

# Expectations for Assessment of Service Quality

Ioan Alfred Letia and Anca Marginean

Technical University of Cluj-Napoca  
Department of Computer Science

WoSS - SYNASC 2010, Timisoara

# Outline

- 1 Grounding
  - Dolce ontology
- 2 Expectations.Support.Enactment
  - Expectation towards support
  - Expectation towards enactment
  - Important properties
- 3 Expectations in Service Enactment
  - Static phase
  - Dynamic phase
- 4 Conclusions

## Problem to Solve

- Quality of a service - difference between client expectations and perceptions
- Requirements:
  - to formalize what is expected
  - to relate what is expected to what is offered
  - to monitor what is perceived
- Solution: EXPECTATIONS based on Dolce ontology

## Problem to Solve

- Quality of a service - difference between client expectations and perceptions
- Requirements:
  - to formalize what is expected
  - to relate what is expected to what is offered
  - to monitor what is perceived
- Solution: **EXPECTATIONS** based on Dolce ontology

# Outline

- 1 Grounding
  - Dolce ontology
- 2 Expectations.Support.Enactment
  - Expectation towards support
  - Expectation towards enactment
  - Important properties
- 3 Expectations in Service Enactment
  - Static phase
  - Dynamic phase
- 4 Conclusions

# Modal Description

- Collective entities: collection of services, regulation community, intentional collective, and normative intentional collective
- Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE)
- Descriptions and Situations (DnS)

# Descriptions and Situations for Expectations

## Modal Descriptions and Satisfying Situations

$ModalDescription(x) \rightarrow Description(x)$

$ModalDescription(x) \rightarrow \exists r, t. (Role(r) \wedge \forall ag, t. Plays(r, ag, t) \rightarrow Agent(ag))$   
 $\wedge Course(t) \wedge Defines(x, r) \wedge Uses(x, t) \wedge AttitudeTowards(r, t)$

$ModalSatisSituation(s) \rightarrow Situation(s)$

$ModalSatisSituation(s) \rightarrow \exists d. ModalDescription(d) \wedge Satisfies(s, d) \rightarrow \exists ag, r, t, ac.$   
 $Agent(ag) \wedge Role(r) \wedge Defines(d, r) \wedge SettingFor(s, ag) \wedge Plays(ag, r) \wedge Course(t)$   
 $\wedge Action(ac) \wedge SettingFor(s, ac) \wedge Uses(d, t) \wedge Classifies(t, ac) \wedge Participant(ag, ac)$

# Outline

- 1 Grounding
  - Dolce ontology
- 2 Expectations.Support.Enactment
  - Expectation towards support
  - Expectation towards enactment
  - Important properties
- 3 Expectations in Service Enactment
  - Static phase
  - Dynamic phase
- 4 Conclusions

# Expectations.Support.Enactment

## Support and Enactment

*Support*  $\sqsubset$  *ModalDescription*

*Enactment*  $\sqsubset$  *Modaldescription*

## Social Extention of Modal Description

$SocialModalDescription(x) \rightarrow \exists r, d. Role(r) \wedge ModalDescription(d)$   
 $\wedge Defines(x, r) \wedge Uses(x, d) \wedge SocialAttit(r, d)$   
 $ExpectationTowards(r, d) \rightarrow SocialAttit(r, d)$

## Expectations

*Exp* *Ag* *ModalDescription* *Ag'* *Action*

# Expectations.Support.Enactment

## Support and Enactment

*Support*  $\sqsubset$  *ModalDescription*

*Enactment*  $\sqsubset$  *Modaldescription*

## Social Extention of Modal Description

*SocialModalDescription*( $x$ )  $\rightarrow \exists r, d. \text{Role}(r) \wedge \text{ModalDescription}(d)$   
 $\wedge \text{Defines}(x, r) \wedge \text{Uses}(x, d) \wedge \text{SocialAttit}(r, d)$   
*ExpectationTowards*( $r, d$ )  $\rightarrow \text{SocialAttit}(r, d)$

## Expectations

*Exp* *Ag* *ModalDescription* *Ag'* *Action*

# Expectations.Support.Enactment

## Support and Enactment

*Support*  $\sqsubset$  *ModalDescription*

*Enactment*  $\sqsubset$  *Modaldescription*

## Social Extention of Modal Description

$SocialModalDescription(x) \rightarrow \exists r, d. Role(r) \wedge ModalDescription(d)$   
 $\wedge Defines(x, r) \wedge Uses(x, d) \wedge SocialAttit(r, d)$   
 $ExpectationTowards(r, d) \rightarrow SocialAttit(r, d)$

## Expectations

*Exp*   *Ag*   *ModalDescription*   *Ag'*   *Action*

# Abstract Expectations

## Expectations

*Exp Ag Support Action*

- they are assumed by the customer, independent of the concrete service
- e.g. a customer expects to be noticed of new offers and changes (Exp client Support upToDate)

*ExpectationSatisfyingSituation  
and (hasSetting some  
(situation  
and (satisfies some support)  
and (hasSetting value upToDate)))*

# Concrete Expectations

## Expectations

*Exp*   *Ag<sub>1</sub>*   *ModalDescription*   *Ag<sub>2</sub>*   *Action*  
(*Message*   *MessageName*)  
*Condition*   *ModalDescription<sub>2</sub>*

- Structure
  - a description of the expected enactment, including the agent responsible for the enactment and the involved action
  - the observable event based on what the owner of the expectation can conclude on the enactment state
  - condition that restricts the situations to where the customer can expect the enactment to take place
- they are built on reading the service description

# Concrete Expectations

## Expectations

*Exp*   *Ag*<sub>1</sub>   *ModalDescription*   *Ag*<sub>2</sub>   *Action*  
(*Message*   *MessageName*)  
*Condition*   *ModalDescription*<sub>2</sub>

- Structure
  - a description of the expected enactment, including the agent responsible for the enactment and the involved action
  - the observable event based on what the owner of the expectation can conclude on the enactment state
  - condition that restricts the situations to where the customer can expect the enactment to take place
- they are built on reading the service description

# Concrete Expectations

## Expectations

*Exp*   *Ag<sub>1</sub>*   *ModalDescription*   *Ag<sub>2</sub>*   *Action*  
*(Message   MessageName)*  
*Condition*   *ModalDescription<sub>2</sub>*

- Structure
  - a description of the expected enactment, including the agent responsible for the enactment and the involved action
  - the observable event based on what the owner of the expectation can conclude on the enactment state
  - condition that restricts the situations to where the customer can expect the enactment to take place
- they are built on reading the service description

# Concrete Expectations

## Expectations

*Exp*   *Ag<sub>1</sub>*   *ModalDescription*   *Ag<sub>2</sub>*   *Action*  
(*Message*   *MessageName*)  
*Condition*   *ModalDescription<sub>2</sub>*

- Structure
  - a description of the expected enactment, including the agent responsible for the enactment and the involved action
  - the observable event based on what the owner of the expectation can conclude on the enactment state
  - condition that restricts the situations to where the customer can expect the enactment to take place
- they are built on reading the service description

## Relevant Properties

- Satisfies - between a situation and a description
  - a situation satisfying an expectation,
  - a situation satisfying a Support or an Enactment,
  - a situation satisfying an Expectation towards Enactment
- realizes - between a situation satisfying a Support and a situation satisfying an Enactment
- isCompatible - between two situations

# Outline

- 1 Grounding
  - Dolce ontology
- 2 Expectations.Support.Enactment
  - Expectation towards support
  - Expectation towards enactment
  - Important properties
- 3 **Expectations in Service Enactment**
  - Static phase
  - Dynamic phase
- 4 Conclusions

# Expectations in Service Enactment - Static Phase

## 1. Creation of expectations

- abstract expectations: stated by the customer
- concrete expectations: created on reading service description
  - interpretation of input / output declaration for atomic processes.
  - interpretation of conditions in process description - conditioned expectations

Results in:

- 1 expectations (1) towards an enactment of the advertised operation by the service provider, (2) conditioned by an enactment of service call by the customer, and (3) observable through receiving the response - situations before receiving the service response
- 2 expectations (1) towards customer believing the result, (2) conditioned by the enactment of the advertised operation by the provider - situations after receiving the service response

# Expectations in Service Enactment - Static Phase

## 1. Creation of expectations

- abstract expectations: stated by the customer
- concrete expectations: created on reading service description
  - interpretation of input / output declaration for atomic processes.
  - interpretation of conditions in process description - conditioned expectations

Results in:

- 1 expectations (1) towards an enactment of the advertised operation by the service provider, (2) conditioned by an enactment of service call by the customer, and (3) observable through receiving the response - situations before receiving the service response
- 2 expectations (1) towards customer believing the result, (2) conditioned by the enactment of the advertised operation by the provider - situations after receiving the service response

# Expectations in Service Enactment - Static Phase

## 1. Creation of expectations

- abstract expectations: stated by the customer
- concrete expectations: created on reading service description
  - interpretation of input / output declaration for atomic processes.
  - interpretation of conditions in process description - conditioned expectations

Results in:

- ① expectations (1) towards an enactment of the advertised operation by the service provider, (2) conditioned by an enactment of service call by the customer, and (3) observable through receiving the response - situations before receiving the service response
- ② expectations (1) towards customer believing the result, (2) conditioned by the enactment of the advertised operation by the provider - situations after receiving the service response

# Expectations in Service Enactment - Static Phase

## 1. Creation of expectations

- abstract expectations: stated by the customer
- concrete expectations: created on reading service description
  - interpretation of input / output declaration for atomic processes.
  - interpretation of conditions in process description - conditioned expectations

Results in:

- 1 expectations (1) towards an enactment of the advertised operation by the service provider, (2) conditioned by an enactment of service call by the customer, and (3) observable through receiving the response - **situations before receiving the service response**
- 2 expectations (1) towards customer believing the result, (2) conditioned by the enactment of the advertised operation by the provider - **situations after receiving the service response**

## Expectations in Service Enactment - Static Phase

2. Comparison between abstract and concrete expectations - based on *realizes* property
  - refrain from service enactment - low quality
  - dialog with the service provider for further information

## Expectations in Service Enactment - Dynamic Phase

- monitoring sent/received messages
  - message sent → the corresponding expectations towards enactment become valid
  - message received → the corresponding expectations towards belief become valid
- monitor confirmed/disregarded active expectations

## Algorithm - Static Phase

```

1: assert abstract expectations
2: create concrete expectations on reading OWL-s files
3: for all  $ea : Exp_{abs}$  do
4:   if  $\exists ec \in Exp_{conc} s.t. realizes(ec, ea)$  then
5:     Good_Quality
6:   else
7:     Infringement_State
8:     if wants_detail then
9:       ask the service provider for more details
10:      add concrete expectation
11:      GOTO 3
12:     end if
13:   end if
14: end for
    
```

## Algorithm - Dynamic Phase

**Require:** *Good\_Quality*

- 1: call the service
- 2: add a situation satisfying an *Enactment* done by the client towards *calling the action  $\alpha$*  of the service: *Enactment client ServiceCall*
- 3: *realisation\_check* { *Exp client Enactment serv\_provider* becomes active }
- 4: **if** *received\_message* **then**
- 5:   add a situation satisfying the *Enactment* towards *action  $\alpha$*  done by the provider *Enactment provider  $\alpha$*
- 6:   update the client situation (in terms of beliefs) according to the service's response { *Exp client Belief  $\alpha_{result}$*  becomes active }
- 7:   *realisation\_check*
- 8: **end if**

## Algorithm - Monitoring Expectation Confirmation

```
1: for all  $exp : Exp_{abs} \setminus Exp_{abs}^{Realized} \cup Exp_{conc} \cup Exp_{belief}$  do  
2:   if ( $realizes(Current\_Situation, exp)$ ) then  
3:      $Good\_Quality$   
4:   else  
5:      $Infringement\_State$   
6:   end if  
7: end for
```

# An example

- A customer needs to make a subscription to a mobile company.

<b>Static phase</b>	
<i>Customer requirements</i>	
Exp customer Support upToDate	Support
Exp customer Support customize	
<i>on reading service description OWL-s/WSDL</i>	
(E1) Exp customer Enactment mobComp bestDescription (Message Output) Condition Enactment customer bestCall	Enactment
(E2) Exp customer Believes bestDescriptionResult Condition Enactment mobComp bestDescription	Belief
<b>Dynamic phase</b>	
<i>on sending the SOAP message</i>	
Situation Enactment customer bestCall	Non-exp
(E1) becomes active	*
<i>on receiving the SOAP message</i>	
Situation Enactment mobComp bestDescription	Non exp
(E2) becomes active	*
<i>on realisation check</i>	
Infringement of Exp customer Support upToDate	Infr

# Outline

- 1 Grounding
  - Dolce ontology
- 2 Expectations.Support.Enactment
  - Expectation towards support
  - Expectation towards enactment
  - Important properties
- 3 Expectations in Service Enactment
  - Static phase
  - Dynamic phase
- 4 Conclusions

# Conclusions

- Captured the meaning of quality in terms of Dolce based expectations
- An algorithm for monitoring quality assessment
- Semantic Interpretation of observations through Description and Situation ontology design pattern

Thanks for your attention!

Thanks for your attention!  
Questions?