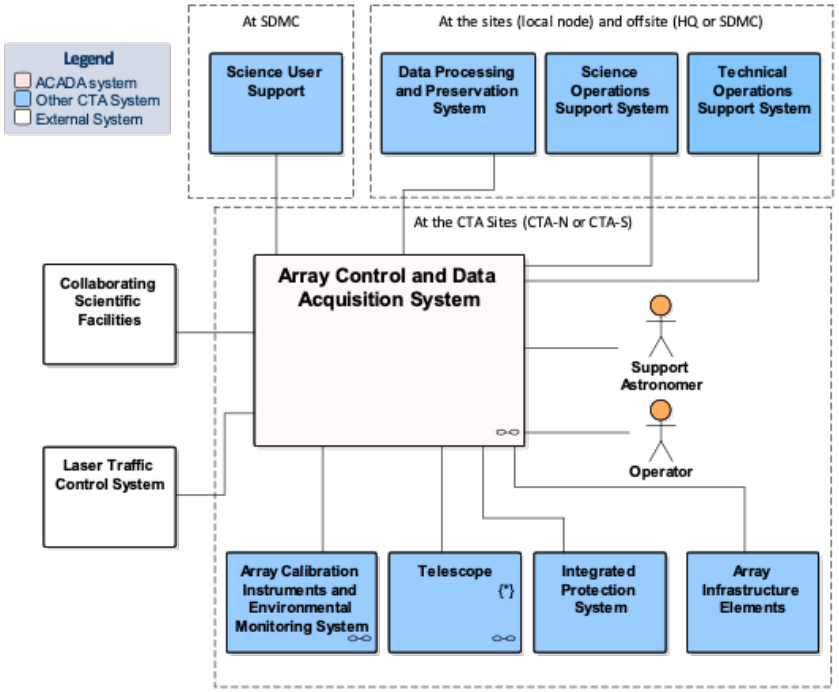
- Decide on the best approach that suits the team conditions.
- A scientific installation SW has its own particularities.
- Funding and staffing scheme.
- …
- Sometimes the choice will be a tailored version of one of the canonical SDLC approaches, or a combination of them.

Our approach:

- ACADA team worked with external experts from Cosylab company to make a proposal suiting the CTA ACADA team needs.
- Cosylab personnel have a long experience from other large scientific installations (ITER, ESS, ALMA,…) and medical equipment.

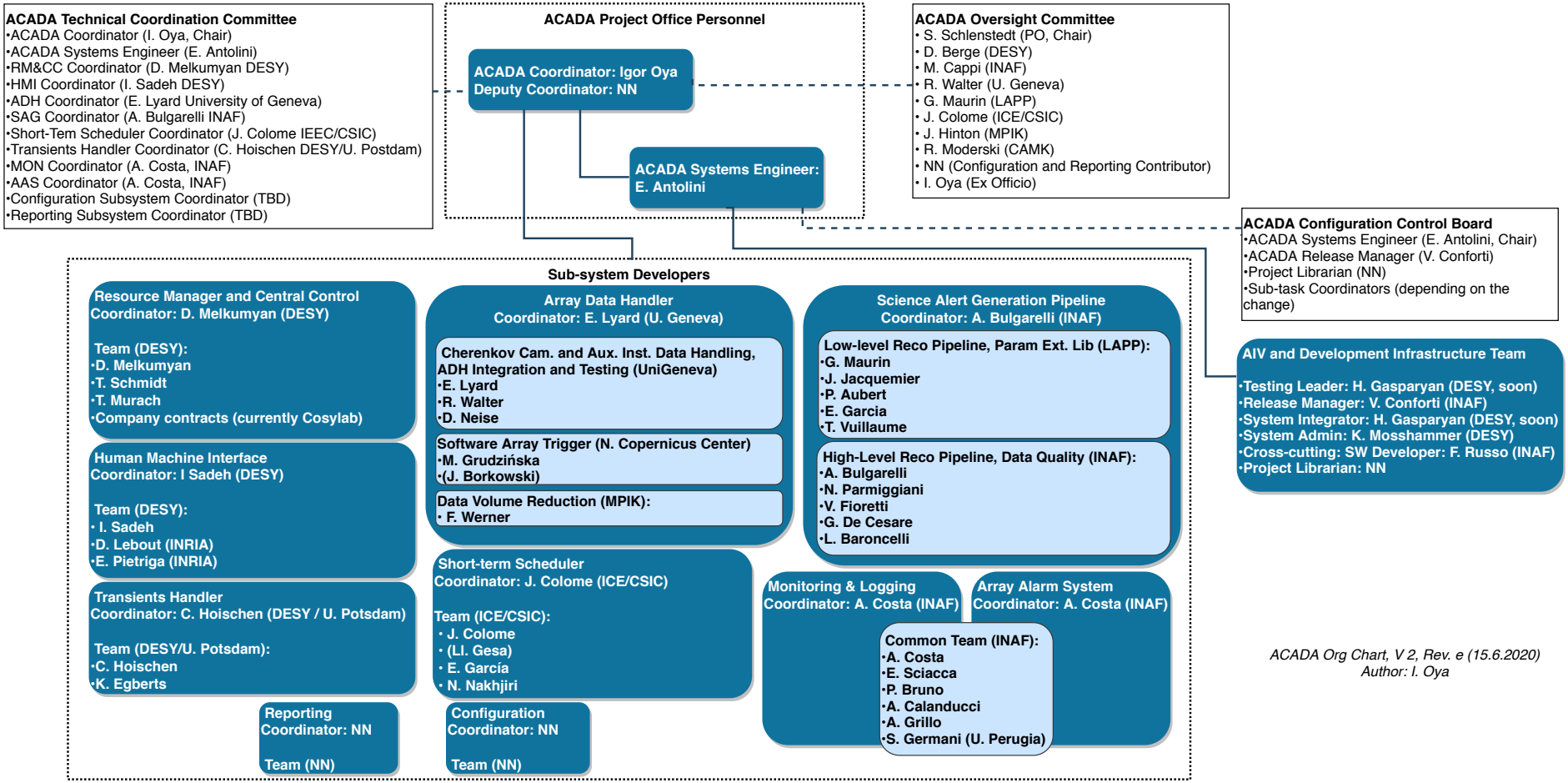
Array Control and Data Acquisition (ACADA)

- System for supervision and control and data acquisition of all telescopes & instruments at both CTA sites.



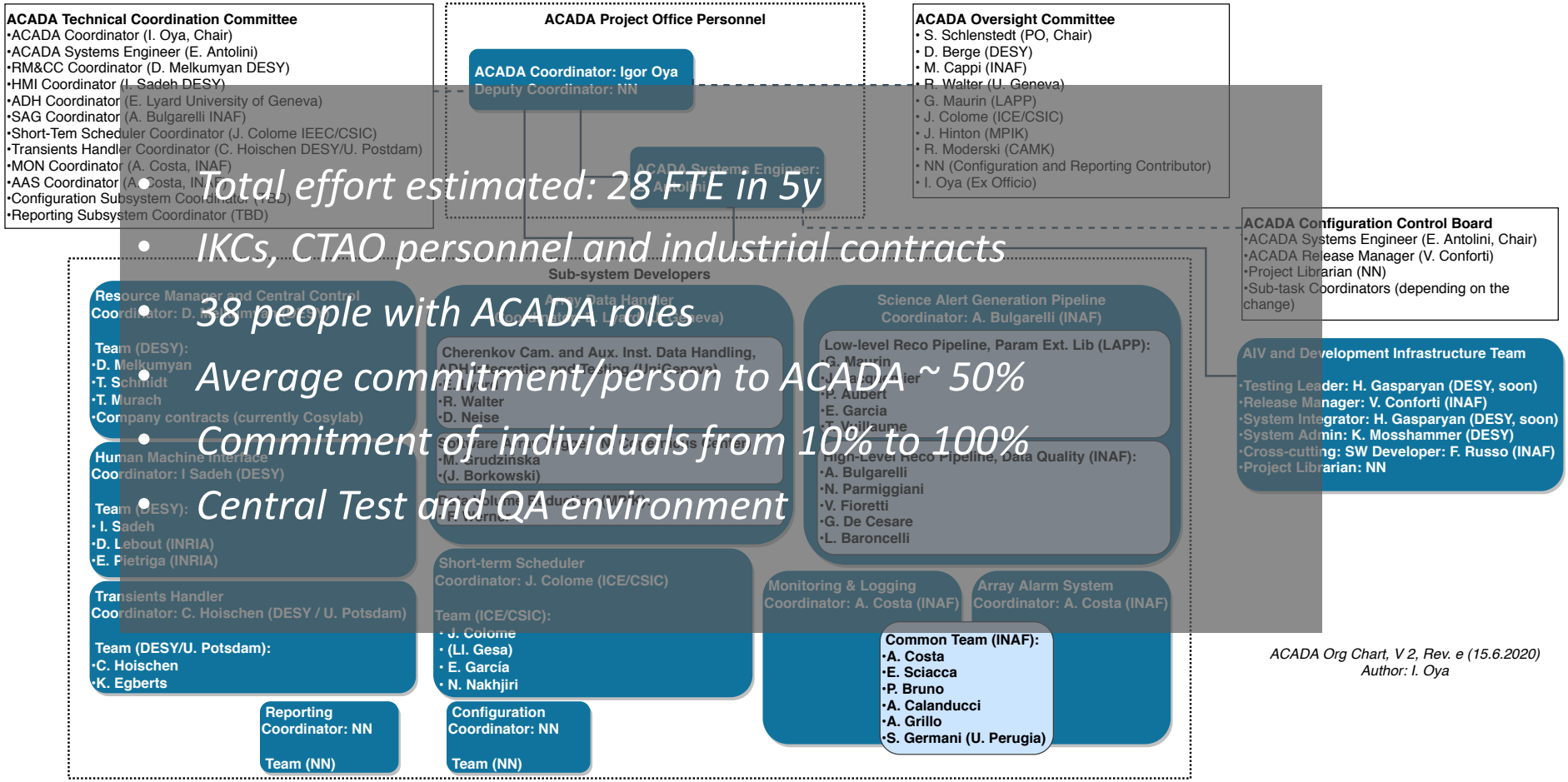
- Equivalent to an industrial control system or SCADA system + embedded scientific online analysis and scheduler.

ACADA:
IKC + Central Office Effort



ACADA Org Chart, V 2, Rev. e (15.6.2020)
Author: I. Oya

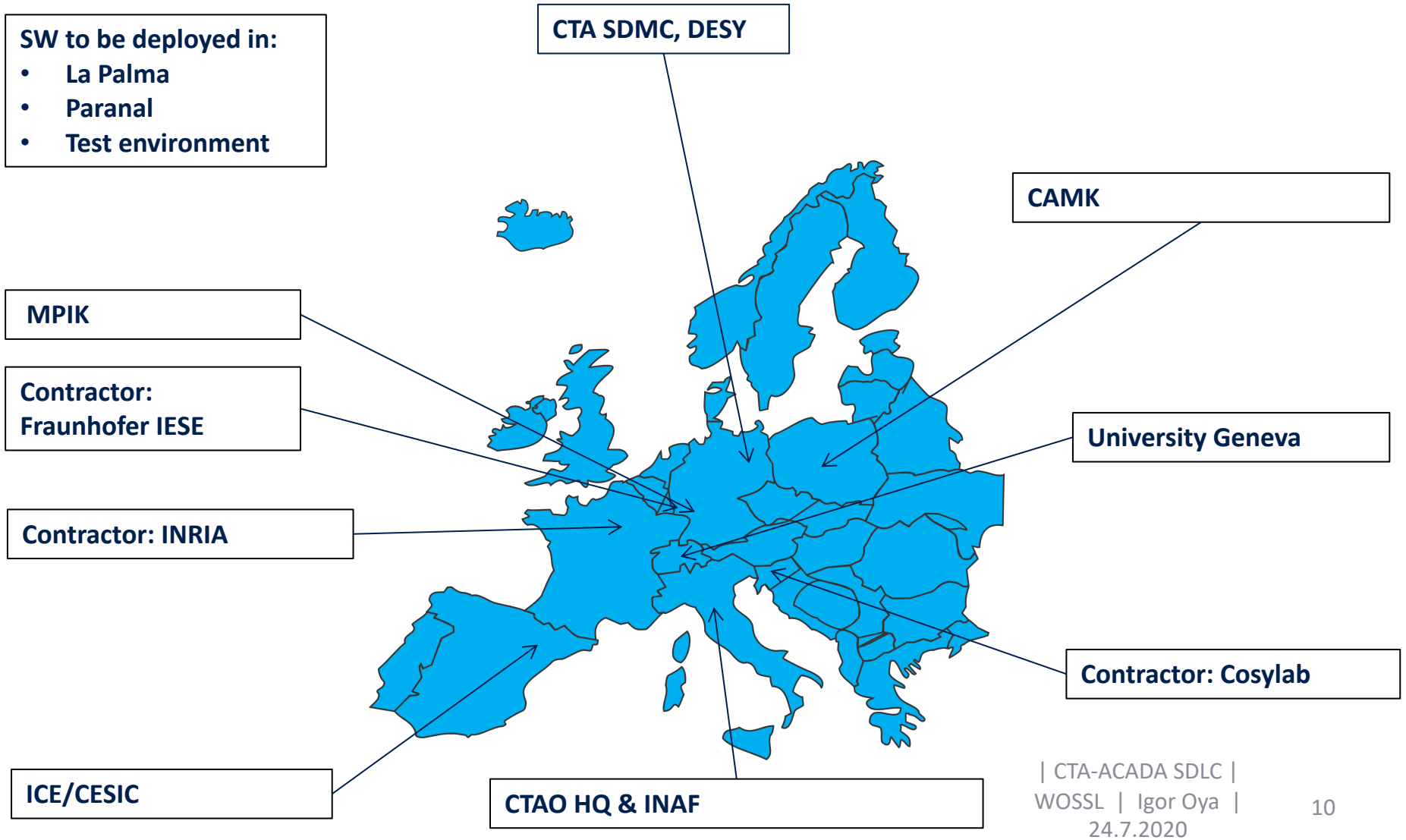
ACADA:
IKC + Central Office Effort



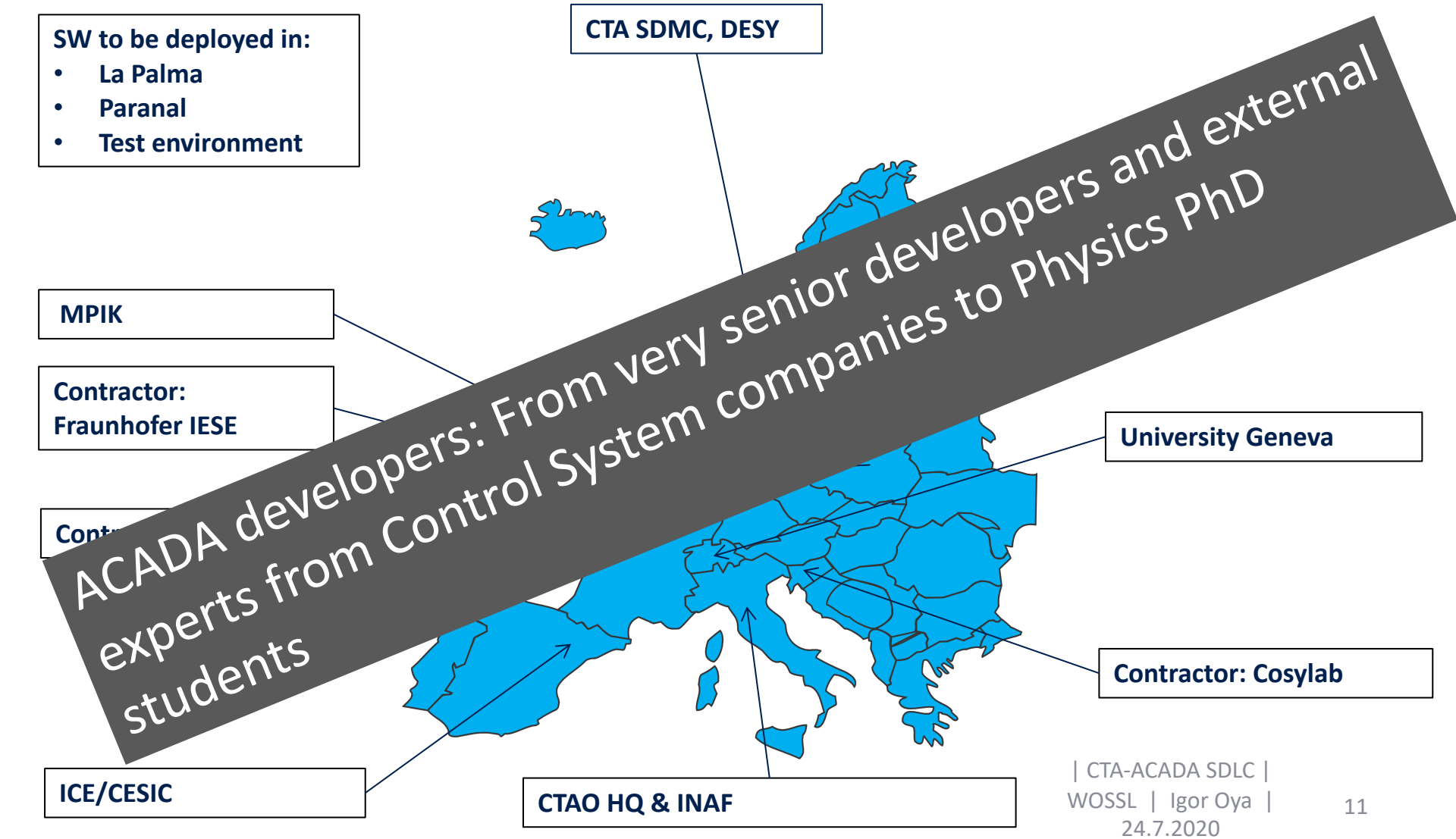
- *Total effort estimated: 28 FTE in 5y*
- *IKCs, CTAO personnel and industrial contracts*
- *38 people with ACADA roles*
- *Average commitment/person to ACADA ~ 50%*
- *Commitment of individuals from 10% to 100%*
- *Central Test and QA environment*

ACADA Org Chart, V 2, Rev. e (15.6.2020)
Author: I. Oya

ACADA Development Team is highly distributed



ACADA Development Team is highly distributed



Current ACADA development and CTA big picture

- ACADA SW development status:
  - Some subsystems and modules are very advanced, with lots of SW written, documented and working with real telescopes.
  - Some contributor have more resources now, others will have later.
  - Some other sub-systems had yet to write their 1st line of code.

- ACADA w.r.t. other of CTA systems:
  - ACADA definition is in general significantly more advanced than surrounding systems.
  - A few ACADA interfaces to other systems are well defined, but for others it is impossible to define at this moment.

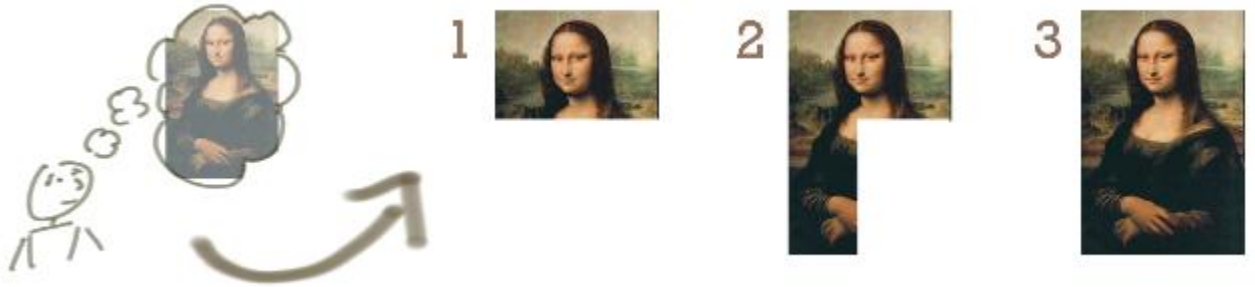
- ACADA schedule vs. project schedule
  - Formal reviews: Preliminary Design Review (PDR), Critical Design Review (CDR), Acceptance Review…
  - Large uncertainties on when ACADA functionalities are needed (e.g. w.r.t. time of telescope deployment).

Based on previous input:

SDLC choice for ACADA

ACADA choice: Iterative and incremental SDLC Model

- Incremental development:



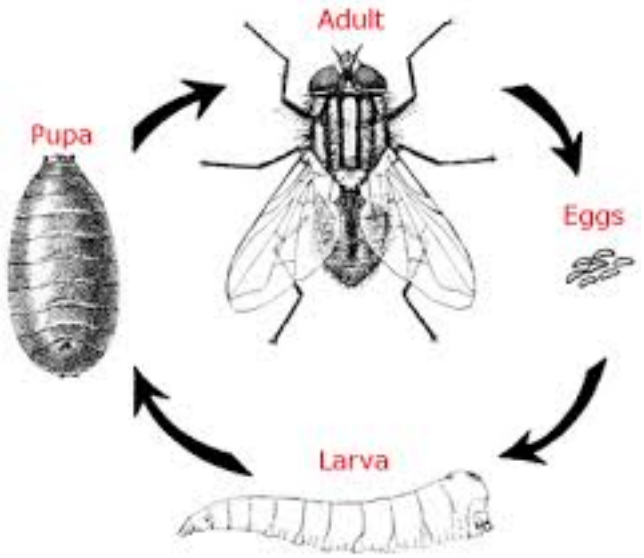
- Iterative development:

Source: Jeff Patton
https://jpattonassociates.com/dont_know_what_i_want/

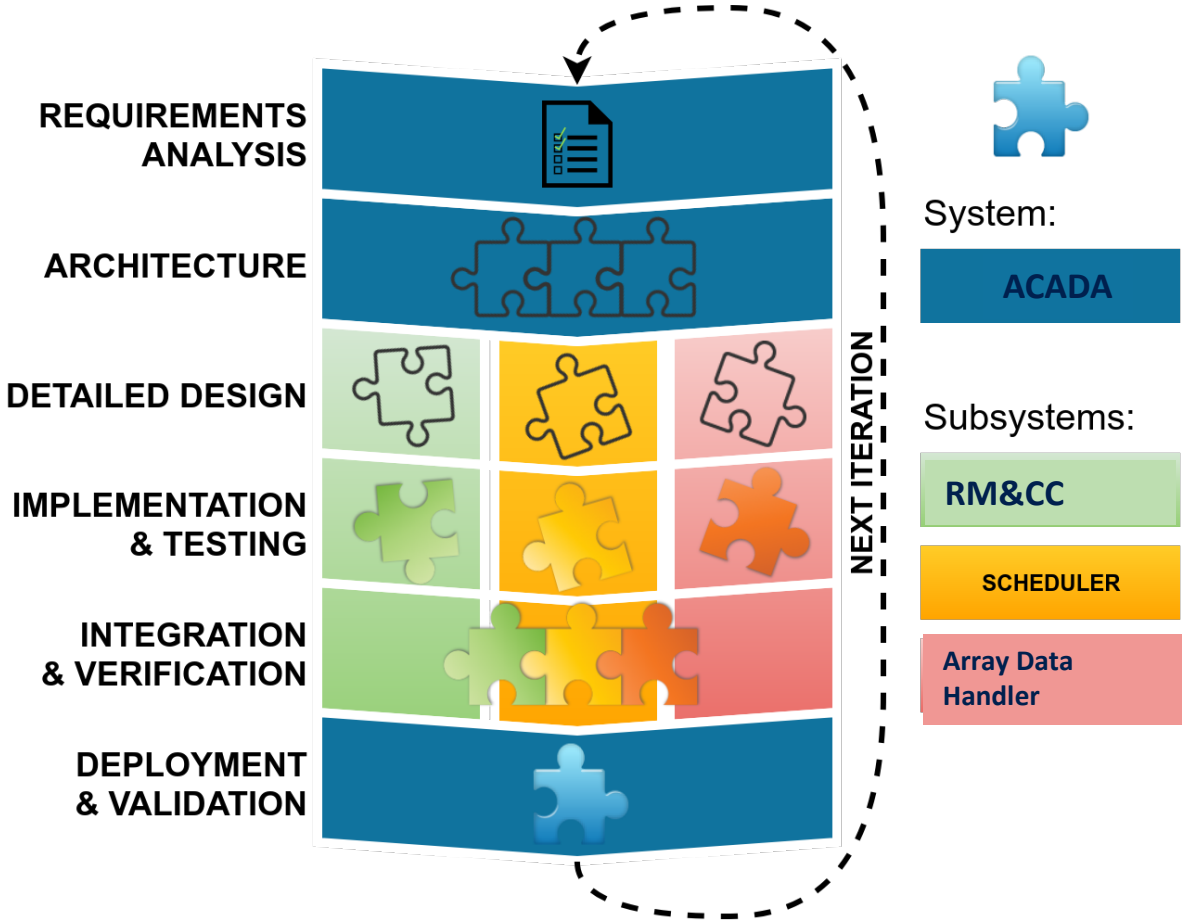
Note: Each contributor team can use the methodology they please as long as the align with the ACADA SDLC at WP level.

Phases in the SDLC

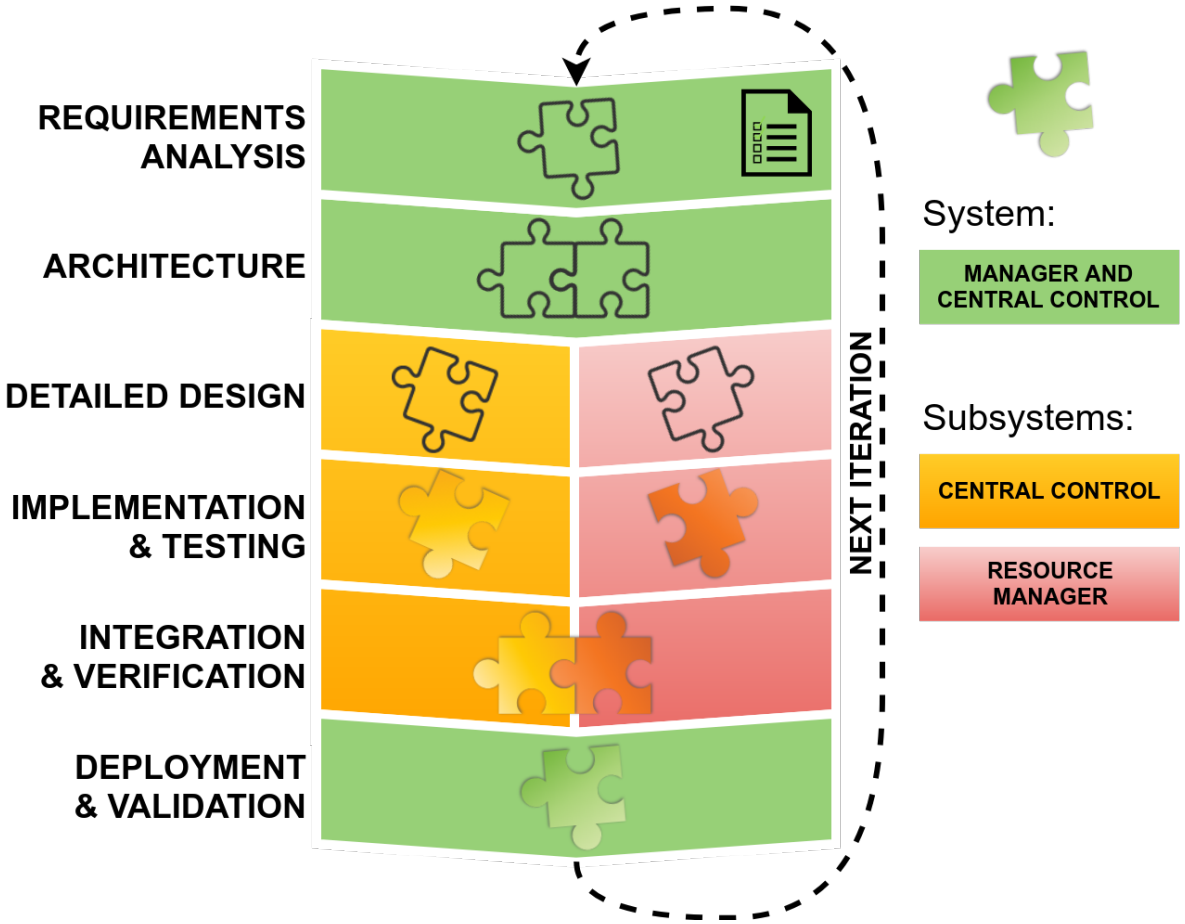
Applied to the ACADA SDLC case



Incremental Iterative SDLC Applied to a Subsystem



Incremental Iterative SDLC Applied to a Subsystem



SDLC Phase Deliverables

Requirements Analysis	Input	• Higher level requirements and use cases
	Output	• Software system requirements and use cases
Architecture	Input	
	Output	• Software architecture specifications
Detailed Design	Input	
	Output	• Detailed software design specifications • Documented interfaces • Unit test case specifications
Implementation & Testing	Input	
	Output	*In case of component:* • Implemented software components • Unit tests • Automated test reports • Code review reports • Draft user documentation *In case of subsystem:* • Integrated software subsystems • Integration test reports • Draft user and deployment documentation
Integration & Verification	Input	
	Output	• Completed integration test plan • Integration test report • Integrated software system • User and deployment documentation
Deployment & Validation	Input	
	Output	• Completed qualification plan • Qualification report • Provisioned, qualified and operational software system • Complete system documentation

• Integration testing specifications / draft plan

• Qualification testing specifications / draft plan

SDLC Roles

Includes ACADA and other CTAO departments personnel

	Auditor	Domain expert	Project manager	Release manager	Requirement analyst	Risk analyst	Software architect	Software designer	Software developer	Software system integrator	Stakeholder	Tester	Testing lead
Requirements Analysis		C	A	R	L	R	C				R		R
Architecture			A	R	C	C	L	C					R
Detailed Design			A	R			C	L	C				R
Implementation & Testing			A	R				C	L				R
Integration & Verification			A	R			C		C	L		R	R
Deployment & Validation	R	R	R	R						L	A		

[A] – Approval authority; [C] – Consultancy; [R] – Active participation, [L] – Lead role

SDLC Implementation: Reviews, Release Plans, Project Milestones

- CTA has a formal review process with PDR, CDR, acceptance reviews etc.
  - Requires to have baseline requirements, architecture designs, detailed design etc in early stages.
  - Even it not intended by our stakeholders, we know that requirements will change during construction and beyond.

- ACADA deliverables are aligned with the CTA project milestones via a *release plan*
  - Specifies scope of each ACADA release in terms of:
    - Requirements, use cases, and interfaces we will support
    - “Relaxation” of quality (non-functional) requirements, e.g. number of supported sub-arrays.
    - QA level of the release.
  - One release every 6 months for a period of 5 years.
  - only next release details are certain: expect frequent changes .

SDLC, Quality Assurance, Requirements Verification and testing

- Our SDLC is aligned with a companion Quality Assurance Plan Document
  - http://icalepcs2019.vrws.de/posters/mompl001_poster.pdf
- Requirements need to be verifiable
- *SDLC stages and testing:*
  - *Requirement analysis:* Qualification test specifications and other verifications are drafted
  - *Architecture:* Integration test plan is prepared
  - *Detailed design phase:* Unit tests are specified
  - *Implementation and testing phase:*
    - Static code analysis
    - Unit tests
    - Continuous integration
    - Code reviews
  - *Integration and verification phase:* integration test plan is followed
  - *Deployment and validation phase:* qualification test plan is created based on qualification test specifications, and executed

SDLC – tooling support

- Continuous integrations and testing: Jenkins, …
- Automatic QA: SonarQube + plugins
- Content / document management system: Confluence, EDMS Web servers, Redmine Wiki, SharePoint
- Issue tracking: Redmine, JIRA
- Requirements analysis tools: Jama
- Test case design tools Jama or MS Office
- Remote connection solutions: VPN, SSH, NX, Windows RDC, …
- Integrated development environment (IDE): Eclipse, …
- Software configuration management (SCM): GitLab, SubVersion…
- Build tools: Maven, CMake,...
- Packaging and distribution tools: RPM/Yum, DEB/APT, MSI, NIX…
- Repository management: Sonatype's Nexus, JFrog's Artifactory, …
- Systems provisioning: RHN Satellite, Spacewalk, Puppet, Ansible…

Legend:
Chosen by CTA - ACADA
Planned or Under investigation
Used now, to be discontinued,
Alternatives

Conclusion

- Choose a SDLC paradigm that fits your project needs.
  - Let the team participate in the choice.
- Understand and specify how the SDLC fits in the overall project structure, schedule and constraints.
- Writing code is not necessarily the main cost of your SW project – take that into account when picking a SDLC approach.
- Document, explain exercise the SDLC with your team.
- Use proper tooling and environment to support it.