Optimization for Self-Adaptive Software Architecture at Runtime

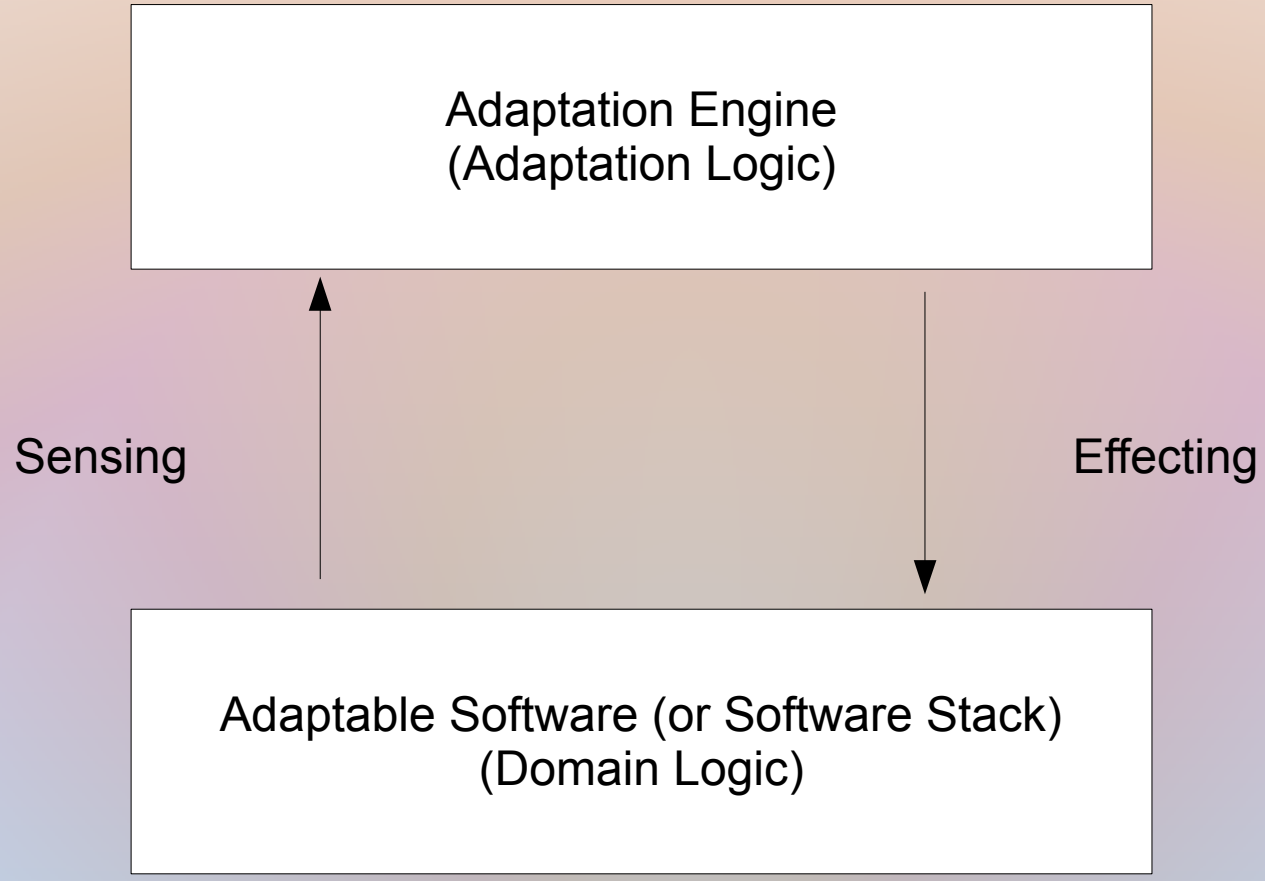
Tao Chen

Self-Adaptive Software

"Software that are able to modify their behaviours and/or structure in response to their perception of the environment and the system itself, and their goals"

De Lemos, Rogério, et al. Software engineering for self-adaptive systems: A second research roadmap. Springer Berlin Heidelberg, 2013

Self-Adaptive Software



Goal of Runtime Optimization in Self-Adaptive Software Architecture

Optimizing the non-functional attributes (e.g., performance, reliability and cost) of the adaptable software, so that their requirements can be better complied.

Goal of Runtime Optimization in Self-Adaptive Software Architecture

- Objective functions:

$$QoS_k(t) = f_k(P_1(t), P_2(t), \ldots P_n(t), \delta)$$

$$Cost(t) = g(P_1(t), P_2(t), \ldots P_n(t))$$

- It is an optimization problem with certain number of objectives:

$$Max/Min(QoS_1(t), QoS_2(t) \ldots QoS_o(t), Cost(t))$$

subject to Service Level Agreement, budget and energy requirements

Problem #1 When to trigger optimization?

Existing work would trigger optimization in self-adaptive software architecture when violation of requirements are detected (either proactive or reactive).

Threshold based approach does not work as it does not resolve the fundamental problems.

What if:

- The violation is trivial?
- The payoff after the optimization is trivial under current circumstance?

Problem #1 Research Questions

- How to quantify the significance of requirements violation?
- How to continually track the changes in requirements violation?
- How to reason about and make trade-off between overhead and payoff when optimization in runtime architecture?

Significance:

In long term, achieving the same or closed adaptation quality with much less number of adaptation, thus less overhead.

Problem #1 Candidate Solutions

- Change detection techniques?
- Machine learning/Ensemble learning?

Problem #2 How to do preference specification easier?

Existing work that apply multi objective optimization in the self-adaptive software optimization reply on the assumption that "it is difficult for stakeholders to correctly specify preference (weights), or the function that update them".

What if:

- There is a comprehensive way to project the stakeholder's preferences?
- There is a way to produce generally good result without preference, if it is really difficult to specify them.

Problem #2 Research Questions

- How to comprehensively translate stakeholders' preference into optimization at architecture runtime.
- How to adopt those preferences in the optimization process?

Significance:

An easier way for stakeholders to influence the optimization process, leading to better productivity in the development of self-adaptive software.

Problem #2 Candidate Solutions

- Model@runtime/Requirement@runtime + optimization algorithms?

- Multi objective optimization for knee points?

Problem #3 How to mitigate the effects of badly defined requirements?

Existing work assume that the requirements of QoS attributes (i.e., constraints) are correctly defined.

What if:

- The engineer who specifies those requirements lack of experience and domain knowledge on self-adaptive software, or they make wrong assumptions?
- - the requirements may be too good, which causes the optimization process struggle to find feasible adaptation decisions.
- - the requirements may be too bad, which might not influence the optimization process but it loses the point of having the requirements.

Problem #3 Research Questions

- How to formulate the optimization problem so that it is resilient to the given requirements?
- How to design the corresponding optimization algorithm?

Significance:

Saving software development effort as the requirements of experience/knowledge of the self-adaptive software are relaxed when specifying the requirements.

Problem #3 Candidate Solutions

- Better constraint handling in optimization?
- Constraints as one or more objectives?

Problem #4 How to apply dynamic optimization to the problem?

Existing work often rely on static optimization, i.e., there is an isolated optimization problems at each point in time.

What if:

- The environment changes are frequent?

Problem #4 Research Questions

- How to formulate the problem as dynamic optimization problem?
- How to adopt dynamic optimization algorithm at software runtime?
- Dynamic optimization vs. static optimization for runtime optimization of self-adaptive software architecture.

Significance:

Potentially improving stability of the software, i.e., quicker response to changes in its states and the environment.

Problem #4 Candidate Solutions

- Optimization algorithms with memory?
- Optimization algorithms with machine learning?

Problem #5 How to perform meta-optimization at the adaptation engine?

Optimization for self-adaptive software architecture work at the problem domain. This has lead to intelligent adaptation engine, which consists of number of intelligent component.

What if:

- Not all of the components are necessary at some points in time?

Problem #5 Research Questions

- How to quantify the usefulness of an intelligent components?

- How to reason about the needs of those components?

Significance:

Achieving less overhead and simplifying the design process of self-adaptive software.

Problem #5 Candidate Solutions

Conclusion

- When to trigger runtime optimization in self-adaptive software architecture?
- How to help stakeholder to better project preferences into the runtime optimization?
- How to improve the stability of runtime optimization against the quality of requirements?
- How to achieve dynamic optimization for self-adaptive software architecture at runtime?
- How to achieve meta-optimization for self-adaptive software architecture at runtime?