KAIST

SELAB KAIST

SoSE'19

Spectrum-Based Fault Localization on a Collaboration Graph of a System-of-Systems

Yong-Jun Shin, Sangwon Hyun, Young-Min Baek, and Doo-Hwan Bae
{yjshin, swhyun, ymbaek, bae}@se.kaist.ac.kr

Software Engineering Lab,
Korea Advanced Institute of Science and Technology (KAIST)

2019.05.22

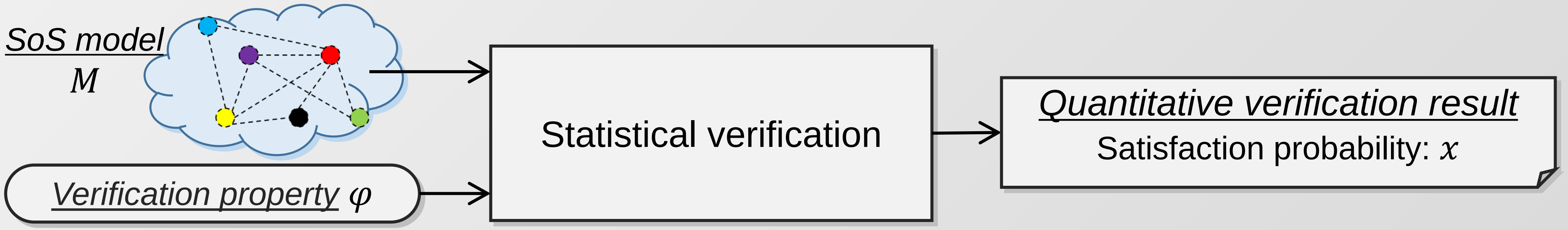
OUTLINE

1. **Introduction**
2. **Background**
3. **Overall Approach**
4. **Approach**
5. **Evaluation**
   A. Experimental Setup
   B. Evaluation Results
6. **Conclusion & Discussion**

Introduction

How to debug an SoS failure

- **Detect failure of an SoS goal or a violation of a verification property with statistical verification.**
  - We can get quantitative verification results utilizing statistical model checking for SoS verification[1,2].



- **Find the causes of the failure in the SoS. (Cont'd)**
  - Find the locations of the failure inducing entities and fix them.

[1] D. Seo, D. Shin, Y.-M. Baek, J. Song, W. Yun, J. Kim, E. Jee, and D.-H. Bae, "Modeling and verification for different types of system of systems using prism," in Proceedings of the 4th International Workshop on Software Engineering for Systems-of-Systems. ACM, 2016, pp. 12– 18.
[2] A. Mignogna, L. Mangeruca, B. Boyer, A. Legay, and A. Arnold, "SoS contract verification using statistical model checking," arXiv preprint arXiv:1311.3632, 2013.

Finding causes of a failure in the SoS

Related work: typical approaches to localize faults in Software Eng.

Category	Approach	References	Limitation
Static analysis	Analyzing a program code without executing the program. Supported by techniques of minimizing analysis area	(Neelofar 2017), (Santelices 2013), (Binkley 2004)	Engineers should have a white-box system. It is hard to apply to a large-scale system.
Dynamic analysis	Analyzing artifacts of program execution, e.g. execution log or testing results.	(Pham 2016), (Sharama 2013), (Rish 2005)	Instrumentation or probing is required. Analyzing the large size of execution results is required.

Limitations of existing approaches for SoS fault localization

- Localization granularity of existing approaches is not scalable for SoS which is a large complex system.
- Collaborative behavior of CSs (constituent systems) in an SoS is not considered.

Motivation & Goal

Motivation

- A fault localization technique for SoS is needed,
  - that efficiently reduces SoS debugging cost.
  - that is applicable to an SoS, a large complex system consisting of multiple black-box constituent systems (CSs).
  - that considers collaborative behaviors of CSs of an SoS.

Goal

- We propose a fault localization technique for SoS, extending a spectrum-based fault localization (SBFL), an existing fault localization approach.
  - It localizes CSs and their interactions that induce failure of an SoS.
  - It requires only statistical verification results and abstracted models of an SoS.
  - The abstracted model represents collaborations in an SoS.

Background

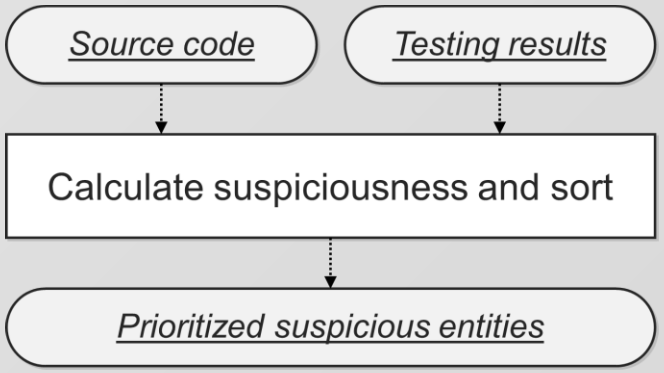
Spectrum-based Fault Localization (SBFL)

- **An example of applying SBFL (Tarantula[1]) to a program unit**

	Test cases							
`mid() {`	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3	*Sus.	rank
`Int x, y, z, m;`								
`1: Read("Enter 3 numbers:",x,y,z);`	●	●	●	●	●	●	0.5	7
`2: m = z;`	●	●	●	●	●	●	0.5	7
`3: If (y<z)`	●	●	●	●	●	●	0.5	7
`4: if (x<y)`	●	●			●	●	0.63	3
`5: m = y;`		●					0.0	13
`6: else if (x<z)`	●				●	●	0.71	2
`7: m = y;` ***//***bug***//***	●					●	0.83	**1**
`8: else`			●	●			0.0	13
`9: if (x>y)`			●	●			0.0	13
`10: m = y;`			●				0.0	13
`11: else if (x>z)`				●			0.0	13
`12: m = x;`							0.0	13
`13: print("Middle number is:",m); }`	●	●	●	●	●	●	0.5	7
Pass/Fail Status	P	P	P	P	P	F		

*suspiciousness

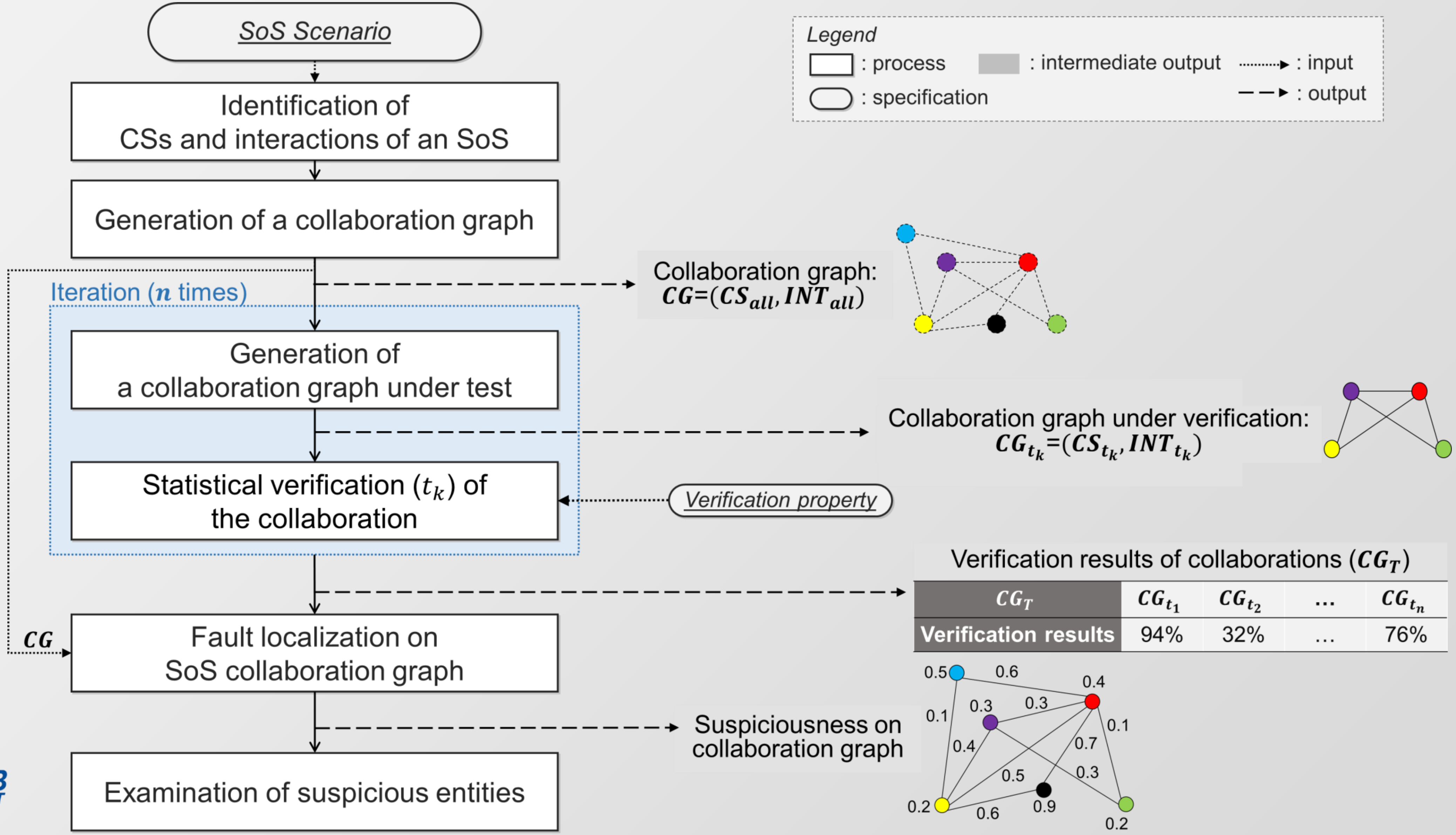
$$suspiciousness(e) = \frac{\frac{failed(e)}{totalfailed}}{\frac{pased(e)}{totalpassed}+\frac{failed(e)}{totalfailed}}$$



[1] Jones, James A., and Mary Jean Harrold. "Empirical evaluation of the tarantula automatic fault-localization technique." Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, 2005.

Overall Approach

Overall process of SoS fault localization



Verification results of collaborations (CG_T)

CG_T	CG_{t_1}	CG_{t_2}	…	CG_{t_n}
Verification results	94%	32%	…	76%

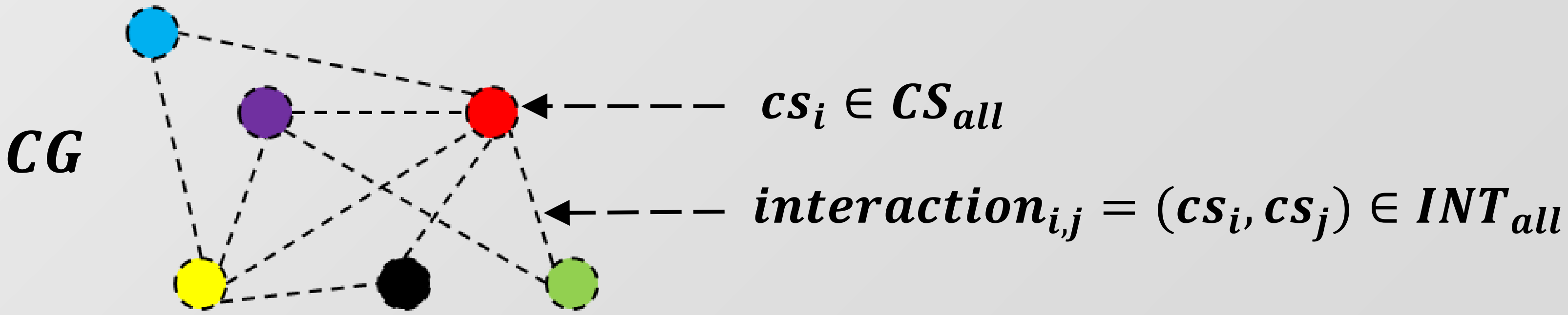
KAIST

SELAB KAIST

Approach

Input 1: Collaboration Graph of an SoS

- **A model of collaborations in SoS**
  - It is an abstraction of an SoS representing collaborations of constituent systems (CSs) to achieve SoS goals.
  - It shows a structure of the collaboration on a graph, a CS as a node and an interaction between two CSs as an edge of the graph.
- $\boldsymbol{CG = (CS_{all}, INT_{all})}$
  - CS_{all}: a set of all CSs in an SoS, a set of nodes in a graph
  - INT_{all}: a set of all possible interactions between two CSs in an SoS, a set of edges in a graph



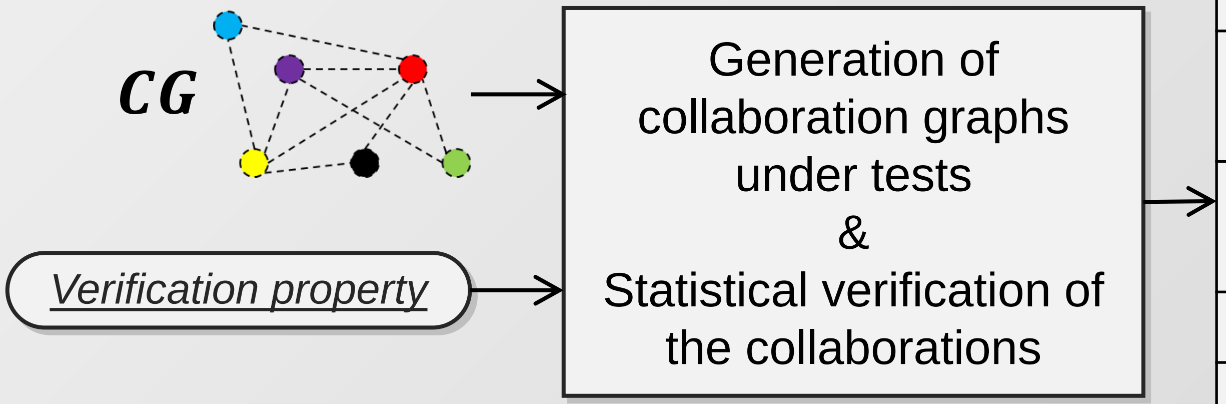
Input 2: Collaboration Graphs under Verification

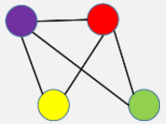
Models of diverse collaborations in SoS

- Diverse sub collaboration graphs are verified to show partial and diverse collaborations in an SoS and its goal achievement.

$CG_{t_k} = \left(CS_{t_k}, INT_{t_k}\right)$, a subgraph of $CG = (CS_{all}, INT_{all})$

- $CS_{t_k} \subseteq CS_{all}$
- $INT_{t_k} \subseteq INT_{all}$



k	CG_{t_k}	Verification results
1		**91%**
2		**85%**
…	…	…
n		**30%**

SBFL with statistical verification results of SoS

Suspiciousness calculation of each entity (node/edge)

- an extension of Tarantula[1] to utilize statistical verification results.

$$suspiciousness(e) = \frac{\frac{failedProb(e)}{totalFailedProb}}{\frac{pasedProb(e)}{totalPassedProb} + \frac{failedProb(e)}{totalFailedProb}}$$

- $passedProb(●) = 2.1$
- $failedProb(●) = 1.9$
- $totalPassedProb = 3$
- $totalFailedProb = 3$

Collaboration graph of SoS (CG)	Collaboration graph under test (CG_{t_k}) 1	2	3	4	5	6	*Sus.	Rank
Node (CS_{all}) {								
CS1 ●	●		●		●	●	0.48	6
CS2	●	●		●			0.4	7
CS3 // fault			●	●		●	0.73	2
CS4	●	●		●	●	●	0.48	5
}								
Edge (INT_{all}) {								
CS1-CS2	●						0.1	9
CS1-CS3			●				0.6	4
CS2-CS3				●			0.7	3
CS2-CS4	●	●					0.25	8
CS3-CS4 // fault				●		●	0.8	1
}								
Goal achievement probability	0.9	0.6	0.4	0.3	0.7	0.1		
	0.1	0.4	0.6	0.7	0.3	0.9	*Suspiciousness	

[1] Jones, James A., and Mary Jean Harrold. "Empirical evaluation of the tarantula automatic fault-localization technique." Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, 2005.

Spectrum-Based Fault Localization on a Collaboration Graph of a System-of-Systems

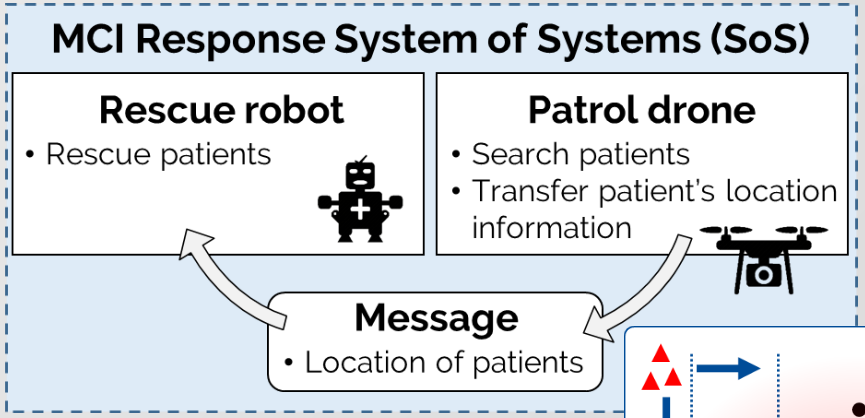
Evaluation

- Experimental Setup
- Evaluation Results

Experimental setup

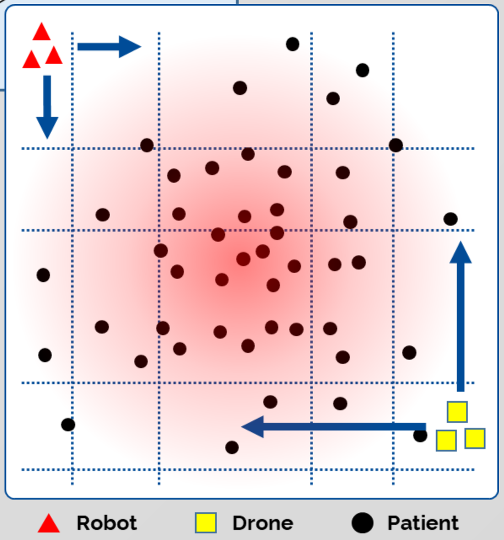
Experimental scenario

- Mass Casualty Incident (MCI) response SoS with autonomous rescue robots and patrol drones.
  - *Goal*: rescuing more than 80% of patients
  - Some robots and drones are selected and collaborate to achieve the goal.



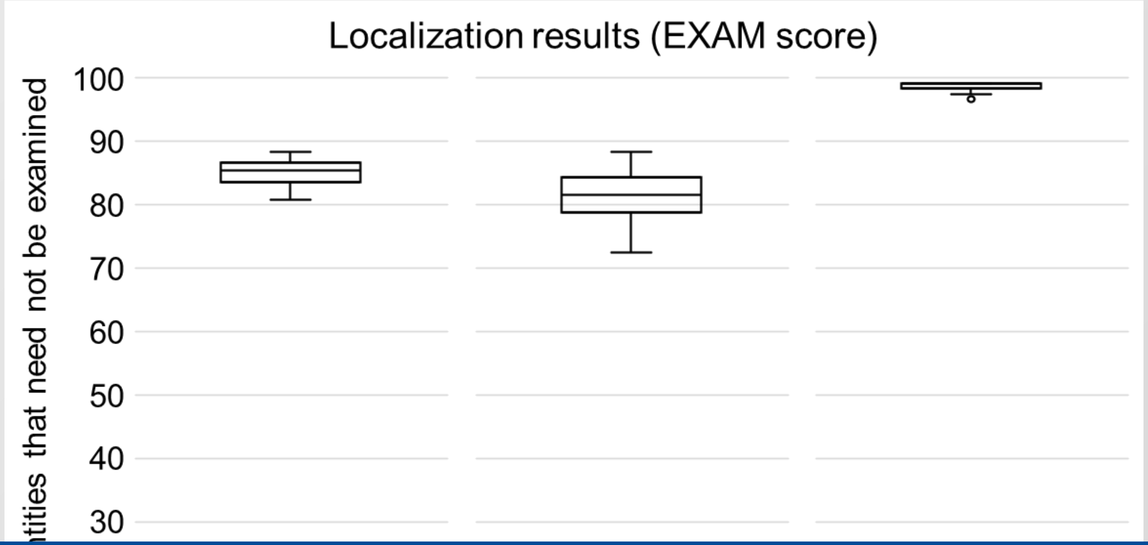
Artificial faults

- Scenario 1 (faulty CS – robot)
  - Lower patient rescuing probability
- Scenario 2 (faulty CS – drone)
  - Lower patient discovery probability
- Scenario 3 (faulty interaction – message delay)
  - Delayed message transmission



Evaluation results (1/2)

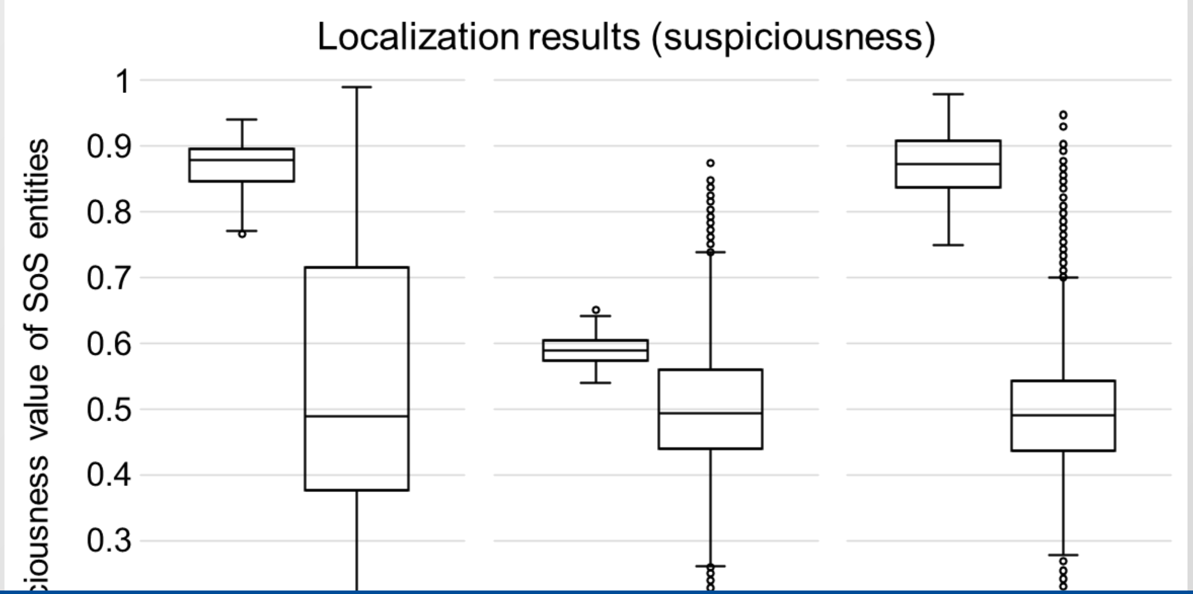
- **RQ1. Does our approach effectively localize faulty CS or interactions, so that reduces SoS debugging cost?**
  - In the three scenarios, 88% of debugging effort was reduced on average.



Using the SBFL approach, debugging cost can be dramatically reduced with given collaboration graphs and its verification results.

Evaluation results (2/2)

- **RQ2. Does suspiciousness value effectively classify faulty/normal SoS entities?**
  - In most cases, the calculated suspiciousness is a good indicator for engineers to compare the debugging priority with the probability that an entity is faulty.



In the given scenarios, calculated suspiciousness helps to effectively identify entities that induce failure of SoS.

Spectrum-Based Fault Localization on a Collaboration Graph of a System-of-Systems

Conclusion

Conclusion & Future Work

Conclusion

- We proposed an SBFL technique for SoS.
  - It effectively localizes CSs and their interactions that induce failure of an SoS.
  - It is applicable with only collaboration graphs of an SoS and their statistical verification results.

Future work

- We showed the feasibility of applying SBFL to SoS engineering, but the evaluation was limited to given scenarios. It will be extended to more complex SoS scenarios in future work.

Thank You.

Spectrum-Based Fault Localization on a Collaboration Graph of a System-of-Systems

Yong-Jun Shin, Sangwon Hyun, Young-Min Baek, and Doo-Hwan Bae
{yjshin, swhyun, ymbaek, bae}@se.kaist.ac.kr

Korea Advanced Institute of Science and Technology (KAIST)

2019.05.22

END OF FILE

Spectrum-Based Fault Localization on a Collaboration Graph of a System-of-Systems