An Approach to Web Services Oriented Modeling and Validation

Yujian Fu, Zhijiang Dong & Xudong He

Florida International University

April 16, 2006

Outline

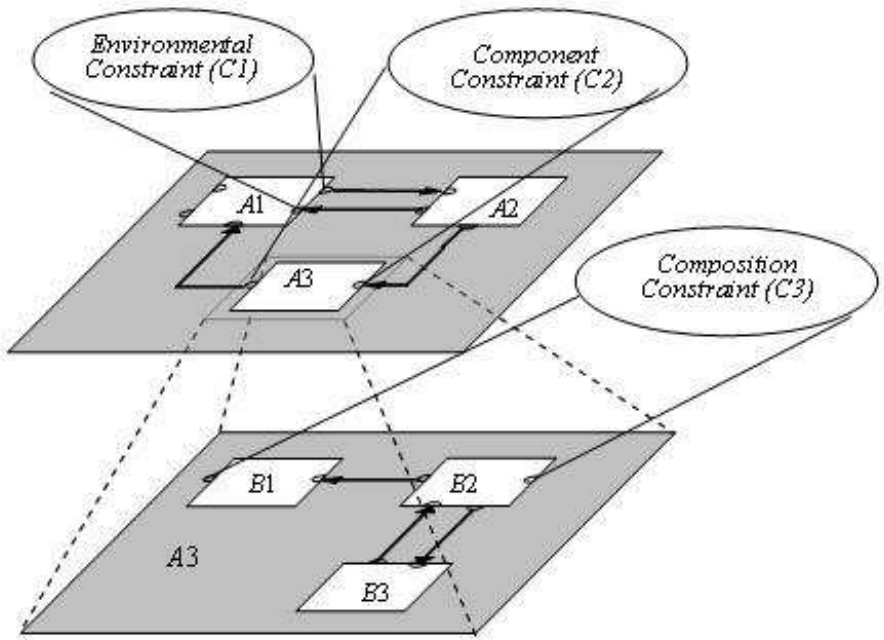
- Motivation
- Preliminaries
- Our Approach
  - Service Oriented Software Architecture Model (SO-SAM) +
  - Runtime Checker
- Case Study
  - Properties for Webbed Application
  - Experiment results
- Conclusions

Motivations

- Current web description languages (WDLs)
  - Less address the web integration and monitoring
  - Most of existing tools test only one or some aspects of Webbed applications
  - Need for verification frameworks and tools to ensure successful deployment of webbed applications
- Current software architecture description languages (ADLs)
  - Have no sufficient ability in the description of web services characteristics, e.g., service request, register, provide and location …
  - Absence of sufficient tool support to create webbed applications of high quality
- Service oriented architecture
  - Without formal semantics
  - Absence of sufficient tool support to create webbed applications of high quality

Preliminaries

- SAM – Software Architecture Model

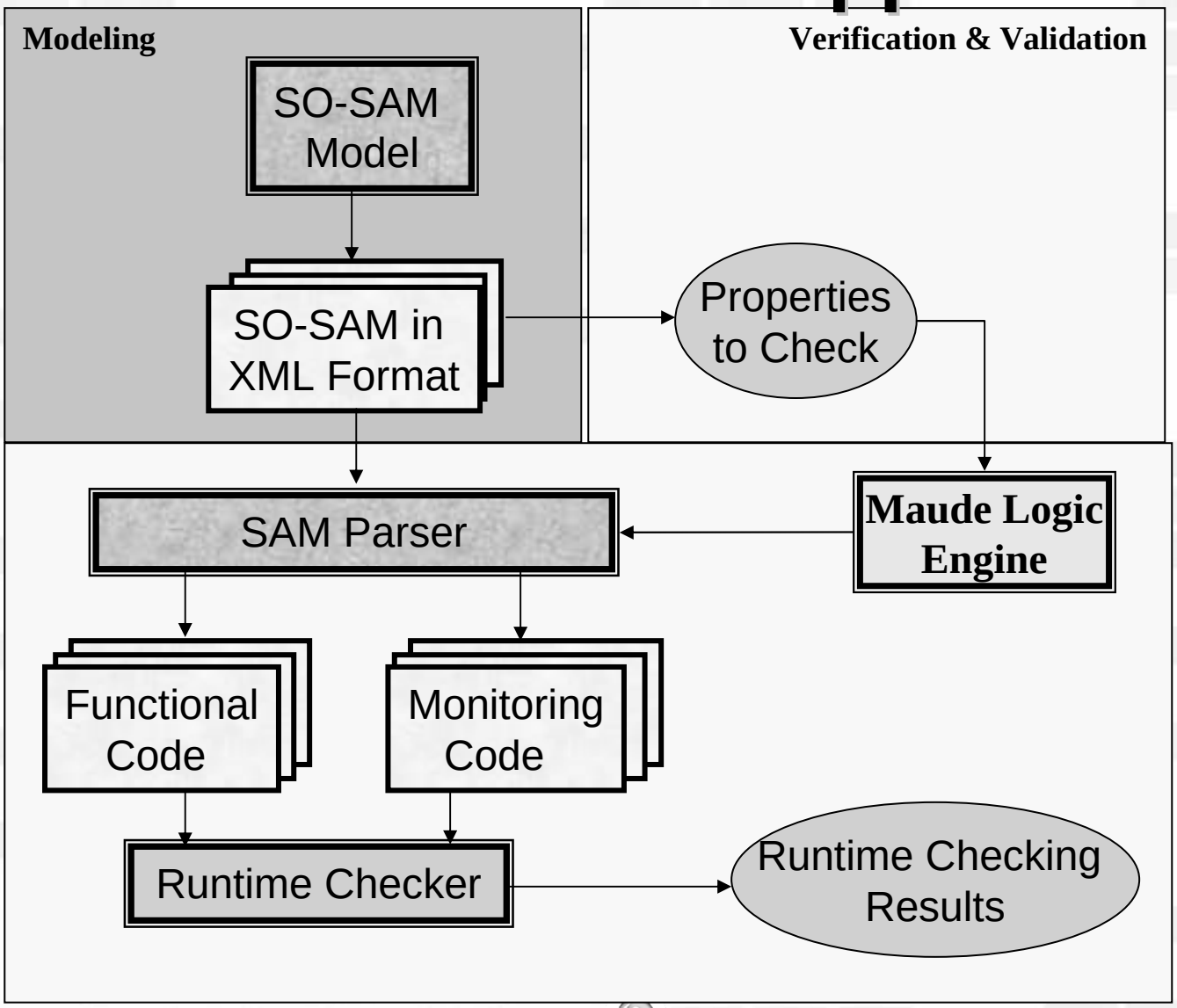


- Composition C = {C_m, C_n, C_r}
- Mapping function f
- Behavior Model – Petri net (PN)
- Property Specification – Temporal Logic (TL)
- Connect through port

Our Approach

- Objective
  - To develop an integrated framework and use available prototype tools to automate verification of properties of webbed applications
- Approach
  - The use of mathematically proven methods (SO-SAM) for formal specification of web services
  - The methodology is based on the Petri nets and Temporal logic
  - Identify the initial and final ports as well as the service sort to describe the web applications
  - Specifying properties for webbed applications using temporal logic
  - Verifying the correctness of the system model

Structure of Approach



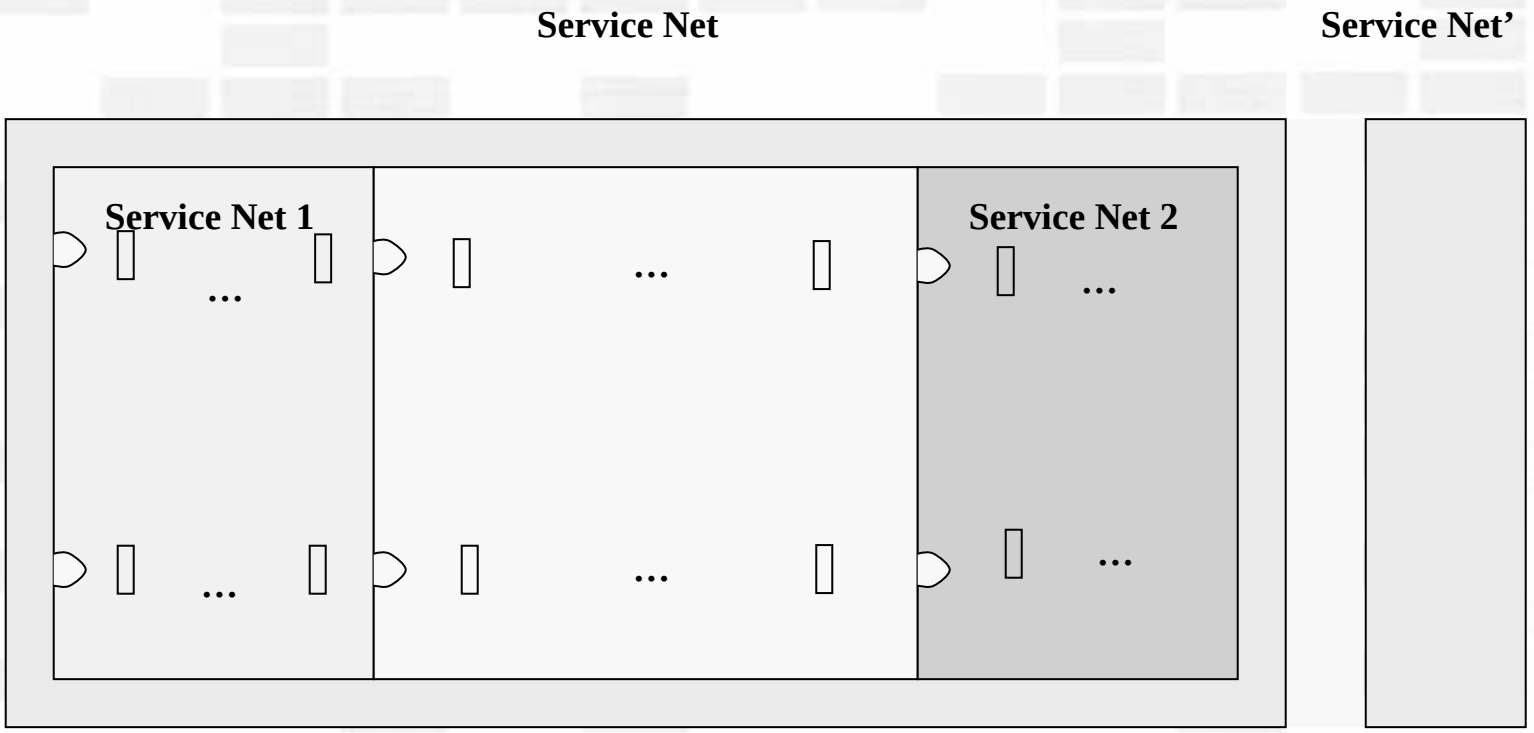
Runtime Checking

- Light-weighted formal verification techniques
  - Mainly on the implementation level
  - Properties have to be formally defined
  - Check the current execution path
- Model Checking approach is
  - Automatic
  - Exhaustive search
  - Suffering the state explosion problem

SO-SAM Model

- SO-SAM model is an extension of SAM
  - Service component: Initial ports, final ports
  - Composition: initial ports of one service must connect with the finial ports of another or vice versa
  - Service sort
  - Each connector cannot be a composition
- Static semantics are Petri net semantics with the new service signature (service sort)
- Dynamic semantics are Petri nets' dynamic semantics
- This extension reuses most of Petri nets' & SAM's semantics. The integration of services can be reached by the execution of the model either statically or dynamically.

A Simple Example



An Online Shopping Example

Customer

Customer browses on the web for some products shown in the website.
The customer clicks on the AddToShoppingCart button (hyperlink), which is taken through a secure shopping cart software and checkout process (hosted on Payment Online secure servers) where credit card transaction information can be entered. After processing the credit card online, authorization responses are then returned to the customer to show the transaction status. If the transaction is successful the customer confirms this transaction, the system generates the order and shipping information and informs customer through email or browser. Otherwise, the transaction is failed, and the error message is shown to customer through browser.

ShoppingCart

Shopping cart keeps a list of a customer's items and indexes each item. When the checkout button sends the message to the cart, all items of the customer (identified by customer ID) will be sent to the retailer's online platform.

Warehouse

The web keeps all avaiable information of the products of retailers, such as product name, category, price, etc. If the quantity of a required item exceeds the storage in the warehouse, the whole transaction is discarded. Otherwise, order can be generated.

OrderProcessing

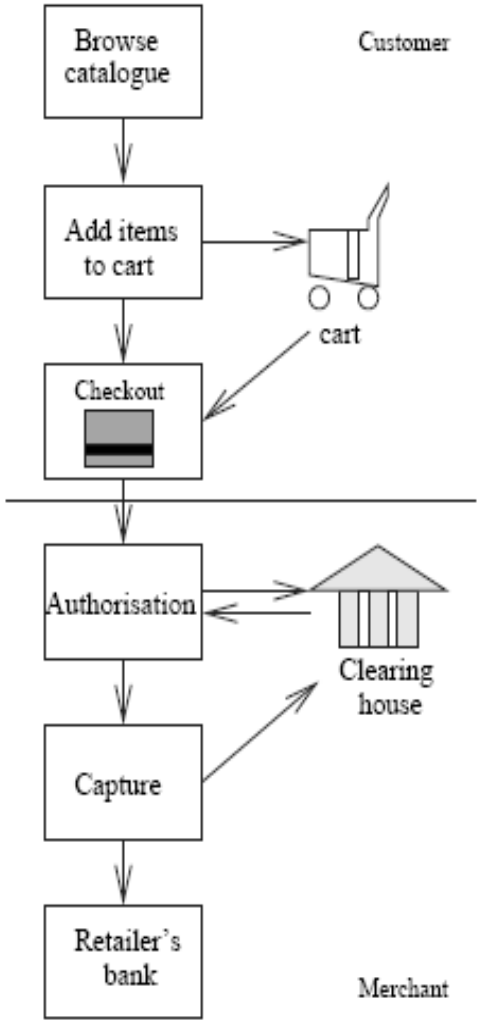
The web calculates the total price of all items that a customer requested, and prevalidates the customer information such as credit card number, credit line, expiration date. It then sends the form of credit card information out and waits for response. If the credit card information is valid, the order is processed and the related information, i.e., order number, is sent to the customer. Otherwise, the transaction is failed.

CreditCardProcessing

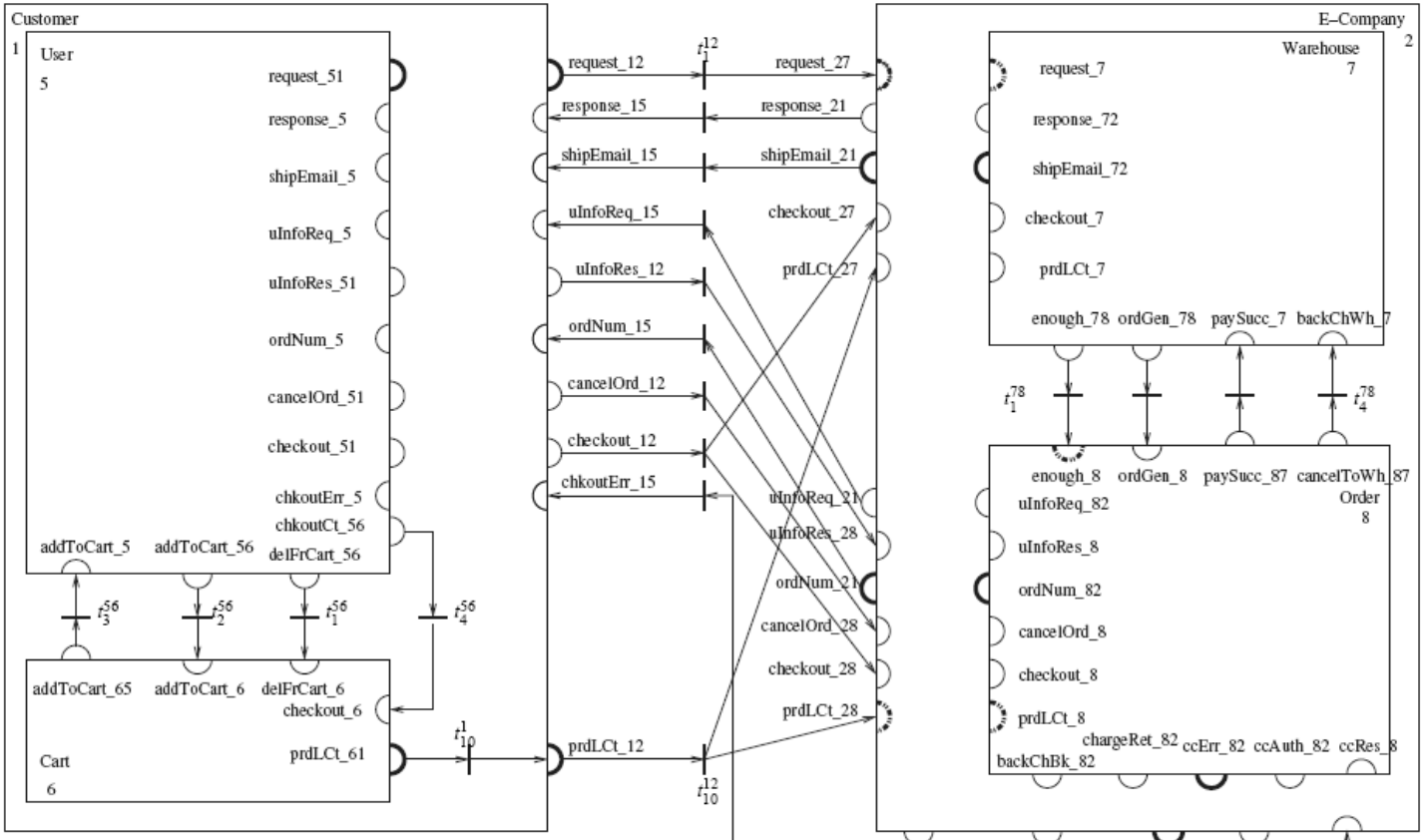
The authorisation department checks the credit card information, such as the card has not been reported stolen and there is sufficient credit on the card. Then reports to the retailer's company.

RetailerBank

The service provided here is collecting or refunding money from or back to customer.



SO-SAM Model of Online Shopping



$\Phi(request\_12) = \Phi(request\_27) =< S_S, uid, req, category >$
$\Phi(response\_15) = \Phi(response\_21) =< S_S, uid, prdName, category, price >$
$\Phi(shipEmail\_15) = \Phi(shipEmail\_21) =< S_S, uid, ship >$
$\Phi(uInfoReq\_15) = \Phi(uInfoReq\_21) =< S_S, uid, uReq >$
$\Phi(uInfoRes\_12) = \Phi(uInfoRes\_28) =< S_S, uid, name, ccNum, limit, ccExp >$
$\Phi(chkErr\_15) = \Phi(ccErr\_21) =< S_S, uid, ccErr >$
$\Phi(ordNum\_15) = \Phi(ordNum\_21) =< S_S, uid, ordNum >$
$\Phi(cancelOrd\_12) = \Phi(cancelOrd\_12) =< S_S, uid, cancel, ordNum >$
$\Phi(checkout\_12) = \Phi(checkout\_27) = \Phi(checkout\_28) =< S_S, uid, checkout >$
$\Phi(prdLCt\_12) = \Phi(prdLCt\_27) = \Phi(prdLCt\_28) =< S_S, uid, prdName, category, price >$
$\Phi(backChBk\_24) = \Phi(backChBk\_4) =< S_S, uid, ordNum, price >$
$\Phi(ccAuth\_23) = \Phi(ccAuth\_3) =< S_S, uid, name, ccNum, limit, ccExp, price >$
$\Phi(ccRes\_32) = \Phi(ccRes\_28) =< S_S, uid, ccNum, ccStatus >$
$\Phi(backCh\_43) = \Phi(backCh\_3) =< S_S, uid, ccNum, totP >$
$\Phi(charge\_34) = \Phi(chargeRet\_4) =< S_S, uid, ordNum, totP >$

General Webbed Application Modeling Properties

- They are not specific to a particular application domain
- Related to the logical organization of the whole application rather than ergonomics/design of individual pages
- Structure properties: related with the topology of the model.
- Dynamic properties: related with the behavior (state changing) of the model.
- Real-time properties: evaluate the system in terms of time response.

Structural Properties

- The services of each outgoing port in a component have same names as the incoming port of its connector.
- Each outgoing port with request message must have been responded either positively or negatively.
- Each final port must be reachable from the initial port. This property ensures that each service is not superfluous.
- If several final ports exist simultaneously, then these final states are the common states of the transaction.

Dynamic Properties

- Liveness check of web application is necessary to make sure that the service request can get response so that the transaction can continue.
- Deadlock freedom of web application means that operations of a web service do not have circular waiting.
- Examples

$$\Box((checkout\_12(< S_S, uid, \text{“}checkout\text{”} >) \wedge prdLCt\_12(< S_S, uid, pname, category, price, quantity >)) \rightarrow (\Diamond(ordNum\_15(< S_S, uid, ordNum >) \wedge shipEmail\_15(< S_S, uid, \text{“}shipEmail\text{”} >)) \vee chkoutErr\_15(< S_S, uid, \text{“}ccErr\text{”} >)))$$

Results

- Most of properties we verified are return true.
- Not all properties can be verified by runtime verification.
  - Safety properties, e.g., $\Box p$, can be verified
  - Liveness properties, e.g., $\Diamond p$, is hard to be verified
  - There are three types of results: true, false, unsure.

Discussions

- SO-SAM is an executable component-based service-oriented architecture model.
- Many interesting web service properties can be explored to analyze service composition, integration, and reusing.
- Runtime monitoring provides a new coordination of web service model validation and fills in the gap between service oriented model and implementation.
- More service properties need to be explored to satisfy service composition and integration features