Workflow Variability for Autonomic IoT Systems

By Damian Arellanes, Kung-Kiu Lau
Summarized by Kaushik Gedela
Framework of Presentation

- Introduction
- DX-MAN model
- Composition Connectors as Variability Operators
- Feedback Control Loops
- Case Study
Introduction

• Workflows represent IoT systems composed of billions of services with an overwhelming number of interactions. Thus, it becomes infeasible to manually manage such systems as the scale and complexity increases.
• Autonomicity is a crucial desideratum for the management of complex large-scale IoT systems operating in highly dynamic environments. It is a property that allows adapting behaviour at runtime to different contexts with minimal or no human intervention.
• Changing behaviour at runtime in highly variable environments is a complex and challenging task. For that reason, variability-based autonomicity was a requirement
This paper extends the semantics of the DX-MAN service model with autonomic capabilities for IoT systems

*model* that combines variability with behaviour in the solution space, while providing an infinite number of workflow variants for composite IoT services

an approach that avoids dynamic reconfiguration (by using non-deployable and executable only workflows).
**DX-MAN Model**

algebraic model for IoT systems where services and exogenous connectors are first-class entities

\[ S := A | C \]

service S is a stateless distributed software unit with a well defined interface, this can either be Atomic (A), or Composite (C)

A service defines a workflow space \( W \) which is a non-empty (finite or infinite) set, where each \( w \in W \) is a workflow variant that represents an alternative service behaviour
Atomic Services

- An atomic service $A$ is a tuple $\langle IC, O \rangle$ consisting of an invocation connector $IC$ and a non-empty finite set $O$ of $j$ primitive operations.
- It is formed by connecting an invocation connector with a computation unit.
- To satisfy an external request, an invocation connector is responsible for executing a workflow in $W$. 

![Diagram of Atomic Services and Workflow Space]
Composite Services

- a composite service is a tuple $\langle CC, \rangle$ consisting of:
  - a composition connector $CC$ that invokes multiple workflows defined by the composite service
  - a non-empty finite set which is a family of non-empty (finite or infinite) sets of sub-workflow spaces, these can either be atomic sub-service or composite sub-services

- A composite service is a variation point which defines a new non-empty (finite or infinite) workflow space $W$ using the sub-workflow spaces via algebraic references
Workflow Selection

- A composition connector CC is a variability operator that defines the alternative behaviours of a composite service.
- At design-time, an abstract workflow tree is automatically created for a composite service, as a result of composition. It represents the hierarchical control flow structure of a composite service, where n leaves are atomic workflows, composite workflow spaces or any combination thereof.
- A concrete workflow tree enables the selection of a workflow variant at runtime. It particularly sets specific values for the customizable control flow parameters of an abstract workflow tree.
**Sequencer**

- A sequencer connector SEQ uses the Kleene star operation to allow the repetition of n elements, resulting in infinite sequences. It then defines an infinite workflow space for a composite service.
- A sequencer is a function defined as:
  - SEQ: \(\rightarrow W\)
For example: We choose robot 3, the abstract workflow tree and workflow tree would be:
Parallelizer

- A parallelizer connector PAR allows the execution of multiple elements in parallel.
- As it supports element repetition, it defines $\infty$ parallel workflows for a composite service s.t. each $w_i \in W, i=1,\ldots, \infty$ is a workflow executing all the elements in parallel.
- PAR: $\mapsto W$
For Example: The workflow and abstract workflow of home 5 would be as shown using the parallelizer.
Feedback Control Loops

- In DX-MAN, workflow spaces represent the adaptation space of a composite service, since they provide a wide range of workflow variants, each representing a different behaviour.
- DX-MAN does not require to link the variability model with the behavioural model, as those dimensions are mixed in the semantics of a composite service.
- The selection of workflow variants (i.e., changing behaviour) takes place at runtime whenever the context changes. This is done by building the concrete workflow tree that best adapts to the current context.
MAPE-K

- we use Monitoring, Analysis, Planning, Execution and Knowledge (MAPE-K) which endow composite services with autonomicity
- MAPE-K is a feedback control loop consisting of multiple sensors, a monitor, an analyzer, a planner, an executor, an effector and a knowledge base

shows that a MAPE-K loop manages a composite service and collects information from the external context
MAPE-K cont.

- The MAPE-K components are able to read and update the knowledge base which stores relevant information for realizing autonomic behaviour.
- By default, the knowledge base stores the abstract workflow tree for the managed composite service.
- Monitor uses sensor data to build a context model for the external environment.
- Analyzer then determines if a new behavior is required.
- Planner then determines the best workflow variant given current settings.
- Executor transforms it into a concrete workflow tree matching the structure of the abstract workflow tree.
Why MAPE-K?

MAPE-K loops are not structured hierarchically as they never interact. Instead, they only select a workflow for the managed composite service (at any level in the hierarchy) and they execute new workflows (when control is blocked in the managed composition connector) or replace an existing workflow with a “better one” (when control has already passed through).

This is not the case in regular IoT systems as the DX-MAN model will continuously build larger and larger hierarchies.
Case-study: Smart Home

2 different values are considered and 1 constraint:

1. Tidiness and Energy Consumption
2. User Interaction

User Interaction is only ever required for 1 task and that is cooking

Tidiness must always add up to 1

Overall Utility is calculated through this equation

\[ U(w_i, \phi) = \omega_1 \cdot f_1(w_i, \phi) + \omega_2 \cdot f_2(w_i, \phi) + \omega_3 \cdot f_3(w_i) / (\omega_1 + \omega_2 + \omega_3) \]
Conclusions

- A MAPE-K loop selects the composite service behaviour (i.e., the workflow variant) that best adapts to the current context. As workflows are non-deployable and executable only, the executor changes a composite service behaviour by executing the selected variant instead of dynamically reconfiguring the whole workflow.
- DX-MAN currently enables control flow variability, making it suitable for operations that do not require data, e.g., door closing.
- DX-MAN is suitable for closed environments.