Verification of SoII Architecture Using Colored Petri Nets

1Punitha Devi C., 2Dr. V. Prasanna Venkatesan, 3Diwahar S. and 4Meiappane A

1 Department of Computer Science and Engineering, Pondicherry University, Ponducherry, INDIA.
2 Department of Banking Technology, Pondicherry University, Ponducherry, INDIA.
3 Department of Computer Science and Engineering, Pondicherry University, Ponducherry, INDIA.
4 Department of Information Technology, Manakula Vinayagar Institute of Technology, Pondicherry, INDIA.

A B S T R A C T
Verification ensures that the work products of software architecture meets the requirements and follows design principles and specifications stated for it. This process is done to make sure that the product is being built in the right way. This in turn leads to correction of errors and check for accuracy at the design phase itself. This process has been adopted for verifying service oriented information integration (SoII) architecture in this paper. The architecture brings about a generic and a comprehensive solution to the needful consumers to consume it for their purpose. The layers and components have been identified and designed, whereas verification of the design is the prime objective of this paper. Among the various simulation tools and verification standards collared petrinets where found suitable and being used to verify this SoII architecture.

INTRODUCTION
Software architecture has gained significance in providing a new way to construct and maintain effectively large scale complex systems. (Bass et al, 2003) Building such complex and large scale systems are difficult and error prone. Software architecture enables one to model the system in a detailed way to enable analysis of behaviour of the system, identification of logical faults and easy to implement design pattern. (Clements et al, 2002) It is because techniques and tools that assure correctness and reliability are far behind while compared to increasing growth of size and complexity in software systems. Need arises for a proper design that leads to higher percentage of concentration in revealing errors prior to implementation. (Christinehofmeister et al, 2007) Evaluation and verification of architectures is highly necessary and to ensure correctness and accuracy in the earlier stage of development process. (B. Tekinerdogan et al, 2001) Formal automated verification tools prove to be promising in analysing the interface issues, correct behavioural and performance before implementation and cost effective by fixing problems at the design stage of the development lifecycle. Such verification methods are process algebra, petri nets, temporal logic, automata, label transition systems, real time logic etc (Kurt Jensen et al, 2007). Among these Petri nets has been identified to best suit our verification requirements of SOII architecture. (Houshang Darab et al, 2009)

2. Colored Petri Nets (CPN):
Colored Petri nets are particularly well suited for modeling and analyzing large and complex systems for several reasons: they have an intuitive graphical representation; they are executable; hierarchical models can be constructed; it is possible to model the time used by different activities in a system; and mature and well-tested tools exist for creating, simulating, and analyzing coloured Petri net models. (Kurt Jensen et al, 2007) This tool is used to check whether this model flow works correctly and there is no deadlock occurs in this model. Coloured Petri Nets (CP-nets or CPN) is a graphical oriented language for specification, verification of systems, design, and simulation. (J.Wang et al, 2011) The relationship between CP-nets and ordinary Petri Nets is similar to the relationship of high-level programming languages and assembly code. In theory, they have accurately the
same computational power. On the other hand, the high-level language offers a lot more modelling power, because they have enhanced structuring amenities, e.g., types and modules. (J. Wang et al., 2011)

Petri Net is a set of nodes and arcs. There are two types of nodes: places and transitions, which signify the state of the system and the occurrence of events, correspondingly. (Vijay Gehlot, Carmen Nigro, 2010) Most of the development of Petri Nets emerges the need of specifying synchronization situations, collective resource conflicts and asymmetric systems (alternative routes); For example important events in the illustration of a manufacturing setting. In manufacturing systems, places would represent operations (e.g., process, transportation, reparation), and transitions symbolize events, such as termination of a job processing or a machine breakdown. Arcs are directed, and connect places with transitions (or transitions with places). Tokens reside in places and represent the truth of the condition associated with such a place. The firing process induces a token’s flow among places; when a transition fires, tokens from all its input places are removed and put into the transition output places. A transition can only be fired if it has been enabled (i.e. there are sufficient tokens at its input places). (Kurt Jensen et al., 2007; J. Wang, 2007)

The advance of CP-nets has driven the desire to develop a modelling language – at the same time theoretically it is handy enough to use in practice for the various systems of the size and difficulty found in industrial projects. To achieve this, the strength of Petri nets is combined with the programming languages. (Elhillali Kerkouche et al., 2010) Petri nets provides the primitives for the depiction of the synchronization of parallel processes, while programming languages provide the primitives for the definition of data types and manipulates the data values.

3. SOII Architecture:

This proposed architecture aspires to give a top level view of information integration with SOA that could be adopted are made suitable to any information integration problem. The proposed architecture is an abstract one that can be realized as a frame work and made concrete when applied on a domain or application or problem. The architecture is described along with its components in (Punitha et.al, 2014). Banking scenario is taken for verifying this architecture. The prime objective of this architecture is to provide a comprehensive model, which is generic, flexible and can be used for all information integration process in any domain.

3.1 Design of CPN Model for SOII Architecture:

In order to better represent the SOII architecture using petri net, Object Oriented Methodology (OOM) is used. This approach is used in (Elhillali Kerkouche et al., 2010; Vinai George Biju and Santanu Kumar Rath, 2010, Bhawana Agarwal, 2012). Initially it is designed using object oriented design and it is converted into petri net model for the proposed service oriented information integration architecture. Requirements for SOII architecture is gathered, and represented using use case diagram.

3.2 Use Case Diagram of CPN model for SOII Architecture:

Figure shows the use case diagram of CPN model for the verification purpose of SOII architecture. The requirement for verification is specified along with the interaction user and the system. The user here may be application system, a web portal or device who initiates the system. The request information is provided by the user and the response information is obtained back from the system to the user.

![Fig. 1: Use case diagram of CPN Model for SOII architecture.](image-url)
The request is processed to know the associated data to be retrieved and the data. The data to be converted into information is done through a modelling process. The information is modelled suitably to be converted to service. Then the service is created and integrated with the other information service to obtain the requested information. The integrated information is formulated for the required format and given as a response to the user. The interaction of the system is clearly stated here and next the associated objects are to be modelled for it to be represented in CPN.

3.3 Class Diagram of the CPN model for SOII architecture:

The classes associated with the use case model depicted in figure are brought out here. The attributes and the operations which reflect the functionality exposed through the classes is stated in the diagram.

![Class Diagram](image)

**Fig. 2:** Class Diagram.

The class diagram displayed above brings out the complete picture of the SOII architecture that is to be modelled and verified with CPN. The intentions and the functionality of the classes are explained below.

3.4 Request Information:

**Request Handler:** The request is analysed with context to give the information that are to be retrieved with relevance to the request using the methods below

- Get context details ( ) method to retrieve the context details of the request
- Decomposes request ( ) method to split the request into a set of associated Business information
- Organiser request ( ) method to structure the various information that are to be retrieved

**Service Request Handler:** The request is processed as information service request

- Get information service request context( ) method to align the several information with service context
- Decomposes request ( ) method to split the information as information services

**Information request handler:** The information service request are to be analysed for appropriate information.

- Get information request context( ) method to align the several information services with information context
- Decomposes request ( ) method to split the information services as appropriate request information
- Organises request( ) method to structure the information with relevance to the business rules and access policies.

**Data request Analyzer:** The information request is to be analysed for appropriate data.

- Get data request context( ) method to align the several information with relevant data context
- Decomposes request ( ) method to split the information as one or more data
- Organises request( ) method to structure the data with relevance to the access policies.

**Acquisition of Data:**

**Data request modeller:**

The data pertaining to the requested information is structured to the form so as to be associated with data sources.

- Convert data request ( ) method to convert the data request into related data
- Check access rights ( ) method to check for the access policies for the data to be retrieved and accessed

**Facilitator:** The moderator or federator to facilitate data retrieval from distributed heterogeneous data sources.

- Get modeled data ( ) method to structure the data to be used in the query
- Get abstract query ( ) method to retrieve the abstract query based on the data
- Organise query ( ) method to formulate the structured query with data and conditions
- Retrieve data ( ) method to invoke the concrete query to acquire data
- Combine data ( ) method to integrate data
Database interface manager:  
Facilitates data source interactions and acts as interface between the architecture and data sources.  
- Create concrete query ( ) method to formulate the concrete query with respect to data source.  
- Execute query ( ) method to hit the data sources by executing the query  

Query Analyser: To form the abstract query for the data request.  
- Interpret request ( ) method to break the data request into allied data  
- Generate execution plan ( ) method to form the steps to get the data from various data silos  
- Create abstract query ( ) method to formulate the abstract query for each of the data  

Modelling Information:  
Information Modeller: To structure, transform and organize the data into information  
- Transform data ( ) method to add semantics and business context to convert data into information  
- Organisedata ( ) method to structure the information in the required form bundled along with its business rules.  

Modelling Service:  
Information service modeller:  
To organize the information with service rules pertaining to agreements and contracts  
- Build candidate service ( ) method to create the abstract service interface and implementation procedures  
- Get candidate service operations ( ) method to form the related operations pertaining to the information  

Creating service:  
Information Service factory:  
- Create concrete service ( ) method to make the abstract candidate service into concrete by applying the language template and technology constraints  
- Publish service ( ) method to publish the service into registry  

Information service manager: To incorporate ownership rights and reusability constraints  
- State ownership ( ) method that assigns the ownership details  
- Get access rights ( ) method that sets the access rights  
- Get service scope ( ) method to get the scope of the information service  

Strategy Selector: To uphold the processed information request  
- Get process request ( ) method to obtain the process request information  
- Verify service link ( ) method to check for all the information is retrieved and integrated.  
- Generate response ( ) method to formulate the integrated information  

Integrating service:  
Service Integrator:  
This is the integration engine which integrates all the information services  
- Integrate services ( ) method to integrated one or more information services.  
- Adherence check ( ) method to check for all the information is used for integration  

Build response:  
Response Builder:  
To formulate the response with relevance to the request and deliver the integrated information as service  
- Formulate response ( ) method to build the response with adherence to the request and delivery channel.  
- Publish response ( ) method to publish the integrated information as service in the registry.  

3.5 Collaboration diagram of CPN for SOII architecture:  
The architecture follows service principles and the style adopted to form this proposed approach is SOA. The communication between components or objects is through messages. The collaboration is only exchange of messages.  
The course of action of the CPN model starts with the user request for information as a string and this is broken down and looked up for the needed information and the associated data in the various data sources and combined to form a information which is structured and presented as service information.  
The data sources are text files that are maintained with the data, where the transitions look upon for processing. The input is a data file that contains information aligned with the context. Likewise the net could be implemented for various time for obtaining one or more information and then they are combined to produce the required integrated information.
Building a Scenario for CPN model of SOII architecture:

The CPN Model for SOII architecture has to be designed in such a way to incorporate the components of the architecture as static and dynamic elements. The rectangle boxes indicate the dynamic elements and the oval represent the static elements which just hold the value or data. The process of any scenario would follow all the components; thereby verify the functionalities of the elements. The general steps of the scenario would be as follows:

The user or system input is broken up and structured through the information request handler and service request handler:

1. The input is made into a data and the associated query is formulated.
2. The query which is string now converted to data through data sources.
3. If more number of data is involved then simultaneous calls are sent to formulate query and data retrieval.
4. The retrieved data is added with semantics and modelled information modeller.
5. The information is modelled as service in service modeller.
6. The service manger places the conditions for using the information.
7. The strategy selector holds the information and waits for all the other information to be collected by going through the above said procedure again and again.
8. The service integrator integrates the information received from the strategy selector.
9. The response builder produces the result to the user or system or application.

4.1 Formal Definition of Non Hierarchical CPN for SOII Architecture:

The main objective is to define formal definition of the non hierarchical SOII architecture CPN model by defining the attributes used in the model (Kurt Jensen and Lars M Kristensen, 2009)

Formal Definition of Non-hierarchical Coloured Petri Net:

A non-hierarchical Coloured Petri Net is a nine-tuple CPN = \text{where}

1. \text{is defined as finite set of States.}
2. \text{is defined as finite set of conversion such that .}
3. \text{is defined as set of directed lines.}
4. \text{is defined as finite set of non-empty data sets.}
5. \text{is defined as finite set of typed identifiers such that for all identifiers .}
6. \text{is defined as data set function that contains a data set to each state.}
7. \text{is a condition function that contains a condition to each conversion such that .}
8. \text{is an line expression function that assigns an line expression to each line such that .}
9. \text{is an boot function that assigns an boot expression to each place such that .}

The above given formal definition of the syntax of CPN vary from the formal definition in a slight ways. In CPN, parallel lines are not directly allowed in the definition of a CPN, and it has been improved to include a set of variables V. The change has to be included to build the formal definition concur with how the user develop a CPN model using CPN Tools. The users are allowed to declare the variables in such a way that they can emerge as open variables in the guards and line expressions.

5. Verifying properties of the SOII architecture CPN Model:

The required properties of SOII architecture CPN model are identified and formal techniques are used to verify them by using state space analysis. Three key properties are defined for SOII architecture to ensure the
request are aligned with relevance to context and the needed information are obtained and integrated as per the request, to complete the process without any deadlock. The properties chosen to verify the SOII architecture are

- **P0 (symmetry): different scenario and its I/O are unordered:**
  This property can be verified by different possible information integration scenarios and check whether the automation of the process works correctly.

- **P1 (no deadlock): the model may always process different incoming context requests:**
  This property can be verified by checking whether the CPN works for the number of information required and integrated without indefinite looping and as well as by observing the liveness property which give the deadlock marking as result of the model.

- **P2 (fairness): every place and transit of CPN model are detected and processed:**
  This property can be verified by observing the fairness property of state space analysis report. It should contain no infinity sequence which mean that the model work completely as well as correctly by processing the functions exposed in transitions, placing the outcome in places.

P0, P1, P2 are difficult to verify, despite it can be achieved through the execution of number of test cases and also to examine all possible execution orders.

6. Case Study Implementation of SOII Architecture:

The case study chosen is a situation for debit card information integration. This scenario is specific to enable a transaction in A Bank with the help of the B Bank Debit card.

This is one scenario which comes in handy when a person is not able to use his A bank card due to some reason but needs to access the account with the help of B card for a particular transaction. Hence here customer information is retrieved from B Bank and made use to validate the customer in Bank A, and issue a OTP to the customer for enabling this transaction.

For this scenario the bank A is assumed to be Indian Bank
Bank B is assumed to be ICICI Bank and
Third party OTP service provider
Hence N= 2 and initiated by Indian bank. The working flow of the complete process is portrayed explicitly in the CPN net figure, which is derived from the general SOII architecture CPN Model.

Figure

The CPN model of the SOII architecture for the debit cards information integration workflow is shown in CPN net figure. The CPN can be divided into three parts: INDIAN Bank, ICICI Bank, and OTP Generation (CPN net figure).

- The process starts from Indian bank ATM, customer request placed to request handler in Integration Layer as integrated information as a service (IIaaS). If the requested service is available, it will pick from global registry and gives it back immediately. If the requested service is not available then it passes the request to request handler in Integration layer of ICICI bank for getting the Customer Information Record (CIR).

- When ICICI bank receives the request it send to request handler in Service Layer and it is placed in information request in Information Layer. Information request checks its logs whether this information is already available or not. If it not available it hands it to Information Modeller.

- Information modeller will model the request by adding domain knowledge and semantic ontology. This modelled information is handed to Data Request Analyser and analyser will analyse the request and hands over to modeller to model the data request. This request is handed over to facilitator.
Now facilitator, sends this request to query analyser, this will form the query based on the request. This query can be a specific query or a broken query. This query is given back to facilitator.

Facilitator places it in database interface manager to fetch the exact data from the federated database or virtualised data. This data is passed to information modeller through facilitator.

Information modeller receives it and adds business rules and policy to that information. This is placed in service modeller. Service modeller will model the information into service based on the service rules of the vendor. This is passed as a candidate service to service factory.

Once, service factory receives this candidate service it is changes to service based on the service template available and given to service manager.

Service manager will place the service in service repository and gives it to strategy selector. Where service is selected based on some strategy. Then it goes to integrator and goes out as an IaaS to Request handler of Indian bank. Now, CIR is given to INDIAN bank.

Once INDIAN bank receives the CIR, it checks for the customer existence in the database based on the customer’s unique id. It passes the request to information layer and information modeller will model the request by adding semantics and domain knowledge to it and gives it to request modeller.

Request modeller will model the request in such a way to get a query to find the specific customer from the database and gives it to query analyser through facilitator.

Query analyser will form a query based on the request and gives it to database interface manager. Database will revert back with the required information and information modeller will add business policy and rules to it and gives it to service modeller.

Service modeller delivers a candidate service based on the service rules of the vendor. Once it send candidate service to service factory it initiates the OTP service, which means the information of the customer got from ICICI matches with INDIAN bank.

Service factory builds a service based on the service template and gives it to service manager. It will hand over to strategy selector. Strategy selector will select the strategy to integrate all three services (i.e. CIR service from ICICI, Service from INDIAN Bank and OTP service). Once OTP service reaches the Indian bank all the services are integrated in service integrator and given it as Integrated Information as a Service (IaaS).

7. Verification Results:

The Fairness Properties in petrinet gives No infinite occurrence sequences. Hence there is no dead lock in the model.

Liveness Properties

<table>
<thead>
<tr>
<th>Dead Transition Instances</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Live Transition Instances</td>
<td>None</td>
</tr>
</tbody>
</table>

8. Conclusion:

The SOII architecture that has been designed needs to verified and evaluated to prove its completeness and working state. Petri nets are one formal method that has a graphical representation and can also be used as a simulator to verify a model or architecture. In order to verify the design of the architecture, first the object oriented modelling is used to model the requirements and functionalities. Hence the use case, class and collaboration diagrams are formulated in order to make the architecture verifiable. The CPN model is designed and verified for the proposed SOII architecture for a banking scenario.

REFERENCES


