Modeling and Analyzing Openness Trade-Offs in Software Platforms
A Goal-Oriented Approach

Mahsa H. Sadi and Eric Yu
Department of Computer Science
University of Toronto

Feb 28th, REFSQ 2017, Germany

Open Innovation
Software organizations open up their processes and platforms to external developers
To use external ideas and paths to market

External developers
A part of a software ecosystem offering applications


Example – Mobile Platforms

Other Examples of Open Platforms

Problem Statement
Opening up platforms is a non-trivial problem

Openness requirements are in competition with security, performance, controllability

e.g. Distributing features versus maintainability


Research Objective

To help identify suitable design strategies for opening up a software platform

Contributions of this Study

1. Requirements and Concerns in Open Software Platforms

2. Modeling and Analysis of Requirements in Open Software Platforms
Requirements and Concerns in Open Software Platforms

Openness Requirements

The specific concerns and quality requirements that openness introduces on platform designs

Business-Level Openness Requirements – Example

Customer-Related Objectives

Stickiness of the Platform

Growing the network size of complementary applications hardens switching to a different platform


System-Level
Openness Requirements – Example

**Extensibility**
The ease of adding a new application to a platform
Can be further refined into:
- Composability, Deployability, Stability, Configurability

[References]

General Concerns in Designing
Software Platforms – Example

**Security**
Possible defective or malicious code in external applications may disable the overall system
- The integrity of platform services and data
- The confidentiality and privacy of the end-users’ data
- Correct operation of features developed by multiple parties

[References]

Summary of the First Part

**Two Characteristics of the Identified Requirements**
- Non-Functional, related to:
  - Quality requirements (e.g. Accessibility, Extensibility)
  - Business objectives (e.g. Stickiness, Ecosystem gravity)
- Interacting with each other:
  - Competition
  - Synergy
Contributions of this Study

1. Requirements and Concerns in Open Software Platforms

2. Modeling and Analysis of Requirements in Open Software Platforms

The Proposed Approach

Non-Functional Requirements Analysis Method


The Case Under Study - 1

AUTOSAR Platform

An Operating System for automotive platforms

**Functionalities:** Controlling the electronic units of a vehicle
(e.g. the speed, the brakes, the windows)

Opened up to a variety of 3rd party applications
(Certified, uncertified)


The Case Under Study - 2

The Specific feature under study

The design of Data Provision Service

- How platform communicates data with 3rd party apps
- How 3rd party applications communicate data with each other

**Objective:** Revisit the Design of Data Provision Service
Step 1: Identifying and Prioritizing Requirements - 1

**Domain-Specific Design Requirements**

- Dependability: Many embedded domains have stringent dependability requirements. These domains are probably not the first adopters of an ecosystem-based approach to software development. However, if that was the case, the embedded platform would satisfy real-time requirements for the execution of individual applications, integrity requirements, high availability, and mechanisms to eliminate undesired feature interaction if several applications interact with the same actuators.


Step 1: Identifying and Prioritizing Requirements - 2

**Domain-Specific Design Requirements**

- Dependability
- Security [Platform] | High
- Performance | High
- Integrity
- Availability
- Response Time
- Access-Time

Step 1: Identifying and Prioritizing Requirements - 3

Step 1: Identifying and Prioritizing Requirements - 6
Step 2: Identifying the Design Objective and Alternative Design Options

**Design Objective:** To provide data service to 3rd Party applications.

- **Centralized Data Provision**

  - All data is stored and exchanged through the platform (but most data is isolated to a single application) through an API. There are no other mechanisms provided for information exchange between applications besides through the platform. Data and provided services from hardware abstractions are accessed through the API by either an explicit get/put and/or subscribe both at run-time. i.e. data is pulled by those applications needing it and not pushed to them. There is also an API to determine the available data set at run-time on a particular embedded device.

  - The platform offers a predictable arbitration mechanism when several applications want to access the same actuator and use a particular service. The arbitration is implemented by the board.


Step 3: Evaluating design options against the design requirements

**Design Requirement:** Response Time

**Evaluation:** (−)

All data operation requests should pass through a central gateway and a queue controlled by the platform.

Centralized Data Provision

Analyzing the Fulfillment of Design Requirements in Centralized Data Provision
Centralized Data Provision

Performance is sacrificed for Openness.

Comparison of the Design Alternatives

Step 2: Identifying the Alternative Design Options

**Design Objective:** To provide data service to 3rd Party applications

Alternative 1: Centralized Data Provision

Alternative 2: Semi-Centralized Data Provision

Discussion

**Performance:** Crucial for real-time Operations of the platform

**Composability and Deployability:** Crucial for accommodating 3rd party applications overtime

A combination of centralized and semi-centralized data provision can be used.
Summary

Objective:
To identify suitable open platform design strategies

Proposed Solution:
1. Consider openness as a class of non-functional requirements
2. Refine it in parallel with other important concerns
   Use a goal-oriented requirements analysis approach
3. Use it as criteria for selecting a suitable design option
   Analyze potential trade-offs
   Balance the fulfillment of the requirements

Future Work
1. To assess the applicability of the proposed method in real-world open software platform projects
2. To provide knowledge support for refining openness requirements
3. To enrich the analytical capabilities of the proposed method for determining effective openness design strategies
4. To compare with peer approaches, such as ATAM

E-mail: mhsadi@cs.toronto.edu