TU Wien

A Methodology for the Design of System of Systems (SoSs)

H.Kopetz

January 2011

Outline

- Introduction
- System of Systems
- Architectural Style
- Autonomic Component
- Emergence
- Conclusion

Why are SoS Becoming so Important?

The available technology (e.g., the Internet) makes it possible to interconnect *independently developed systems* (*legacy systems*) to form new *system-of-systems (SoS)* .

The integration of different *legacy systems* into a SoS promises more efficient economic processes and improved services.

Examples: Power distribution, Car-to-car communication, air-traffic control, banking systems

Monolithic System versus SoS

	Monolithic System	System of Systems (SoS)
Sphere of control	single organization	different organizations
Subsystems are	obedient	autonomous
Composability mechanism	integration *control*	interoperation *influence*
Interface Control	single organization	international standard
Structure	hierarchy	mesh
Goal orientation	single goal	multiple goals
Evolution	coordinated	not coordinated
Emergence	controlled	unanticipated
Architectural Style	same in all subsystems	different

.

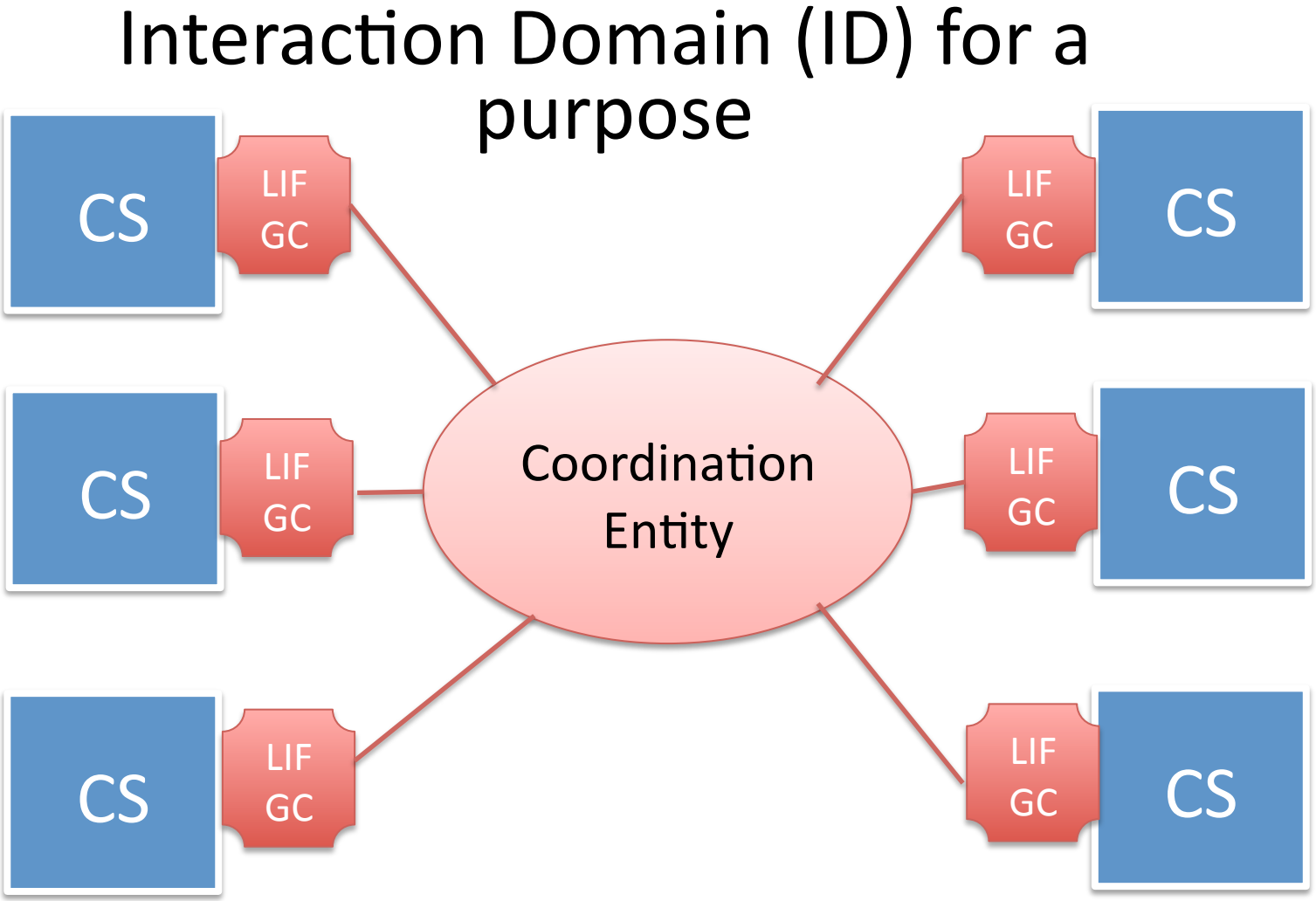
Purpose of an SoS

Constituent Systems (CS) are integrated by an Interaction Doman (ID) to realize a well-defined purpose:

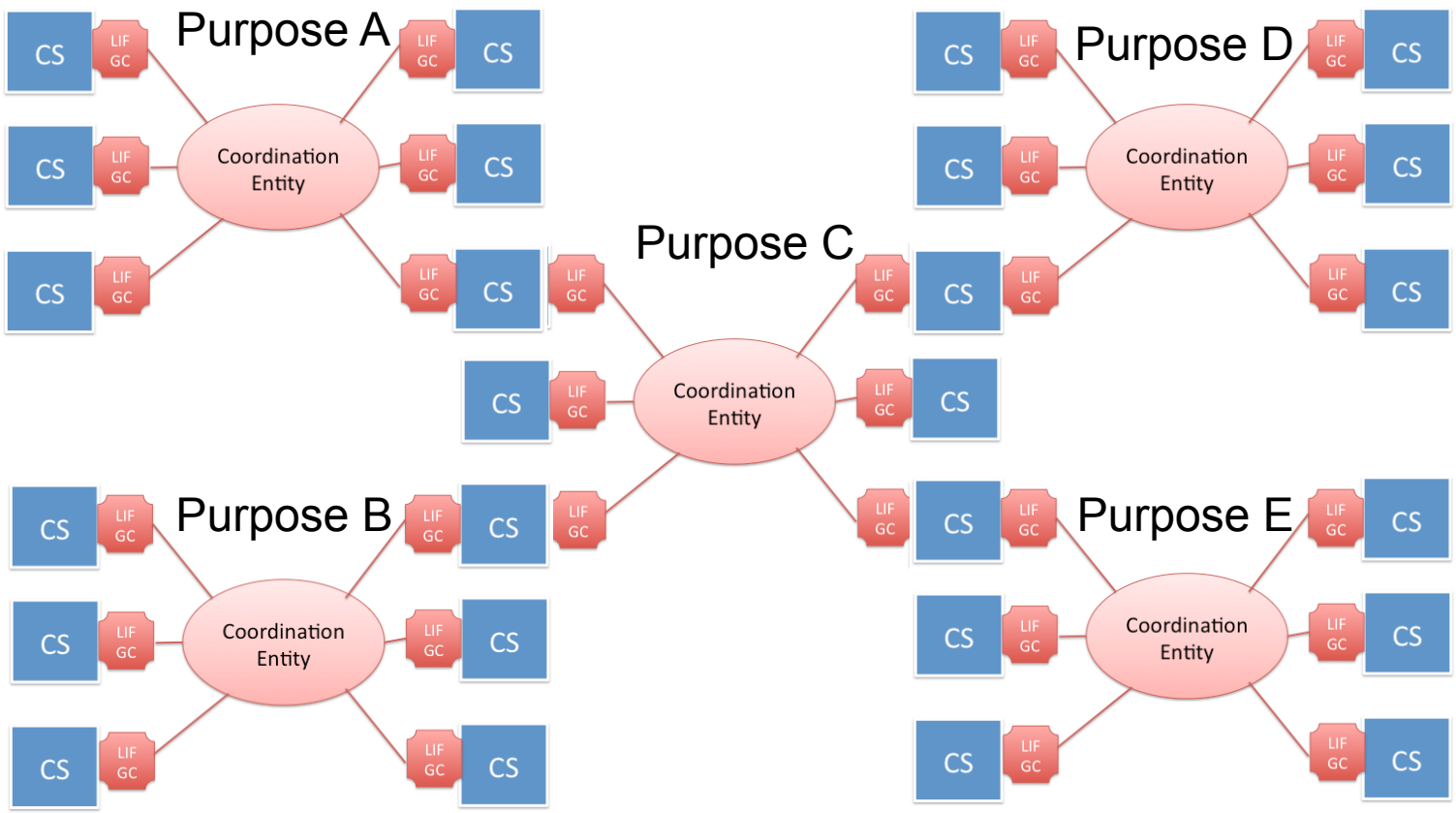
- The purpose is achieved by *voluntary cooperation* and not by *enforced control*.
- The SoS must provide incentives that a CS contributes to the purpose of the SoS.
- In an SoS it can be chosen dynamically which CS is to provide the needed services.

Example: Electric utility energy provider

Structure of an SoS



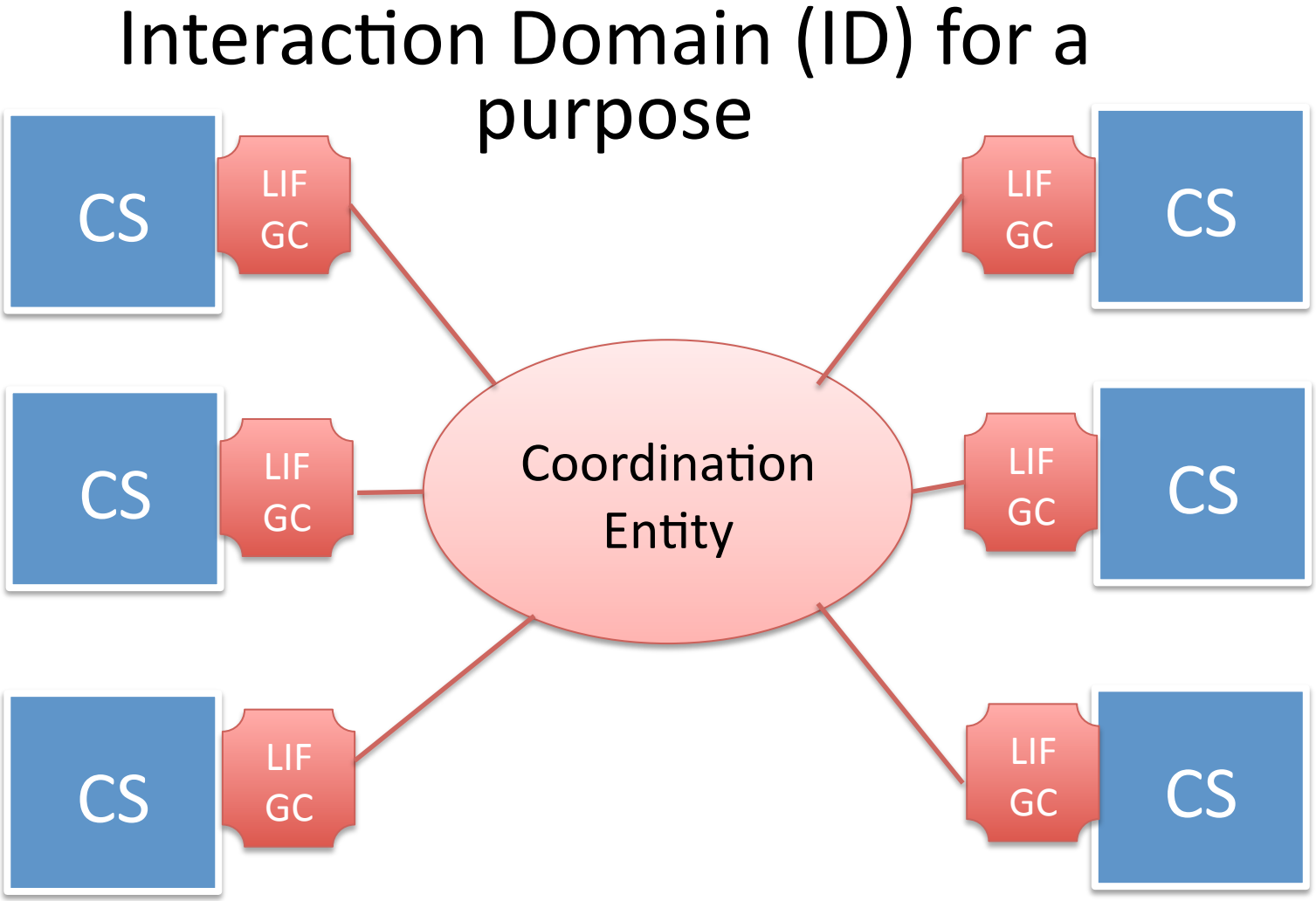
A CS can be involved in more than one ID



Evolution

- Constituent (legacy) systems evolve according to their own objectives, which are not always in line with the SoS objectives.
- The SoS must cope dynamically with this divergence of objectives and try to optimize the overall utility.

Structure of an SoS



Services of the Coordination Entity (CE)

- *Coordination service*
- *Configuration service*
- *Security service*
- *Discovery service*
- *Diagnostic services*
- *Continuous validation service*

The *relied upon interface properties* of the Linking Interfaces must be precisely specified and continuously monitored –requires a precise gateway LIF specification.

Linking Interface (LIF) Specification

The integration of the legacy systems is achieved by message exchanges across well-defined *Linking Interfaces (LIF)*.

A linking interface specification consists of three parts:

- Transport specification
- Syntactic Specification
- Semantic Specification

Transport LIF Specification

The transport of *uninterpreted bit streams* is the subject of the *transport specification*.

It covers:

- Addressing
- Authentication of sender
- Temporal issues (timing, flow control, congestion control)

The Internet protocols provide a universal solution to the transport problem of non time-critical data.

Syntactic LIF Specification

The syntactic LIF specification establishes the *syntactic interoperability* of legacy systems:

- structures the *bit stream* of a message into *syntactic units*
- assigns local *names* to the syntactic units
- Involves the end systems, not the transport system

Semantic LIF Specification

The semantic LIF specification establishes the *semantic interoperability* of legacy systems

- It assigns *meaning* to the *syntactic units* established by the syntactic LIF specification by referring to an *interface model*.
- *Semantic content* must be captured, despite differences in the representation in the different legacy systems
- Difficult, if different conceptualizations are maintained in the different legacy systems.

Architectural Style

- Under the term *architectural style* we subsume all *explicit* or *implicit principles*, *rules* and *conventions* that are used in the development of a system.
- Different organization deploy different architectural styles.
- Whenever two systems, based on different architectural styles, are connected*, property mismatches* surface at the interfaces.

Examples of a *Property Mismatch*

- *Endianness of data*:—the ordering of the subunits (e.g., bits or Bytes)
- *Naming Incoherence* : Different names for the same concept or the same name for different concepts
- *Representation*: Different representations of the same physical quantity (e.g., temperature)
- *Concepts*: Different conceptualization of reality

Coherent End-Systems

If the end systems are *coherent*, i.e., if there are no property mismatches between the end systems, then the end-systems can be connected by any appropriate communication system without a *mediator*.

Incoherent End-Systems: Semantic Content

Consider the two variables

T-Luft = 30

and

T-air = 86

on the surface there is a *property mismatch*: *different names* and *different values*

but the *semantic content* of both variables is the same.

How to Resolve Property Mismatches?

Property mismatches occur if the end system are *non-coherent*, i.e. based on different architectural styles.

There are two techniques to resolve property mismatches:

Worldwide Standardization *(Esperanto)*

or

Mediator (Gateway Component)

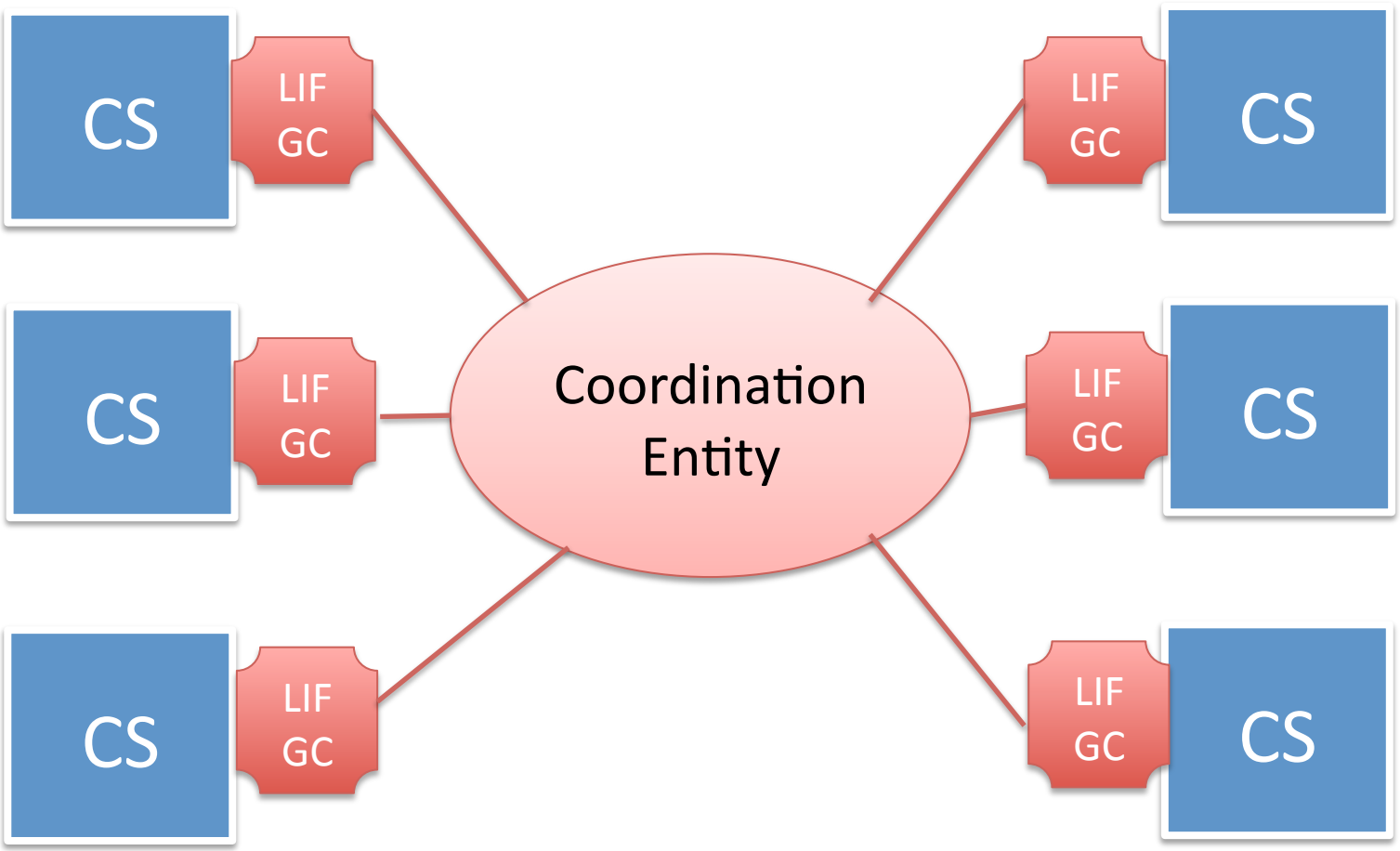
World-wide Standardization

- Development of *Domain Ontologies* to establish a *universally accepted conceptualization* and *name space* of an application domain (e.g., electricity load balancing)

- *Ontological Commitment* to use the specified ontology in each of the legacy systems.

Structure of an SoS

Interaction Domain for a purpose



Mediation by a Gateway

Whenever two systems that are developed according to different architectural styles are connected, a mediator, i.e. a *gateway component,* must be provided to resolve property mismatches.

Mediator--Gateway

Emergence

- The interactions of legacy systems give rise to unique global properties at the system level that are not present at the level of the subsystems—the *emergent properties*.
- Emergent properties are irreducible, holistic, and novel—they disappear when the system is partitioned into its subsystem.
- Emergent properties can appear unexpectedly or they are planned. In many situations, the *first appearance* of the emergent properties is *unforeseen* and *unpredictable*.

Prior and Derived Properties

- When dealing with emergence, it is helpful to distinguish between the *prior properties* of the components and the new *derived properties* that come about by the interactions of the components.
- In many cases the prior properties and the derived properties can be of a completely different kind.
- It often happens that the *derived properties* open a completely new domain of science and engineering that requires the formation of novel concepts that capture essential properties of this new domain.

Example: Worldwide Banking System

The world-wide interconnection of and the autonomic trading among the banks has resulted in complexities and the emergence of a *world-wide banking crisis* that cannot be reduced to the behavior of any individual bank.

Alan Greenspan:

"...if I didn't understand it, and I had access to a couple of hundred PhDs, how the rest of the world is going to understand it sort of bewildered me."

quote from A.R. Sorkin, *Too Big To Fail*, p.90

Research Issues in *Embedded SoS*

- Emergence
- Safety and Security in dynamic SoS
- Robust Service in the face of system evolution
- Semantic interface specification
- Autonomic components and knowledge representation
- Opportunistic flexibility

Conclusions

- The interconnection of existing legacy systems into Systems-of-Systems opens a set of fascinating new *research topics.*
- The provision of *stable global SoS services*, in the face of *system failures*, *intrusions*, and continuous system evolution, is a most important challenge.
- The development of an understanding of the topic of *emergence*, which at present is not well understood, requires a substantial effort from the research community.