Modeling Families of Public Licensing Services: A Case Study

Guillermina Cledou Luis. Barbosa

HASLab INESCTEC and Universidade do Minho

FormaliSE2017

Outline

1. Part I
   - Context and motivation
   - Modeling families of services with FTA
   - Verifying properties of families of services with Uppaal

2. Part II
   - Analyzing FTA
   - Enriching FTA

Service Delivery in E-Government

Challenges

- Rapid development
- Service integration
- Cost reduction
- Conformance with laws and regulations

In Practice

- Ad-hoc ICT solutions disregarding common functionality and shared processes

Service Delivery in E-Government

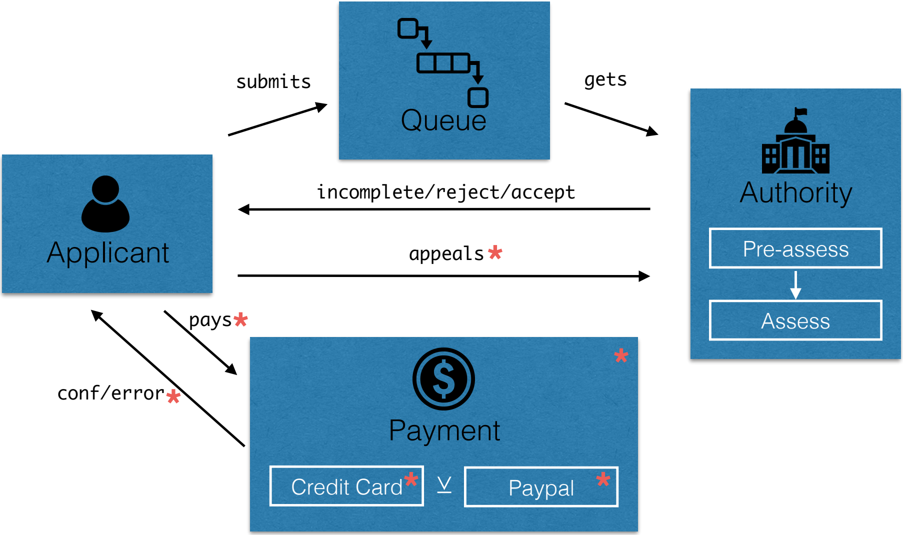
Challenges

- Rapid development
- Service integration
- Cost reduction
- Conformance with laws and regulations

In Practice

- Ad-hoc ICT solutions disregarding common functionality and shared processes

Public Licensing Services



How can we address existing challenges?

Challenges

$$\left.\begin{array}{l}\bullet\ \text{Rapid development}\\ \bullet\ \text{Service integration}\\ \bullet\ \text{Cost reduction}\end{array}\right\}\ \text{Software Product Lines}$$

$$\left.\bullet\ \text{Conformance with laws and regulations}\right\}\ \text{Formal Methods}$$

Software Product Line

A *set* of software systems that share a high number of *features* while differing on others, where concrete configurations are derived from a core of common assets in a prescribed way.

Feature

- A characteristic or behavior of a system that is visible to the user.
- e.g., *pay*, *cc*, *pp*, ...

Feature model

- Expresses valid feature combinations, i.e., the set of systems that can be derived from the SPL
- e.g., {{*pay*, *cc*, *pp* }, {*pay*, *cc* }, {*pay*, *pp* },{}}

Software Product Line

A *set* of software systems that share a high number of *features* while differing on others, where concrete configurations are derived from a core of common assets in a prescribed way.

Feature

- A characteristic or behavior of a system that is visible to the user.
- e.g., *pay*, *cc*, *pp*, ...

Feature model

- Expresses valid feature combinations, i.e., the set of systems that can be derived from the SPL
- e.g., $\{\{$*pay*, *cc*, *pp* $\}$, $\{$*pay*, *cc* $\}$, $\{$*pay*, *pp* $\}$,$\{\}\}$

Software Product Line

A *set* of software systems that share a high number of *features* while differing on others, where concrete configurations are derived from a core of common assets in a prescribed way.

Feature

- A characteristic or behavior of a system that is visible to the user.
- e.g., *pay*, *cc*, *pp*, ...

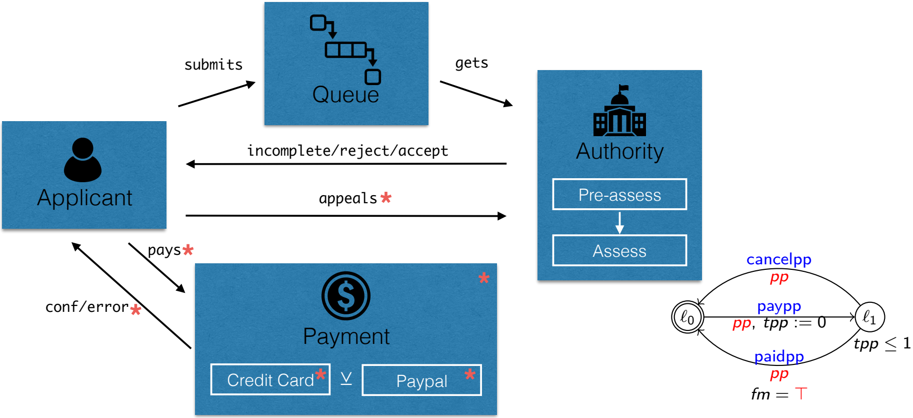
Feature model

- Expresses valid feature combinations, i.e., the set of systems that can be derived from the SPL
- e.g., {{*pay*, *cc*, *pp* }, {*pay*, *cc* }, {*pay*, *pp* },{}}

A Modeling formalism for SPL

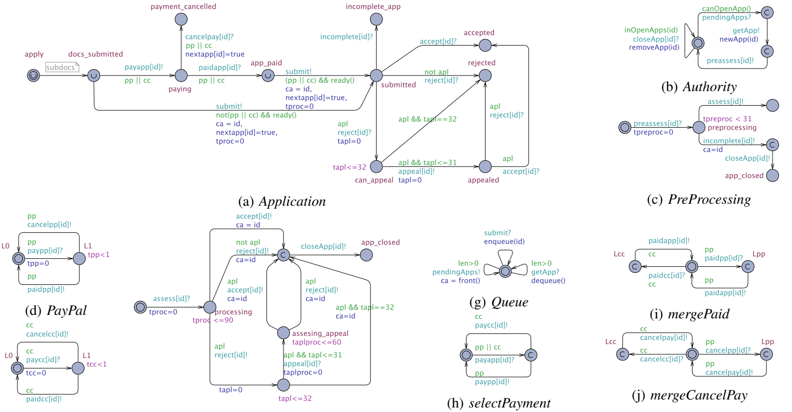
Feature Timed Automata (FTA)

- Extends Timed Automata with *variability*
- Enables the verification of the entire SPL by capturing its behavior in a single model



Uppaal

- Real-time model checker
- Used by academics and industry



(a) *Application*

(b) *Authority*

(c) *PreProcessing*

(d) *PayPal*

(e) *CreditCard*

(f) *Processing*

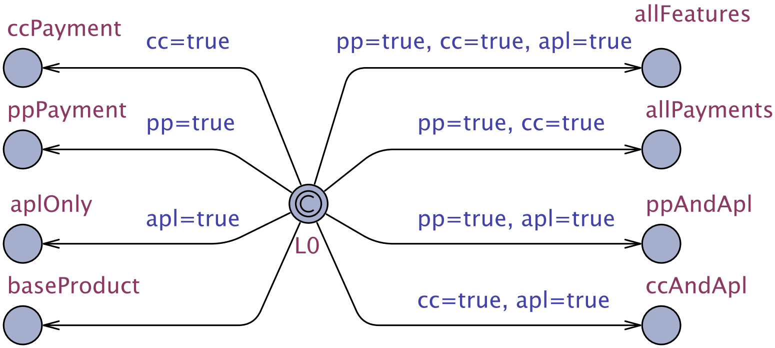
(g) *Queue*

(h) *selectPayment*

(i) *mergePaid*

(j) *mergeCancelPay*

Modeling variability in Uppaal



Verifying properties in Uppaal

Example properties

- An application eventually results in accepted, rejected, incomplete, or canceled
- An application is processed within 121 days
- An application can not be opened by more than one authority
- ...

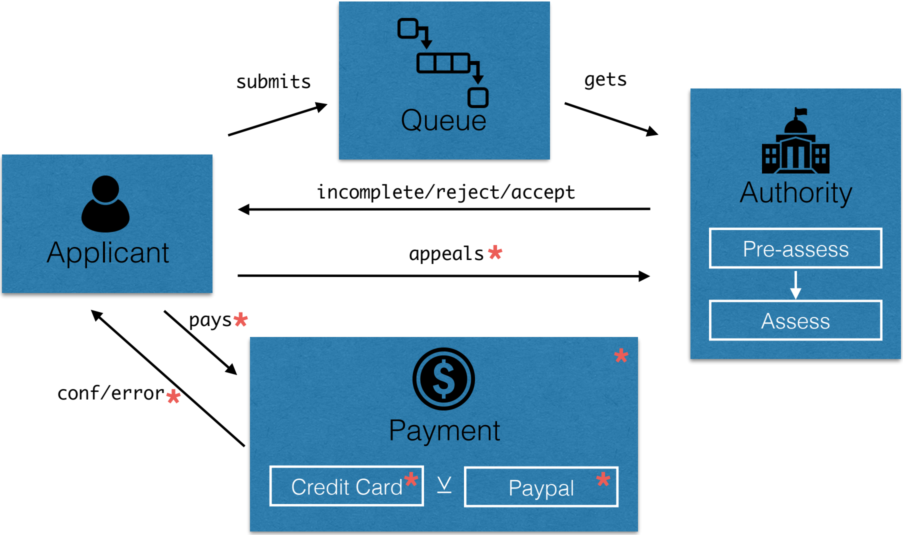
<table>
<tr><th>Property</th><th></th></tr>
<tr><td>Liveness</td><td><code>ap0.apply --> (ap0.accepted || ap0.incomplete_app || ap0.payment_cancelled || ap0.rejected)</code></td></tr>
<tr><td></td><td><code>(mergeCancelPay(0).Lpp || mergeCancelPay(0).Lcc) --> ap0.payment_cancelled</code></td></tr>
<tr><td>Reachability</td><td><code>!cc --> !(exists(i:app_id) (CreditCard(i).L1 || mergeCancelPay(i).Lcc || mergePaid(i).Lcc))</code></td></tr>
<tr><td>Safety</td><td><code>A[] ap0.submitted imply ap0.tproc <= 90+31</code></td></tr>
<tr><td></td><td><code>A[] ap3.appealed imply ap3.tapl <=60</code></td></tr>
<tr><td></td><td><code>A[] forall(i:app_id) !(auth0.inOpenApps(i) && auth1.inOpenApps(i))</code></td></tr>
</table>

Outline

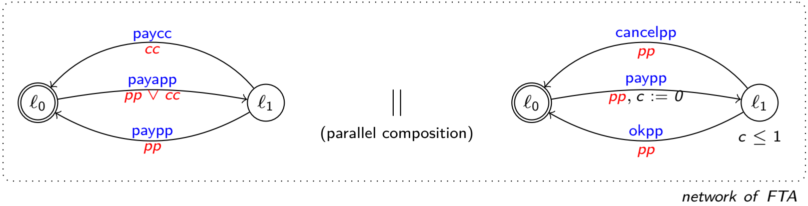
1. Part I
   - Context and motivation
   - Modeling families of services with FTA
   - Verifying properties of families of services with Uppaal

2. Part II
   - Analyzing FTA
   - Enriching FTA

Analyzing FTA



Analyzing FTA



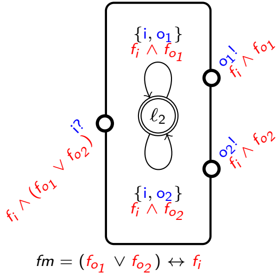
Feature Timed Automata (FTA)

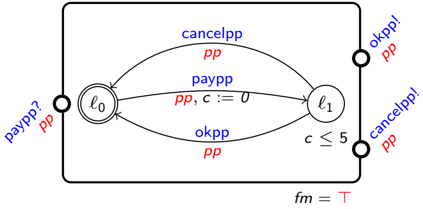
- Disregards modular and compositional aspects of SPL development
- Implicit communication points
- Lack of variability composition
- Lack of reusable common orchestration mechanisms

Extending FTA

Interface Featured Timed Automata (IFTA)

- Extends FTA with interfaces that restrict they way automata are composed
- Multi-action transitions to simplify design



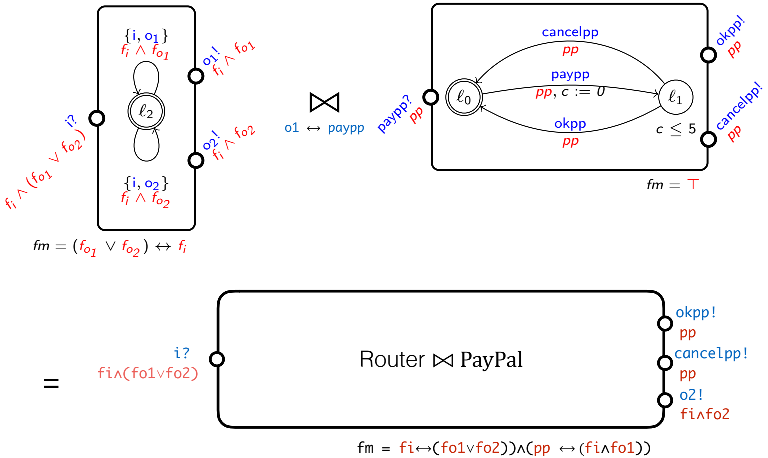


- ?,! denote inputs and outputs interfaces, respectively.
- each interface has associated an inferred *feature expression*.

Extending FTA

Interface Featured Timed Automata (IFTA)

- Explicit communication points + composition of variability



Implementation

Scala DSL: `https://github.com/haslab/ifta`

- Specification of IFTA
- Uppaal
- Interactive representation
- Dot

Conclusions

E-Government

- Unexplored domain with respect to SPL + Formal methods

FTA

- Allows to simplify the modeling and verification of families of timed automata
- Can be enriched to reason about variability during composition

IFTA

- *Multi-action transitions* simplify design
- *Interfaces* enables reasoning about variability + visual feedback
- *Composition* takes into account the feature models
- Limitations in the implementation
  - Uppaal doesn't work very well with sequence of committed states
  - Size of IFTA composition can growth quickly

Questions?

Conclusions

E-Government

- Unexplored domain with respect to SPL + Formal methods

FTA

- Allows to simplify the modeling and verification of families of timed automata
- Can be enriched to reason about variability during composition

IFTA

- *Multi-action transitions* simplify design
- *Interfaces* enables reasoning about variability + visual feedback
- *Composition* takes into account the feature models
- Limitations in the implementation
  - Uppaal doesn't work very well with sequence of committed states
  - Size of IFTA composition can growth quickly

Questions?