Specification and Verification of Context-dependent Services

Naseem Ibrahim, Vangalur Alagar, and Mubarak Mohammad

Department of Computer Science & Software Engineering
Concordia University, Montreal, Canada
{n_ibrah,alagar,ms_moham}@cse.concordia.ca

Agenda

- Background
- Motivation
- Research Goal
- Formal Model of Service
- Service Composition
- Verification Approach
- Related Work
- Conclusion

Background

- *Service Oriented Computing* (SOC) aims to increase the efficiency, agility and productivity of an enterprise by utilizing services as the primary means through which functionality is represented.

Background

- A *service* is an autonomous, platform-independent software program that can be *described*, *published*, *discovered*, and *composed*.

- A *service contract* establishes the terms of engagement with the service, provides technical constraints and requirements, and provides semantic information the service owner wishes to make public.

Background

- A composite service is defined as a coordinated aggregate of services.
- Service composition is a method to achieve composite services. In principle, service composition is performed to produce new complex services, each with a new functionality.
- Service composition can be divided into
  - Static composition, and
  - Dynamic composition.

Background

- *Context* is defined as the information used to characterize the situation of an entity. This entity can be a person, a place, an object or an event.
- An example is the contextual information of this presentation which includes:
  - Location: Reykjavik University.
  - Speaker: Naseem Ibrahim
  - Date: June 9, 2011
  - Time: 11:10 am

Motivation

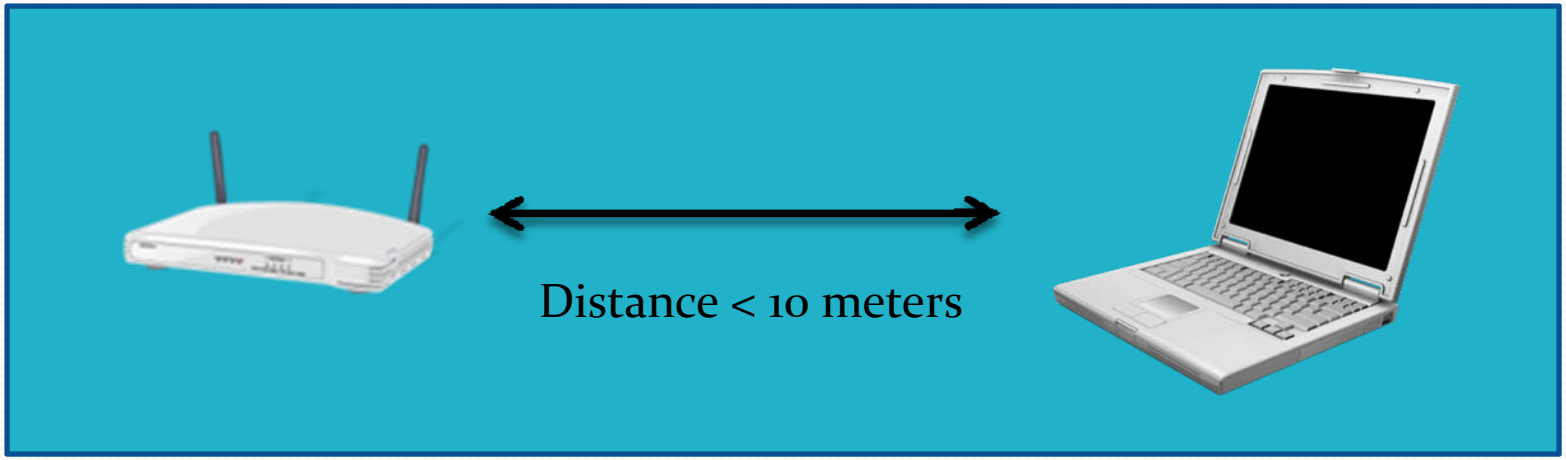
- There is a service oriented system in which:
  - There are multiple service providers that provide the same service.
  - Each service provider provides the service with a set of quality constraints.
- The service requester should be able to get the service that best matches its quality requirements throw the system.

Motivation

- To obtain the best service match, functional, non-functional and contextual information should all be considered during the service publication, discovery and selection process.

Motivation

- A service cannot guarantee its contract in all situations. The relationship between the service contract and the related contextual information on which the service can guarantee its contract plays a crucial role in service provision.



Motivation

- Formal methods are necessary for the specification of the services, their contracts, the related contextual information, and their composition.

- Without a formal basis it is impossible to verify the correctness conditions for service compositions and the satisfaction of contractual obligations in service provisions.

Motivation

- A great amount of research has been done in the area of service specification, verification, publication, discovery, selection and composition.

- The issue with current approaches is that they do not consider the relationship between the service contract and the related contextual information.

Research Goal

- Our research aims to define a formal framework for services with context-dependent contracts that enable:
  - the formal specification of services,
  - the publication of services,
  - the discovery of published services,
  - the selection and composition of services, and
  - the verification of the property that provided service satisfies requested service and the verification of the composition result.

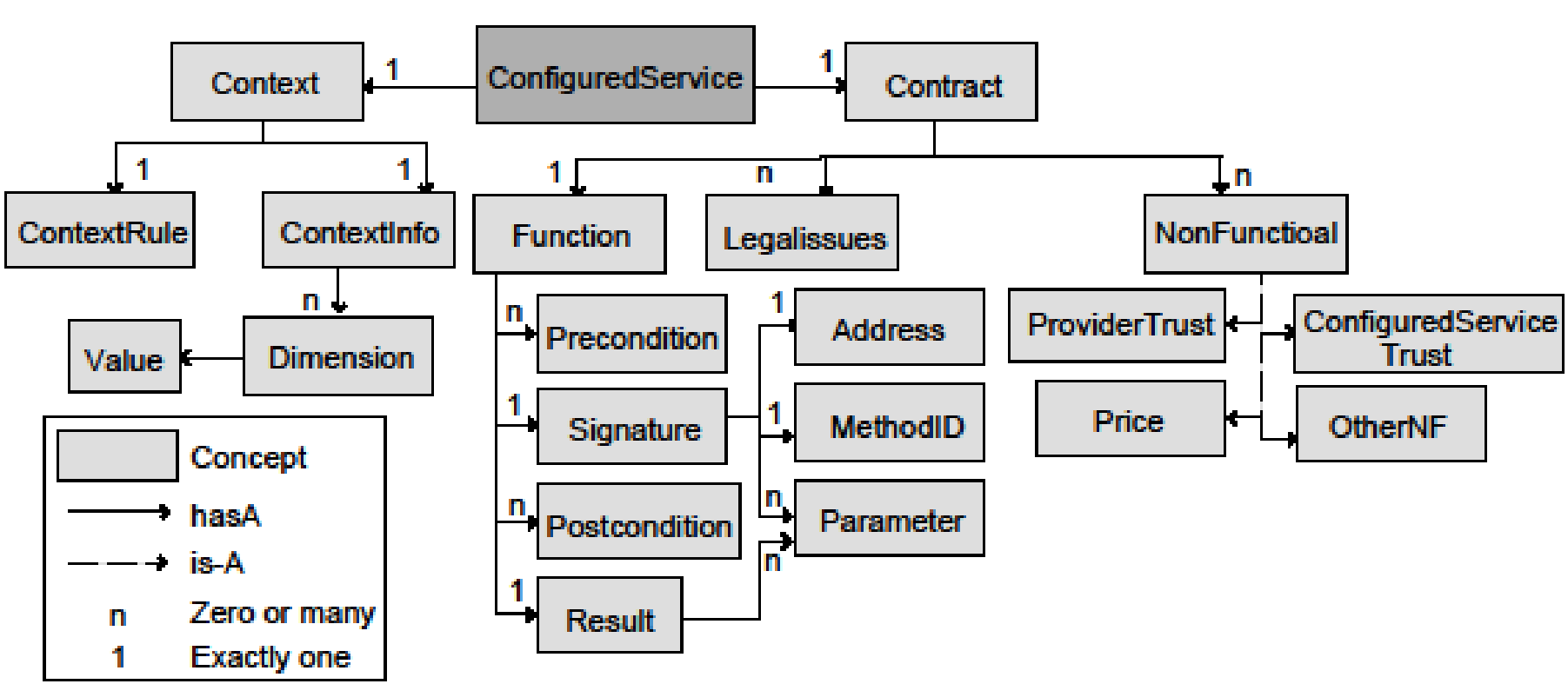
Paper Contribution

1. Defining a *formal model* for the specification of services with context-dependent contracts.

2. Defining *a compositional theory* for the composition of services with context-dependent contracts.

3. Defining *a formal verification approach* for the verification of the composition result.

Formal Model Of Service



ConfiguredService-Contract

- Function: which includes
  - The signature which includes:
    - identifier,
    - address, and
    - parameters.
  - The result,
  - Preconditions, and
  - Postconditions

ConfiguredService-Contract

- Nonfunctional properties: which includes:
  - Price,
  - ConfiguredService Trust: which includes:
    - Safety,
    - Security,
    - Availability, and
    - Reliability .
  - Provider Trust.

ConfiguredService-Contract

- Legal Issues: The legal rules that constrains a contract includes:
  - Business rules, such as refund conditions, interest and administrative charges, and payment rules
  - Trade laws, such as the service requesters rights, privacy laws, and censor rules.

ConfiguredService-Context

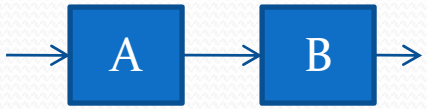
- The ConfiguredService context defines the contextual information of the ConfiguredService.
- The context rules define the contextual information related to the Service Requester that should be true for the Service Provider to guarantee the contract associated with the ConfiguredService.

Service Composition

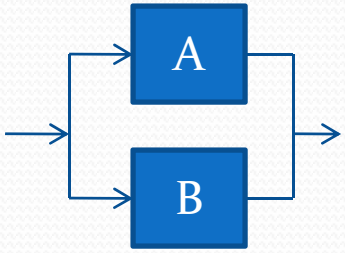
- Composing services includes defining the execution logic of the participating ConfiguredServices, which includes:
  - Defining the composition constructs.
  - Defining the ConfiguredService resulted from applying a composition construct on ConfiguredServices.

Composition Constructs

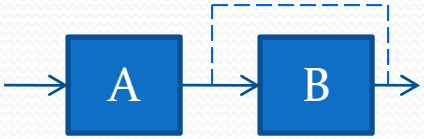
- Sequential composition construct A>>B.



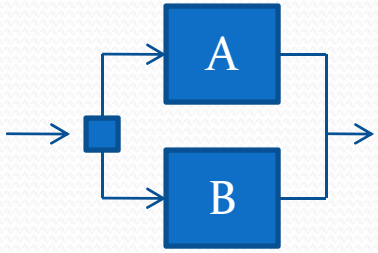
- Parallel composition construct A||B.



- Priority construct A≺ B

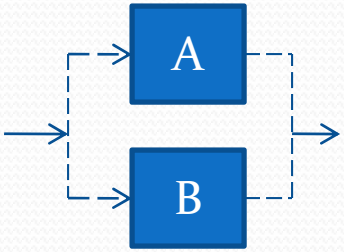


- Composition with no order A◊B

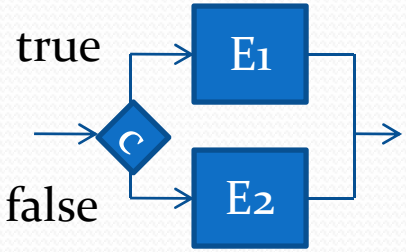


Composition Constructs

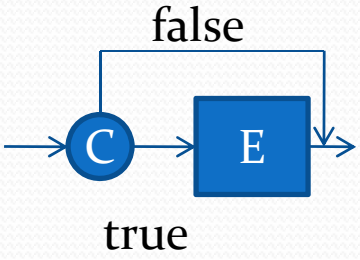
- Nondeterministic choice construct A ≀ B



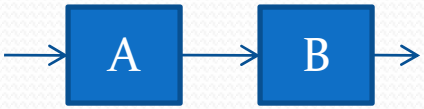
- Conditional choice construct E1 $\triangleright_c$ E2



- Iteration construct (while) $E_{\circ c}$

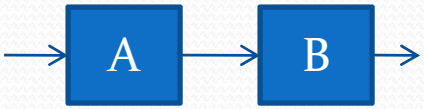


Sequential composition construct A>>B



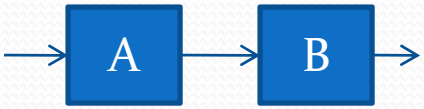
- Contract:
  - Function:
    - Preconditions:
      - If B requires more constraints: defined as the union of preconditions of A, and preconditions of B that are postconditions of A.
      - If B does not require more constraints: defined as the preconditions of A.
    - Postconditions:
      - If A is observable: defined as the union of the postconditions of A and B.
      - If A is not observable: defined as the postconditions of B.

Sequential composition construct A>>B



- Contract:
  - Nonfunctional:
    - Safety (timeliness): defined as the sum of A and B.
    - Safety (data): defined as the union of A and B.
    - Security: defined as the union of A and B.
    - Availability: defined as the sum of A and B.
    - Reliability: defined as the minimum of A and B.

Sequential Composition Construct A>>B



- Legal: defined as the union of A and B.
- Parameters:
  - Input: defined as the union of the input parameters of A, and input parameters of B that are not output parameters of A
  - Output: defined as the union of the output parameters of A and B.
- Attributes: defined as the union of A and B

Formal Verification

- It is essential to verify that the functional behavior of the service composition meets the requirements of the service requesters while taking into consideration the nonfunctional, legal and contextual conditions.
- Instead of defining a new verification tool to verify the service composition a transformation approach is followed.
- A formally defined service composition can be automatically transformed into a model understood by the model checking tool UPPAAL.

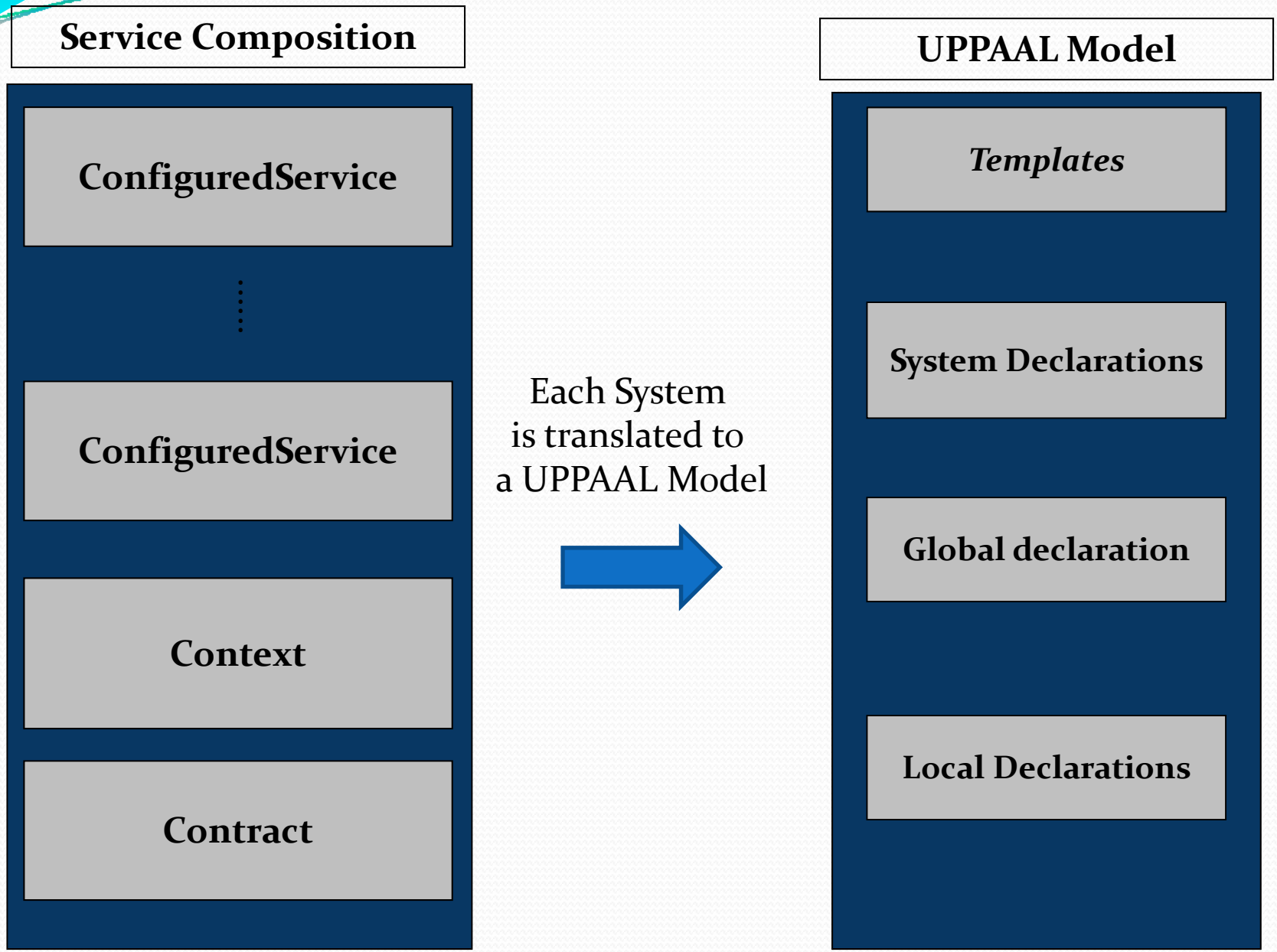
UPPAAL

- UPPAAL is a tool for the modeling, validation and verification of real-time systems represented as a collection of synchronized extended timed-automata.
- It is well established, and has been around for 13 years.
- It has been used for the formal verification of applications in different domains.

Transformation Rules

- The transformation rules are divided into:
  - Transformation rules for generating the *global declaration.*
  - Transformation rules for *ConfiguredServices*.
  - Transformation rules for *compositions flow*.

Transformation Rules



Transformation Rules

ConfiguredService

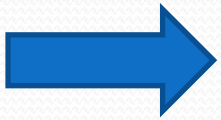
Contract

1. **Function**
2. **Nonfunctional**
3. **Legal**

Context

1.Context Info
2.Context Rules

Each *ConfiguredService* is Translated to one template



UPPAAL template

Locations (L)

Actions (A)

Edges (E)
Expressions:
1.Select
2.Guard
3.Sync
4.Update

Invariants (I)

Clocks (K)

Transformation Rules- ConfiguredService

- *ConfiguredService* transformation will include defining a single UPPAAL template for each *ConfiguredService*.
- The template will include two locations and two transitions between them.
- Non-functional constraints, legal rules and context rules are added as guard statements on the transitions.

Transformation Rules- Composition flow

- The composition flow will be transformed into a single UPPAAL template that represents the composition flow. This template is synchronised with the templates corresponding to the individual *ConfiguredServices*.

- The verification is performed on the set of all templates.

Formal Verification

- We need to verify:
  - The behavior of the composition is correct with respect to functionality.
  - The behavior of the composition is correct with respect to nonfunctional properties.
  - The context rules are not contradictory, and met for each *ConfiguredService*.
  - The legal rules are not contradictory, and met for each *ConfiguredService*.

Related Work

- There are many approaches for the formal specification of services and their compositions.
- All of them formally model the service functional behavior.
- All of them model the composition of the functionality.
- Only view model the nonfunctional properties as part of the service formal definition.

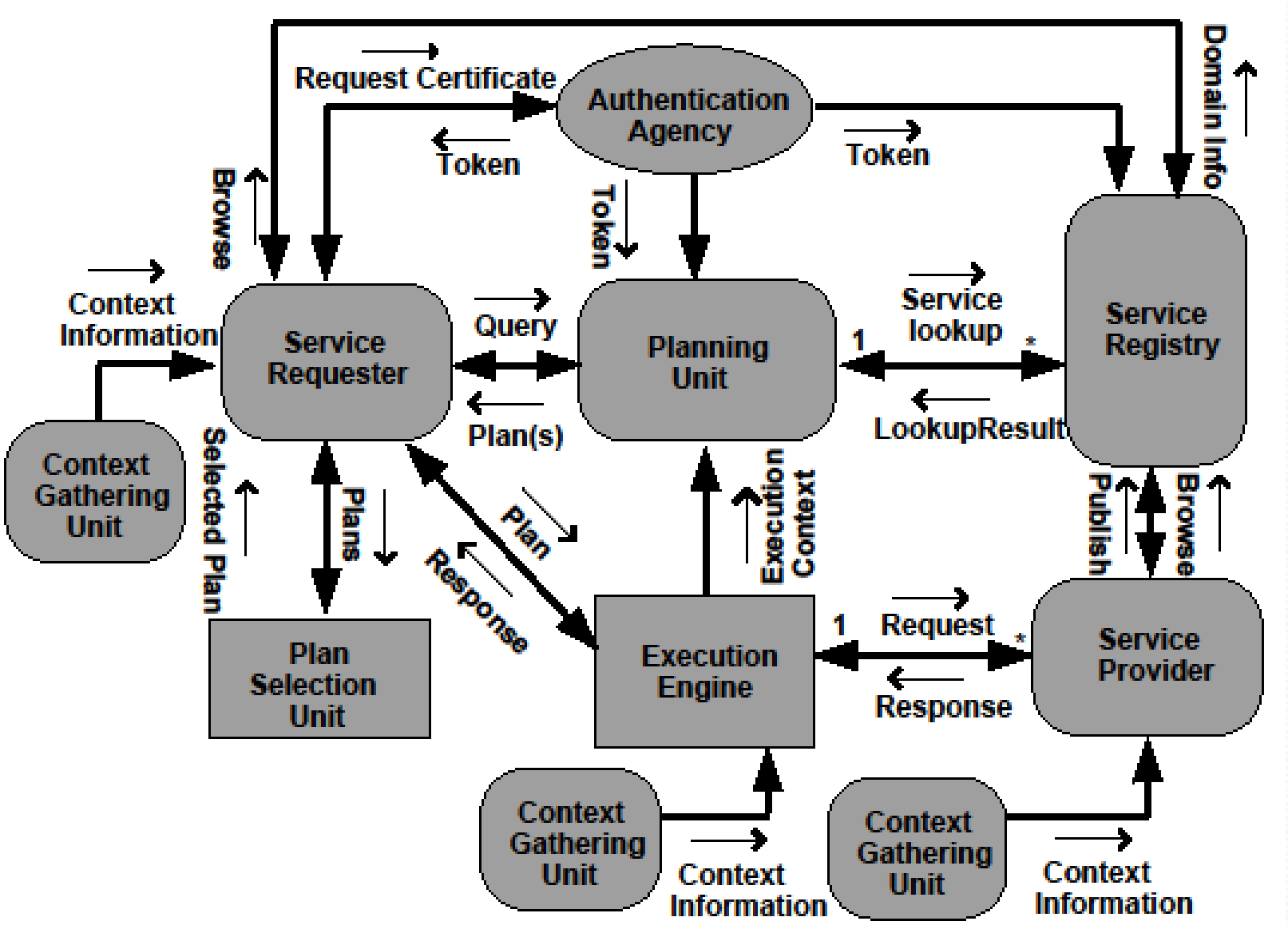
Related Work

- To our knowledge, our approach is the only approach that consider all the previous properties and in addition it:
  - Includes the context as a part of the service definition and considers the relationship between the context and the contract.
  - Includes the legal rules as a main element of the service contract.
  - Defines the composition of all the element of the service including functionality, nonfunctional properties, legal rules and contextual information.

Conclusion

- We have proposed a *formal model* for the *specification*, and *composition* of services with *context dependent contracts*.
- We have also presented a *formal verification approach* using the model checking tool UPPAAL.
- The work presented in this paper is part of *the formal framework for the provision of context-dependent services (FrSeC).*

Conclusion- FrSeC Architecture



FrSeC Characteristics

1. Support for dynamic selection.
2. Support for dynamic planning.
3. Specification of nonfunctional requirements.
4. Support semantic information.
5. Support for contextual information.
6. Support for compensation provision.
7. Replanning support.
8. Use of formal methods.
9. Trusted transactions.

Future Work

- We plan to:
  - Define a dynamic composition approach that automates the service composition process at execution-time.
  - Investigate dynamic reconfiguration issues arising out of defaults and dynamic compositions of services.
  - Develop a set of tools that automate the composition and verification process.

Thank You!