Composition Algorithm Adaptation in Service Oriented Systems

14th European Conference on Software Architecture(ECSA), 3rd Context-aware, Autonomous and Smart Architectures International Workshop (CASA), 15 September, 2020

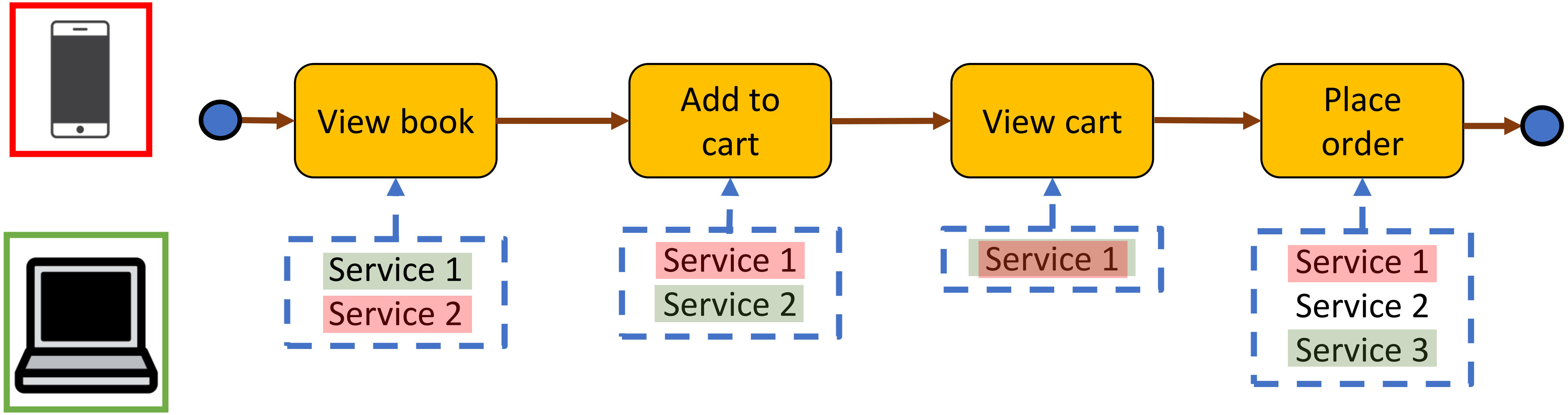
Niranjana Deshpande, Naveen Sharma

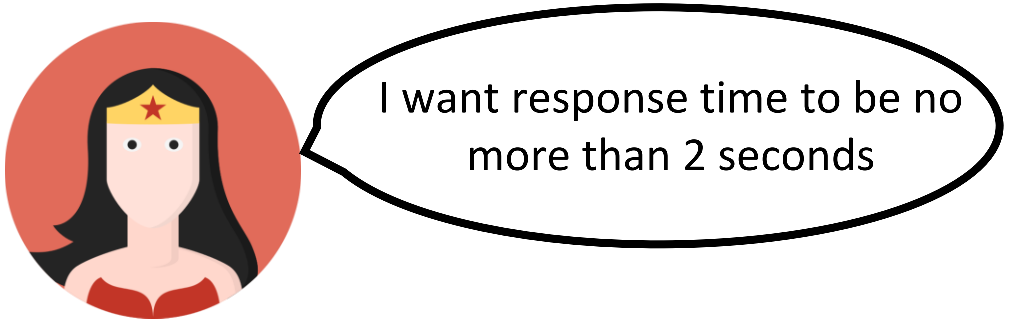
Rochester Institute of Technology

nd7896@rit.edu

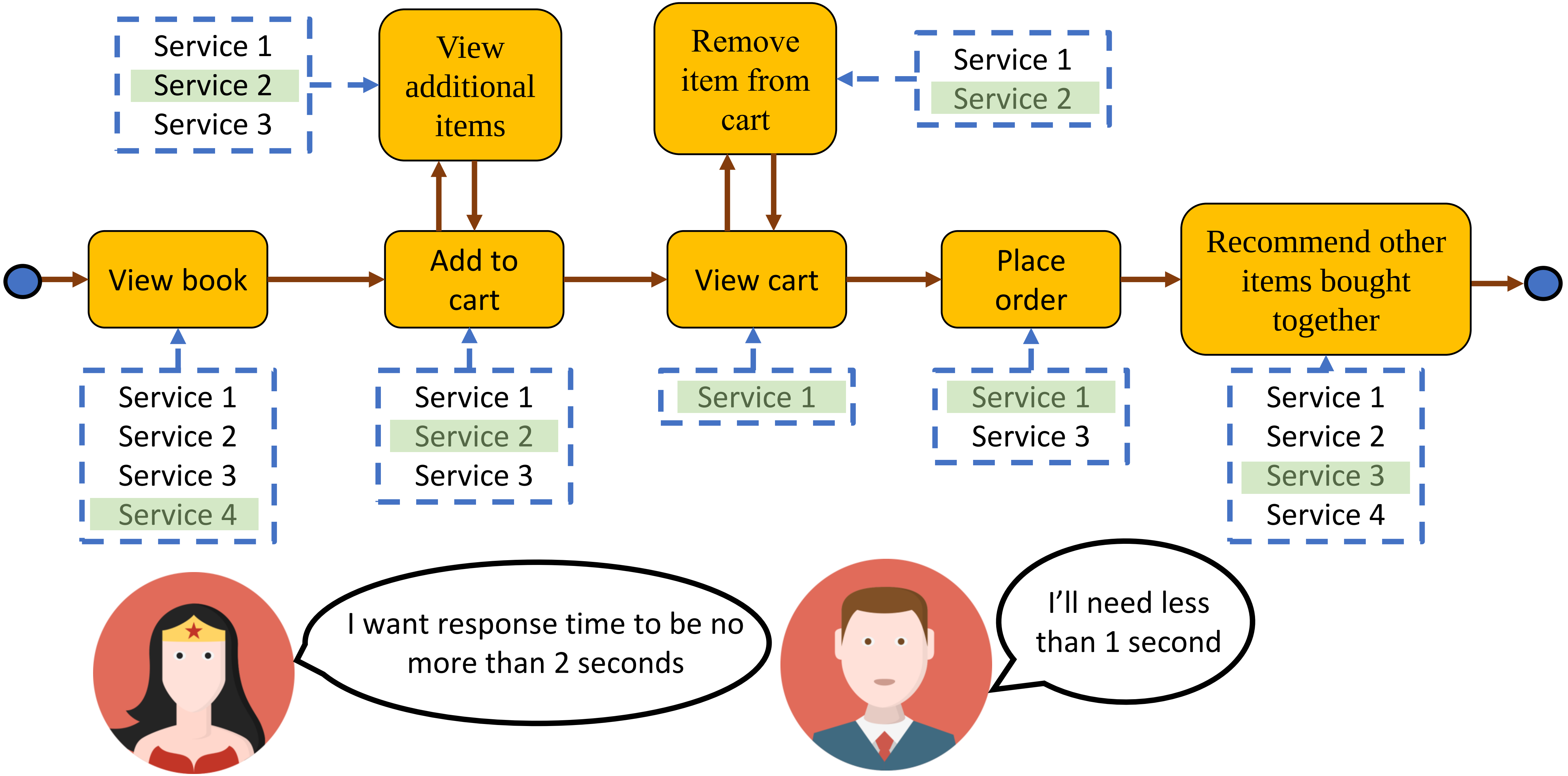
This material is based upon work supported by the National Science Foundation under Award No. 1943002.

Motivation


View book
Add to cart
View cart
Place order
Service 1
Service 2
Service 1
Service 2
Service 1
Service 1
Service 2
Service 3

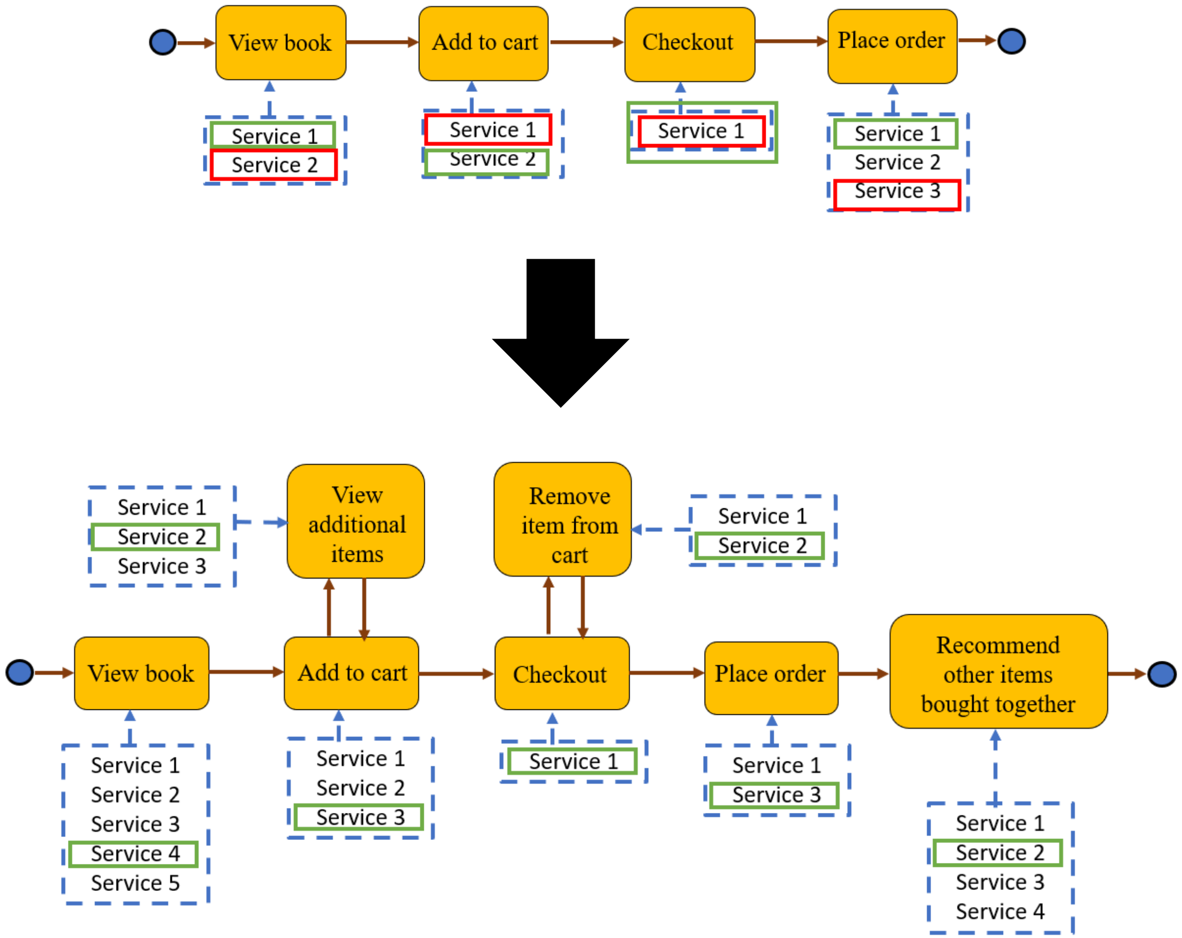

I want response time to be no more than 2 seconds

Service Composition: Dynamic Environment



Service Composition: Considerations

- Compositions need to meet nonfunctional user needs while constrained by compute resources
- Due to dynamic environment, composition needs to be adaptable
- *So, services are re-selected using a predetermined algorithm*



Current Approaches

- Most approaches perform adaptation at the services level or propose new composition algorithms (Wang et al. 2016, Ali et al. 2015, Schuller et al. 2014, Al-Helal et al. 2014, Ardagna et al. 2011)

- Current Approaches
  - Platforms
  - Algorithms

Current Approaches: Platforms

- MOSES (Cardellini et al., 2017)
  - Address adaptation at the services level using linear programming (LP) formulations
  - Support for dynamically adapting coordination patterns is also provided
- QoSMOS (Calinescu et al., 2009)
  - Selects candidate services at runtime or allocates resources to services for execution to meet Quality of Service (QoS) requirements
- Integrating reinforcement learning with multi-agent techniques (Wang et al. 2017)
  - Proposes adaptive service selection at runtime using a Markov Decision Process
  - Multi-agent techniques have communication overhead
  - Can easily become compute intensive

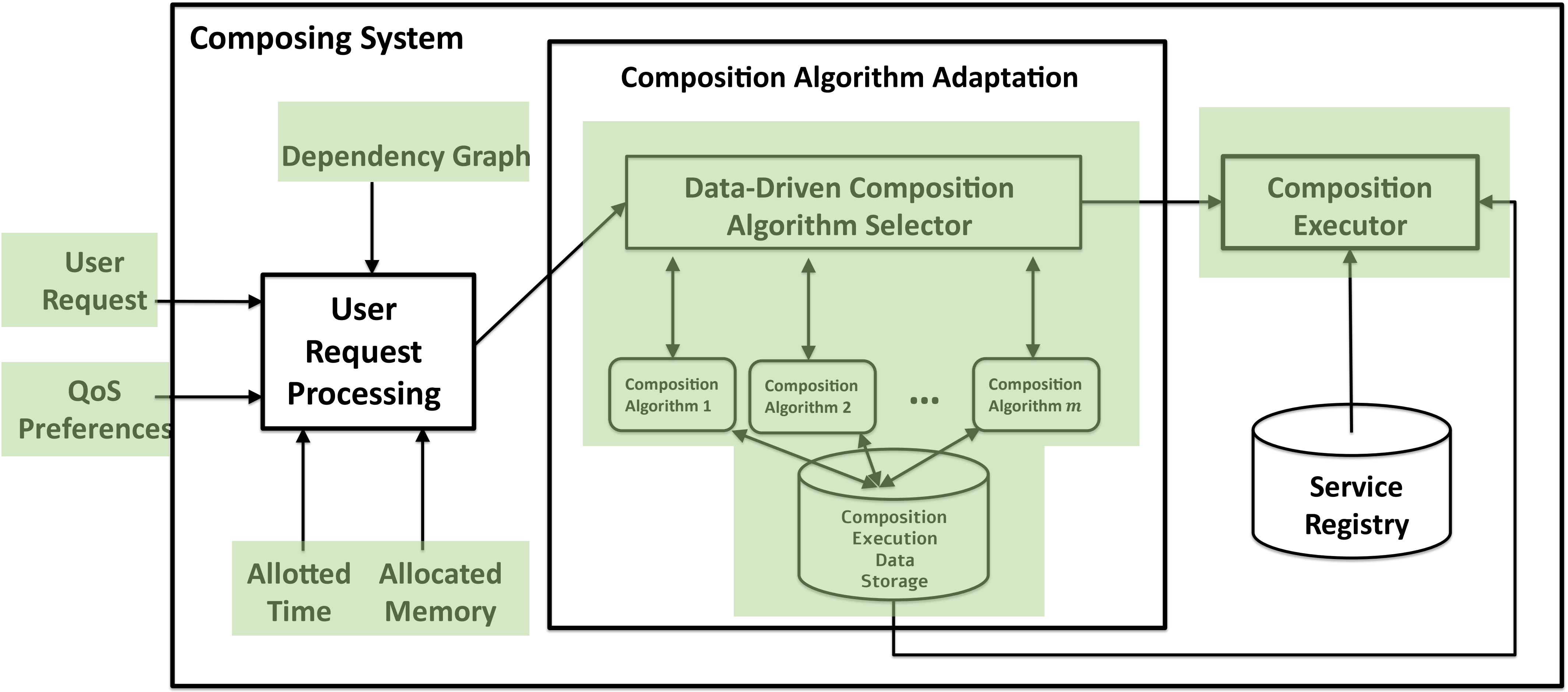
Current Approaches: Composition Algorithms

- Alrifai et al. 2009
  - Decomposes global constraints into local constraints to select services
- Trummer and Faltings, 2011
  - Propose dynamic algorithm selection for a set of batched user requests
  - Focus is on recommending algorithms to minimize cost
  - Recommendations are most recent executions

Research Question

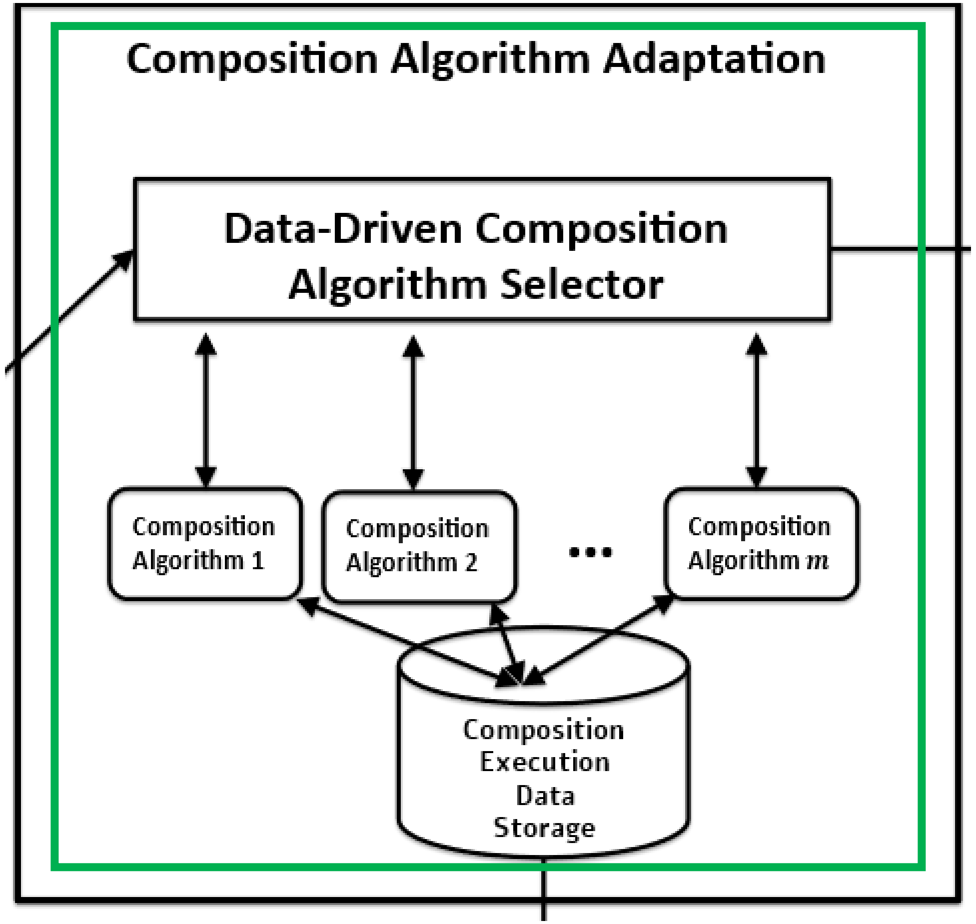
- Given a wide variety of available service composition methods, we focus on this question: how can we determine the right method for a given service composition task?
  - Goal 1: How can we determine the right composition method for a given service composition task?
  - Goal 2: How does selecting a composition algorithm on a per-instance basis perform compared to a pre-selected algorithm?

System Overview



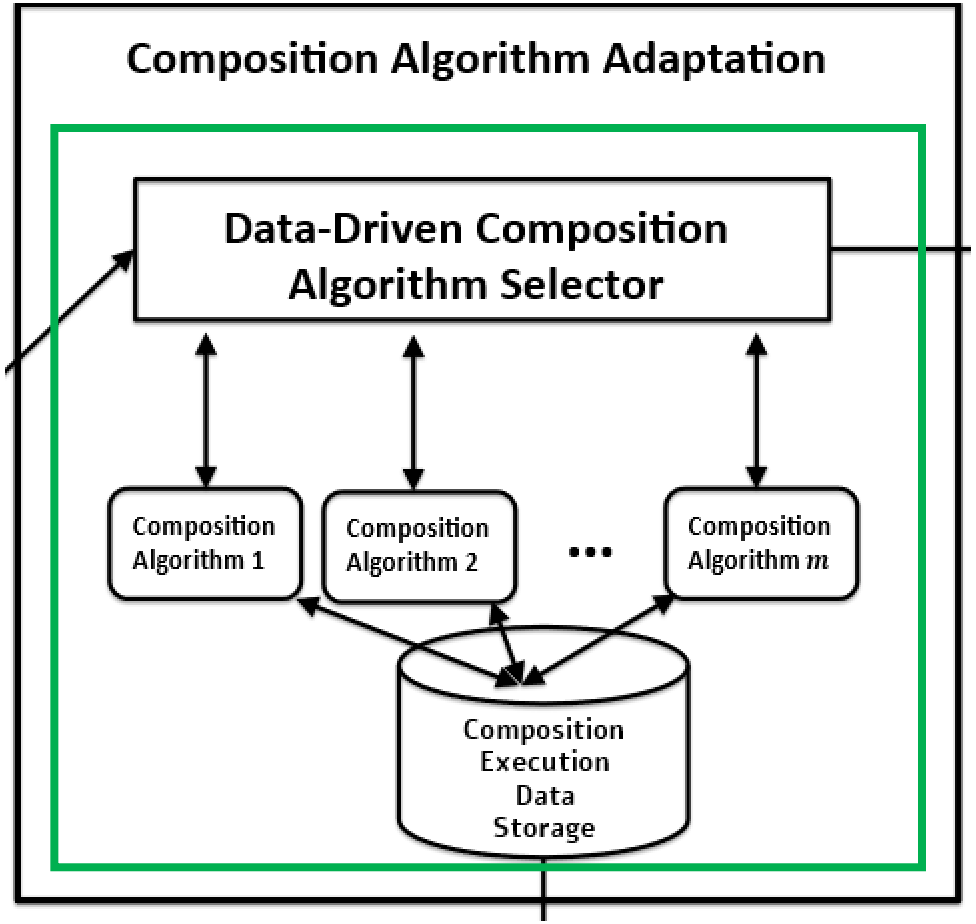
Composition Algorithm Selector

- Steps followed
  - Dataset creation
  - Classifier selection
  - Evaluation of classifier-based selector



Composition Algorithm Selector

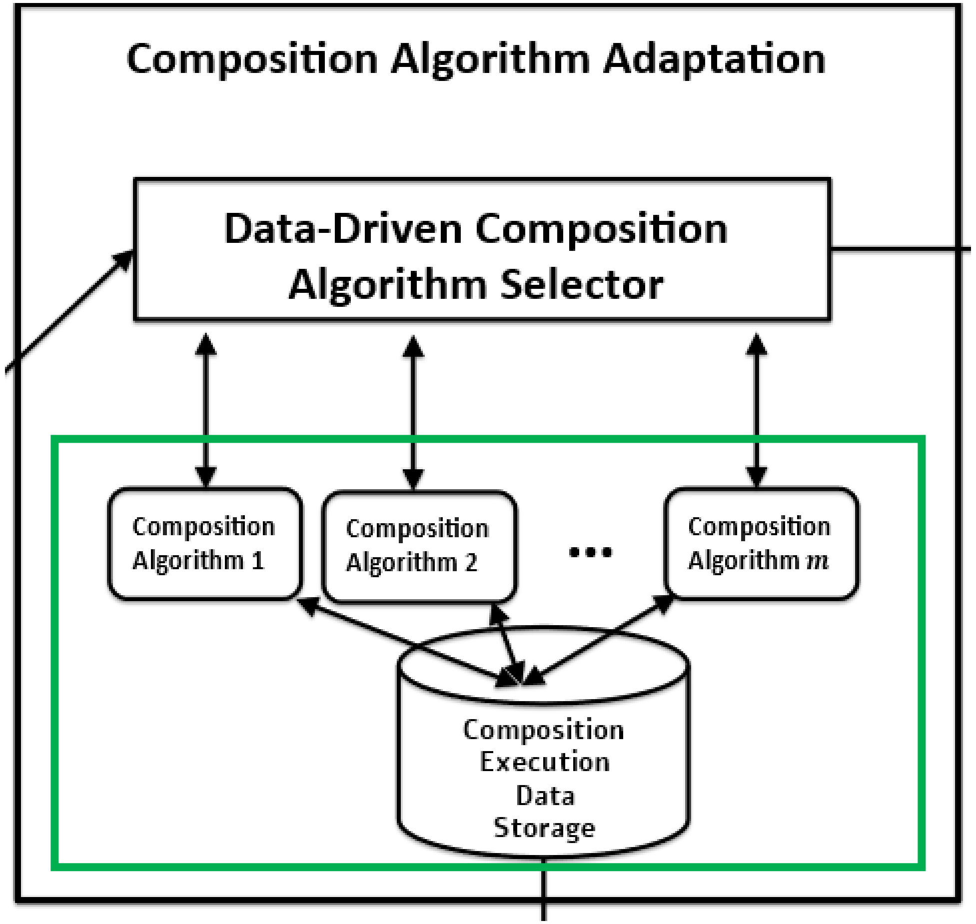
- <u>Goal 1:</u> How can we determine the right composition method for a given service composition task?
- We examine the use of classifiers as a selector
- Composition algorithms considered
  - Multi-Constraint Shortest Path (MCSP) (Yue et al., 2007),
  - Ant Colony System (ACS) (Zhang et al., 2010) and
  - Genetic Algorithm (GA) (Trummer and Faltings, 2011)



Composition Algorithm Selector

- Dependency graphs
  - Number of Abstract Service (#AS): 5, 10, 20, 30, 40
  - Number of Candidate Services (#CS): 5, 10, 15, 20, 30, 35, 40

- QWS[1] dataset randomly sampled for candidate services
- We use the Lp metric (Zhang et al., 2010) to compute multi-objective solution quality



[1]https://qwsdata.github.io/

Composition Algorithm Selector

- Example dataset entry

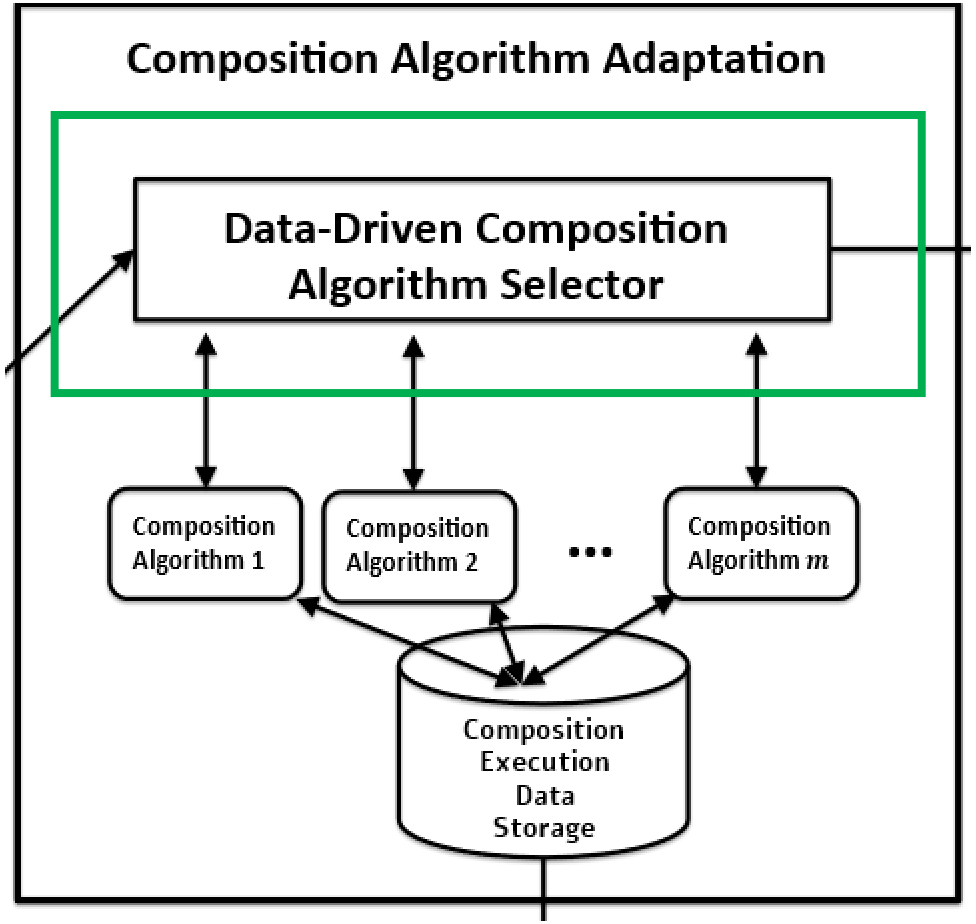
# AS	# CS	Weight 1	…	Weight 8	QoS 1	…	QoS 8	Time (s)	Memory (kB)	Label
20	25	0.125		0.2	0.4		0.8	15	125,000	GA

- Labeling scheme

# AS	# CS	Desired solution quality (Lp)	Delivered solution quality (Lp)	Allotted time	Utilized time	Allocated memory	Utilized memory	Label
20	25	0.7	0.90	15	25	90,000	120,000	MCSP
20	25	0.7	0.60	15	5	90,000	56,000	GA
20	25	0.7	0.71	15	16	90,000	95,000	ACS

Composition Algorithm Selector

- Classifiers considered
  1. Random Forest,
  2. Decision Tree,
  3. Logistic Regression,
  4. Quadratic Discriminant Analysis,
  5. Support Vector Machines
- Dataset shuffled, 70/30 train-test split
- 5 fold cross validation repeated 10 times



Preliminary Results: Classifier Selection

Classifier	Accuracy	F1-score
Random Forest	0.95	0.94
Decision Tree	0.94	0.93
QDA	0.92	0.92
Logistic Regression	0.88	0.88
SVM - rbf	0.51	0.39
SVM - sigmoid	0.52	0.35

- *QDA performance indicates nonlinear decision boundary, different inter-class variances. The best performance was found with regularization parameter was found to be 0.5*
- *Thus, the algorithm selector is Random Forest with 200 trees and a maximum depth of 6.*

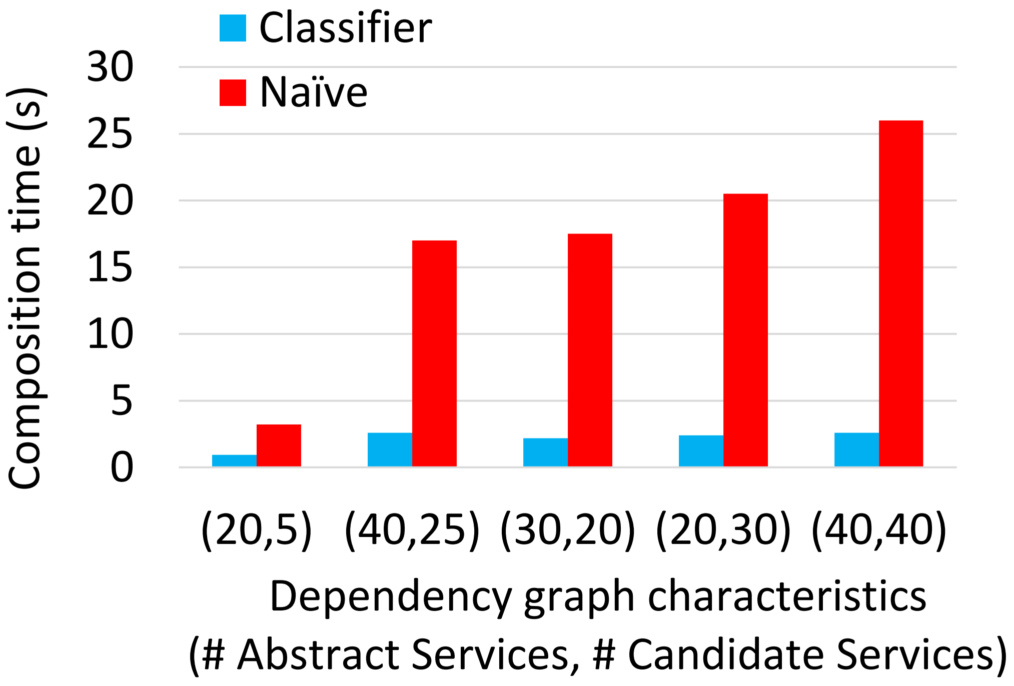
Conclusions on Goal # 1

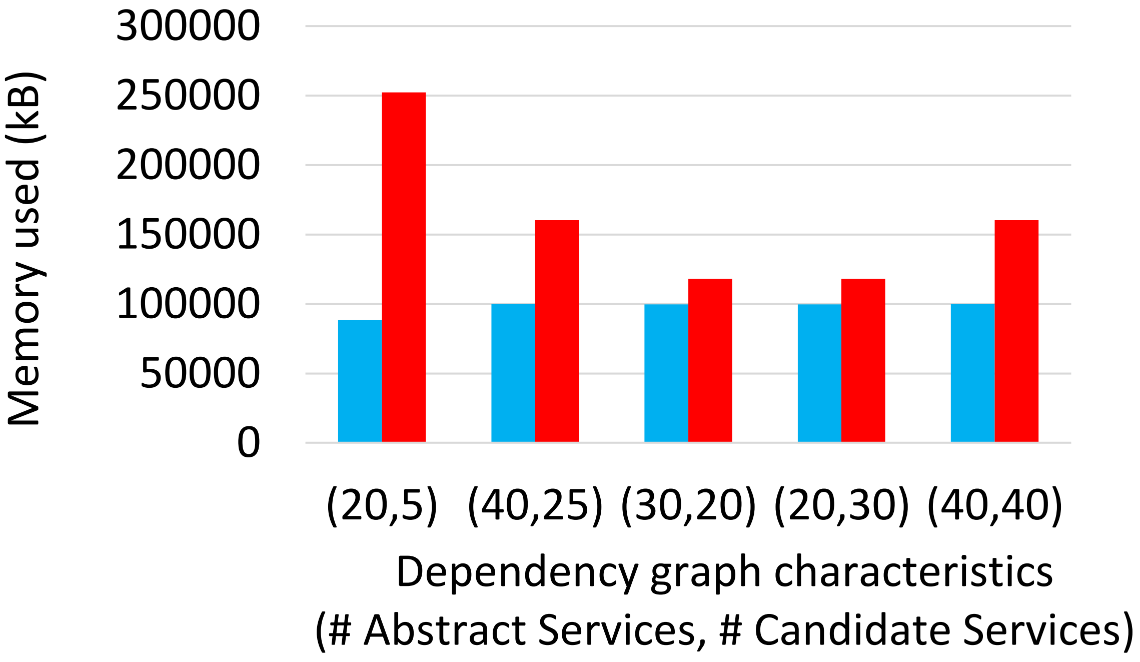
- Goal 1: How can we determine the right composition method for a given service composition task?
- A set of classifiers evaluated
  - Considering different decision boundaries between variables
  - Presence of non-linear decision boundaries indicated
  - Performance observed for one set of solution utilities
- Findings
  - Random forest outperforms the rest

Classifier-based Selector Performance

- Goal 2: How does selecting a composition algorithm on a per-instance basis perform compared to a pre-selected algorithm?
- We compare classifier selection to a naive approach, which selects based purely on solution utility
- Measured time and memory resources used by both selections

Preliminary Results: Usage

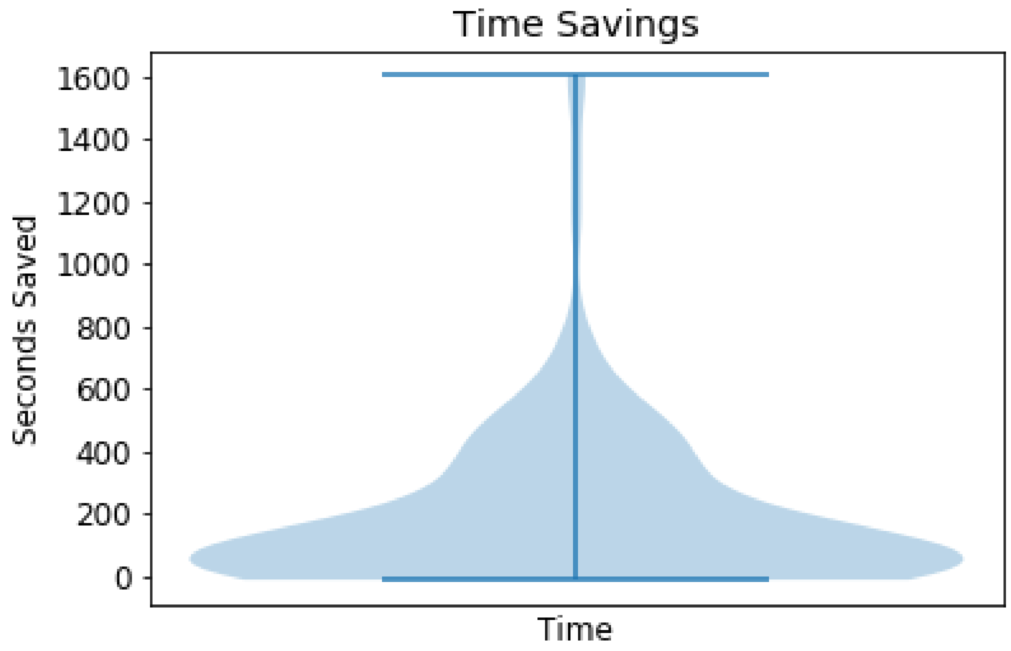


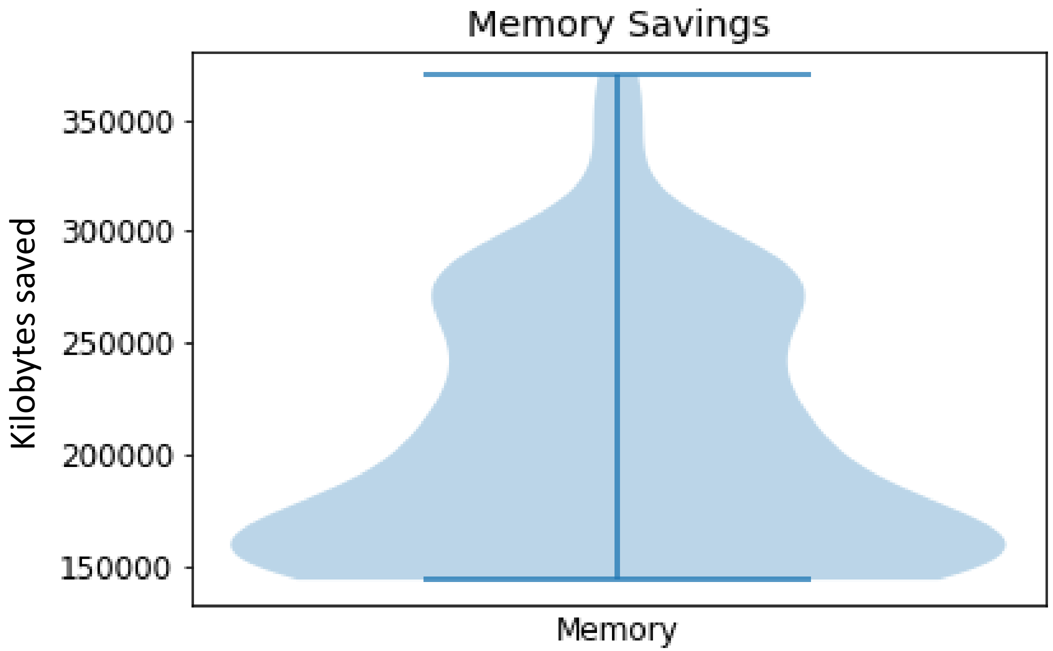


- *On average, classifier-based selection took :*
  - *33% less time*
  - *24.8% less memory*

Dataset Creation | Classifier Selection | Evaluation of Approach

Preliminary Results: On Average





- *Observation: Both time and memory savings observed for the test set. At a minimum about 150000 kB were saved, while time savings peaked approximately around 100 seconds*

Conclusions on Goal 2

- Goal 2: How does selecting a composition algorithm on a per-instance basis perform compared to a pre-selected algorithm?
- Performance compared to a naïve approach
  - Classifier-based selection saves time and memory required for composition
- Findings
  - On average, classifier selected algorithms took
    - 33% less time
    - 24.8% less memory
  - Overheads
    - Selection overhead: ~1% of processing time

Future Work

- Preliminary results demonstrate considerable compute resource savings for various solution utility requirements.
- We will expand our experiments to include
  - Diverse solution qualities
  - Other types of learning algorithms that do not require labels
  - Addition of diverse composition algorithms
- In addition to this, we will deploy our approach as an online feedback loop to be used at runtime.

Summary

- We proposed selection of composition algorithms per composition task
- Preliminary results demonstrate considerable compute resource savings
- Future work includes expansion of experiments include
    - Diverse solution qualities
    - Other types of learning algorithms that do not require labels
    - Deployment as an online feedback loop

References

1. Al-Helal, H., Gamble, R.: Introducing replaceability into web service composition, IEEE Transactions on Services Computing, 7(2), 198–209 (April 2014).https://doi.org/10.1109/TSC.2013.23

2. Al-Masri, E., Mahmoud, Q.H.: Qos-based discovery and ranking of web services. In:2007 16th International Conference on Computer Communications and Networks.pp. 529–534 (Aug 2007). https://doi.org/10.1109/ICCCN.2007.4317873

3. Ali, N., Solis, C.: Self-adaptation to mobile resources in service oriented architecture. In: 2015 IEEE International Conference on Mobile Services. pp. 407–414 (June2015). https://doi.org/10.1109/MobServ.2015.6

4. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web. p. 881–890. WWW ’09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1526709.1526828,https://doi.org/10.1145/1526709.1526828

5. Ardagna, D., Baresi, L., Comai, S., Comuzzi, M., Pernici, B.: A service-based frame-work for flexible business processes. IEEE Software28(2), 61–67 (2011.

6. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering33(6), 369–384 (2007)

7. Ardagna, D., Mirandola, R.: Per-flow optimal service selection for webservices based processes. Journal of Systems and Software, 83(8), 1512–1523(2010).https://doi.org/https://doi.org/10.1016/j.jss.2010.03.045,

8. Bouguettaya, A., Singh, M., Huhns, M., Sheng, Q.Z., Dong, H., Yu, Q., Neiat,A.G., Mistry, S., Benatallah, B., Medjahed, B., Ouzzani, M., Casati, F., Liu,X., Wang, H., Georgakopoulos, D., Chen, L., Nepal, S., Malik, Z., Erradi, A.,Wang, Y., Blake, B., Dustdar, S., Leymann, F., Papazoglou, M.: A service computing manifesto: The next 10 years. Commun. ACM60(4), 64–72 (Mar 2017),http://doi.acm.org/10.1145/2983528

9. Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli,G.: Dynamic qos management and optimization in service-based systems, IEEE Transactions on Software Engineering37(3), 387–409 (May 2011).https://doi.org/10.1109/TSE.2010.92

10. Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Lo Presti, F., Mirandola,R.: Moses: A platform for experimenting with qos-driven self-adaptation policies for service oriented systems. In: de Lemos, R., Garlan, D., Ghezzi, C., Giese, H.(eds.) Software Engineering for Self-Adaptive Systems III. Assurances. pp. 409–433.Springer International Publishing, Cham (2017)

11. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., Mirandola, R.: Towards Self-adaptation for Dependable Service-Oriented Systems, pp. 24–48. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10248-6 ̇2, https://doi.org/10.1007/978-3-642-10248-6 ̇2

12. Cardoso, J.: Approaches to compute workflow complexity. In: Leymann, F.,Reisig, W., Thatte, S.R., van der Aalst, W. (eds.) The Role of Business Processes in Service Oriented Architectures. No. 06291 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum f ̈urInformatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany (2006),http://drops.dagstuhl.de/opus/volltexte/2006/821

References

13. Cho, J., Ko, H., Ko, I.: Adaptive service selection according to the service density in multiple qos aspects. IEEE Transactions on Services Computing9(6), 883–894 (Nov2016), https://doi.org/10.1109/TSC.2015.2428251
14. Fugini, M.G., Pernici, B., Ramoni, F.: Quality analysis of composed services through fault injection. Information Systems Frontiers11(3), 227–239 (07 2009)
15. Gomaa, H., Hashimoto, K., Kim, M., Malek, S., Menasce, D.A.: Software adaptation patterns for service-oriented architectures. In: Proceedings of the 2010 ACM Symposium on Applied Computing. pp. 462–469. SAC '10, ACM, New York, NY, USA(2010), http://doi.acm.org/10.1145/1774088.1774185
16. Jatoth, C., Gangadharan, G., Buyya, R.: Computational intelligence basedqos-awarewebservicecomposition:Asystematicliteraturereview.IEEE Transactions on Services Computing10(03), 475–492 (July 2017).https://doi.org/10.1109/TSC.2015.2473840
17. King, T., Ramirez, A., Rodolfo, C., Clarke, P.: An integrated self-testing frame-work for autonomic computing systems. Journal of Computers2(11 2007).https://doi.org/10.4304/jcp.2.9.37-49
18. Mutanu, L.: State of runtime adaptation in service-oriented systems: what, where, when, how and right. IET Software13, 14–24(10) (February 2019), https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2018.5028
19. Schuller, D., Siebenhaar, M., Hans, R., Wenge, O., Steinmetz, R., Schulte, S.: To-wards heuristic optimization of complex service-based workflows for stochastic qos attributes. In: 2014 IEEE International Conference on Web Services. pp. 361–368(2014)
20. Trummer, I., Faltings, B.: Dynamically selecting composition algorithms for economical composition as a service. In: ICSOC (2011)
21. Wang, H., Chen, X., Wu, Q., Yu, Q., Hu, X., Zheng, Z., Bouguettaya, A.: Integrating reinforcement learning with multi-agent techniques for adaptive service composition. ACM Trans. Auton. Adapt. Syst.12(2), 8:1–8:42 (May 2017).https://doi.org/10.1145/3058592, http://doi.acm.org/10.1145/3058592
22. Wang, L., Li, Q.: A multiagent-based framework for self-adaptive software with search-based optimization. In: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 621–625 (Oct 2016).https://doi.org/10.1109/ICSME.2016.16
23. Wang, L., Li, Q.: A multiagent-based framework for self-adaptive software with search-based optimization. In: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 621–625 (Oct 2016).https://doi.org/10.1109/ICSME.2016.16
24. Yu, T., Zhang, Y., Lin, K.J.: Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. Web1(1) (May 2007),http://doi.acm.org/10.1145/1232722.1232728
25. Zhang, W., Chang, C.K., Feng, T., Jiang, H.y.: Qos-based dynamic web service com-position with ant colony optimization. In: Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual. pp. 493–502. IEEE (2010)

Composition Algorithm Adaptation in Service Oriented Systems

14th European Conference on Software Architecture(ECSA), 3rd Context-aware, Autonomous and Smart Architectures International Workshop (CASA), 15 September, 2020

Niranjana Deshpande, Naveen Sharma

Rochester Institute of Technology

nd7896@rit.edu

RIT

This material is based upon work supported by the National Science Foundation under Award No. 1943002.

NSF