# SoSECIE Webinar

Welcome to the
2019 System of Systems Engineering Collaborators Information Exchange (SoSECIE)

National Defense Industrial Association

*We will start at 11AM Eastern Time*

*Skype Meeting +1 (703) 983-2020, 46013573#*

*You can download today's presentation from the SoSECIE Website:*

*[https://mitre.tahoe.appsembler.com/blog](https://mitre.tahoe.appsembler.com/blog)*

*To add/remove yourself from the email list or suggest a future topic or speaker, send an email to [sosecie@mitre.org](mailto:sosecie@mitre.org)*

# NDIA System of Systems SE Committee

- **Mission**
  - To provide a forum where government, industry, and academia can share lessons learned, promote best practices, address issues, and advocate systems engineering for Systems of Systems (SoS)
  - To identify successful strategies for applying systems engineering principles to systems engineering of SoS
- **Operating Practices**
  - Face to face and virtual SoS Committee meetings are held in conjunction with NDIA SE Division meetings that occur in February, April, June, and August
  - SoS Track at NDIA 22nd Annual Systems Engineering Conference, Grand Hilton Tampa Downtown, Tampa, FL, October 21-24, 2019
    - Conference Info: http://www.ndia.org/events/2019/10/21/22nd-annual-systems-and-mission-engineering-conference

NDIA SE Division SoS Committee Industry Chairs:
Mr. Rick Poel, Boeing
Ms. Jennie Horne, Raytheon
OSD Liaison:
Dr. Judith Dahmann, MITRE

## Simple Rules of Engagement

- I have muted all participant lines for this introduction and the briefing.
- If you need to contact me during the briefing, send me an e-mail at sosecie@mitre.org.
- Download the presentation so you can follow along on your own
- We will hold all questions until the end:
  - I will start with questions submitted online via the CHAT window in Skype.
  - I will then take questions via telephone; State your name, organization, and question clearly.
- If a question requires more discussion, the speaker(s) contact info is in the brief.

## Disclaimer

# 2019 System of Systems Engineering Collaborators Information Exchange Webinars
*Sponsored by MITRE and NDIA SE Division*

***October 22, 2019***
***Modeling Process for the Design of System of Systems Evolution***
*Dr. Jeremy Buisson, Dr. Isabelle Borne and Mr. Franck Petitdemange*

***November 5, 2019***
***Irrational System Behavior in a System of Systems***
*Mr. Douglas L. Van Bossuyt, Mr. Bryan M. O'Halloran and Mr. Ryan M. Arlitt*

***November 19, 2019***
***Multi-Dimensional Classification of System-of-Systems***
*Dr. Bedir Tekinerdogen*

***December 3, 2019***
***Digital Twin Strategies for System of Systems***
*Mr. Michael Borth*

***December 17***
***TBD***

***January 14***
***Framework for Improving Complex System Performance***
*Mr. Chuck Keating*

# Modeling process for the design of SoS evolution

Franck Petitdemange, Isabelle Borne, Jérémy Buisson

IRISA, Université Bretagne Sud

CREC St Cyr, École de Saint-Cyr Coëtquidan

October 22, 2019

## System of Systems engineering

According to Maier (1998)[1], main system-of-systems's characteristics are:

- Operational independence of its constituents
- Managerial independence of its constituents

Following from these characteristics, Maier advocated for several architecture principles:

- Stable intermediate form[2]
- Policy triage[2]
- Leverage the interface
- Ensuring cooperation

[1] Mark W. Maier. "Architecting principles for systems-of-systems". In: *Systems Engineering* 1.4 (1998), pp. 267–284. ISSN: 1520-6858. DOI: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291520-6858%281998%291%3A4%3C267%3A%3AAID-SYS3%3E3.0.CO%3B2-D (visited on 05/27/2019).

[2] Eberhardt Rechtin. *Systems Architecting: Creating and Building Complex Systems*. Englewood Cliffs, N.J: Prentice Hall, 1990. 352 pp. ISBN: 978-0-13-880345-2.

## Evolutionary development

At least, constituent independence precludes any guarantee upon their availability in the SoS

Design of the SoS evolves too:

- Project phasing anticipates forthcoming evolution
- New requirements or context changes may arise
- Arising technologies provide new opportunities in the context of long-lived SoS

Motivates stable intermediate form principle

- Between evolution, the SoS must be capable of operating and fulfilling user purpose

In this talk: operating and fulfilling user purpose during evolution too

## Outline

## Based on French emergency services (simplified)

Departmental Fire and Rescue Service (SDIS): dept 35

Departmental Fire and Rescue Service (SDIS): dept 56

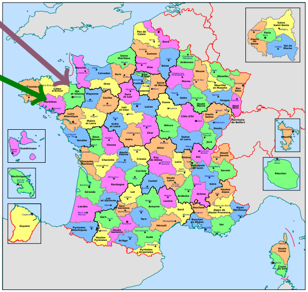
Territorial autonomous entity:
- operation center (CODIS)
- local resources: fire fighters, vehicules, boats

Hospital

Autonomous entity:
- emergency room
- ambulances
- medical emergency service (SAMU)

General Directorate for Civil Security and Crisis Management (DGSCGC)

Min.-of-interior-level entity:
- crisis management
- nationwide resources: helicopters, planes



Nilstilar [CC BY-SA 4.0 (`https://creativecommons.org/licenses/by-sa/4.0`)]

## Scenario

When a call for emergency service is received:

- The territorially competent Departmental Fire and Rescue Service (SDIS) charges its Operation Center (CODIS) to conduct the operation
- CODIS gives directly orders to SDIS's resources, who execute the operation, and CODIS receives resources' reports

On escalation, the operation is reorganized to deal with increased severity:

- Resources from other SDIS, from DGSCGC or from hospital may join the operation
- A group leader may be designated to conduct the operation
- DGSCGC's crisis management may be take over the operation

## Scenario

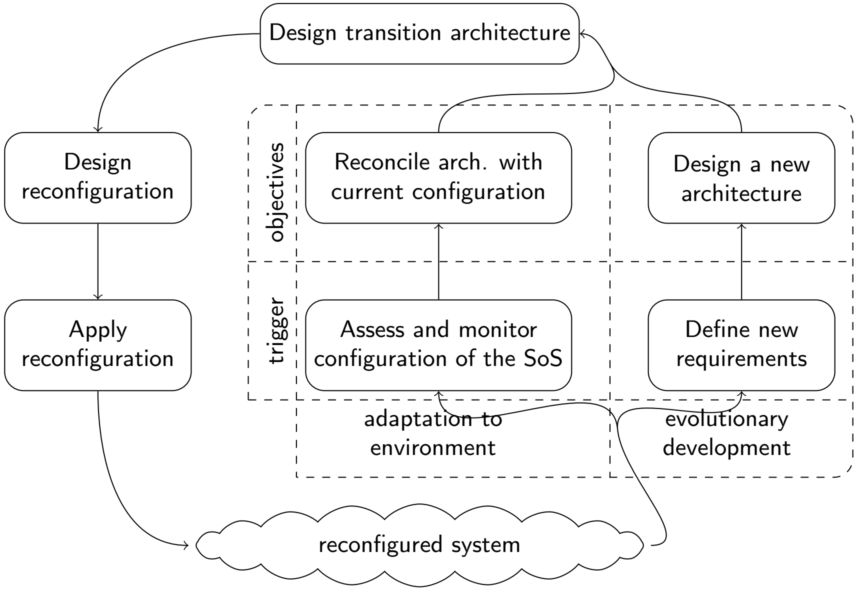
In this talk: the operation is seen as the SoS

- We assume a flood

Evolution occurs when the flood appears to be more serious than initially thought

- In this talk: deploy a group leader to conduct operation
- Beyond the talk, in our complete case study: then deploy additional resources from other organizations

## Overall reconfiguration process

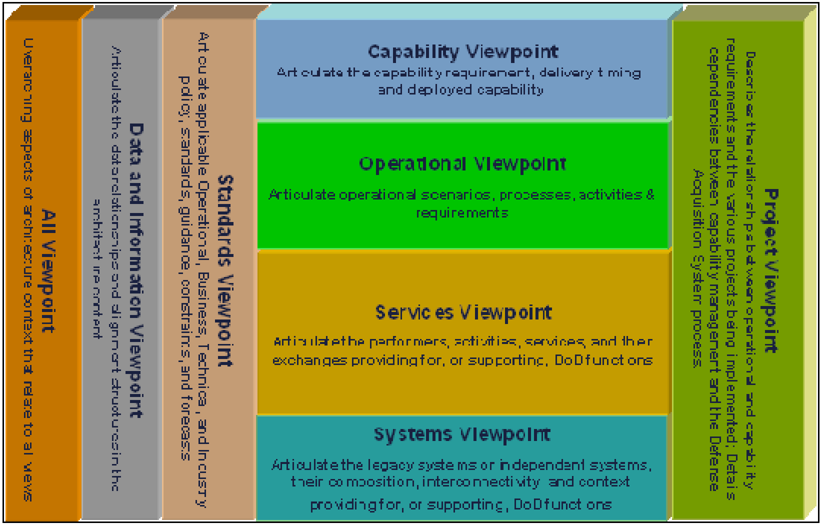

Design transition architecture
Design reconfiguration
Apply reconfiguration
objectives
trigger
Reconcile arch. with current configuration
Design a new architecture
Assess and monitor configuration of the SoS
Define new requirements
adaptation to environment
evolutionary development
reconfigured system

submitted, not yet published

## Outline

## Overview of DoDAF

DoDAF is a comprehensive architecture framework for system of systems



All Viewpoint
Overarching aspects of architecture context that relate to all views
Data and Information Viewpoint
Articulate the data relationships and alignment structures in the architecture content
Standards Viewpoint
Articulate applicable Operational, Business, Technical, and Industry policy, standards, guidance, constraints, and forecasts
Capability Viewpoint
Articulate the capability requirement, delivery timing and deployed capability
Operational Viewpoint
Articulate operational scenarios, processes, activities & requirements
Services Viewpoint
Articulate the performers, activities, services, and their exchanges providing for, or supporting, DoD functions
Systems Viewpoint
Articulate the legacy systems or independent systems, their composition, interconnectivity and context providing for, or supporting, DoD functions
Project Viewpoint
Describes the relationships between operational and capability requirements and the various projects being implemented; Details dependencies between capability management and the Defense Acquisition System process.

## UPDM

UPDM is a comprehensive metamodel for DoDAF (and MODAF)

UPDM is a profile importing and extending SysML and SoaML

- Reuse diagrams and notations
- Extend and specialize concepts

SysML overview:

- The system is decomposed as a tree of blocks
- Blocks interact through connectors binding ports
- Item flow specifies what is conveyed by a connector
- Allocation links elements from various representations of the system

## Modeling the system of systems

When considering system of systems, some issues arise

- Heterogeneity due to independent engineering and management of constituents
- Boundary identification of the system
- Versatility of constituents due to independent management and operation
- Architecture dynamicity due to adaptation to actual environment and to evolving requirements

## Modeling configurations with reconfiguration in mind

To deal with versatility and dynamicity, two levels of modeling

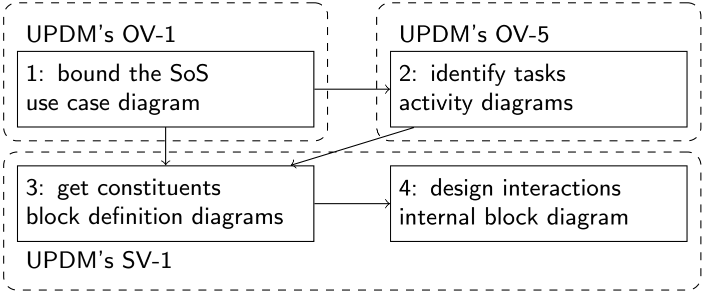
- Rules governing the architecture independently of any operation
  - Describe organization, doctrine, concept of operations
  - Maybe instantiated to concrete architecture
- Concrete architecture for each specific operation
  - Describe actual resources involved in operation and their effective organization

In this talk, we focus on the concrete architecture

- We model rules at the operation-independent architecture using of architectural patterns
- For instance, "layered architecture" style can be used to model hierarchical communication discipline

## Modeling configurations with reconfiguration in mind

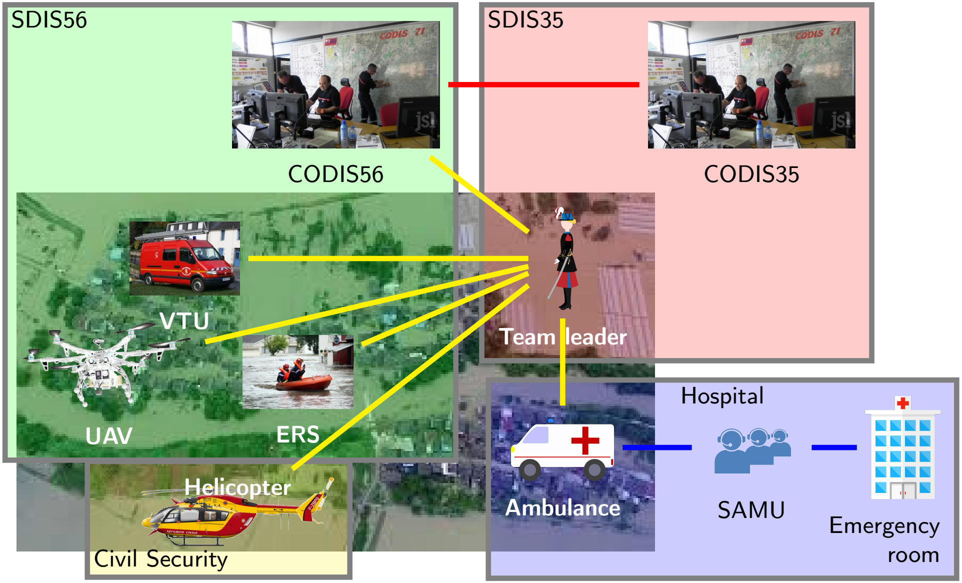
A process yielding to an operation-specific concrete architecture[3]



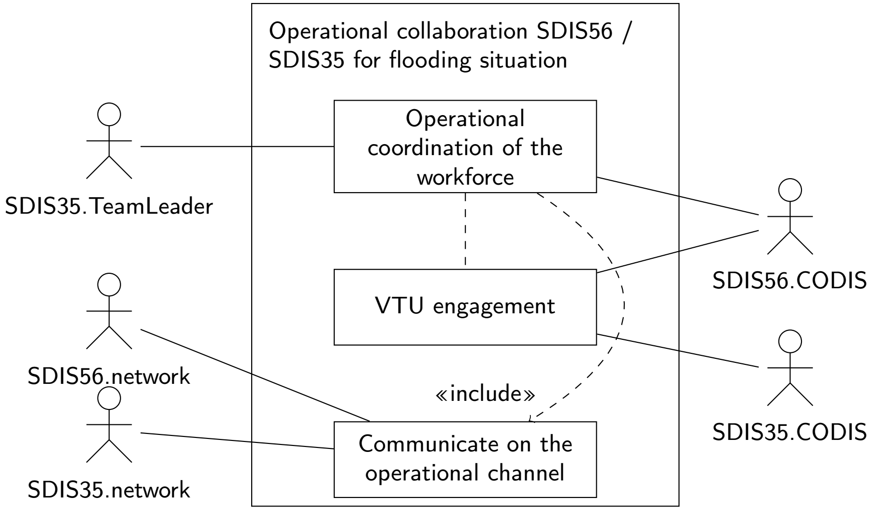
- OV-1: High-level operational concept
- OV-5: Activities and their relationships
- SV-1: Systems and their interfaces

[3] Franck Petitdemange, Isabelle Borne, and Jérémy Buisson. "Modeling System of Systems configurations". In: *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. 2018 13th Annual Conference on System of Systems Engineering (SoSE). June 2018, pp. 392–399. DOI: 10.1109/SYSOSE.2018.8428737.

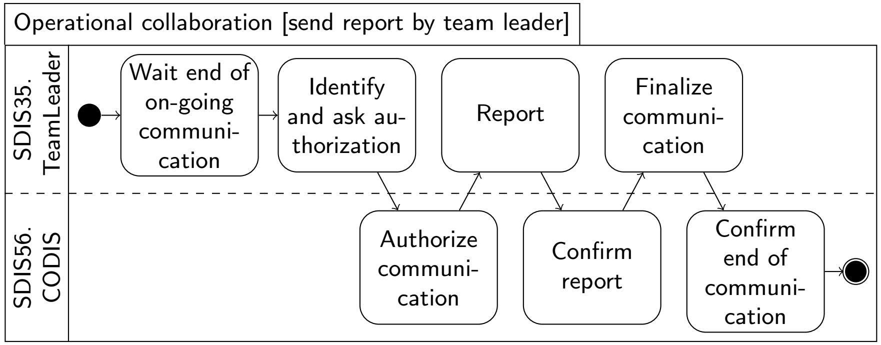
## Illustration: OV-1



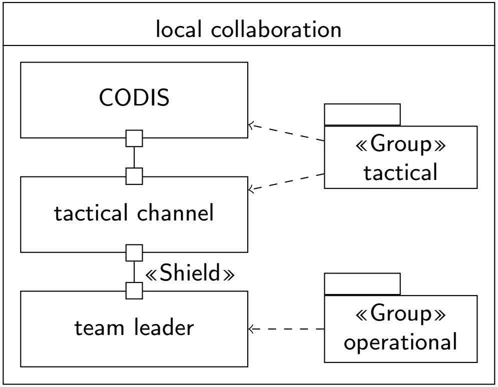
## Illustration: OV-1


Operational collaboration SDIS56 / SDIS35 for flooding situation
SDIS35.TeamLeader
Operational coordination of the workforce
SDIS56.CODIS
VTU engagement
SDIS56.network
SDIS35.CODIS
«include»
SDIS35.network
Communicate on the operational channel

## Illustration: OV-5


Operational collaboration [send report by team leader]
SDIS35. TeamLeader
SDIS56. CODIS
Wait end of on-going communication
Identify and ask authorization
Report
Finalize communication
Authorize communication
Confirm report
Confirm end of communication

## Illustration: SV-1



Architectural primitives[4] are used to model organizational rules

- Layer pattern models hierarchical command chain discipline

[4] Uwe Zdun and Paris Avgeriou. "A catalog of architectural primitives for modeling architectural patterns". In: *Information and Software Technology* 50.9 (Aug. 1, 2008), pp. 1003–1034. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2007.09.003. URL: http://www.sciencedirect.com/science/article/pii/S0950584907001073 (visited on 09/21/2019).

## Summary

While DoDAF is a comprehensive framework, the architecture is aimed to be a set of communication artifacts

- Views are selected to fit to purpose

Towards reconfiguration, we need:

- A model that is representative of the actual deployed system of systems

Motivates targeting SV-1

- Description of actual constituent systems and their interactions

## Outline

## Dynamic reconfiguration

Modifying a system while it is being operated

- Architectural changes include adding/removing constituents/connectors

Not only changing the architecture

- Doing it right!
  - Assuming that there are units of treatment that must be considered as a whole
    - Call it transaction or whatsoever
  - Such units of treatment must occur at either the initial architecture or at the new architecture
    - But do not start with one architecture and complete with another one

## Quiescence

A traditional approach in software systems: quiescence[5]

- Enforce that affected constituents do not communicate through connectors that are going to be removed

Assume constituents can be active or passive

- In passive state, the constituent does not initiate any new unit of treatment

Make affected constituents passive, as well as constituents that may request these affected constituents

[5] Jeff Kramer and Jeff Magee. "The Evolving Philosophers Problem: Dynamic Change Management". In: *IEEE Trans. Softw. Eng.* 16.11 (Nov. 1990), pp. 1293–1306. ISSN: 0098-5589. DOI: 10.1109/32.60317. URL: http://dx.doi.org/10.1109/32.60317 (visited on 08/16/2019).

## In the SoS context

Quiescence is not necessarily well-suited to systems of systems

- Operational and managerial independence vs forcing constituent to passive state
- Still may be relevant for constituents under the control of the SoS

Alternative approaches have been proposed, in the context of software systems[6]

- Forcibly migrate internal state
- Allow several concurrent version
- Wait until inactivity, i.e., quiescence

---

[6] Gísli Hjálmtýsson and Robert Gray. "Dynamic C++ Classes: A Lightweight Mechanism to Update Code in a Running Program". In: *Proceedings of the Annual Conference on USENIX Annual Technical Conference*. ATEC '98. event-place: New Orleans, Louisiana. Berkeley, CA, USA: USENIX Association, 1998, pp. 6–6. URL: `http://dl.acm.org/citation.cfm?id=1268256.1268262` (visited on 08/16/2019).

## Preparation and cleanup as reconfiguration

Preparation and clean up phases of a reconfiguration

- Sometimes as simple as suspending/resuming (parts of) the application
- Sometimes instantiating monitors, managers, and alike
- Sometimes instantiating additional service providers

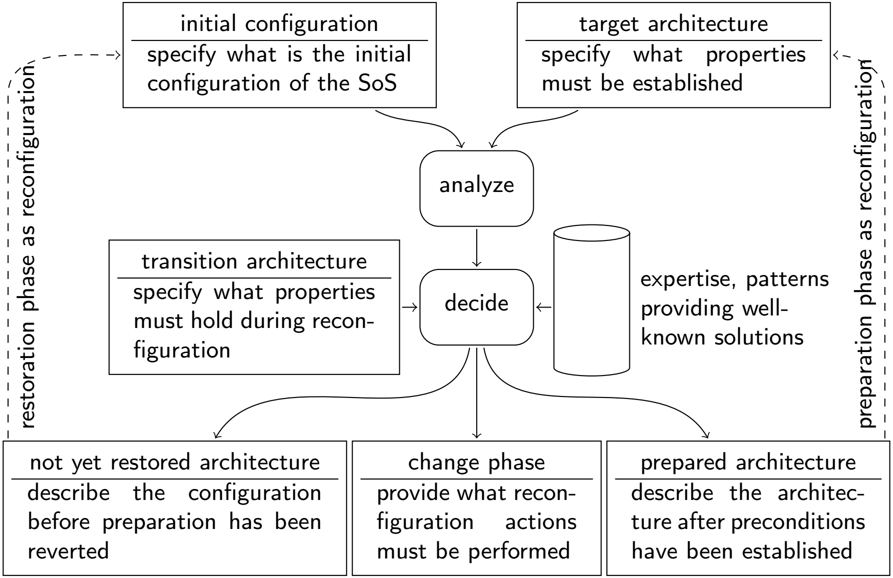
Preparation and clean up phases shall be reconfigurations as well

- Reconfiguration may be compound
- Possibly, each subreconfiguration relies on different mechanisms at various locations in the analysis framework[7]

Need for coarser approaches, i.e., patterns of recurring solutions

[7] Franck Petitdemange. "Développement évolutionnaire de systèmes de systèmes avec une approche par patron de reconfiguration dynamique". PhD thesis. Vannes: Université de Bretagne Sud, Dec. 3, 2018.

## Engineering process to design a reconfiguration


initial configuration
specify what is the initial configuration of the SoS
target architecture
specify what properties must be established
analyze
transition architecture
specify what properties must hold during reconfiguration
decide
expertise, patterns providing well-known solutions
not yet restored architecture
describe the configuration before preparation has been reverted
change phase
provide what reconfiguration actions must be performed
prepared architecture
describe the architecture after preconditions have been established
restoration phase as reconfiguration
preparation phase as reconfiguration

submitted, not yet published

## Reconfiguration pattern

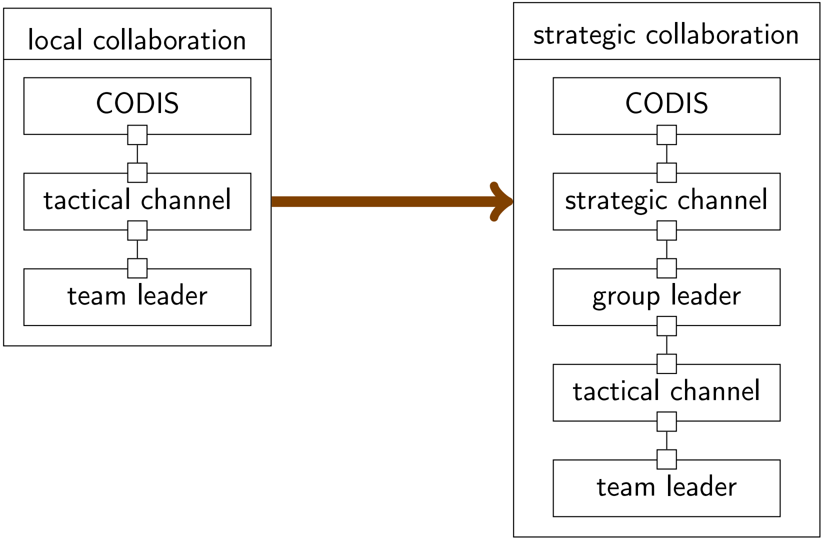
Document well-known solutions to reconfiguration design[8][9]

- Context section
  - Refers to relevant architectural patterns or styles
  - Describes governance constraints and expectations about constituents
- Problem section
  - States structural and behavioral (in)variants, including those of the reconfigured system
- Solution section
  - Specify operations, noticeably those expected to be provided by involved constituents
  - Specify transiently needed constituents

[8] Franck Petitdemange, Isabelle Borne, and Jérémy Buisson. "Approach Based Patterns for System-of-Systems Reconfiguration". In: *2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems*. 2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems. May 2015, pp. 19–22. DOI: 10.1109/SESoS.2015.11.

[9] Franck Petitdemange, Jérémy Buisson, and Isabelle Borne. "Une approche orientée patron pour la reconfiguration de système de systèmes". In: *Techniques et sciences informatiques* 35.6 (Dec. 30, 2016), pp. 665–674. ISSN: 07524072. DOI: 10.3166/tsi.35.665-674. URL: https://tsi.revuesonline.com/article.jsp?articleId=37918 (visited on 08/19/2019).

## Illustration



## Candidate pattern

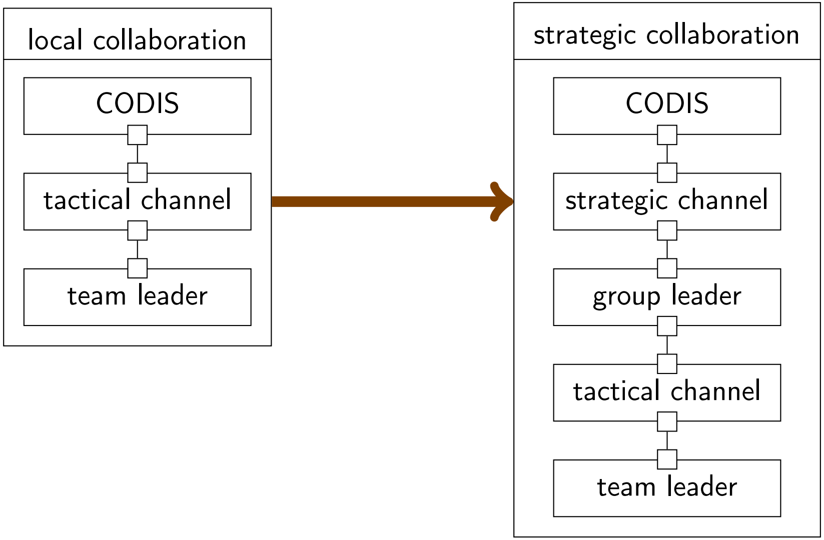
Co-evolution of shared state across initial and final architecture constituents[10]

When a C2 constituent orchestrates actions of (many) other constituents, and this C2 need to be replaced

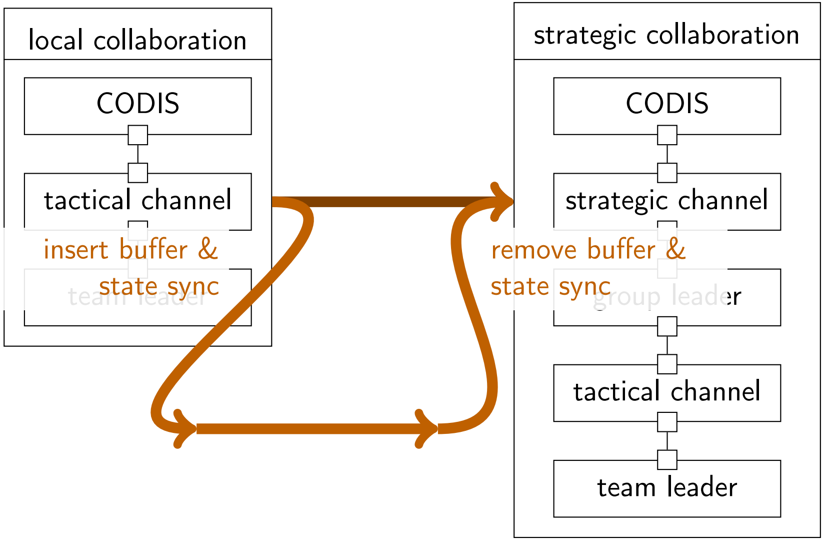
- Avoid blackout while constituents are reconnected to the new C2
- Instead: make the two C2 coexist and collaborate
  - Preparation
    - Integrate the new C2 at the beginning of reconfiguration
    - Deploy facility to synchronize internal state of the two C2
  - Reconfiguration
    - Progressive reconnection of constituents to the new C2
  - Cleanup
    - Dismiss the old C2 and state synchronization facility when all the constituents have been reconnected to the new C2

[10] Franck Petitdemange, Isabelle Borne, and Jérémy Buisson. "Assisting the Evolutionary Development of SoS with Reconfiguration Patterns". In: *Proccedings of the 10th European Conference on Software Architecture Workshops*. ECSAW '16. event-place: Copenhagen, Denmark. New York, NY, USA: ACM, 2016, 9:1–9:7. ISBN: 978-1-4503-4781-5. DOI: 10.1145/2993412.3004845. URL: http://doi.acm.org/10.1145/2993412.3004845 (visited on 10/18/2019).

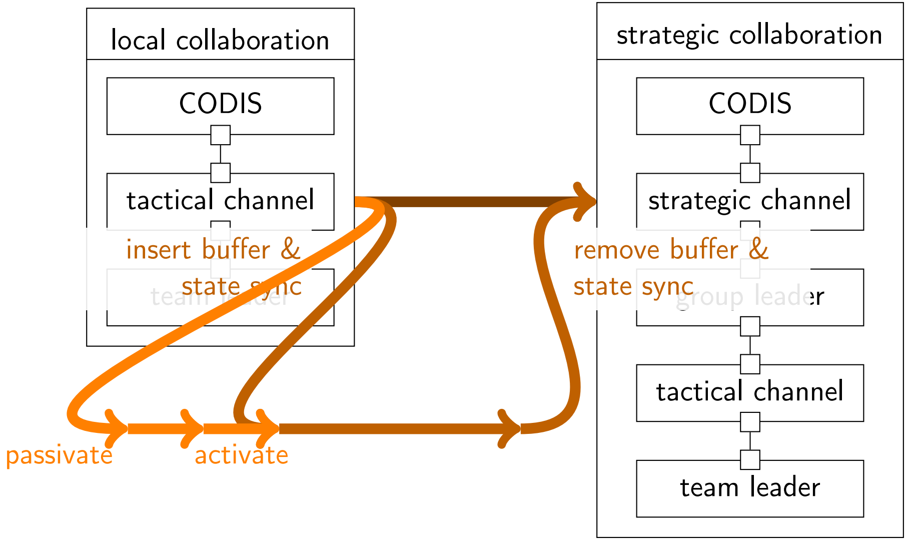
## Illustration



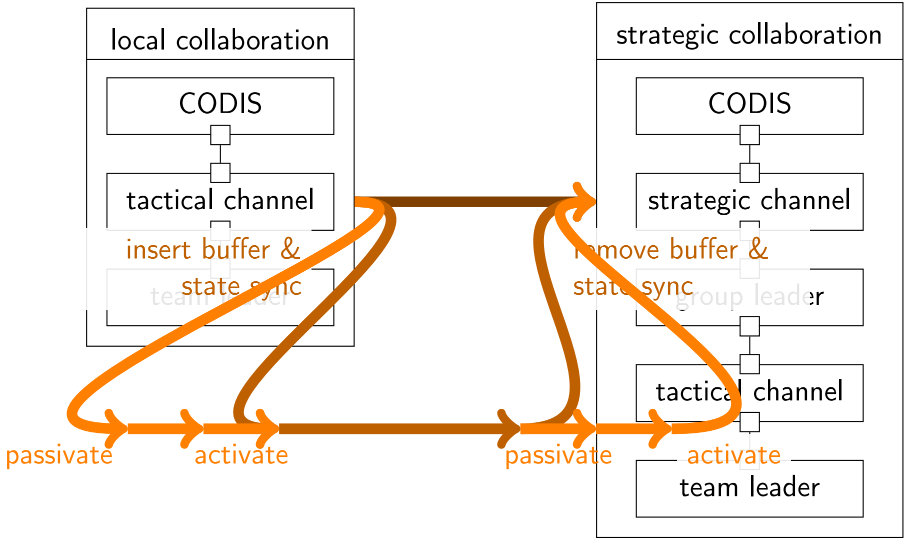
## Illustration



## Illustration



## Illustration



## Summary

Reconfiguration is compound: preparation, desired change and cleanup

- Considering each step a reconfiguration, the global reconfiguration is decomposed top-down
- Until simple enough reconfiguration step, that needs not be further decomposed

Reconfiguration patterns help finding suitable decomposition

- Well-known solutions, with well-known properties, for well-known problems

Designing a reconfiguration turns out to be: finding suitable intermediate architectures

- Issuing reconfiguration operations is known to be automatable

## Outline

## Conclusion

Evolutionary development arise need for SoS reconfiguration

Two aspects have to be dealt with:

- Model the architecture: a process yielding to DoDAF's SV-1
- Design the reconfiguration

Issues that need further work:

- Transition architecture: how do the architect specify the reconfiguration?
  - More than just initial and final architectures
  - Requirement relaxation, temporal modality
- Further experimentation and validation