27th annual INCOSE international symposium

Adelaide, Australia

July 15 - 20, 2017

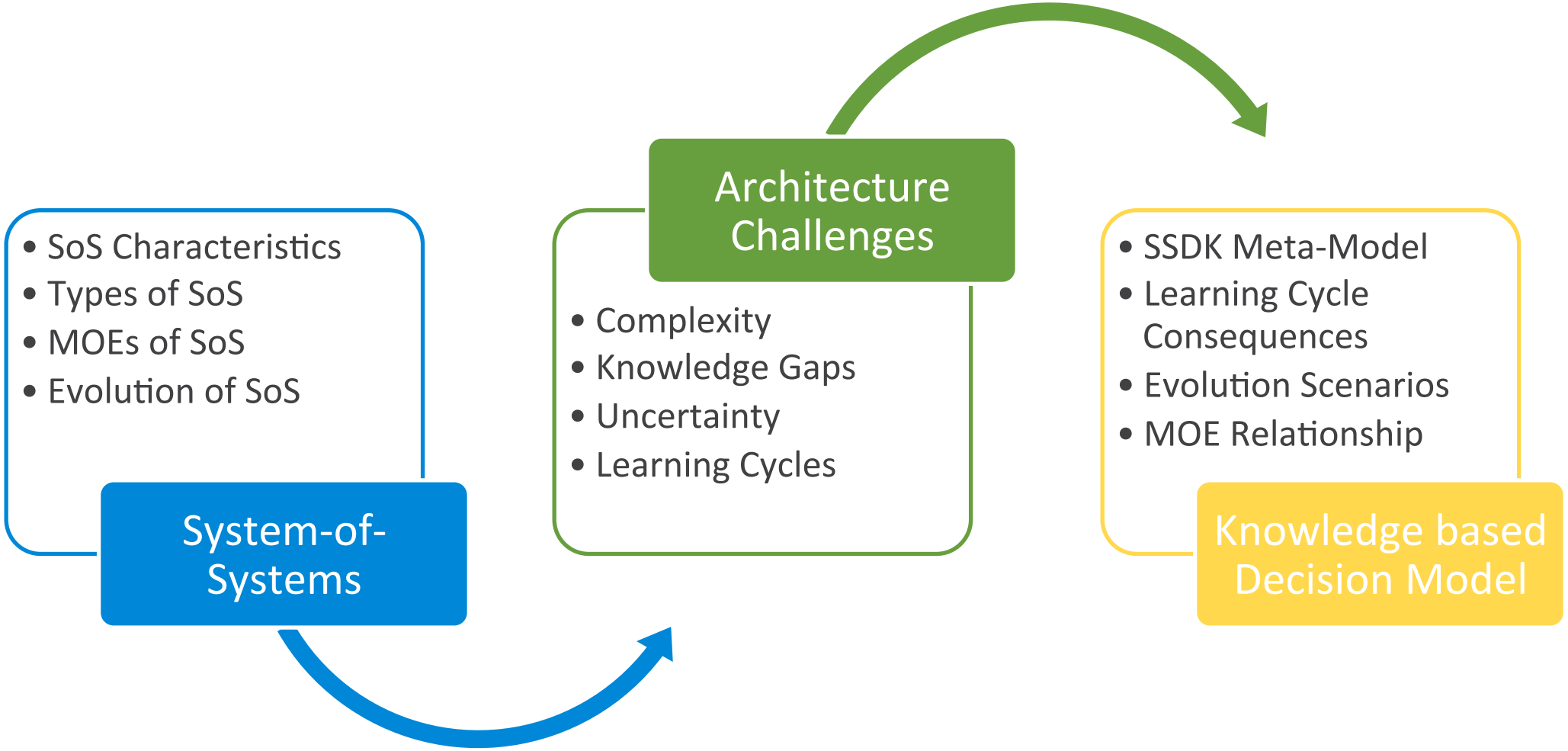
Knowledge Based Decision Model for Architecting and Evolving Complex System-of-Systems

Ramakrishnan Raman
Honeywell Technology Solutions Lab, Bangalore

Meenakshi D'Souza
International Institute of Information Technology, Bangalore

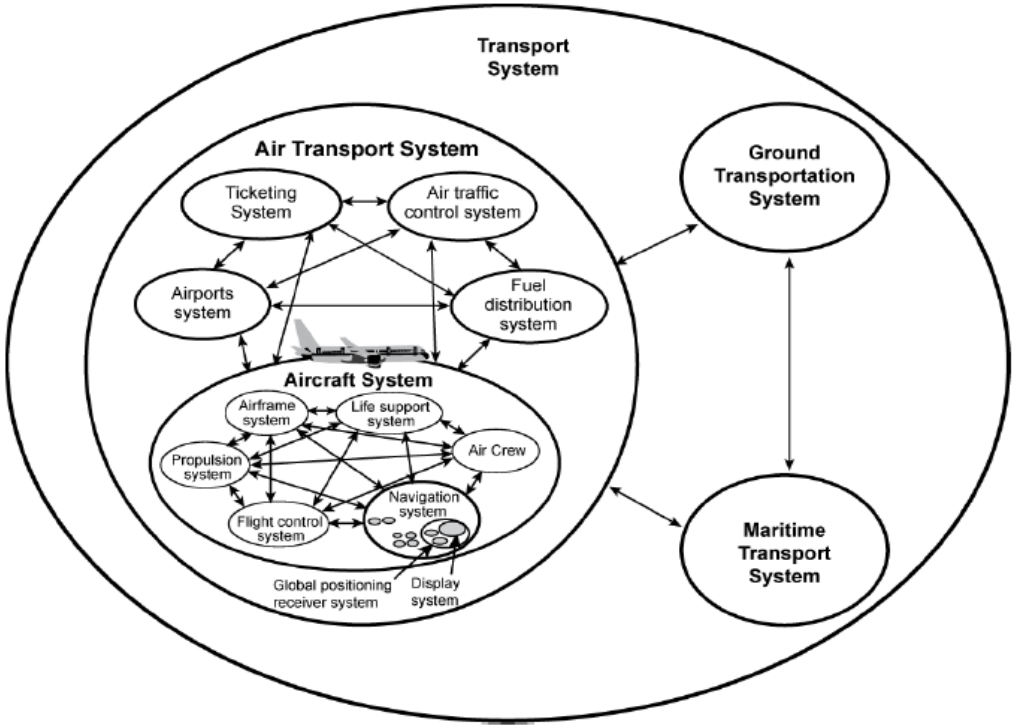
www.incose.org/symp2017

Presentation Outline



SoS: System-of-Systems

- System-of-Systems are systems-of-interest whose system elements are themselves systems - they typically entail large-scale inter-disciplinary problems involving multiple, heterogeneous and distributed systems
- Each system has an independent purpose and viability, in addition to the SoS by itself having an independent purpose and viability
- Typically entail large scale interdisciplinary problems involving multiple, heterogeneous, distributed systems



Source: INCOSE SE Handbook

SoS Characteristics

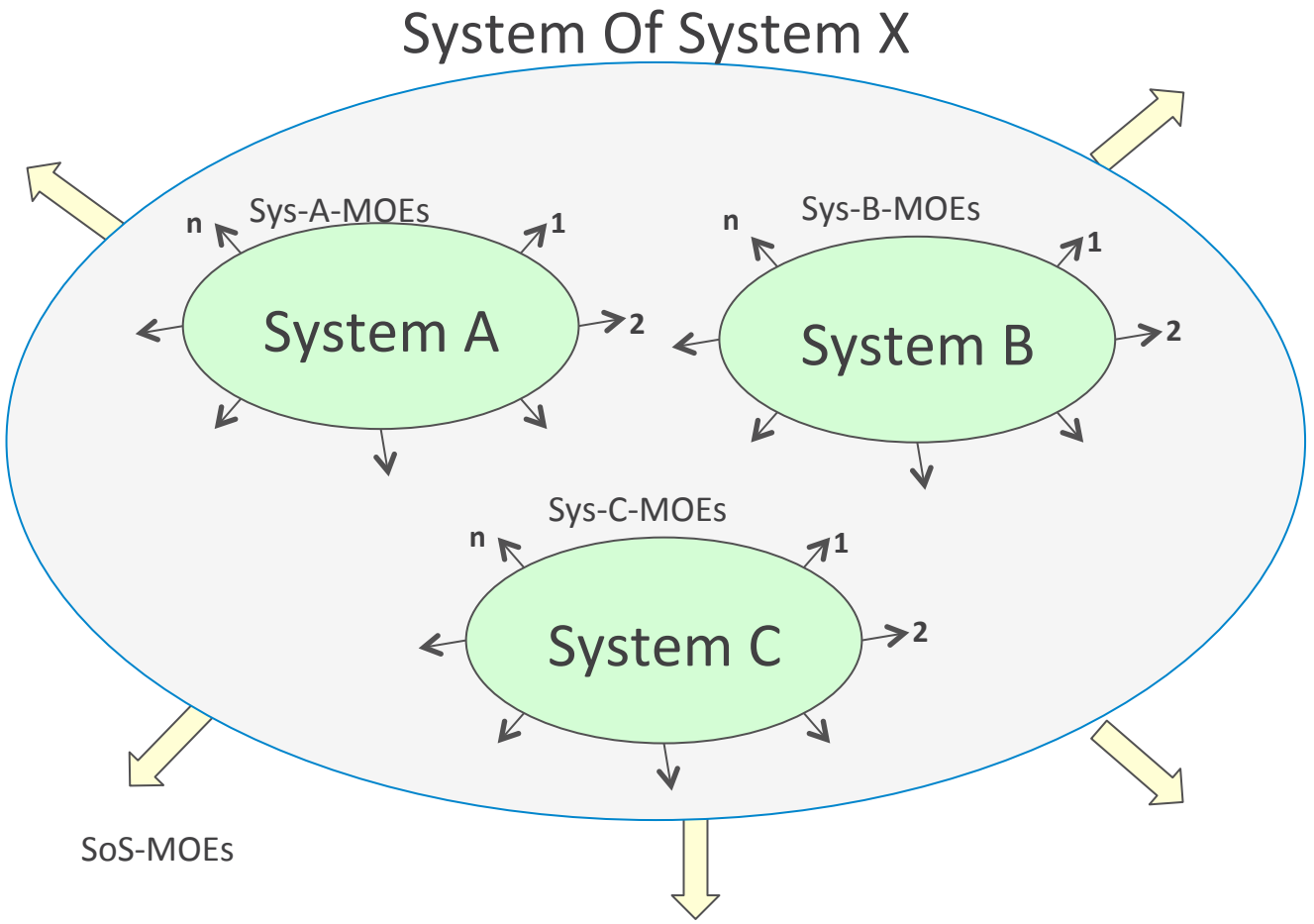
- Operational independence
  - Each constituent system in the SoS possesses the ability to perform by itself, irrespective of the presence or absence of other constituent systems in the SoS
- Managerial independence
  - The constituent systems are acquired and managed independently, and their operational existence is independent of the SoS
- Geographic distribution
  - Large scale geographic dispersion of the constituent systems
- Emergent behavior
  - Behavior that emerges in the SoS may not uniquely reside in any of the constituent systems
- Evolutionary/ adaptive development
  - Often not possible to state that the SoS is complete in all respects and is fully formed

SoS Evolution

- Over time, different changes happen in constituent systems
  - New functions getting added, obsolete functions getting removed, efficient means of realizing some of existing functions
  - Changes would manifest in changes in the emergent behavior of the SoS
- Often, the impact on the SoS is analyzed as an afterthought rather than systematically understanding the same upfront
- Deciding some of the changes can be a complex and daunting tasks
  - Deciding the constituent systems, especially when some of the constituent systems already exist, while others are yet to be designed
  - Different constituent systems may compete for similar functions and may not cooperate seamlessly

MOEs: Measures Of Effectiveness

- Operational measures of success
- Related to the achievement of the mission or operational objective being evaluated
  - In the intended operational environment
  - Under a specified set of conditions
- Manifest at the boundary of the system
- Examples
  - Response time to a user action
  - Time to Alert
  - Availability of the system



Types of SoS

Virtual SoS

- No central authority
- Functions as SoS based on interactions between constituent systems
- SoS is “accepted as-is”
- E.g. ad-hoc integration of internet based services

Collaborative SoS

- Common understanding of overall SoS purpose,
- No central authority that sets the MOEs for the SoS and takes decisions at the SoS level
- E.g. emergency response SoS for crisis situations

Acknowledged SoS

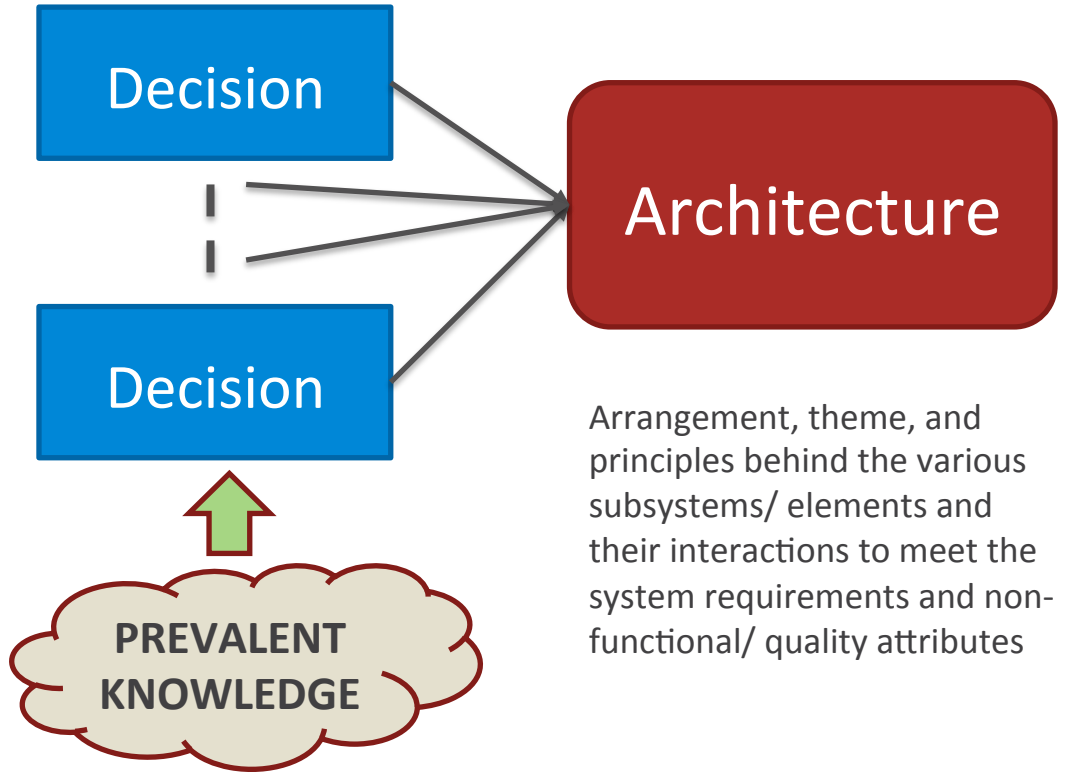
- “Central team” with limited authority
- MOEs for the SoS are more formally recognized and defined
- Constituent team retains significant autonomy
- E.g. e-Commerce SoS online sale and delivery

Directed SoS

- “Central team” with recognizable authority
- MOEs at the SoS level are formally specified
- Decisions at the constituent system level need to address the MOEs of the SoS
- E.g. Healthcare SoS

Architecting an SoS

- The architecture and design of the constituent systems, in essence, shape the architecture, emergent properties and behaviors at the SoS level
- Architectural decisions have a very significant bearing on the subsequent phases of the system lifecycle, including MOEs of the SoS and evolution of SoS
- Knowledge Gap: Something that the design team does not know, but would like to know before making a decision

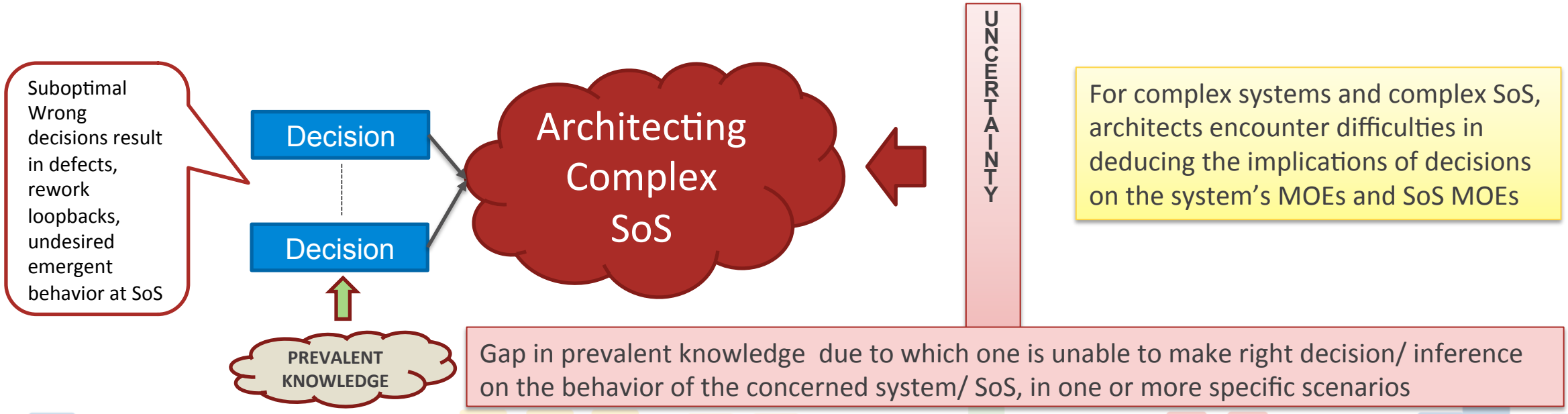


Arrangement, theme, and principles behind the various subsystems/ elements and their interactions to meet the system requirements and non-functional/ quality attributes

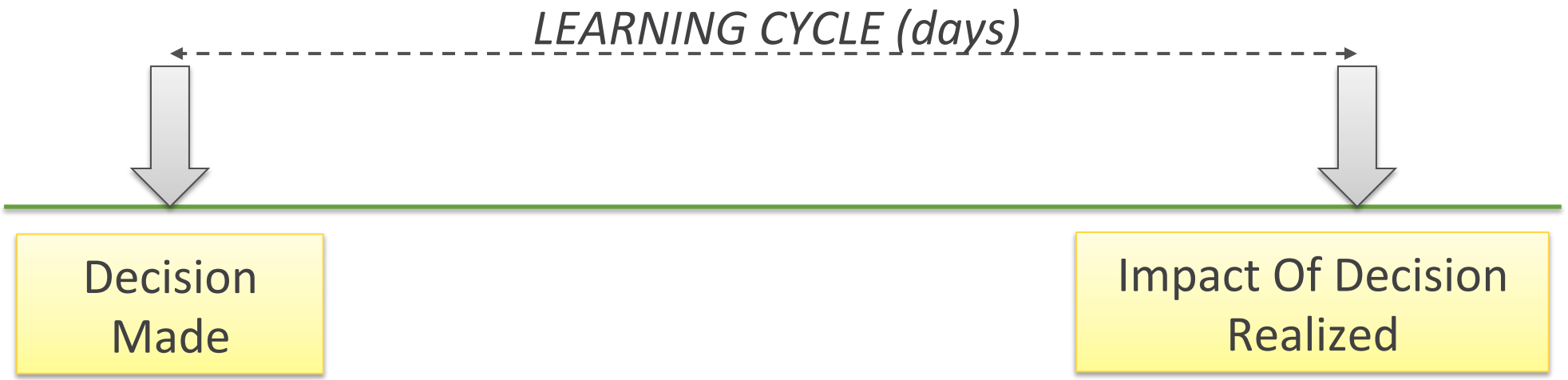
The knowledge available with the design teams on the relevant knowledge areas for applying to the constituent systems and SoS design problem at hand

Architecting Complex SoS

- Complex SoS
  - Multiplex of relationships/ forces/ interactions between constituent systems
  - Difficulties in establishing cause-and-effect chain
  - Very difficult to anticipate the SoS behavior from the knowledge of the constituent system behavior
  - Characteristics: Emergence, hierarchical organization, numerosity….



Learning Cycles

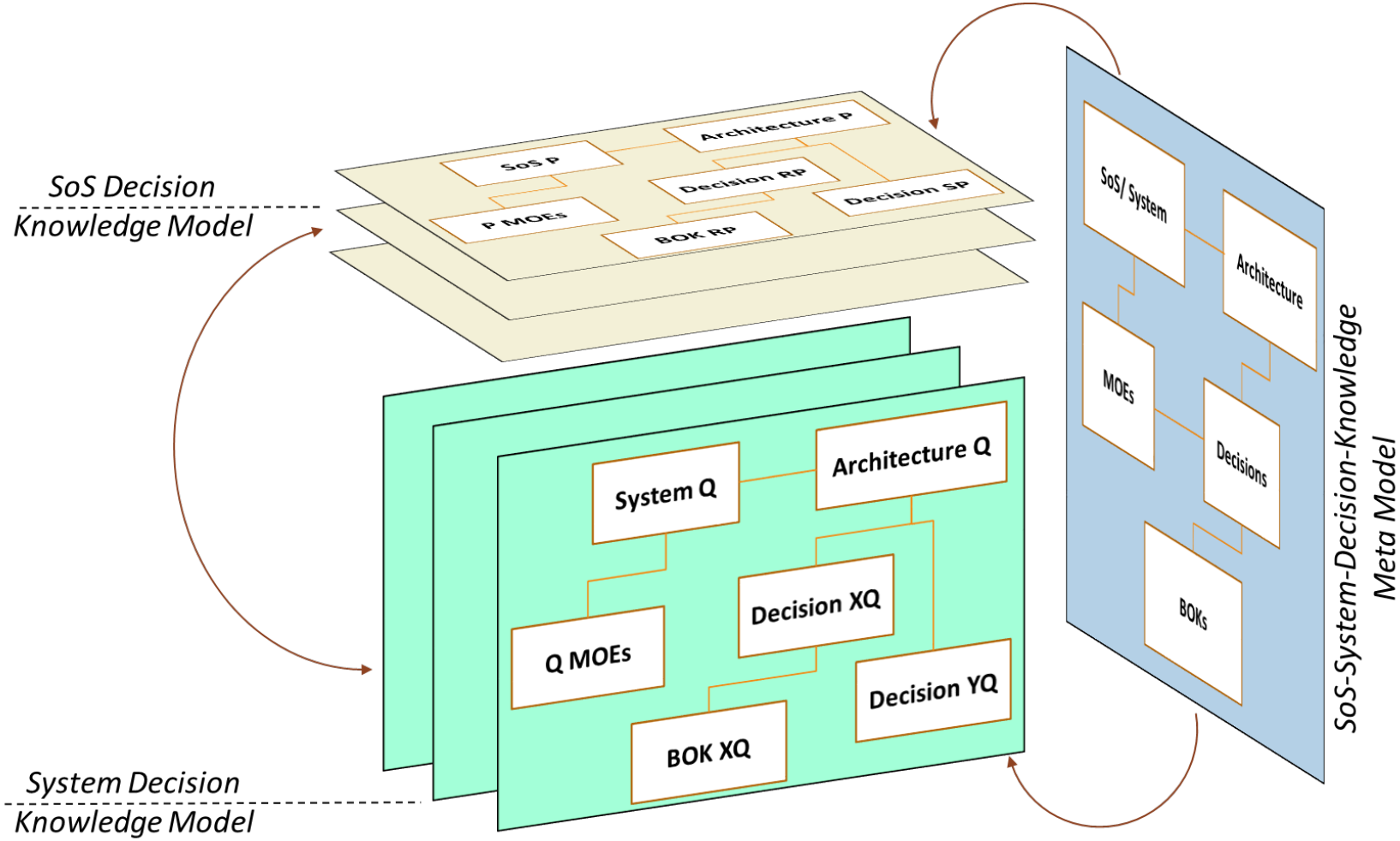


- ❑ Yes, that is the right decision
- ❑ It's ok, though suboptimal, we can proceed
- ❑ No, we need to patch up (surgery)
- ❑ No, we need to redo/ rework (loop back)

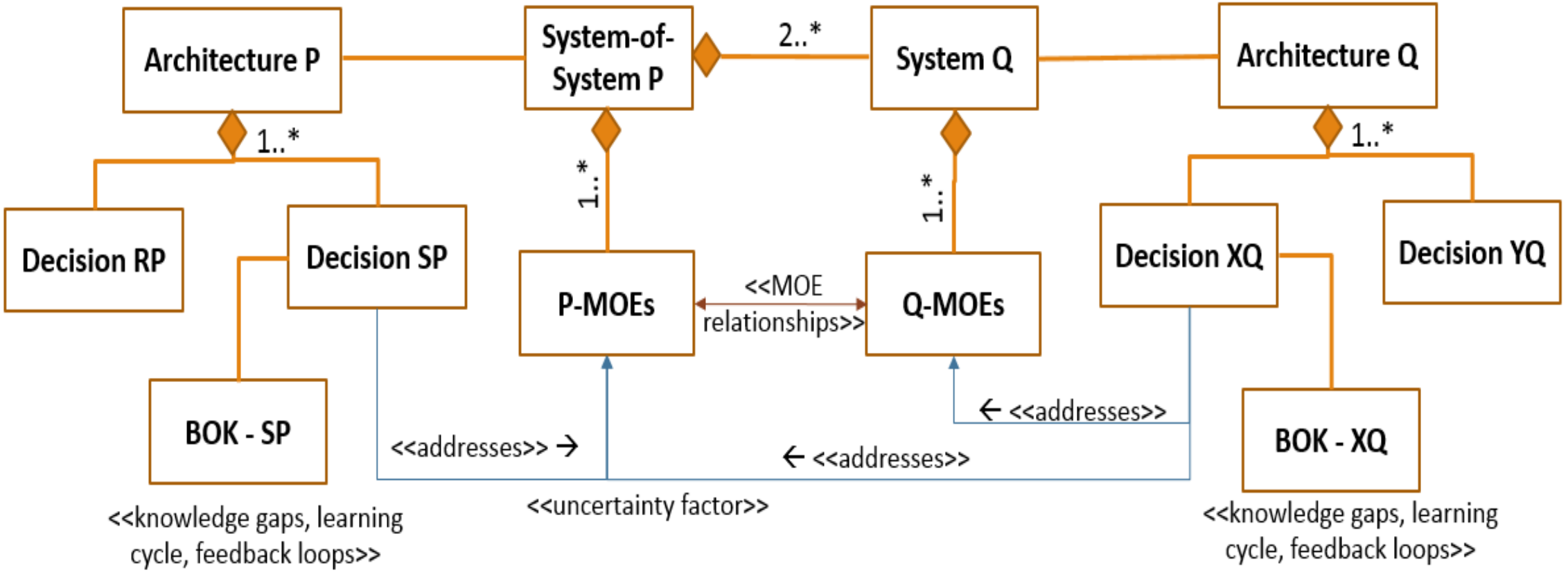
Inadequacies of existing models

- Architecting SoS involves making architectural decisions in spite of inadequate knowledge, both at the constituent system level as well as at the SoS level
- For complex SoS, decisions are made with significant uncertainty in terms of the implications of these decisions on how well the MOEs of the constituent systems are achieved as well as on how well the MOEs of the SoS are achieved
- The intricate aspects of knowledge that drive the architectural decision need to be well integrated into the architectural decision
- When the implications of the decisions surface, there is a need to close loop on the learnings back into further decision making in existing and new constituent system and SoS design

Proposed Knowledge Based Decision Model



SSDK Meta-Model

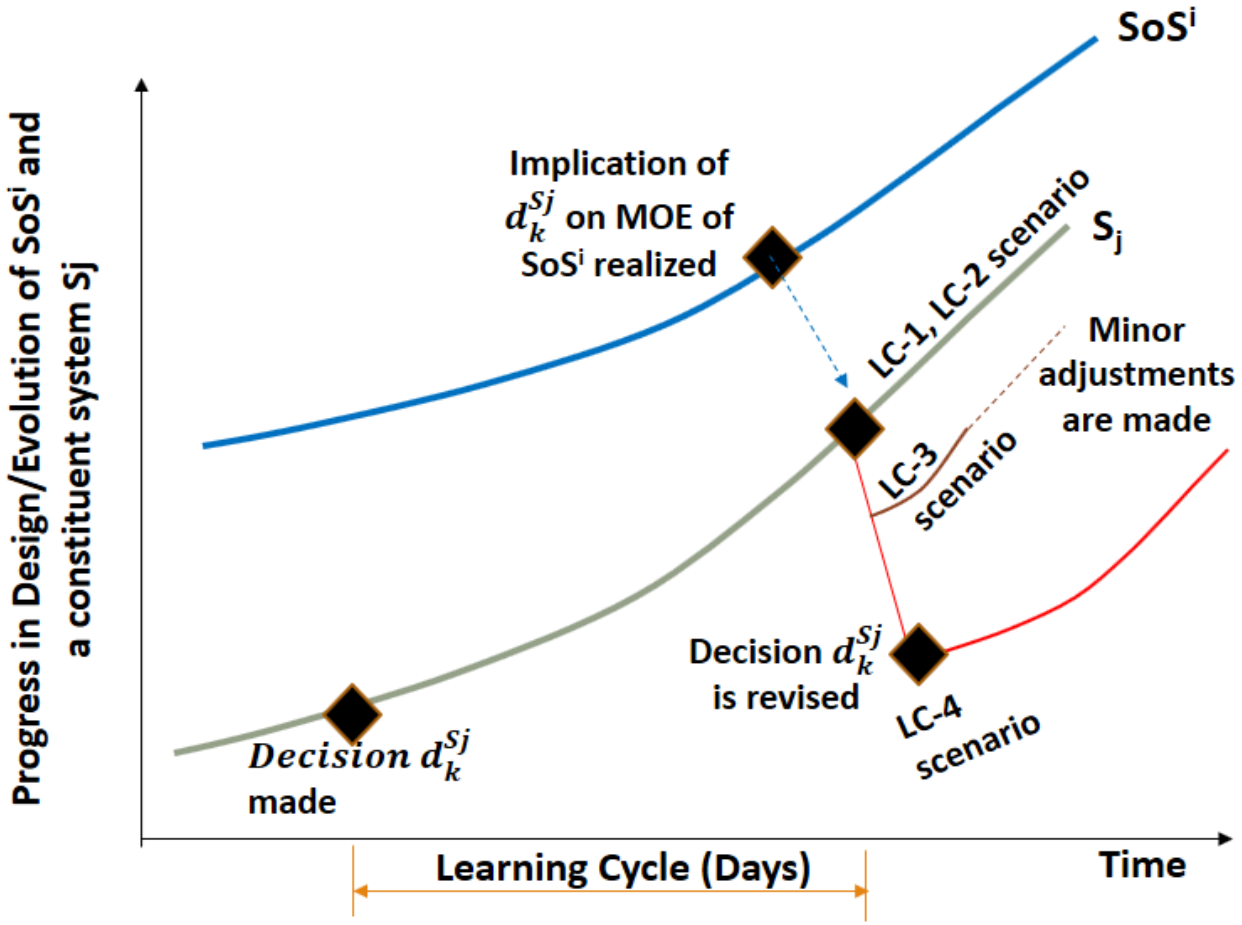


Model Entities across SoS types

Entity	Virtual	Collaborative	Acknowledged	Directed
SoS^i	Emerges	Emerges	Recognized	Managed
MOE^{SoS^i}	-	Emerges	Limited	Controlled
D^{SoS^i}	-	-	Partially Integrated	Well Integrated
$BOK^{d_k^{SoS^i}}$	-	-	Partially Integrated	Well Integrated
$BOK^{d_k^{Sj}}$	System scope	System scope	Partially Integrated	Well Integrated
fUT^{SoS^i}	-	-	Partially Integrated	Well Integrated
$fUT_{S_j}^{SoS^i}$	System scope	System scope	Partially Integrated	Well Integrated

- $SoS\uparrow i$ a specific SoS
- $MOE\uparrow SoS\uparrow i$: set of MOEs of SoS
- $D\uparrow SoS\uparrow i$: Set of architectural decisions for SoS
- $\mathrm{BOK}\uparrow \mathrm{d}\downarrow \mathrm{k}\uparrow \mathrm{SoS}\uparrow \mathrm{i}$: Body of knowledge pertaining to a decision made in specific SoS
- $BOK\uparrow d\downarrow k\uparrow Sj$: Body of knowledge pertaining to a decision made in a constituent system
- $fUT\downarrow S\downarrow j\uparrow SoS\uparrow i$: uncertainty function on how a decision in constituent system impacts MOE of the specific SoS
- $fUT\downarrow\uparrow SoS\uparrow i$: uncertainty function on how a decision made at SoS level impacts MOE of the SoS

Learning Cycle Consequences



CATEGORY	LEARNING CYCLE CONSEQUENCE
LC-1	The decision is the optimal decision, and does not inhibit any of the requirements or behaviours of the constituent system or SoS
LC-2	The decision is not the optimal one, but nevertheless, can be "lived with", i.e. does not impact any critical MOEs of the system or SoS
LC-3	The decision is not the optimal, and it might require some amount of rework or minor correction
LC-4	The decision needs to be significantly reworked, requiring a loop-back to the point where the decision was taken. In extreme cases, the budget or resources required for the rework might far exceed of what is available and allowable

Uncertainty Scenarios

DECREASING UNCERTAINTY

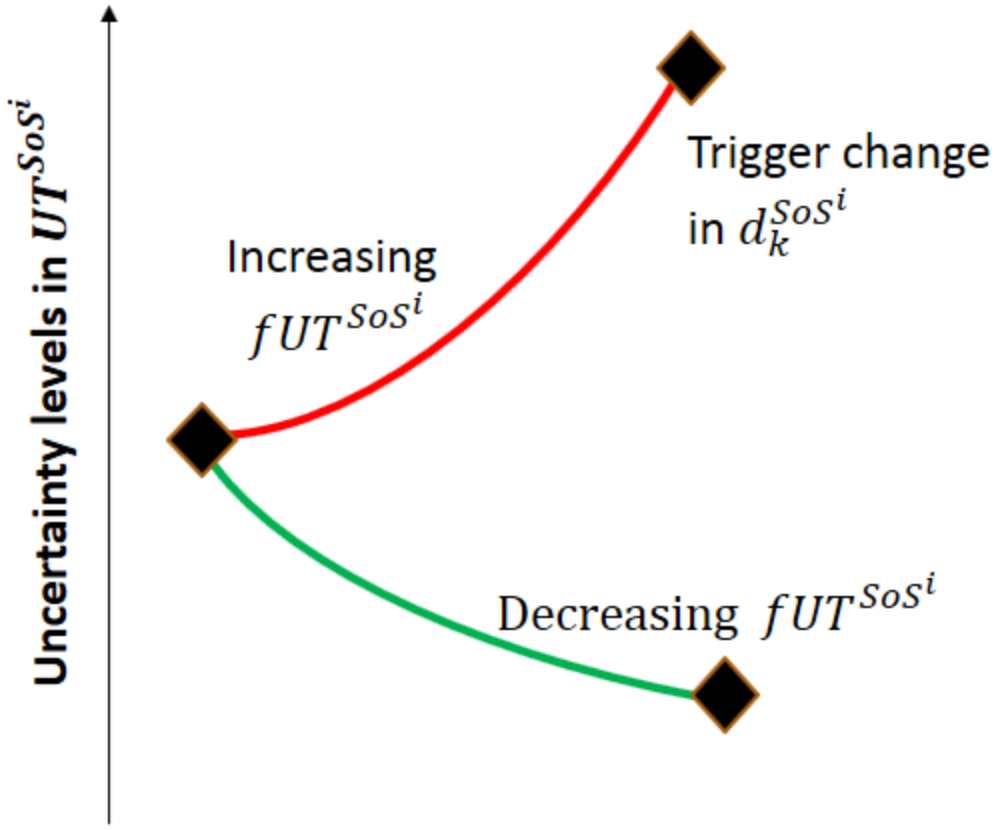
(a) when more knowledge is gained through prototypes, and simulations of the SoS, constituent system, or any specific portion of the SoS,

(b) when subsequent development phases do not encounter any problem that are traceable to the specific decision

(c) when actual build, test, field demonstration, and actual deployment do not encounter any problem that are traceable to the specific decision. This would imply a decreasing uncertainty function

INCREASING UNCERTAINTY

(a) "unknown unknowns" surfacing later, causing increasing uncertainty as development progresses.



Learning Cycle Set

- Learning cycle set is based on actual experience across a set of systems
- Example: of all the implications of $d{\downarrow}k{\uparrow}SoS{\uparrow}i$ for the various constituent systems, LC-1 occurred 25% of the time, while LC-2 occurred 50% of the time and LC-4 was encountered 25% of the time
- Building of this knowledge also ensures that the fidelity of the decision model and decision making process increases with time (increasing occurrence of LC-1 and LC-2), as the design teams learn on the decisions

Learning Cycle Set for decision $d_k^{SoS^i} \in D^{SoS^i}$

$C^{d_k^{SoS^i}}$ = {<LC1, 25%, 30days>,
<LC-2, 50%, 30 days>,
<LC-4, 25%, 50days>}

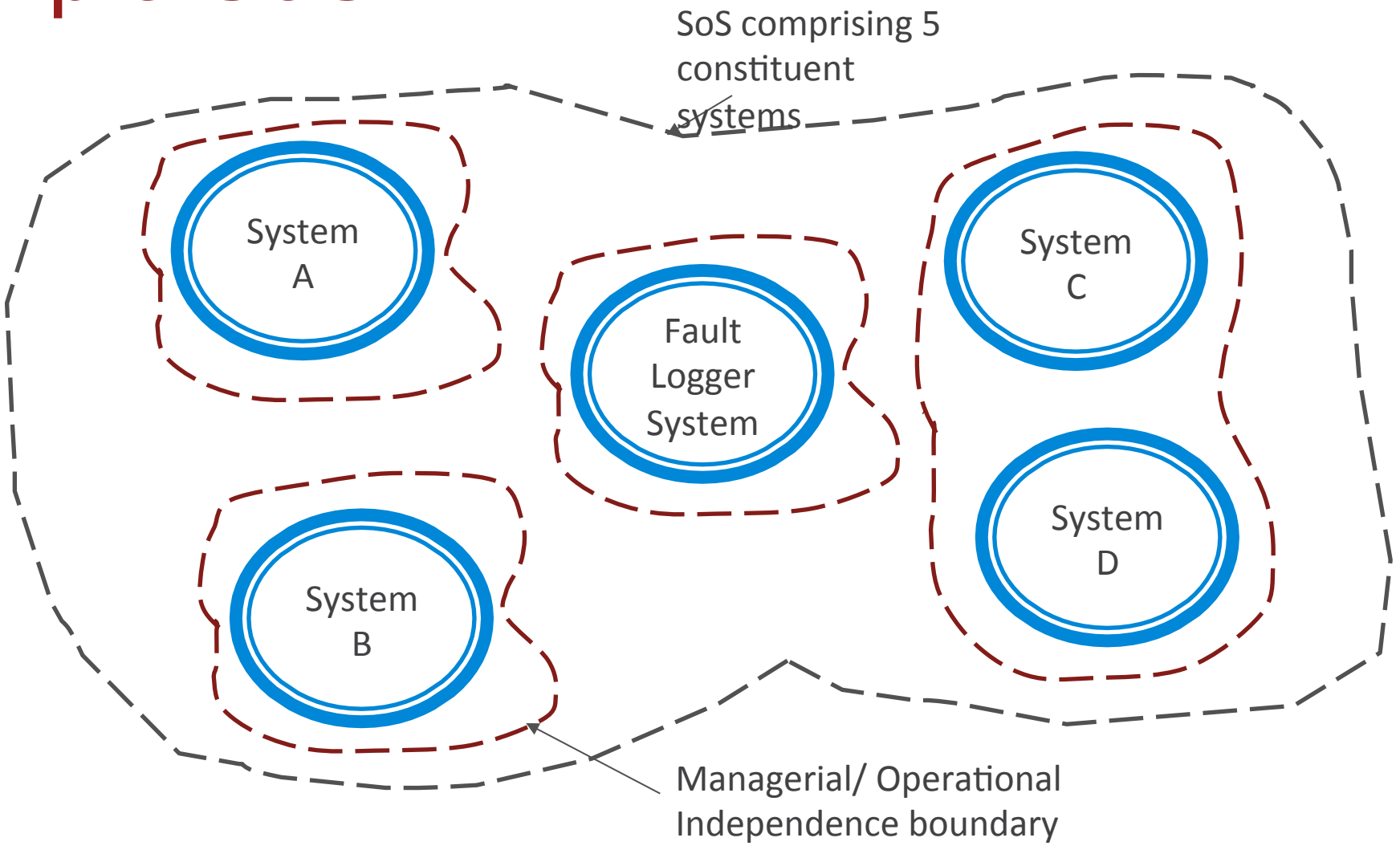
<<consequence, occurrence %,
average learning cycle duration>>

System	LCC	LC
S_k	LC-1	30 days
S_m	LC-2	20 days
S_p	LC-2	40 days
S_q	LC-4	50 days

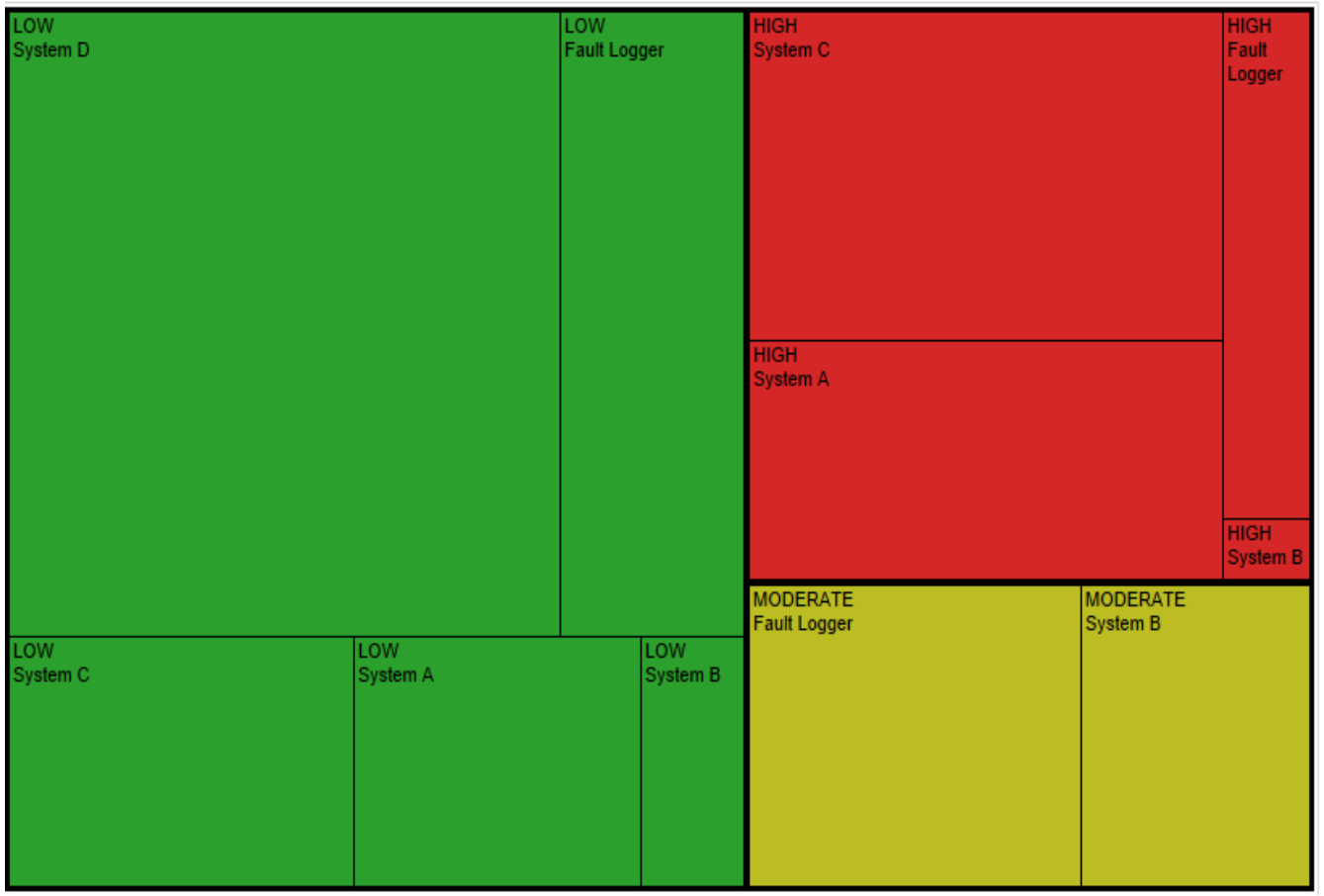
SoS Evolution Scenarios

System Of System	A more efficient and effective means of meeting an existing MOE
	An existing MOE becoming no longer relevant (due to evolution in the purpose of the SoS or of the operational/ mission scenarios of the SoS
	A new MOE becoming relevant, for instance, due to evolution in the objective/ purpose of the SoS or due to the evolution in SoS operational/ mission scenarios
System	A new constituent system, and hence corresponding MOEs getting into the SoS context
	An existing constituent system becoming obsolete and hence being removed, resulting in the removal of the corresponding set of MOEs
	Changes in the relationships of an existing constituent system's MOEs on the MOEs of the SoS

Example SoS


SoS comprising 5 constituent systems
System A
Fault Logger System
System C
System B
System D
Managerial/ Operational Independence boundary

Uncertainty Heatmap



Uncertainty
- LOW
- HIGH
- MODERATE

Decision: NVM Operations

Body Of Knowledge (BOK)	
Knowledge Gaps	•For multi-core systems, what are the optimal data management algorithms for periodic writing? •…
Learning Cycles	• LC1: 30% - 40 days, LC2: 40% - 60 days, LC3: 20% - 30days, LC4: 10%, 50 days
Feedback Loops	• Processor XX leakage power consumption feedbacks post deployment impacted NVM operations •…

Analyzing MOE Relationships

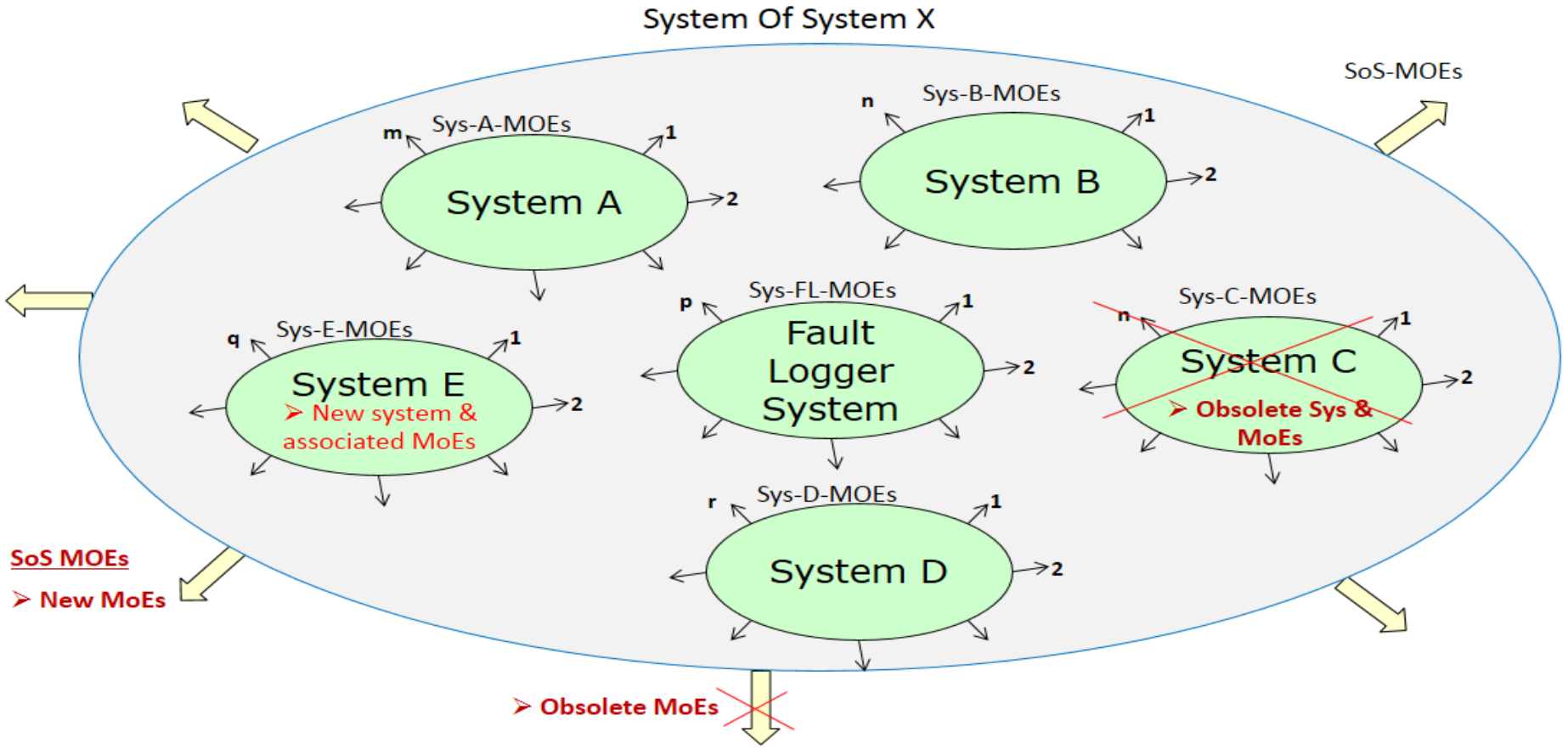
Relative importance of each SoS MOE

System Of System - MoEs	SoS MoE Weight	SysA-MoE-1	SysA-MoE-2	SysA-MoE-3	FaultLogger-MoE-1	FaultLogger-MoE-2	SysB-MoE-1	SysC-MoE-1	SysD-MoE-1	SysD-MoE-2
SoS-MoE-1	9	9	9	7					1	
SoS-MoE-2	7			5	7	7	7		1	
SoS-MoE-3	9	7	9	5					7	
SoS-MoE-4	7	7	7	7	5	5	7	1	7	
SoS-MoE-5	7				7	1	7		9	
SoS-MoE-6	1				1	5	1	5		
SoS-MoE-7	5		5		7	7	7	5		9
SoS-MoE-8	7	9	9	9				7	5	1
Raw score		256	299	255	169	131	183	86	226	52
Relative Weight		15%	18%	15%	10%	8%	11%	5%	14%	3%
Rank		2	1	3	6	7	5	8	4	9

Relative impact of each System MOE on each SoS MOE

System A is a key player in the SoS

SoS Evolution Scenarios



Conclusions

- Model incorporates the uncertainty and knowledge gaps associated with architectural decisions, the learning cycles that occur, feedback loops on the decisions, reflects those back on the uncertainty associated with decision
  - Facilitates the organized collection and leverage of knowledge during the constituent system and SoS development and evolution
  - Aids robust architectural decision making for the organizations designing the constituent systems and SoS
  - BOK enables leverage the prevalent knowledge in the organization to understand the various decisions that need to be made during the design of a SoS, and assess the associated uncertainty
- Future Work
  - Analyze model with respect to variations in the system development life cycles of the constituent systems and the maturity level of the constituent systems on their individual evolution path
  - Effective means of linking the architectural decisions to associated knowledge areas also needs to be explored

27th annual INCOSE international symposium

Adelaide, Australia

July 15 - 20, 2017

www.incose.org/symp2017