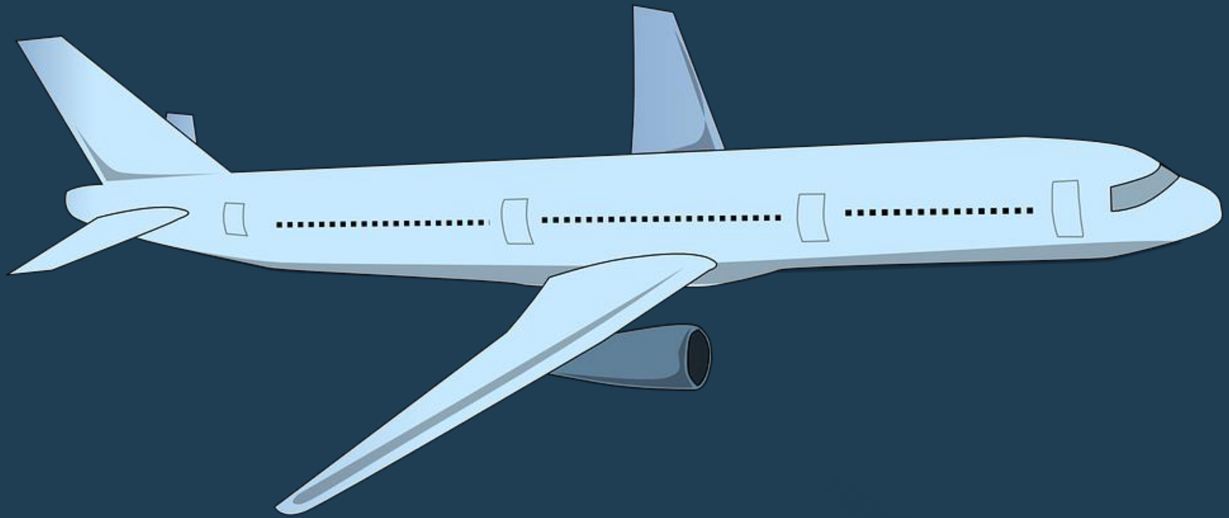
System-of-Systems as an Overarching Paradigm

Rethinking how we engineer systems in a world where IoT, Agile and DevOps are disrupting the way we think about systems

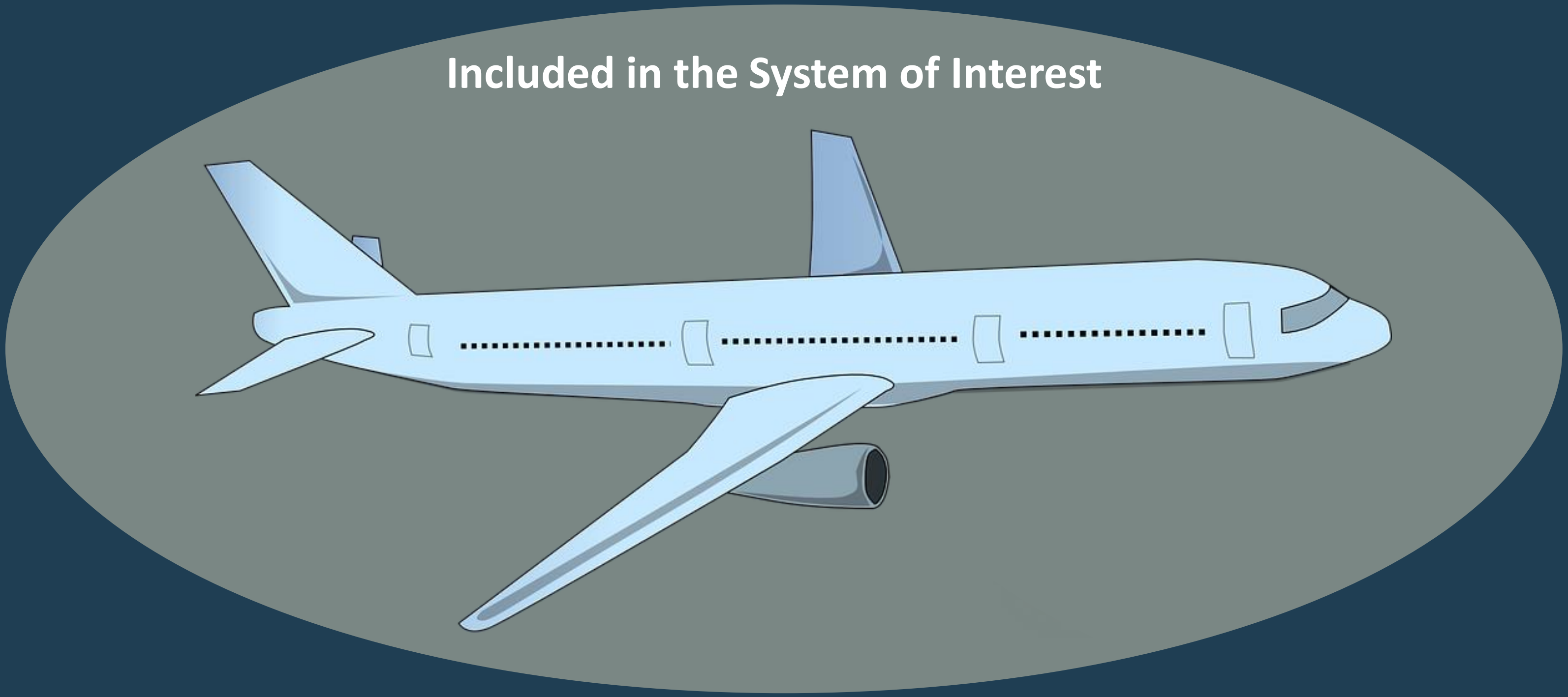
Reggie Cole
Senior Fellow, Lockheed Martin

Systems engineering is a methodical approach for managing complexity and producing trusted systems



Systems engineering has become a bedrock of practices for developing some of the worlds most sophisticated systems

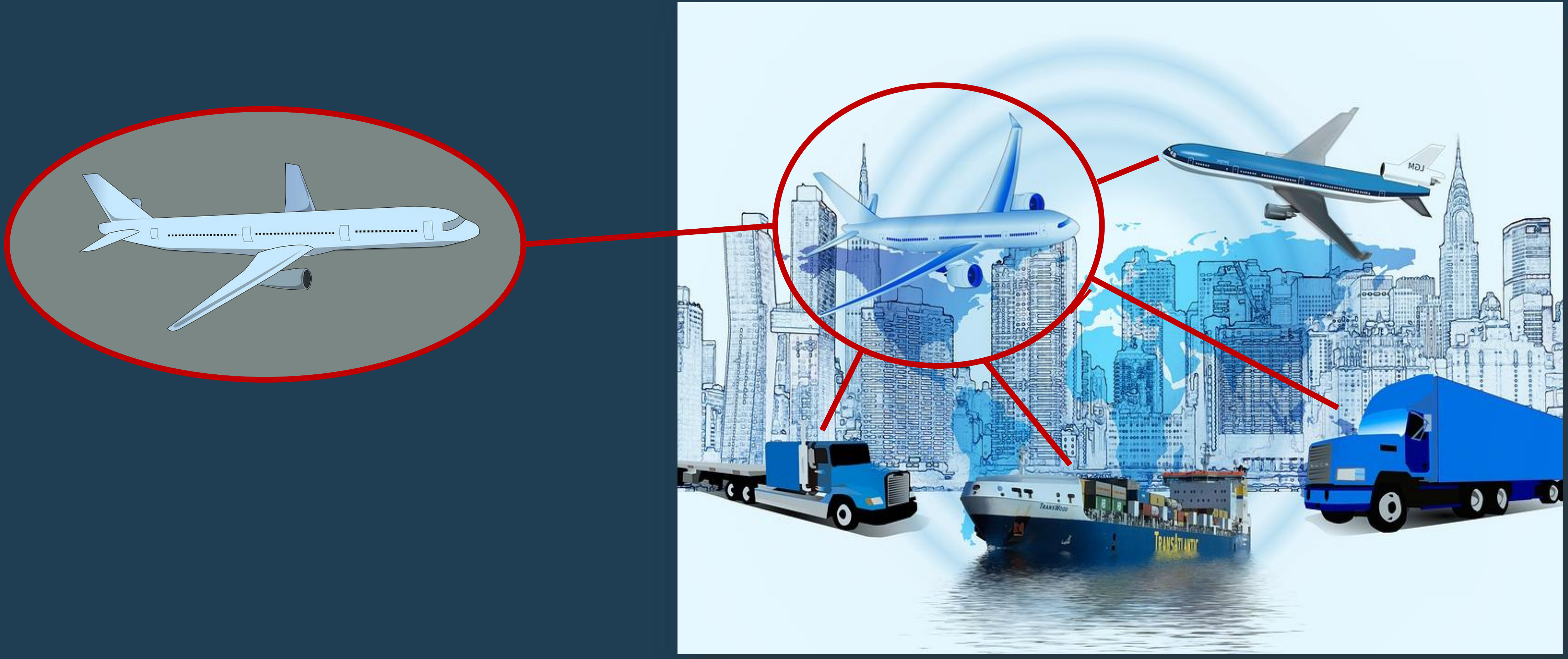
A key element in the practice of systems engineering is defining the context for what’s inside the system of interest and what’s outside



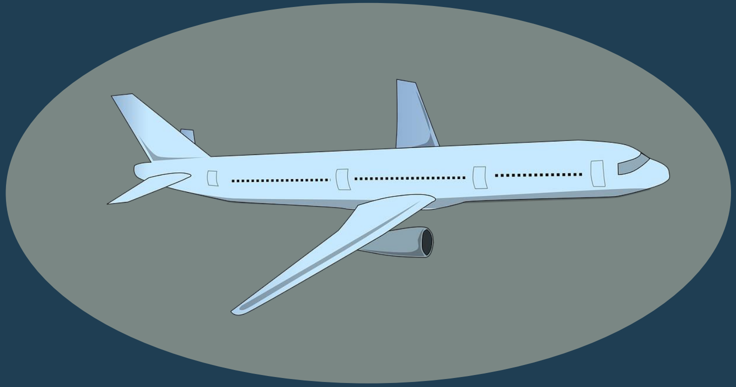
Included in the System of Interest

Outside the System of Interest

We recognize that the system of interest is often part of a larger system-of-systems — and we manage those interfaces accordingly



The field of system-of-systems engineering emerged as a way to deal with the peculiar nature of systems-of-systems

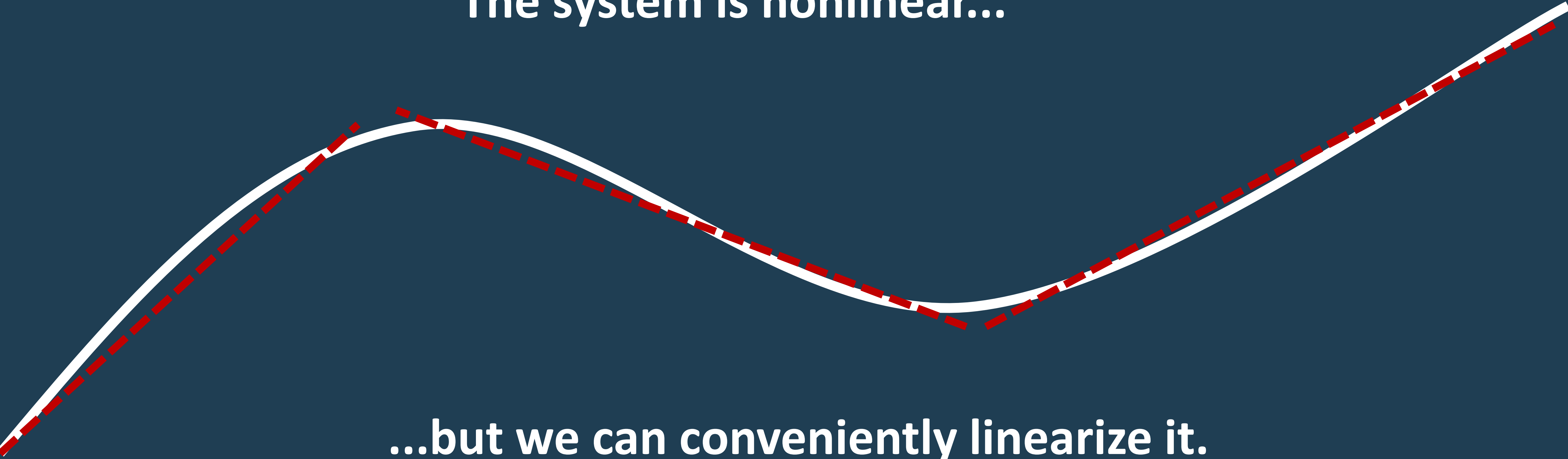


As the systems-of-systems engineering field has evolved, we treated it like a specialty of systems engineering, a special case if you will

I am starting to wonder if we don’t have it backwards.

A nonlinear system analogy:

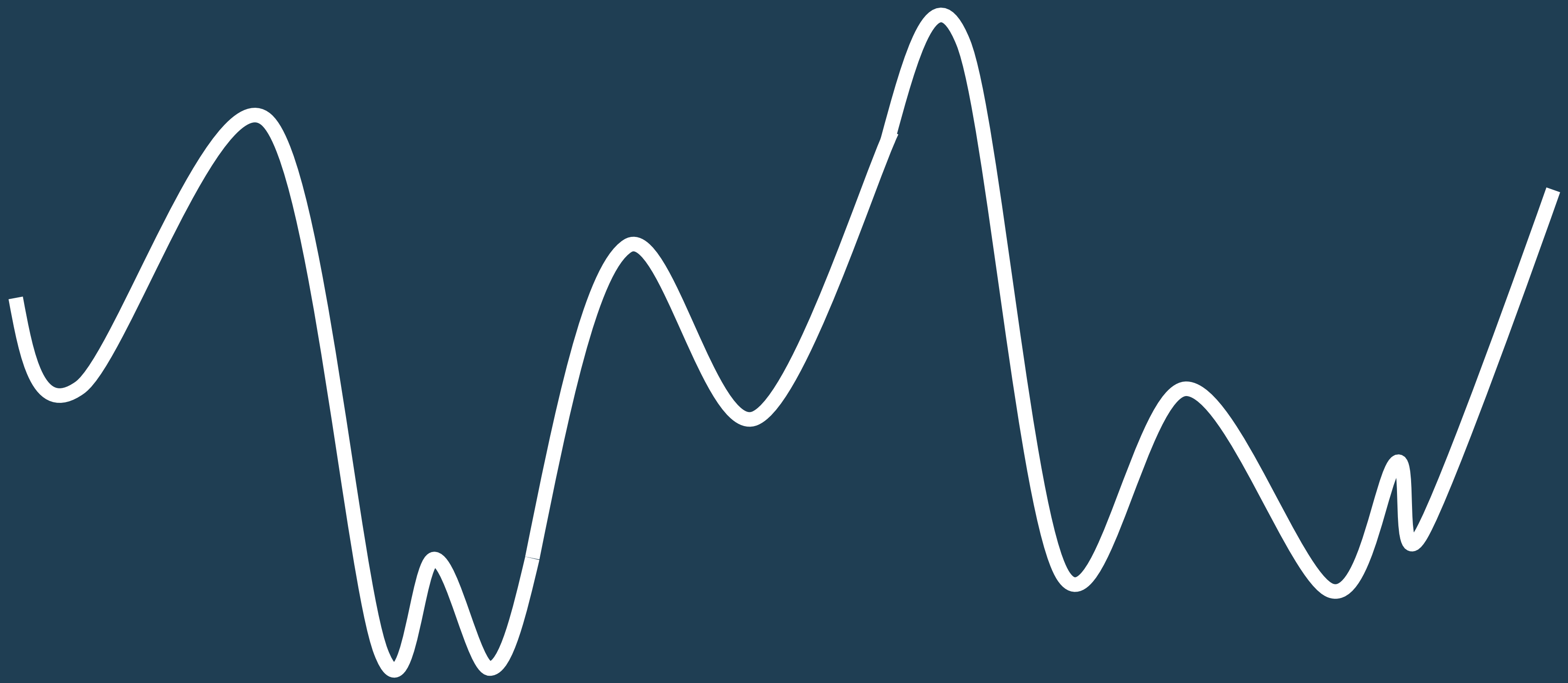
The system is nonlinear...



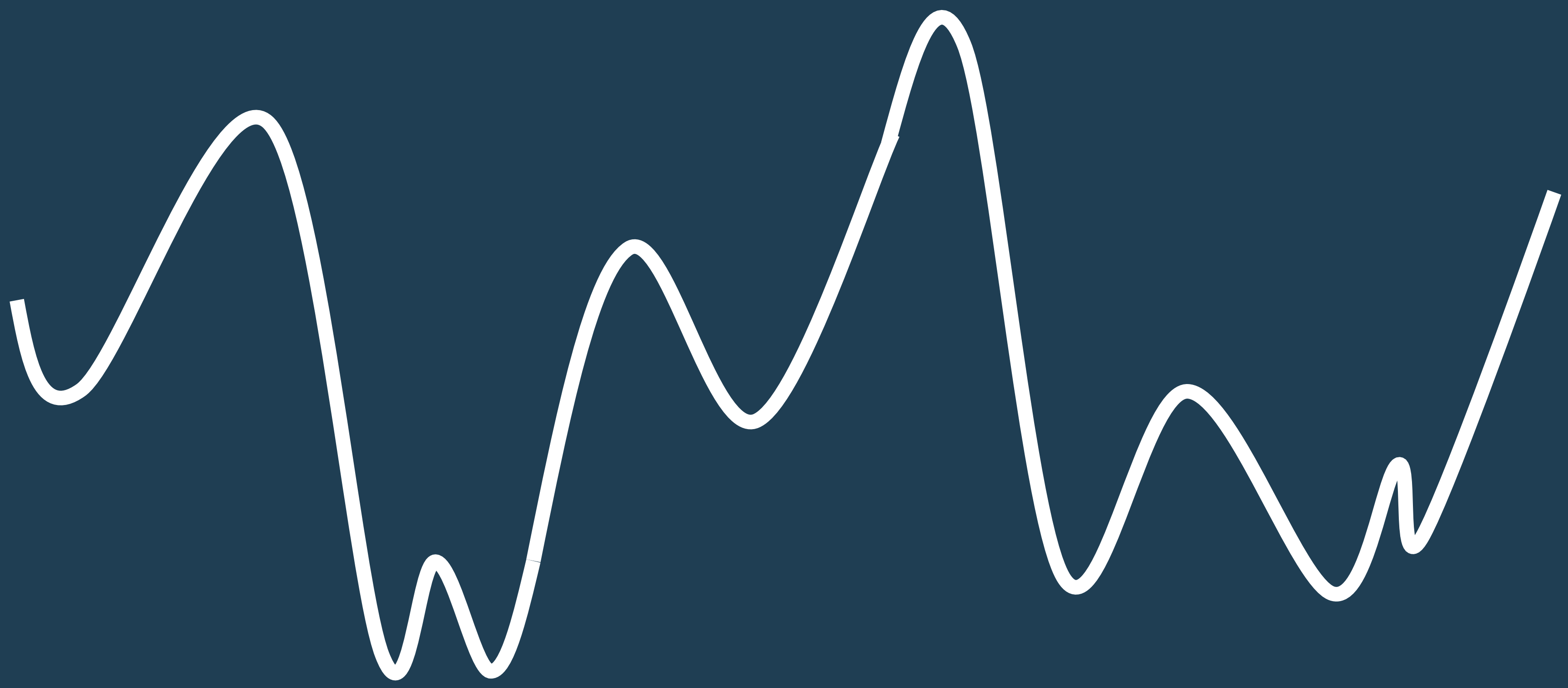
...but we can conveniently linearize it.

A nonlinear system analogy:

As our nonlinear system becomes time-compressed...

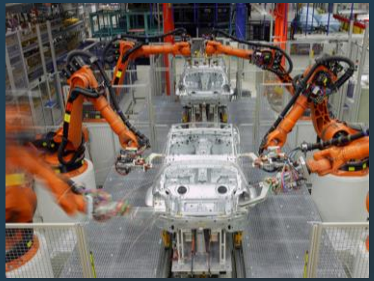


...opportunities for linearization are limited



What if this is what’s happening to systems engineering?

The distinction between systems and systems-of-systems is not hard and fast — it really is a continuum



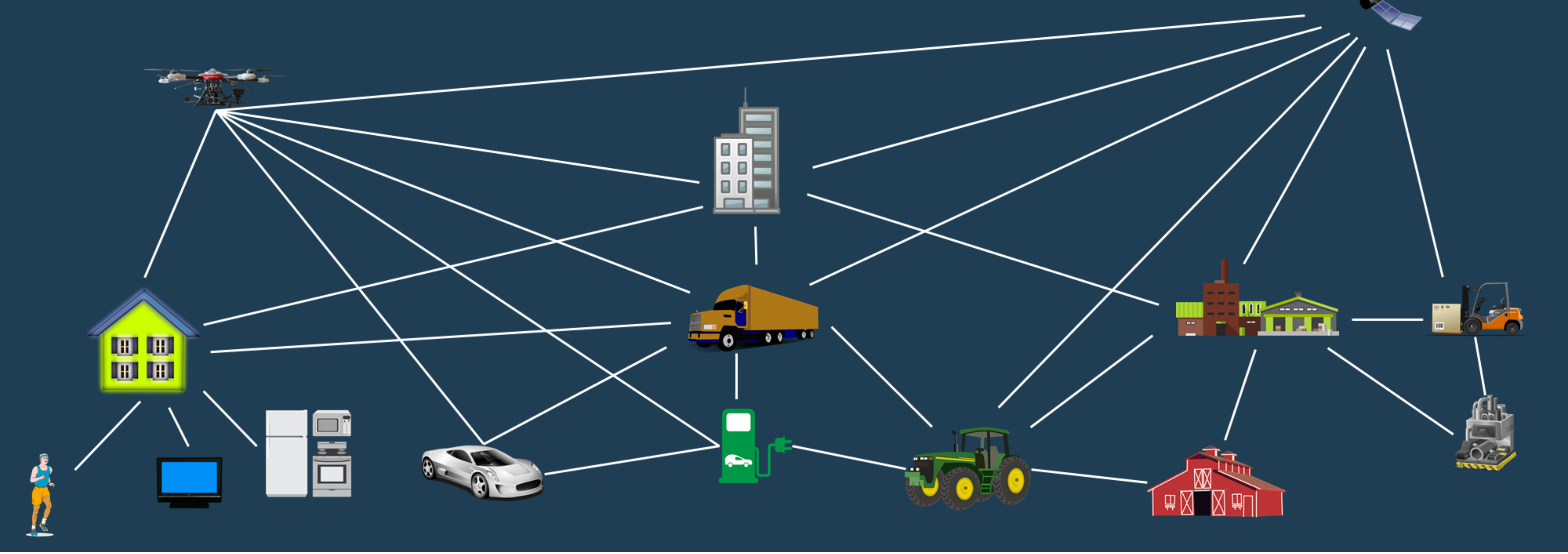
Clearly a System

Increasing System-of-Systems Characteristics

Clearly an SoS

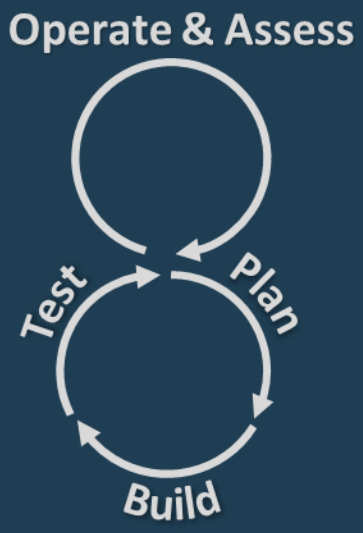
Added to this continuum are the disruptive factors of IoT, Agile and DevOps

Internet of Things



The extension of internet connectivity into physical and cyber devices is driving widespread and accelerating change into system environments.

Agile Development



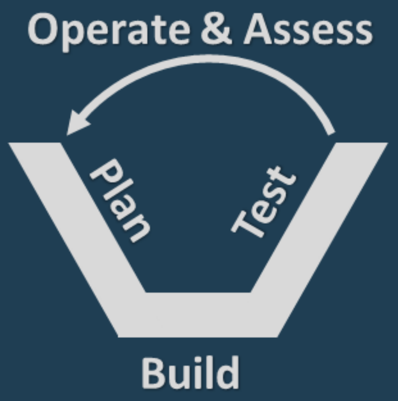
Agile Model

Days/Weeks

These don’t look so different...

The real difference is velocity!

Versus



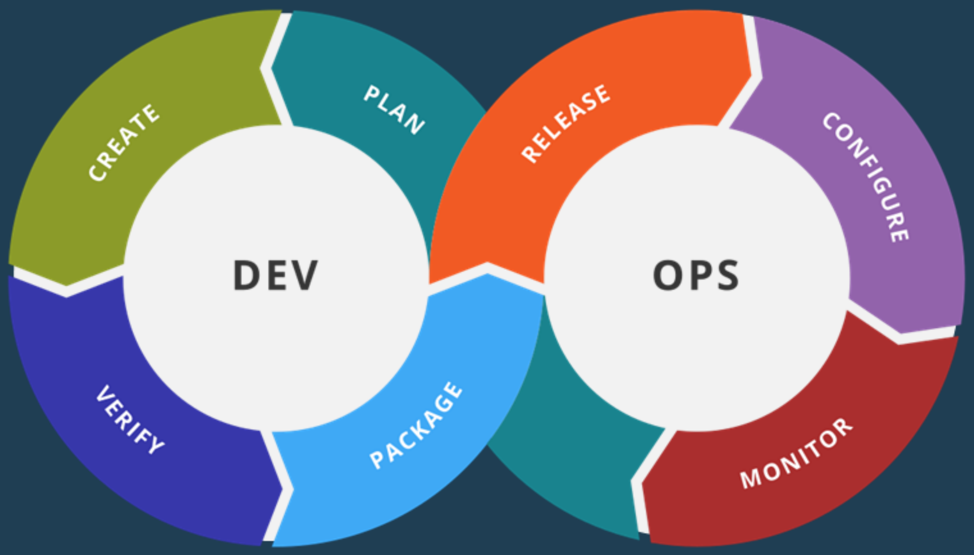
Systems Engineering Model

Months/Years

Agile development is creating a system environment in which the velocity of the “lather-rinse-repeat cycle” results in nearly-continuous change

DevOps Delivery

Seamless Pipeline of Releases to Uses

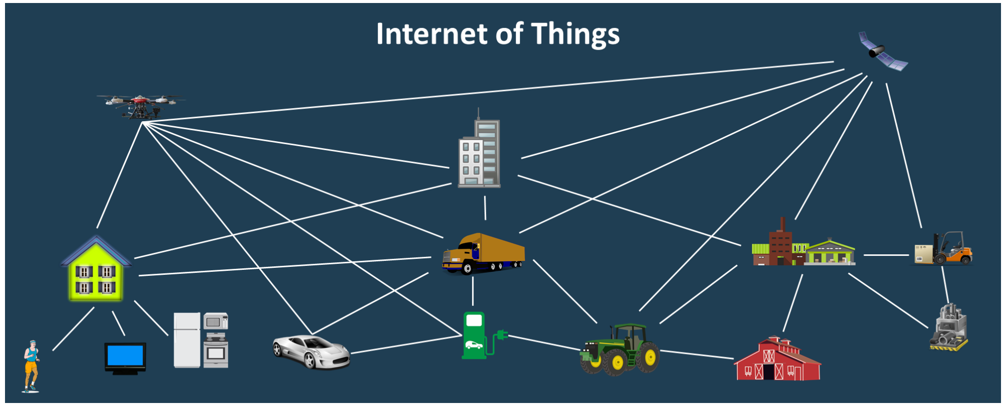


Daily and Even Hourly Releases

Adds Additional Velocity on Top of Agile Development

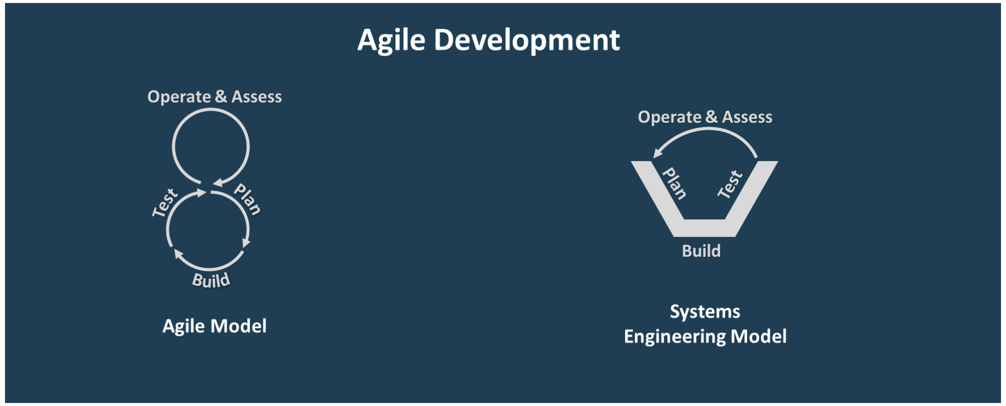
The velocity of the DevOps delivery model adds another level of velocity — turbo-charging the agile development model — on top of an already chaotic system environment

Key Disruptions in the Systems Engineering Environment



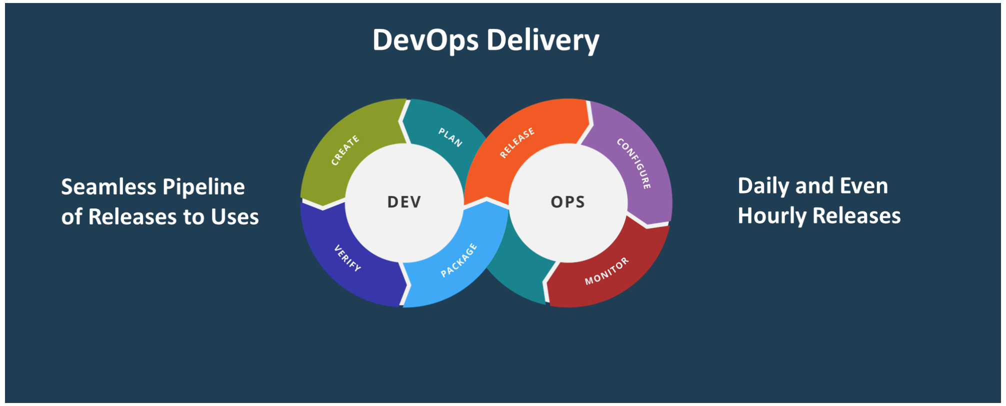
Internet of Things

Increasing Technological Volatility



Agile Development

Increasing Operational Volatility

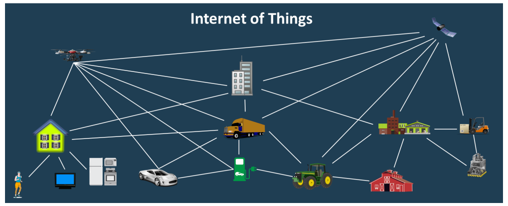


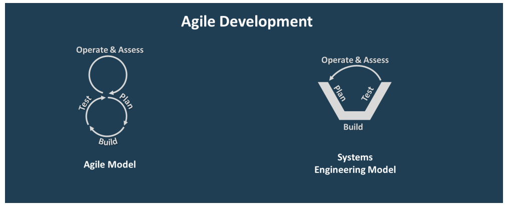
DevOps Delivery

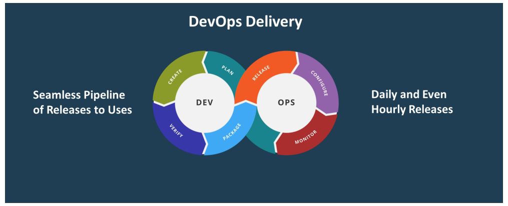
Seamless Pipeline of Releases to Uses

Daily and Even Hourly Releases

How is the system environment changing?


Internet of Things


Agile Development


DevOps Delivery

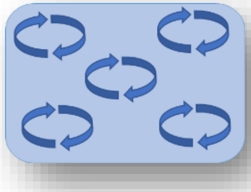
Seamless Pipeline of Releases to Uses
CREATE
PLAN
DEV
VERIFY
PACKAGE
RELEASE
CONFIGURE
OPS
MONITOR
Daily and Even Hourly Releases

The Key Implications

- Explosion of IoT Devices & Connections
- Continuous Changes to Systems
- Increasing Technological Volatility
- Increasing Operational Volatility

Hyperconnectivity — To say that everything is connected to everything is not too far from reality

Technological Disruption — The rate of technological change means that as soon as we establish a baseline, there is pressure to upgrade



Operational Disruption — Operational / business change is pushing system developers to adopt an agile development model

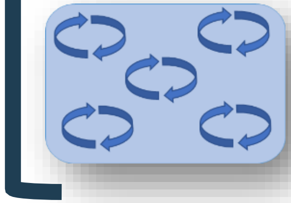
One More Big Change

Hyperconnectivity

To say that everything is connected to everything is not too far from reality

Technological Disruption

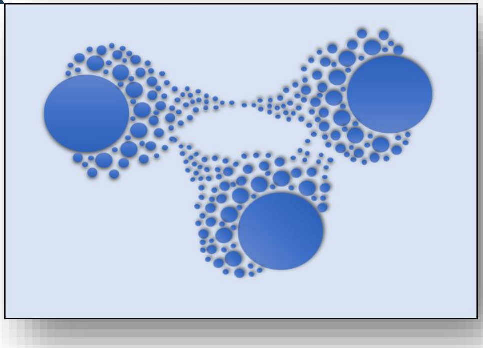
The rate of technological change means that as soon as we establish a baseline, there is pressure to upgrade



Operational Disruption

Operational / business change is pushing system developers to adopt an agile development model

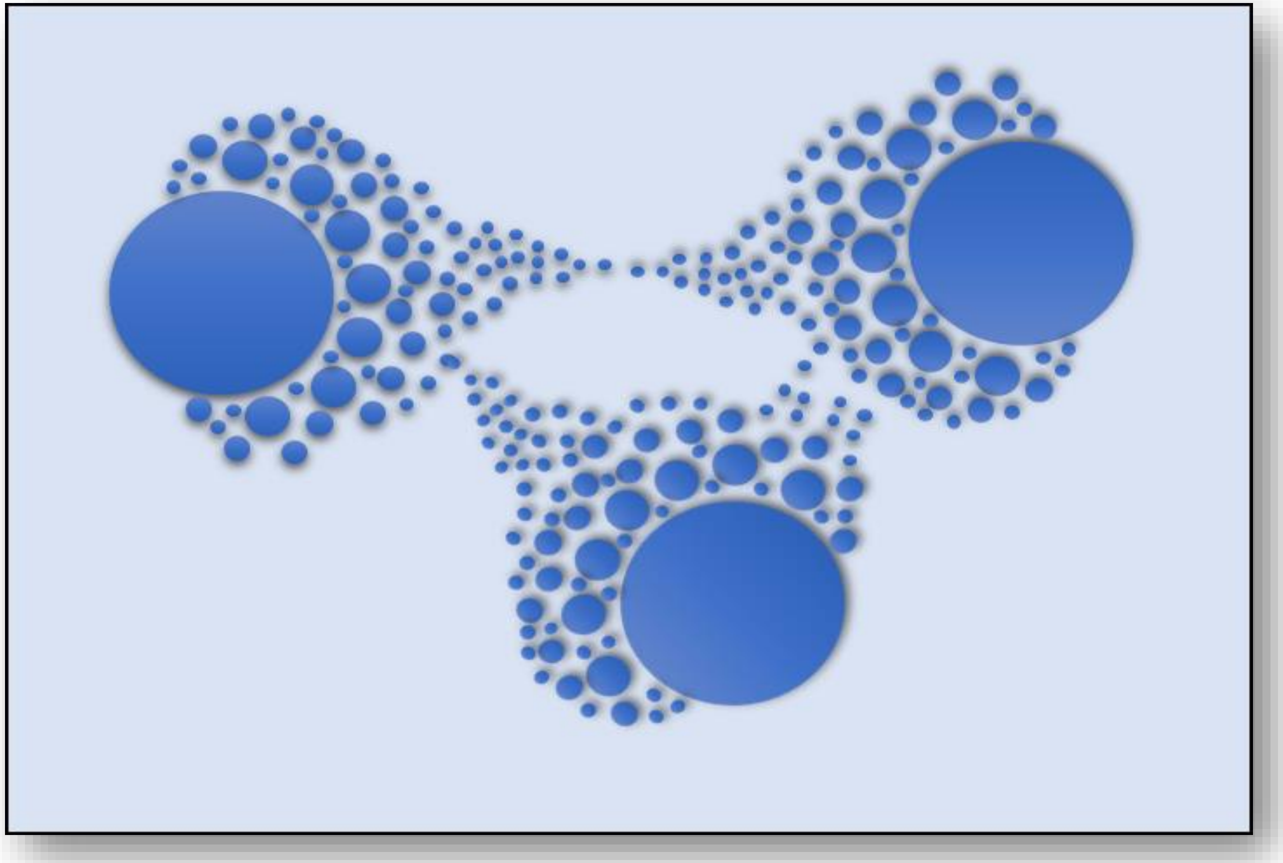
Collectively, these lead to a fourth big implication



Porous System Boundaries

Distinct boundaries between systems is starting to erode, creating porous system boundaries — that is leading to an erosion in system context

Destabilization of System Context

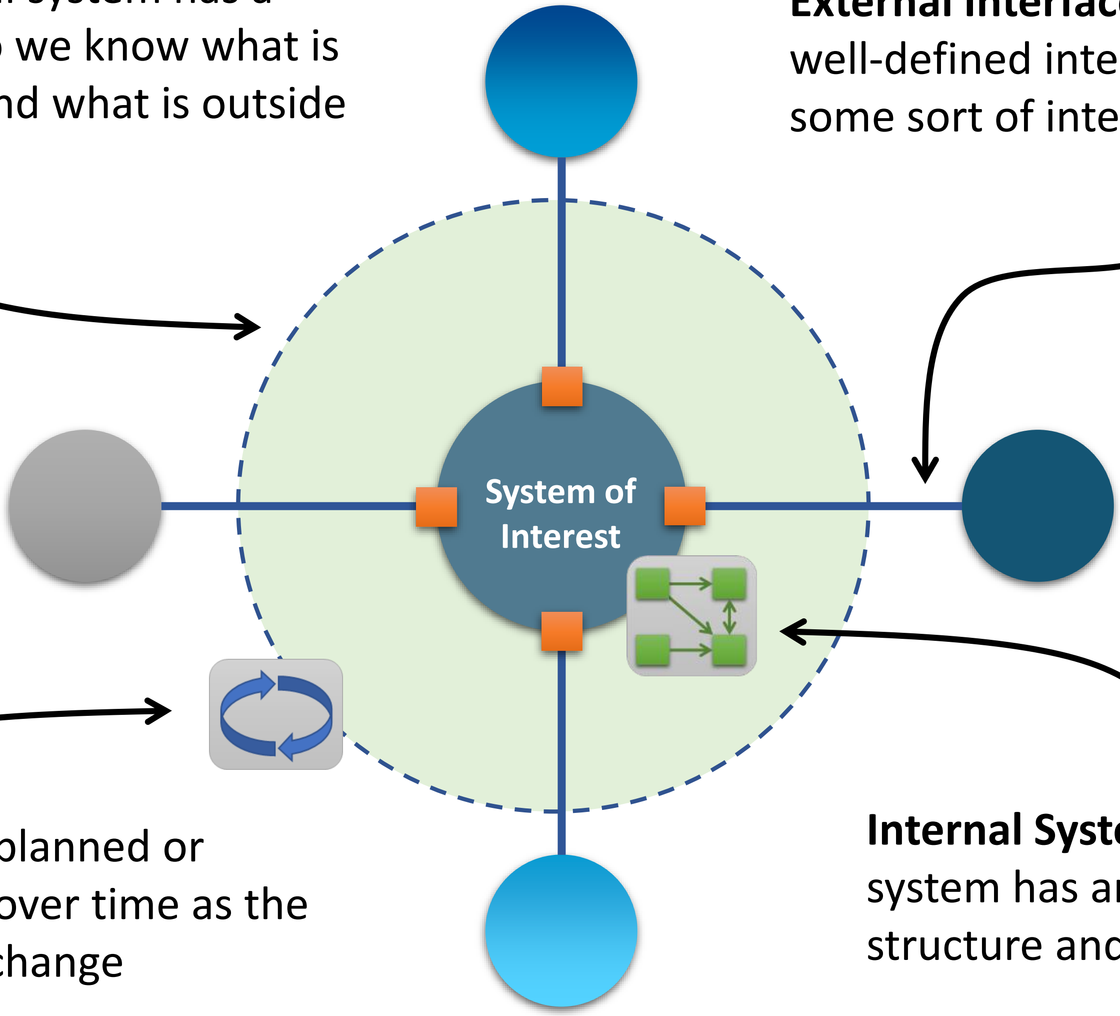


Destabilization of System Context Changes Everything!

The Basic Anatomy of a Traditional System

System Context — A traditional system has a well-defined system context so we know what is inside the System of Interest and what is outside

External Interfaces — A traditional system has well-defined interfaces that are managed using some sort of interface agreement



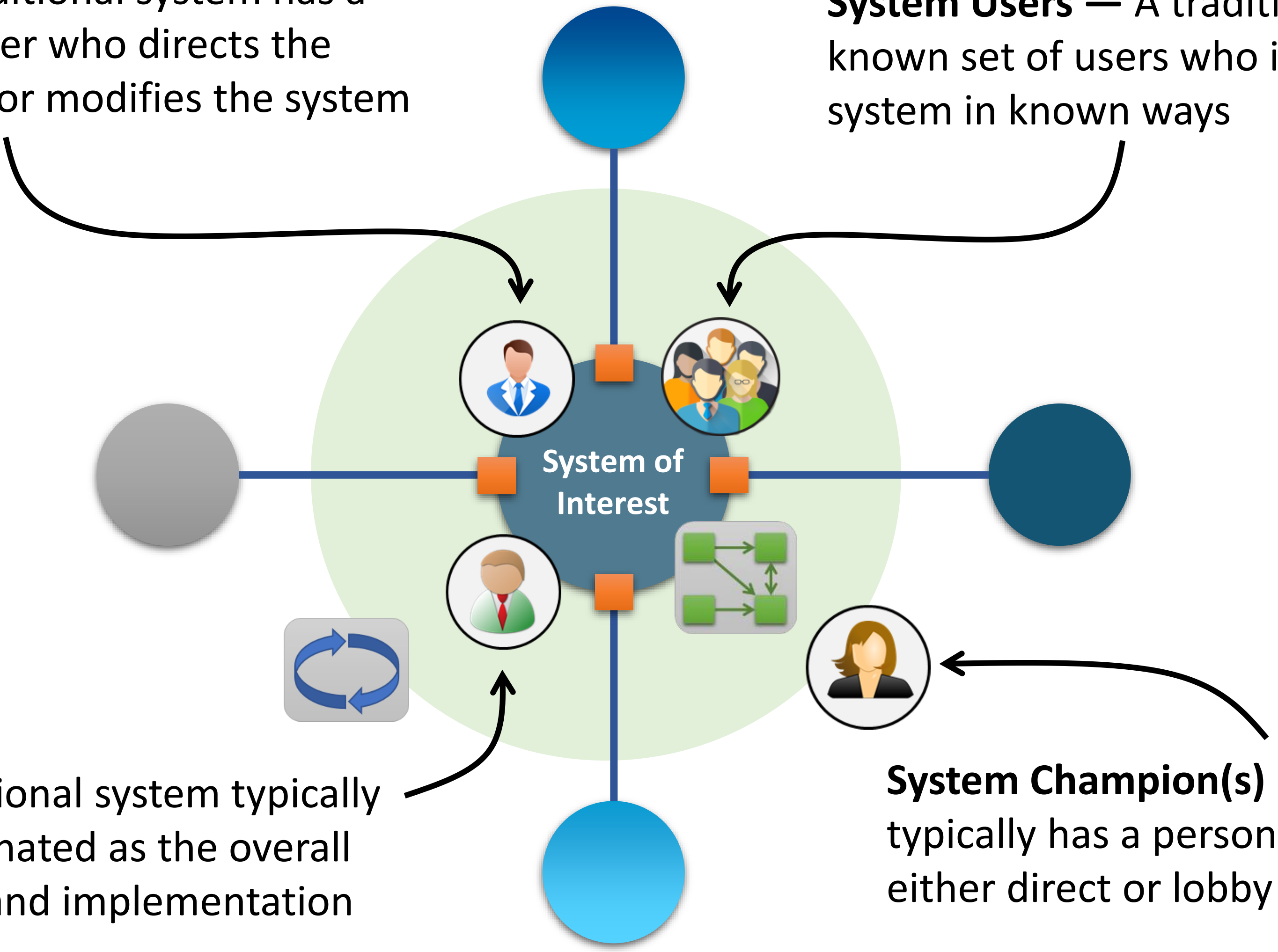
System Evolution — Whether planned or unplanned, all systems evolve over time as the needs of system stakeholders change

Internal System Architecture — A traditional system has an internal system architecture, the structure and behavior of the system

The Anatomy of a Traditional System — With People

System Manager — A traditional system has a designated system manager who directs the team that designs, builds or modifies the system

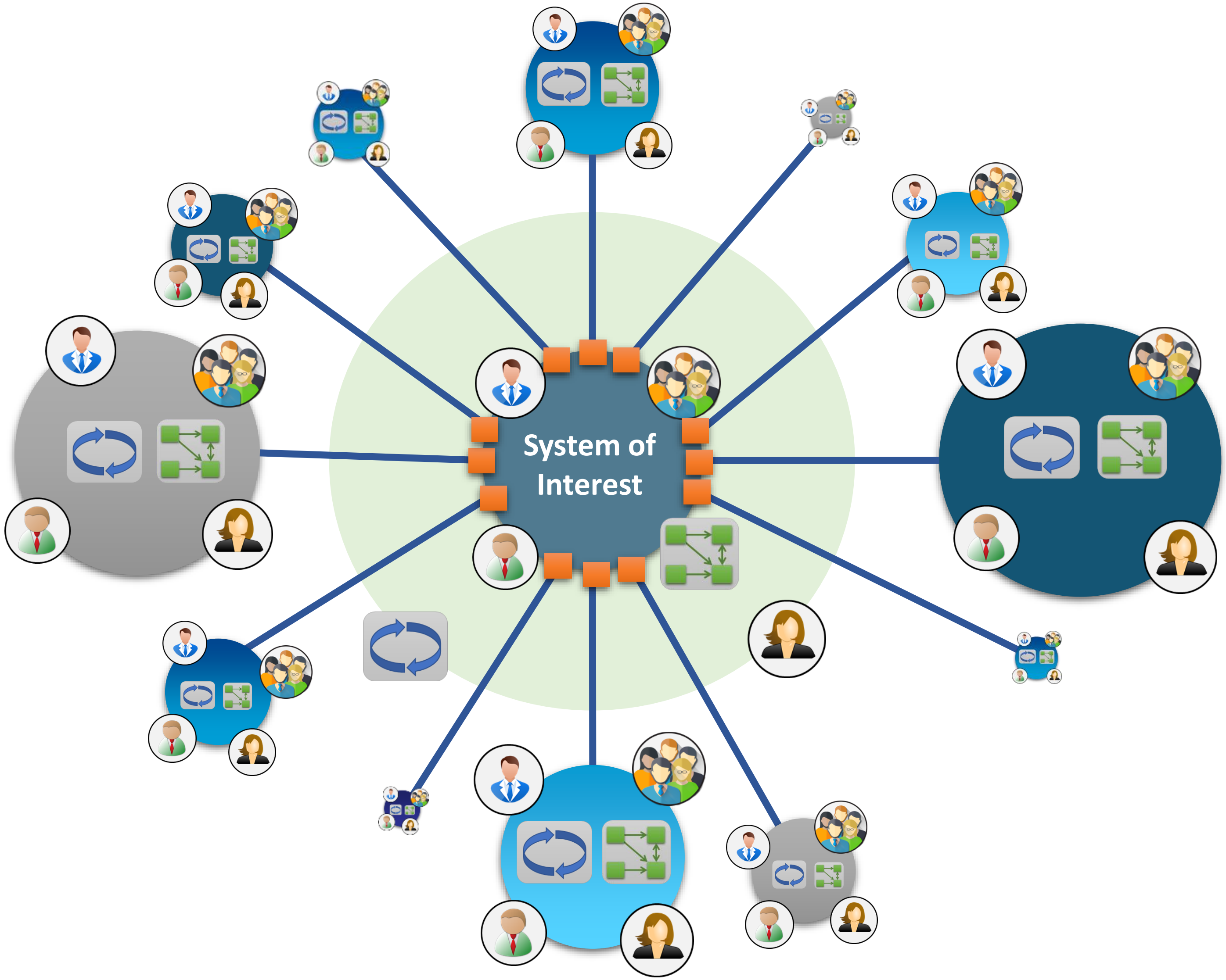
System Users — A traditional system has a known set of users who interact with the system in known ways



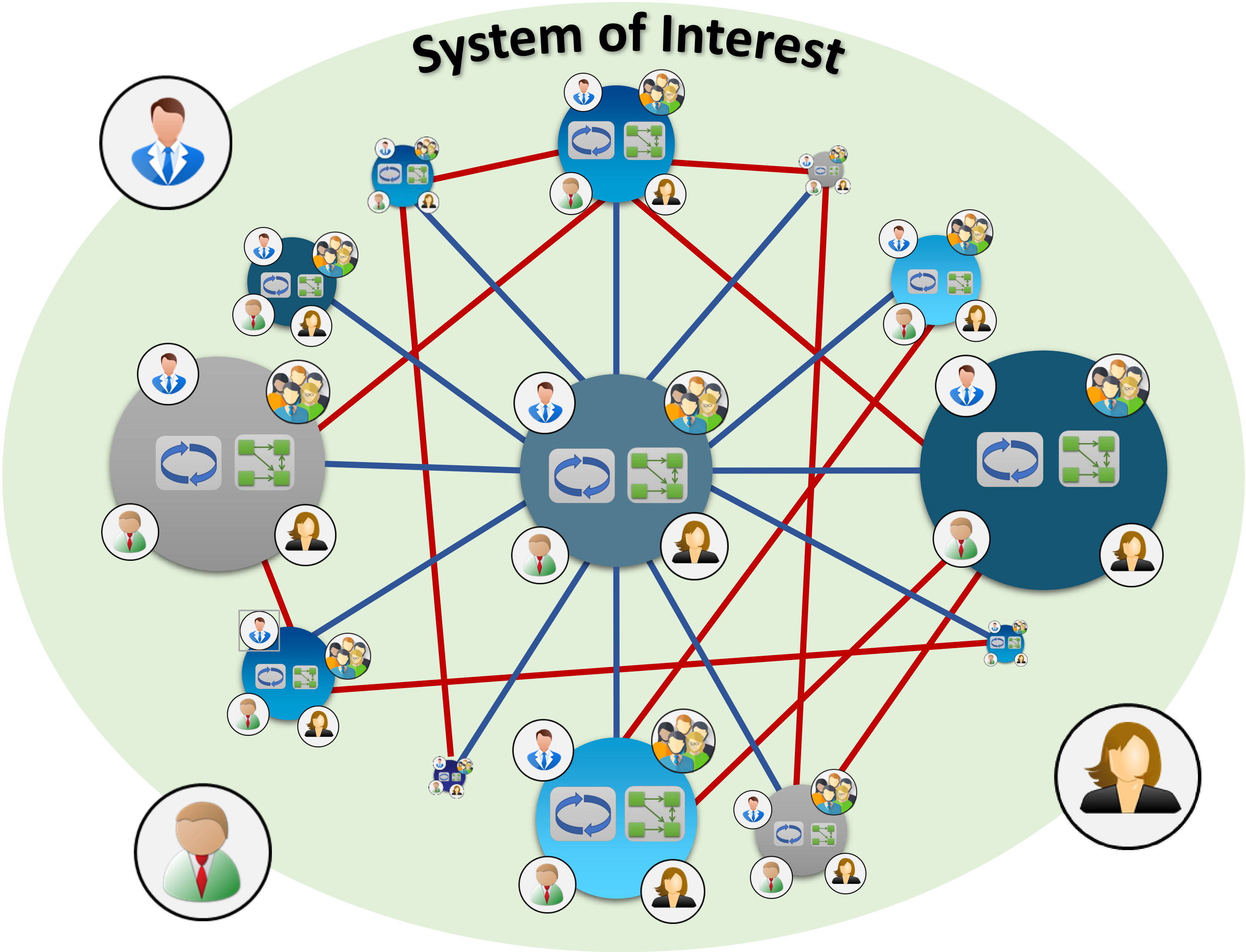
Chief Engineer — A traditional system typically has a person who is designated as the overall technical lead for design and implementation

System Champion(s) — A traditional system typically has a person (or several people) who either direct or lobby for system purpose/changes

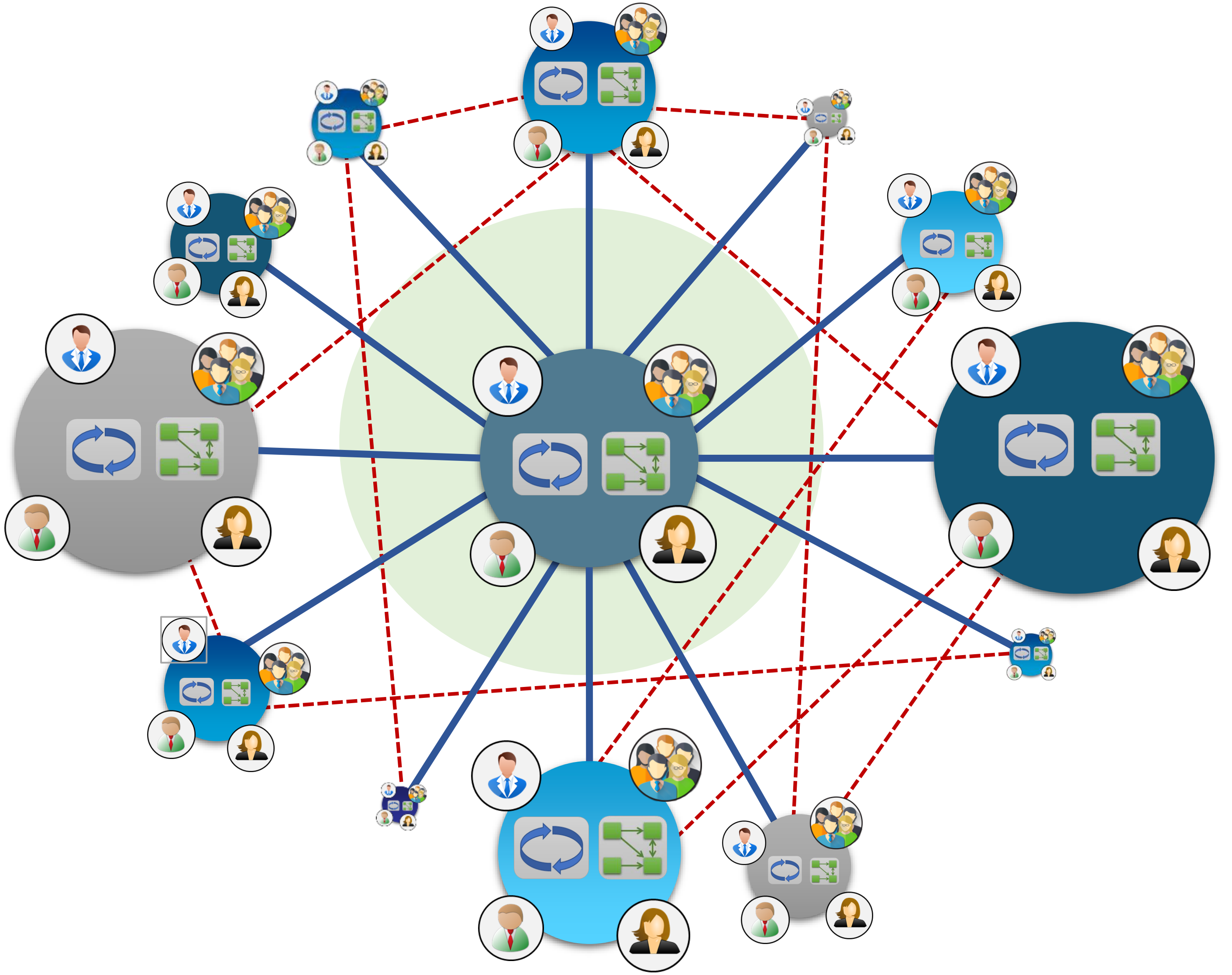
A More Realistic Traditional System



Our System in a System-of-Systems

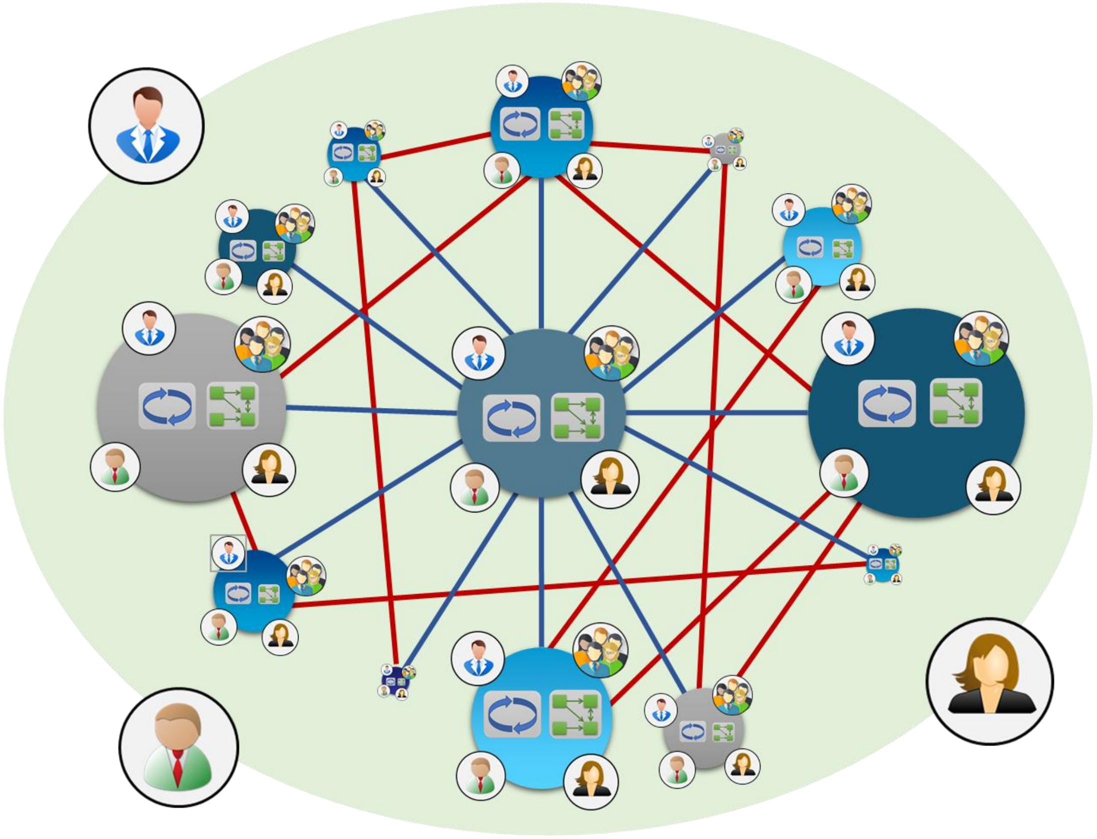


This is Really What Our System Looks Like

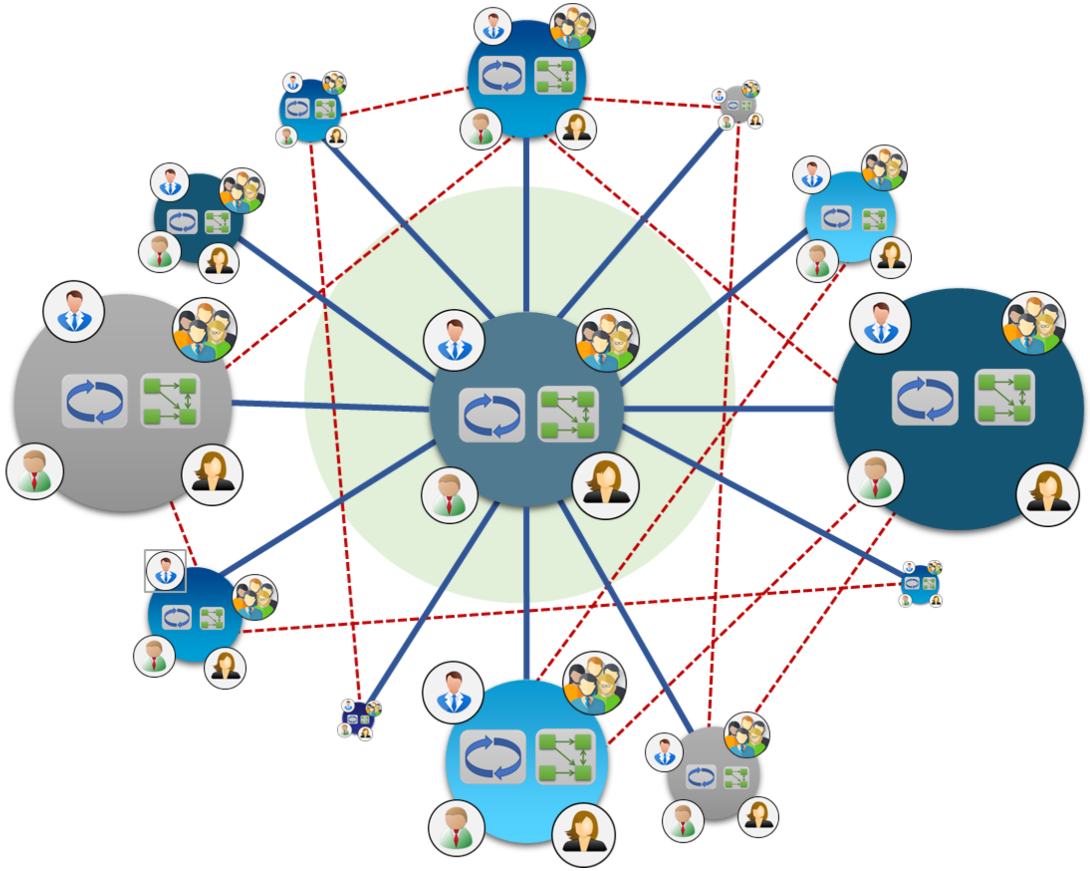


What is the Difference?

System-of-Systems

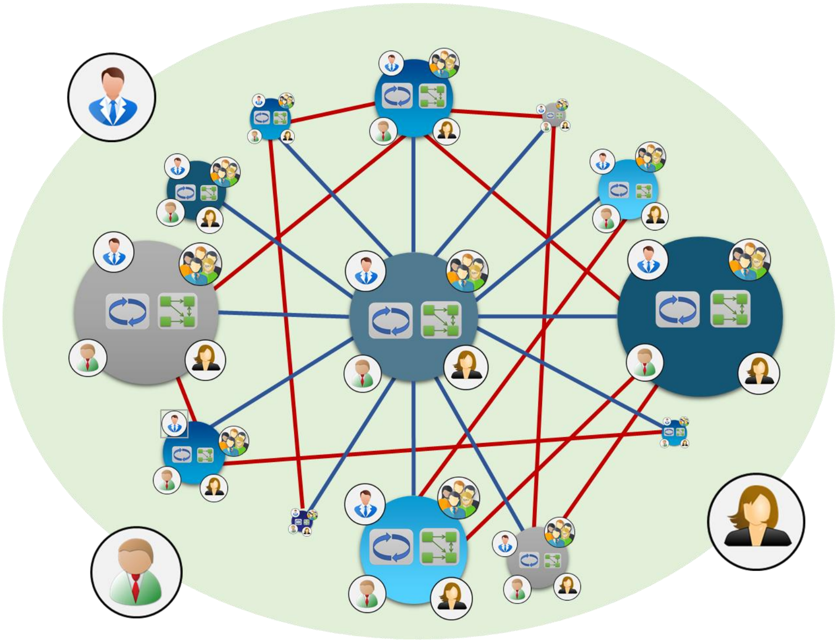


Traditional System



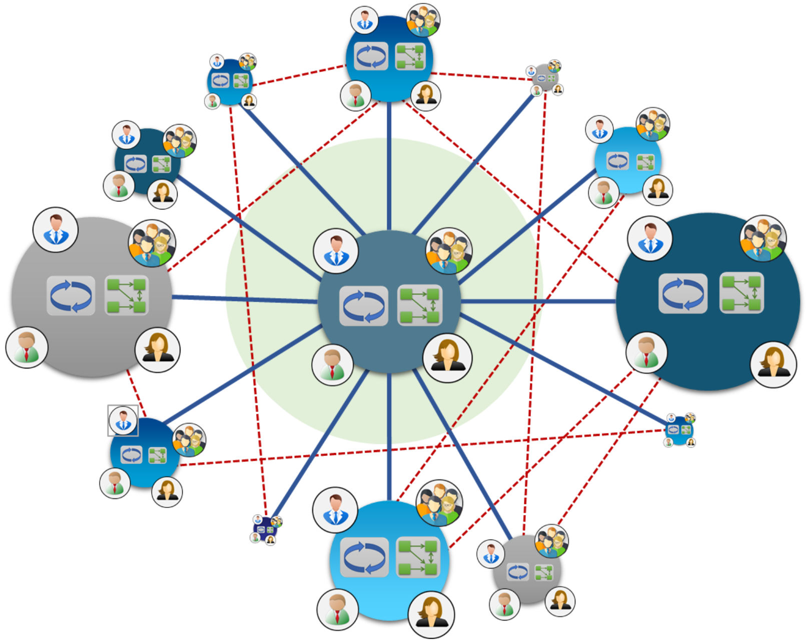
What is the Difference?

System-of-Systems



- This is (at a minimum) an acknowledged SoS
- Perhaps it's even a directed SoS
- There is an acknowledged chief engineer
- The chief engineer may have directive authority
- There is an acknowledged system manager
- The system manger may have directive authority
- The system manger may have budgetary authority

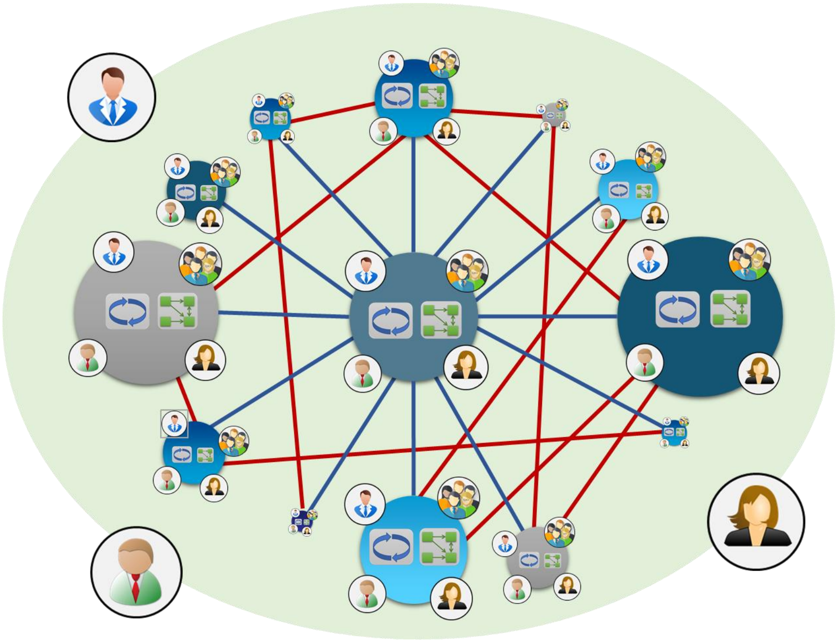
Traditional System



- This is a defacto SoS, like it or not
- Some of the participants may understand that
- Other participants probably will not
- It might even be a collaborative SoS
- No chief engineer or system manager
- No governed management of interfaces
- SoS-level design is based on consensus

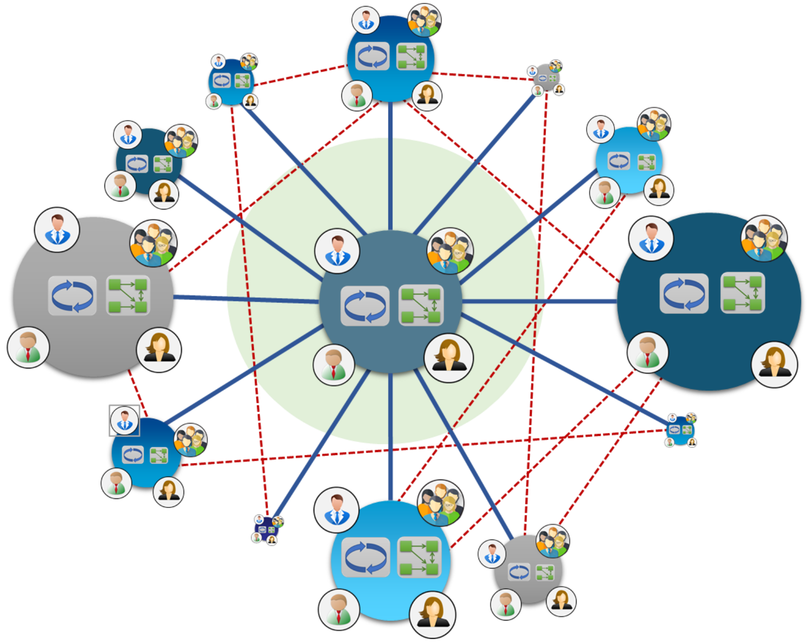
What is the Difference?

An "Official" System-of-Systems



This is still an exceptional case

An "Unofficial" System-of-Systems

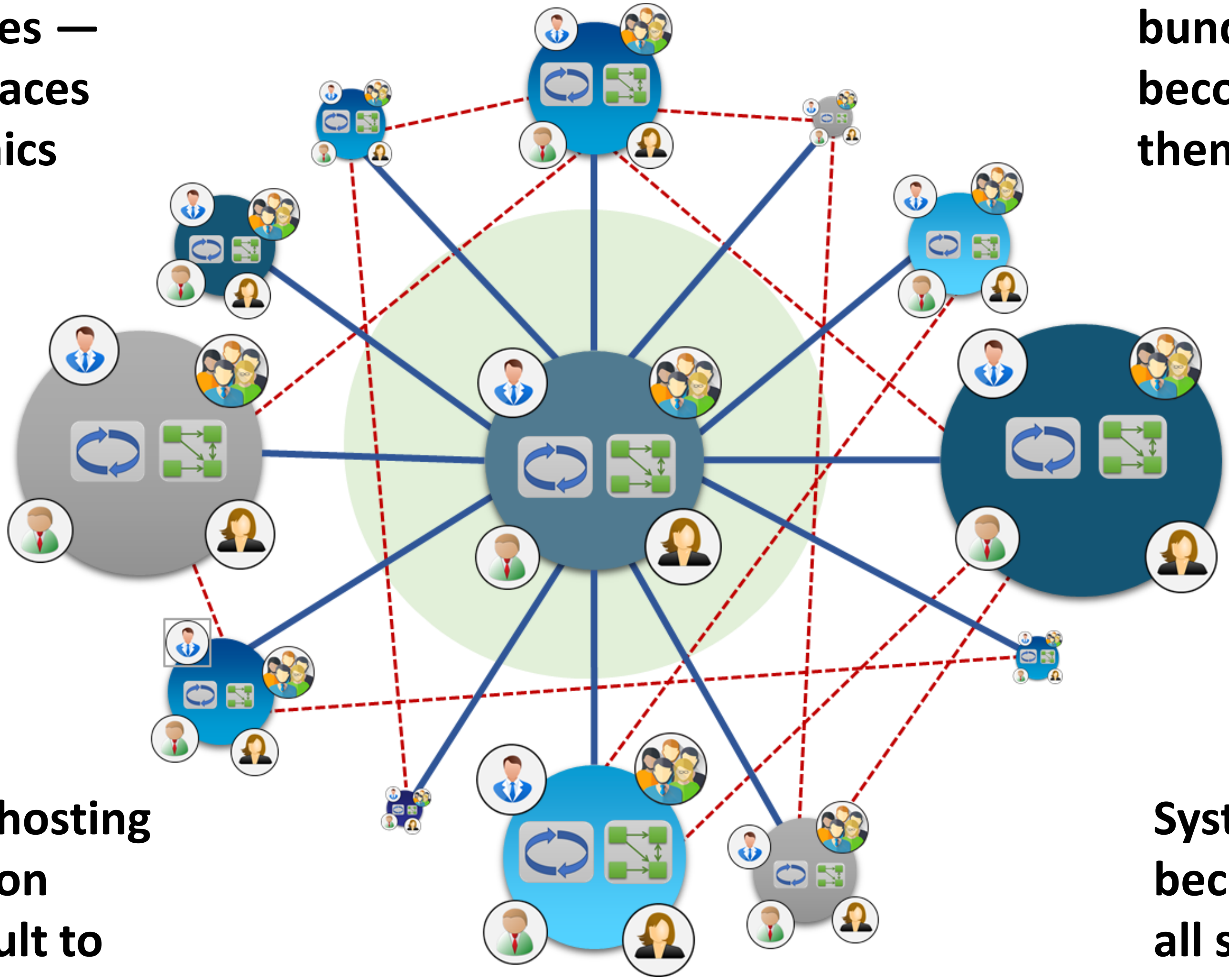


This is becoming more prevalent — in fact, it's becoming the norm

This is the New Anatomy of a System

The primary interfaces are still the primary interfaces — but the secondary interfaces affect the overall dynamics of the system

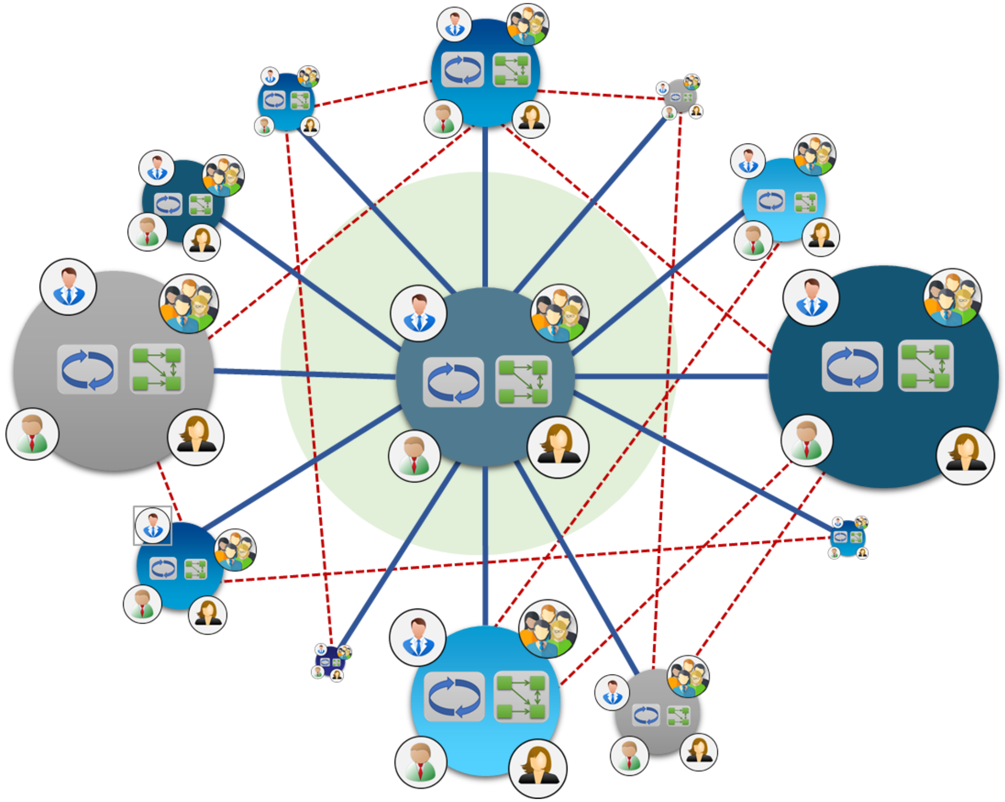
System interfaces are often bundled as services and it becomes difficult to fully isolate them from other interfaces



The trend toward cloud hosting of systems and connection points can make it difficult to separate the concerns of one system from another

System-of-systems dynamics becomes a primary concern for all systems, not just what we think of as systems-of-systems

How This Perspective Changes Systems Engineering



Requirements — Requirements allocation, management and decomposition must account for a loss of "control."

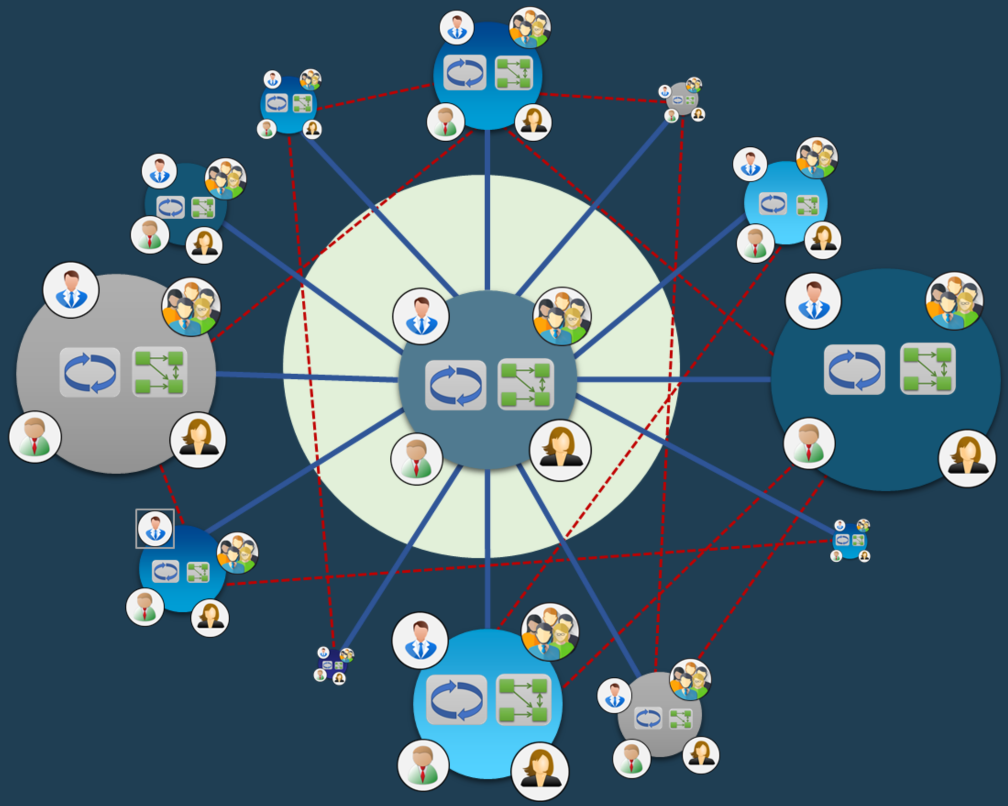
Analysis of Alternatives — Alternatives can no longer be evaluated from a closed-system perspective and need to be continuously re-evaluated.

System Design — System designs need to be evaluated from a system-of-systems perspective, i.e. system dynamics in a system-of-systems context.

Interfaces — System interface management needs to adjust to a more coupled and dynamic environment.

Test & Evaluation — System-of-systems evaluation needs to be the norm, not a special case or excursion.

Change Management — Change management will necessarily become more collaborative.



It’s time to accept that very few (if any) systems actually exist outside of a system-of-systems.

It’s time to consider system-of-systems engineering as an overarching paradigm for all of systems engineering.

Thank you!

I can be reached for follow-ups via email:

reggie.cole@lmco.com