
# Architectural Support for Context-Aware Applications: From Context-Models to Services Platforms

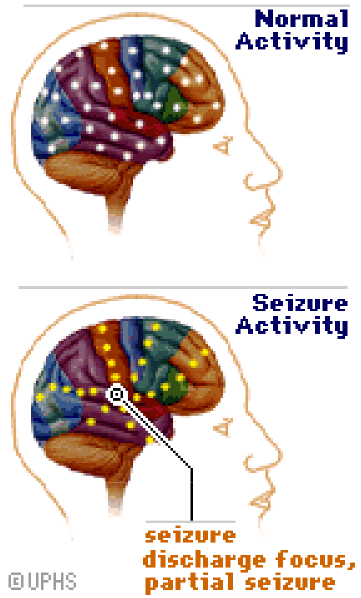
10/05/2012

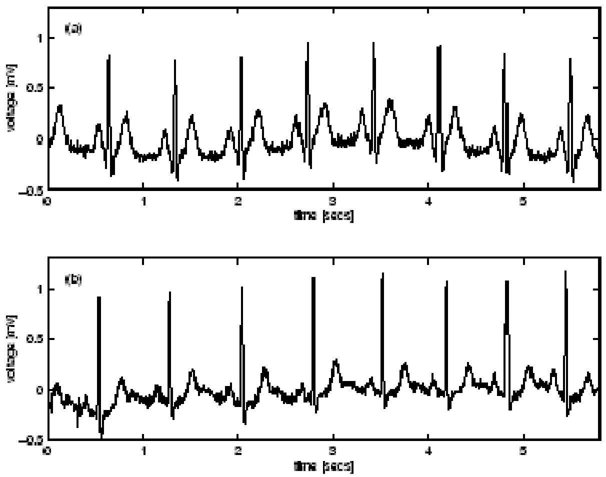
Patrícia Dockhorn Costa

pdcosta@inf.ufes.br

www.inf.ufes.br/~pdcosta

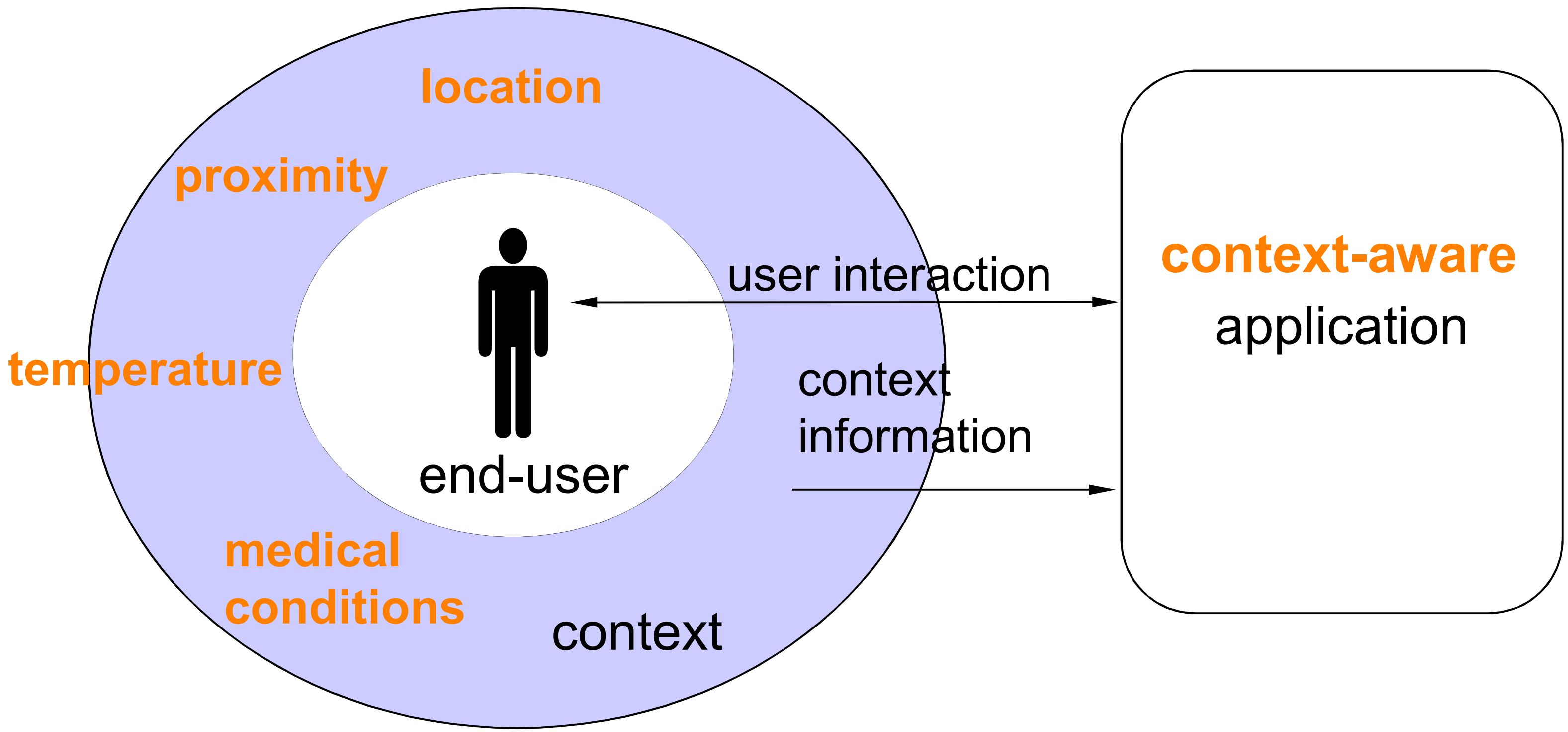
# Healthcare Scenario




- Mr. Janssen suffers from epilepsy
  - limitations in lifestyle
  - need of constant supervision
- Healthcare application
  - detects upcoming seizures
  - informs caregivers
  - increases quality of life

# Context-Aware Application

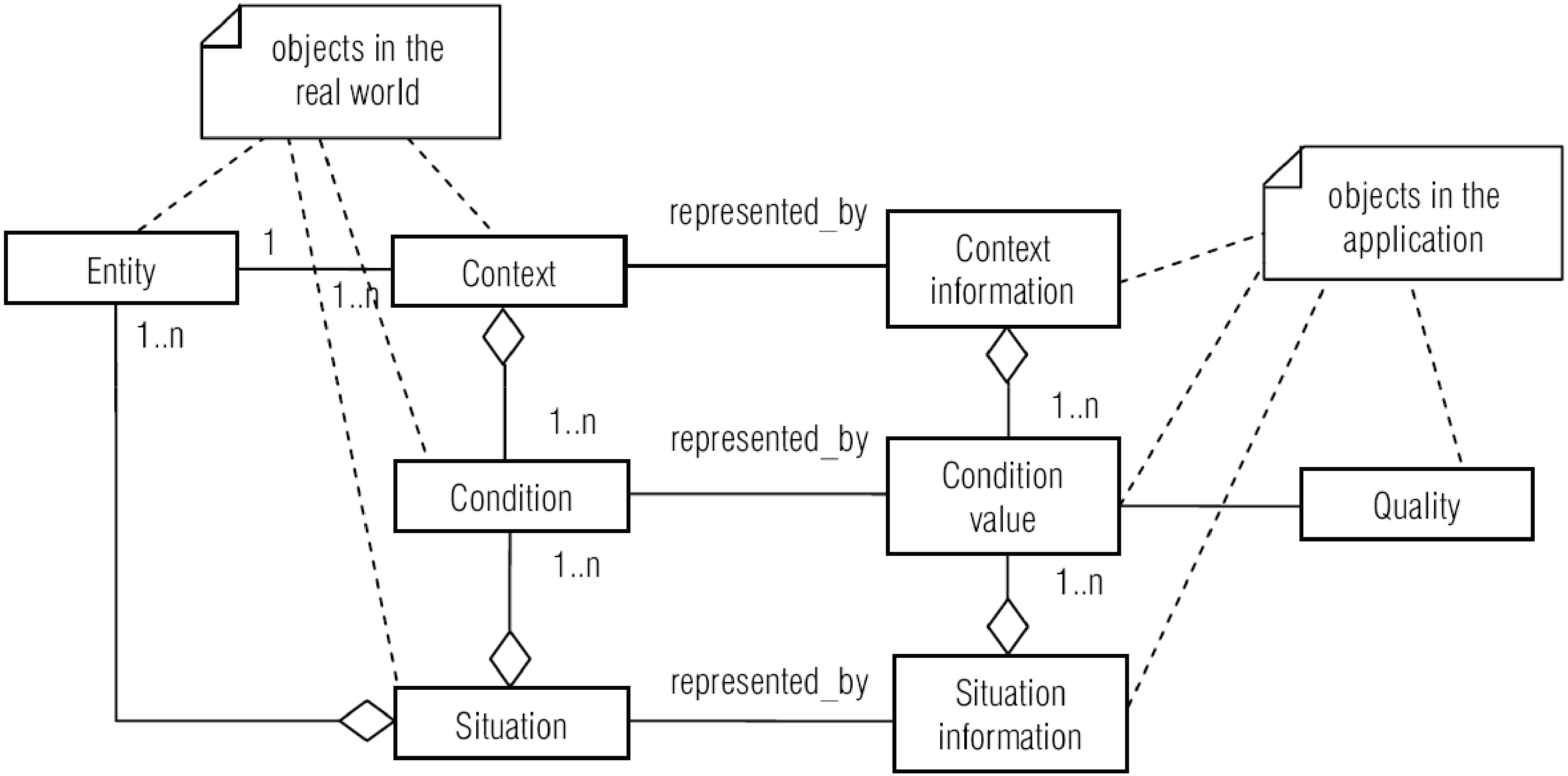



- Context
  - the set of possibly interrelated conditions in which an entity exists

# Context Concepts

- Context
- Context Model
- Context Modelling
- Context Information
- Context-Aware Application
- Situations
- Quality of Context

# Context Concepts



# Challenges

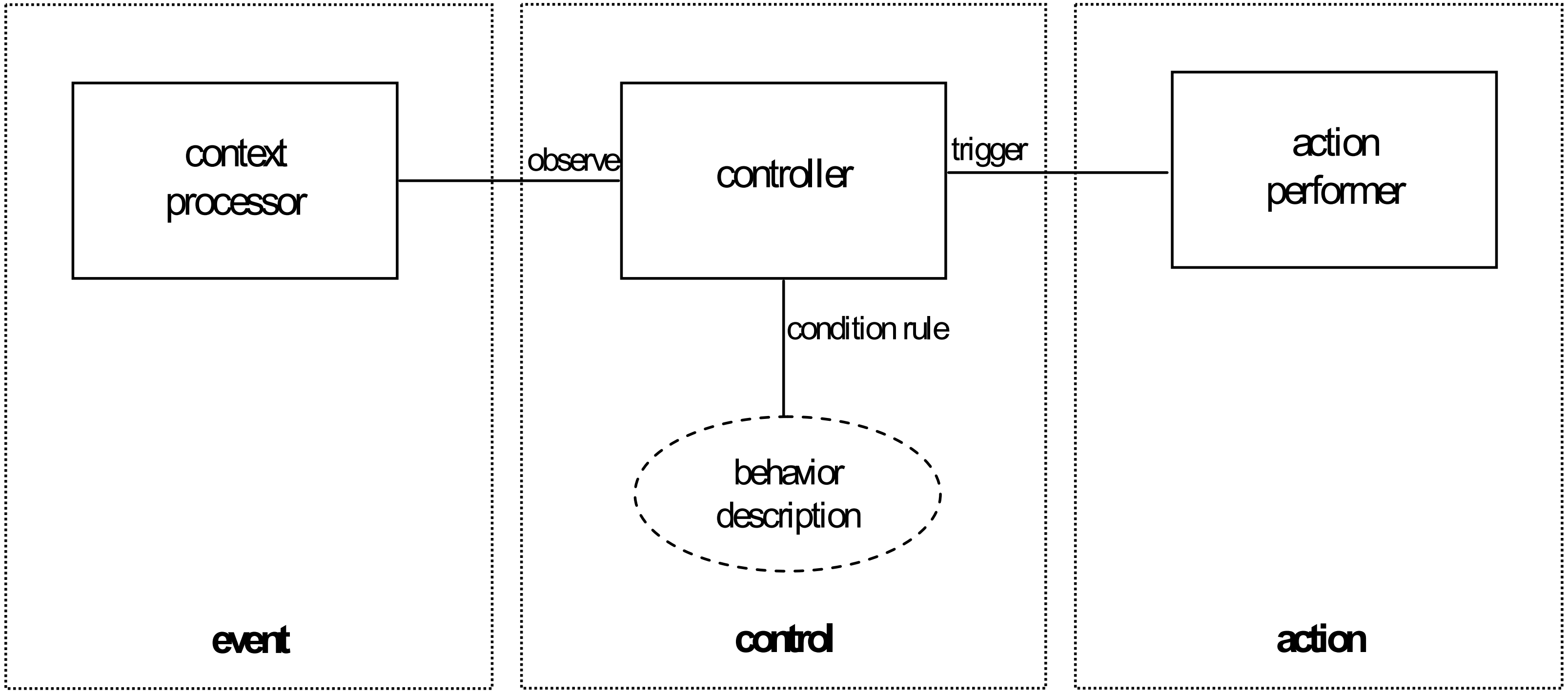
- Capturing context
  - aggregation
  - reasoning
  - inference
- Time sensitive
- Sensors are imperfect (Quality of Context)
- Sensors are distributed
- Applications are distributed (mobile)
- Application adaptation, reactivity
- Security, privacy

# Objective

- Integrated solution for the development of context-aware applications:
  - Reference Architecture
  - Context Handling Platform
  - Context Modelling

# Reference Architecture

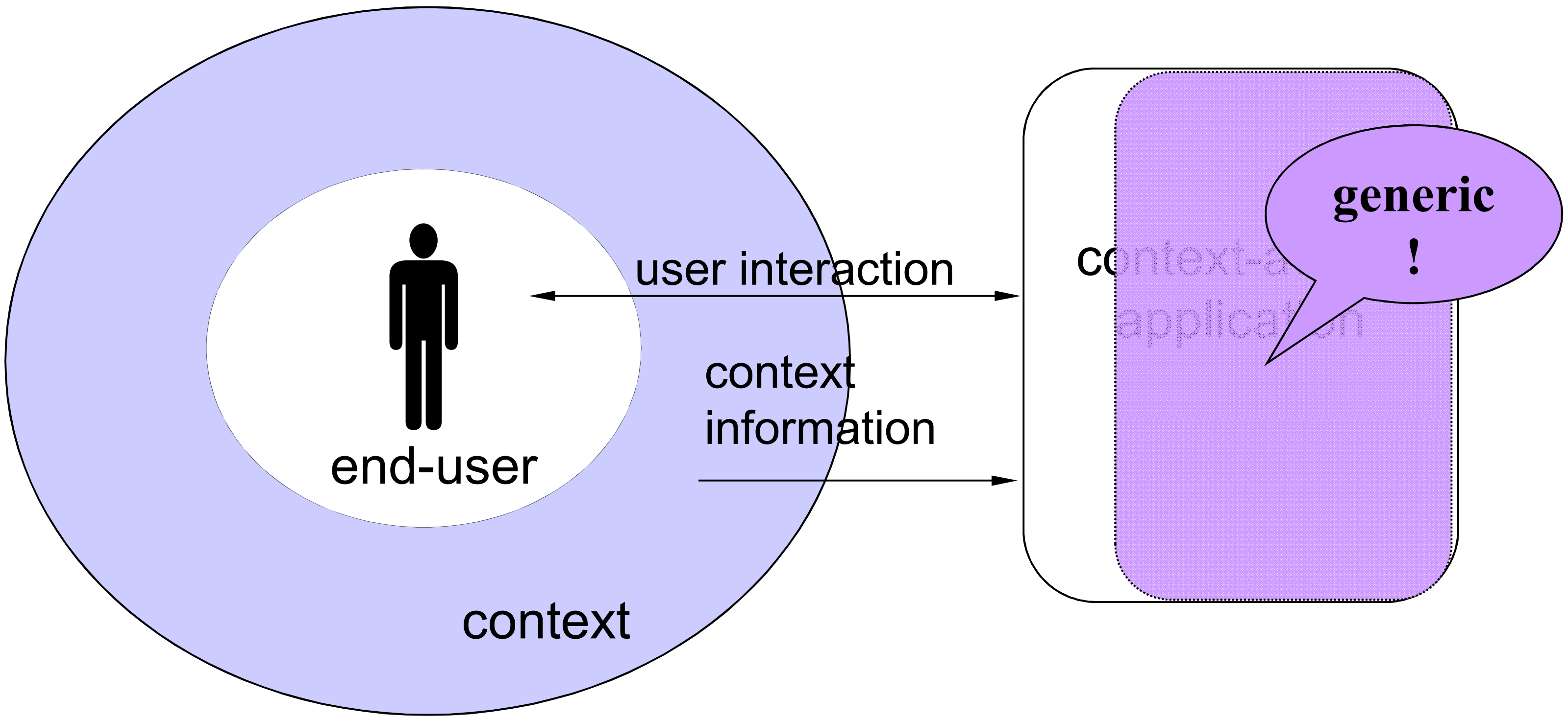
- Context-Aware Patterns
  - Event-Control-Action pattern
  - Context Sources and Managers Hierarchy Pattern
  - Actions Pattern



# Reference Architecture

- Components and Interfaces
  - Context sources
  - Context managers
  - Controllers
  - Action components

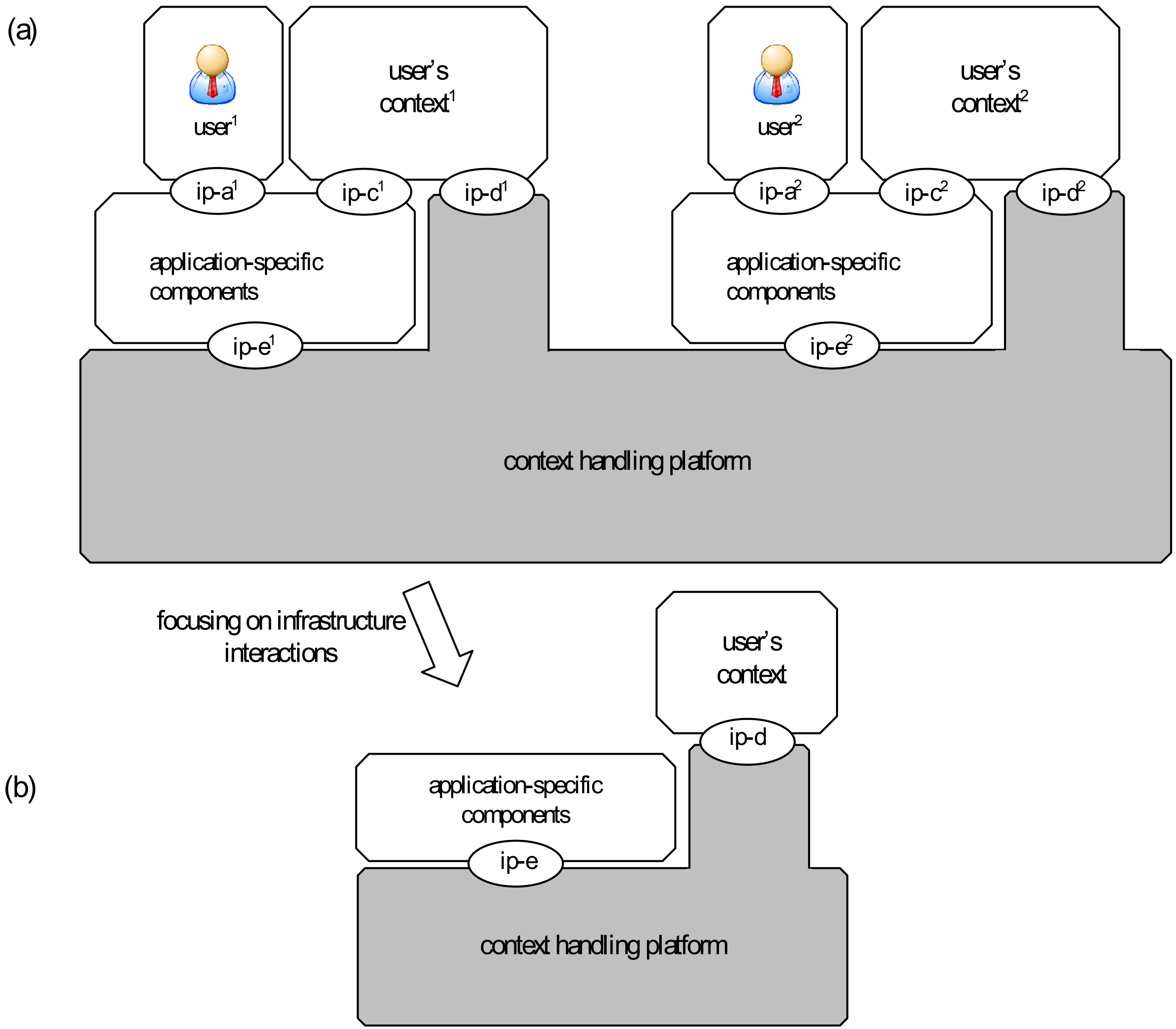
# Context Handling Platform



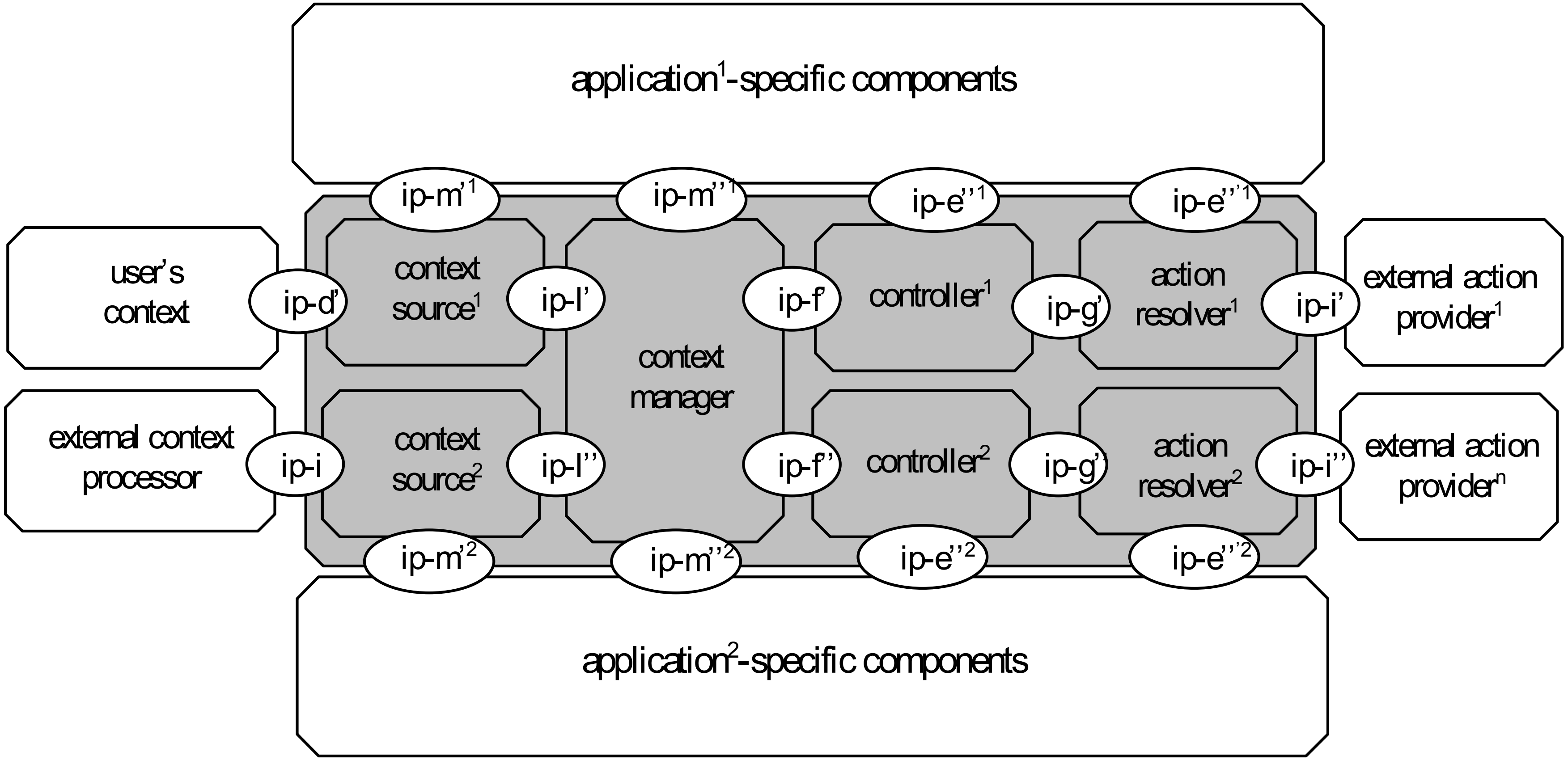
# Context Handling Platform

- Generalizes functionality that can be reused by several context-aware applications
- Gathers context information, performs context processing
- Detects situations in a distributed fashion
- Allows delegation of application rules
- Performs adaptation on behalf of applications

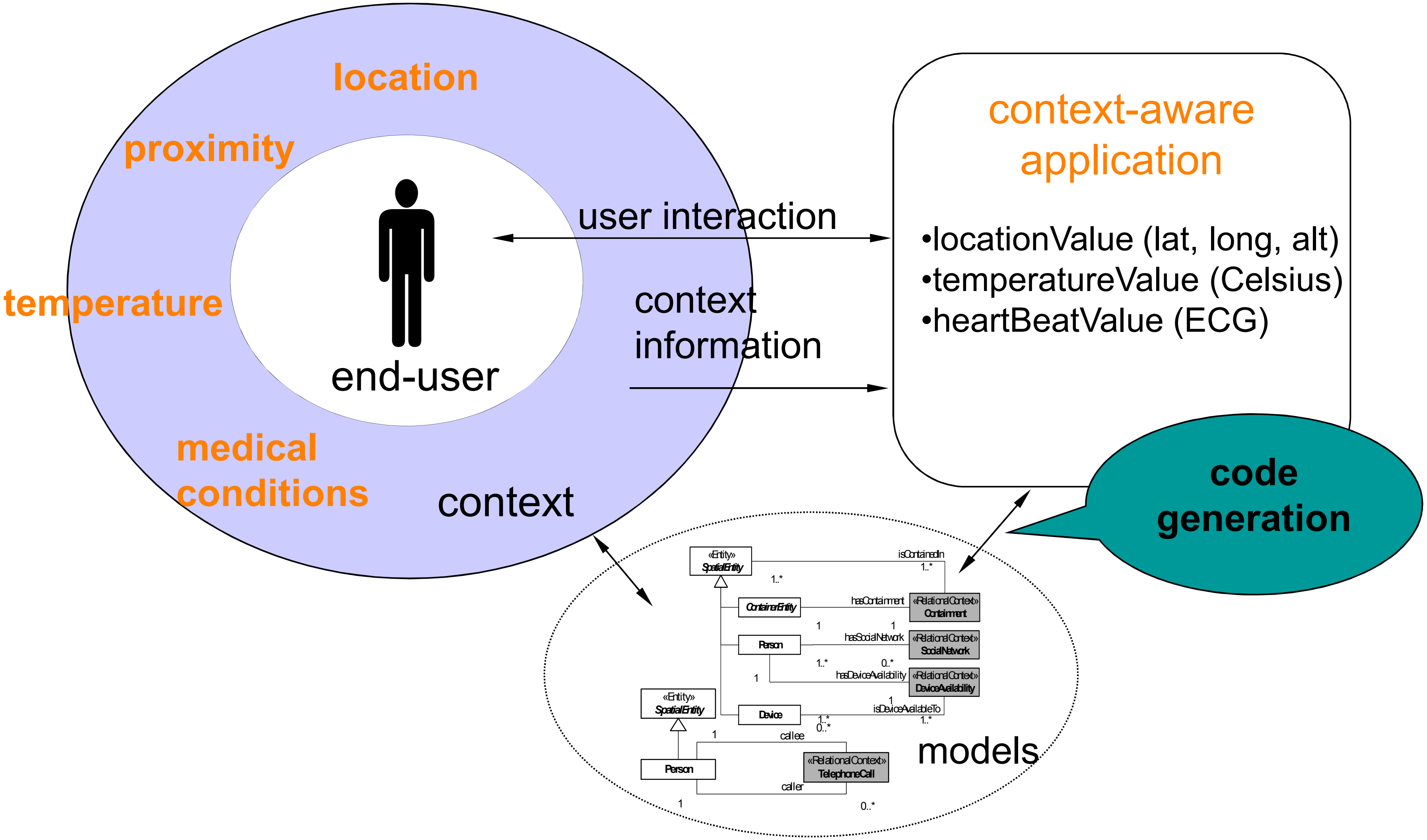
# Context Handling Platform



# Context Handling Platform



# Context Modelling

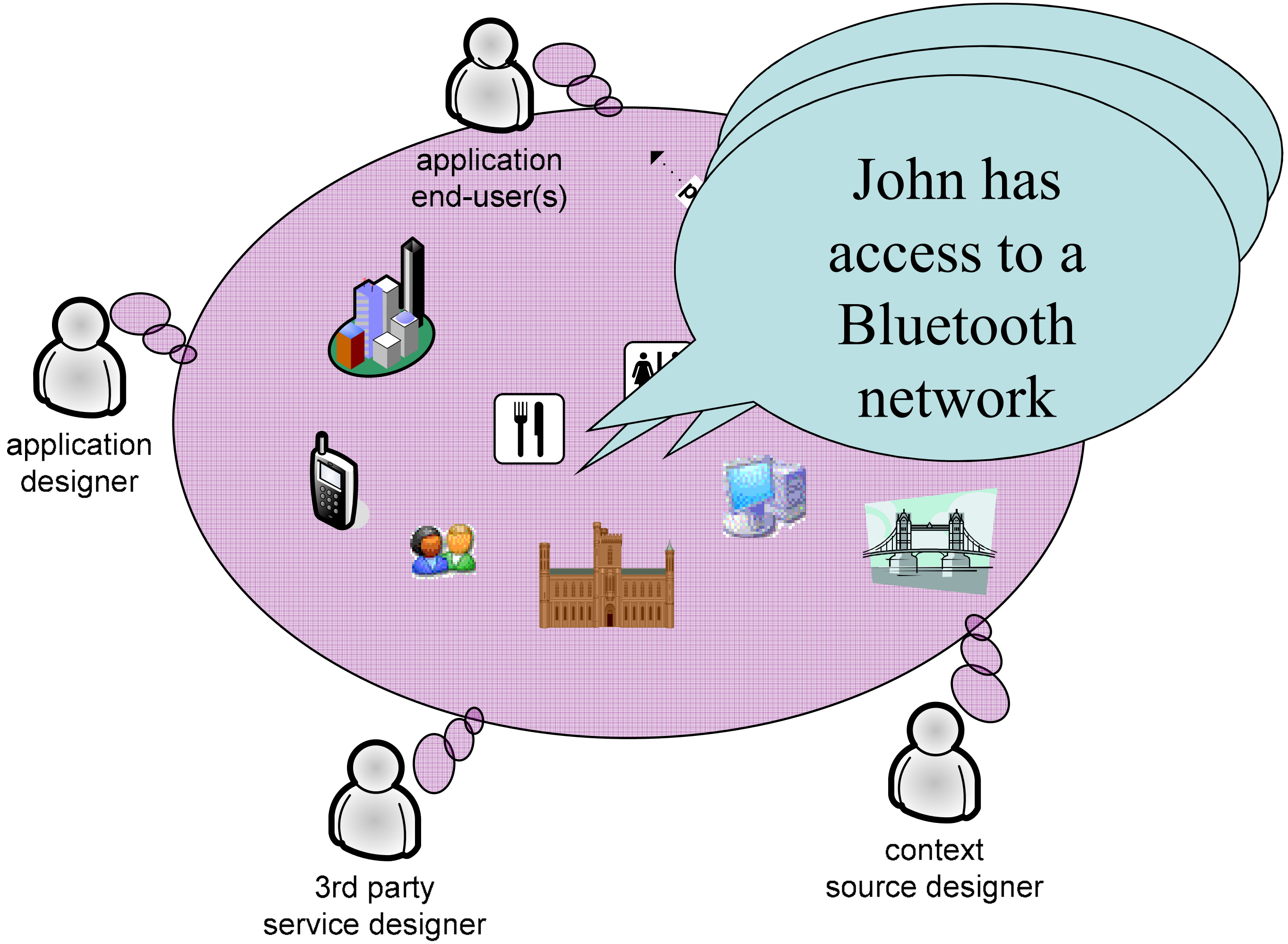


# Context Modelling

- Abstractions to facilitate representation of context information
- Situation modelling (aggregation, reasoning)
- Automatic code generation
- Quality of Context (QoC)

# Application’s universe of discourse and state-of-affairs

- Tourist Application



# Context Modelling Requirements

- In order to support context-aware applications one needs amongst others (meta)models that define
  - Context and situation types and their relationships
  - The "imperfection" of context information (Quality of context)
  - Adaptation rules based on context and situations

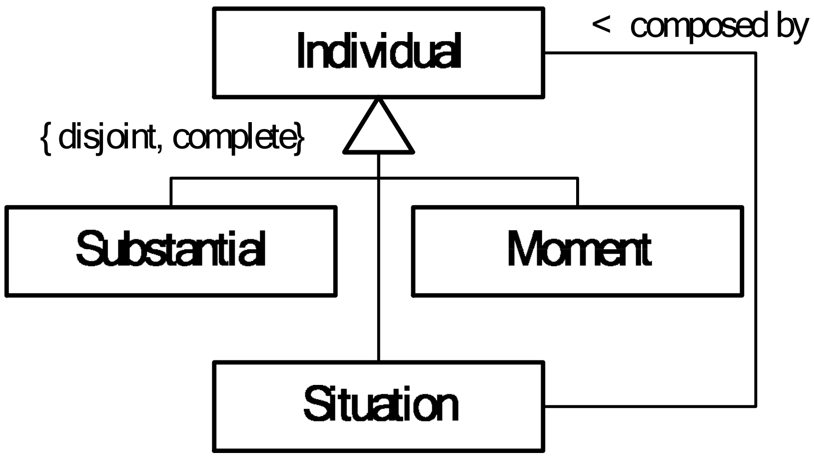
- Context models should:
  - Support common understanding, problem-solving, and communication among the various stakeholders involved in application development
  - Represent context unambiguously

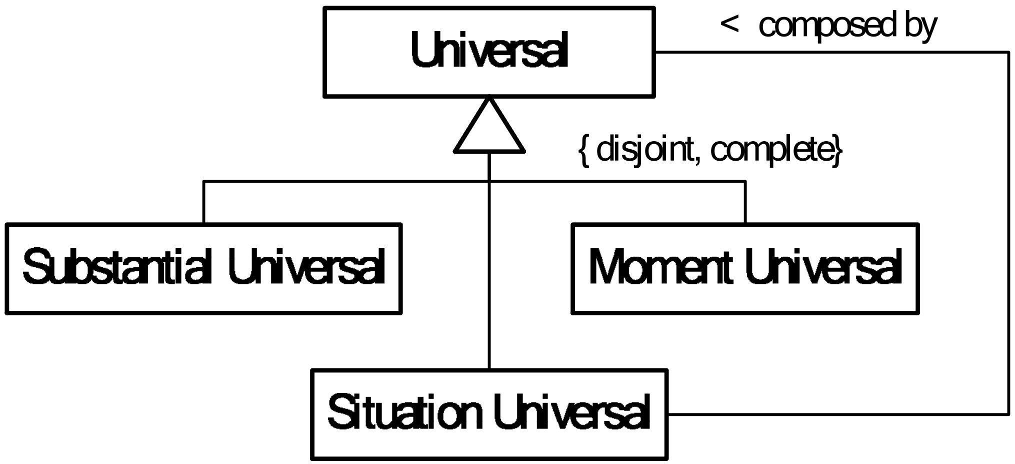
# Goal of Context Modelling

- Provide basic conceptual foundations for context modeling, which allow context-aware application designers to represent (i) relevant elements of a context-aware application’s universe of discourse; and (ii) particular state-of-affairs of interest

- We consider results from foundational ontologies to support our conceptual context modelling approach

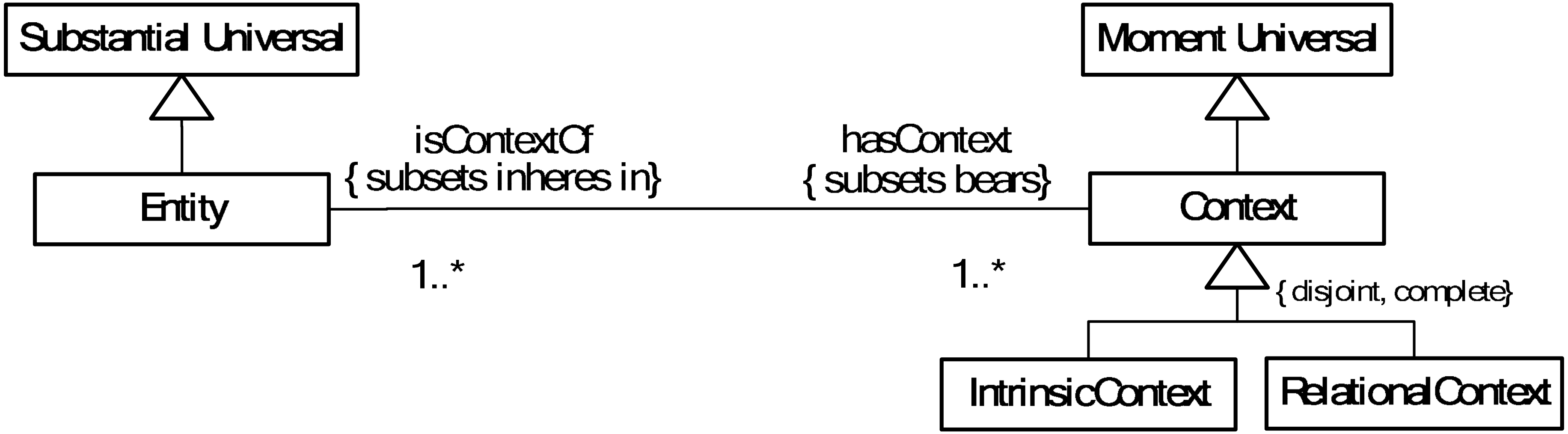
# Foundational ontologies: related work



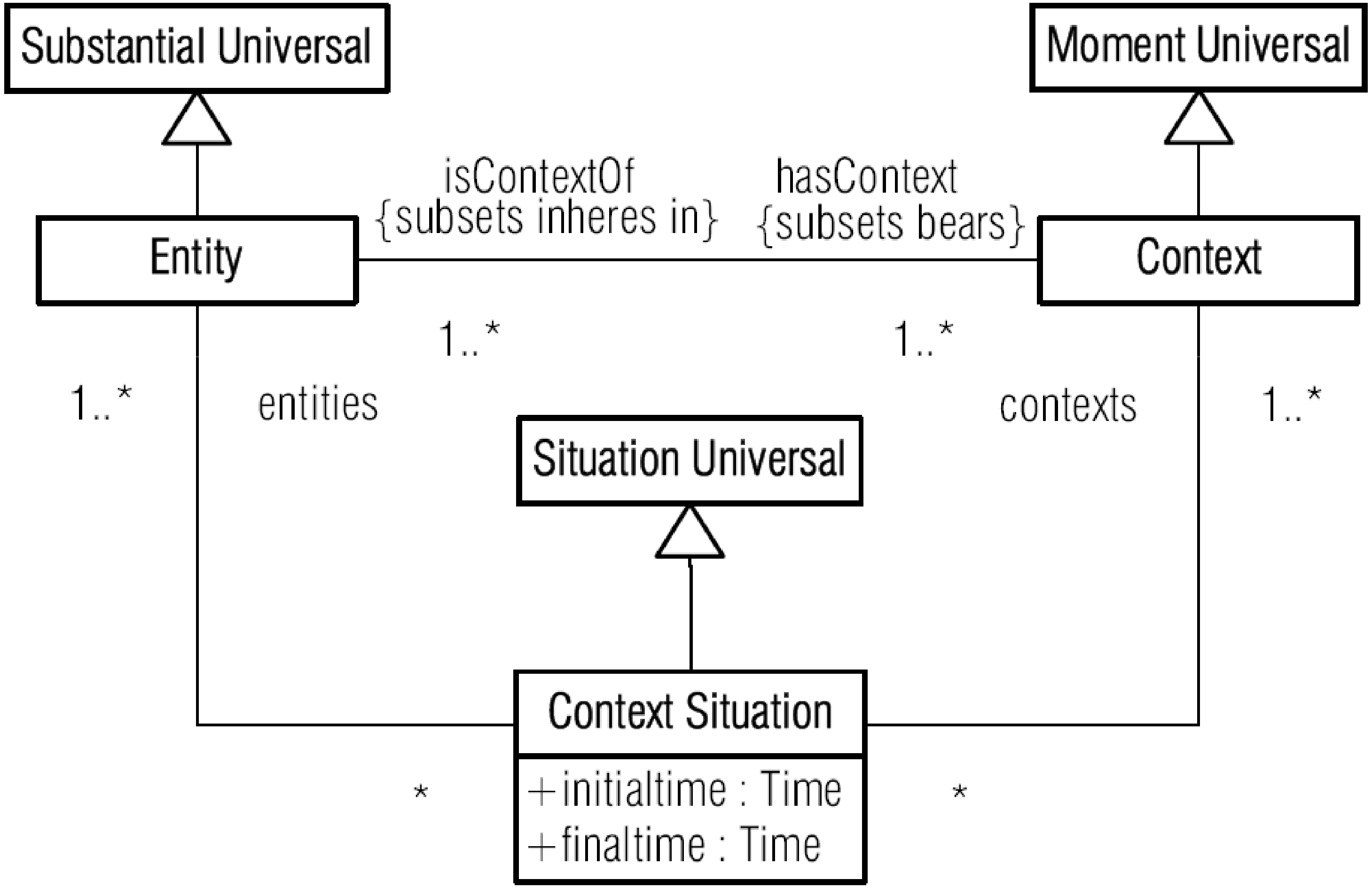



# Foundational Context Concepts

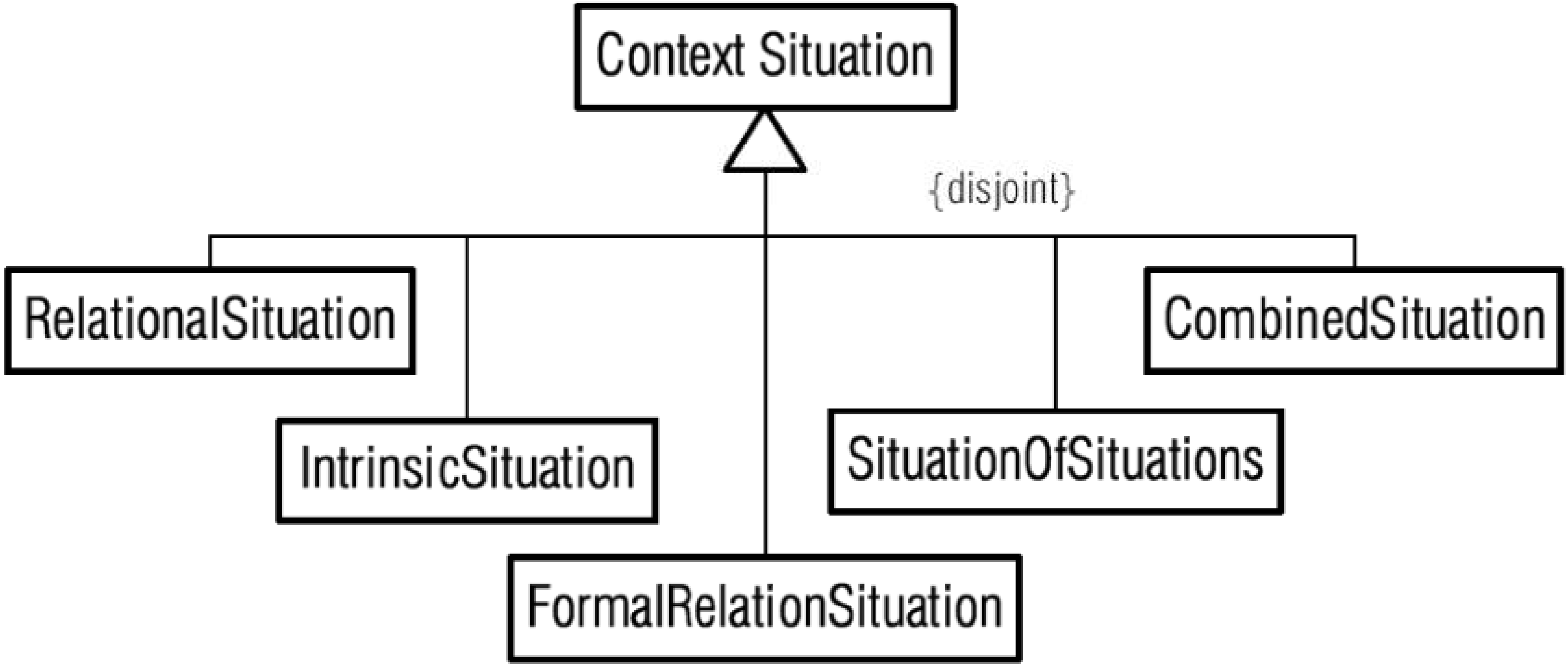
- Context
  - the set of possibly interrelated conditions in which an entity exists



# Foundational Context Concepts: situation



# Foundational Context Concepts: situation



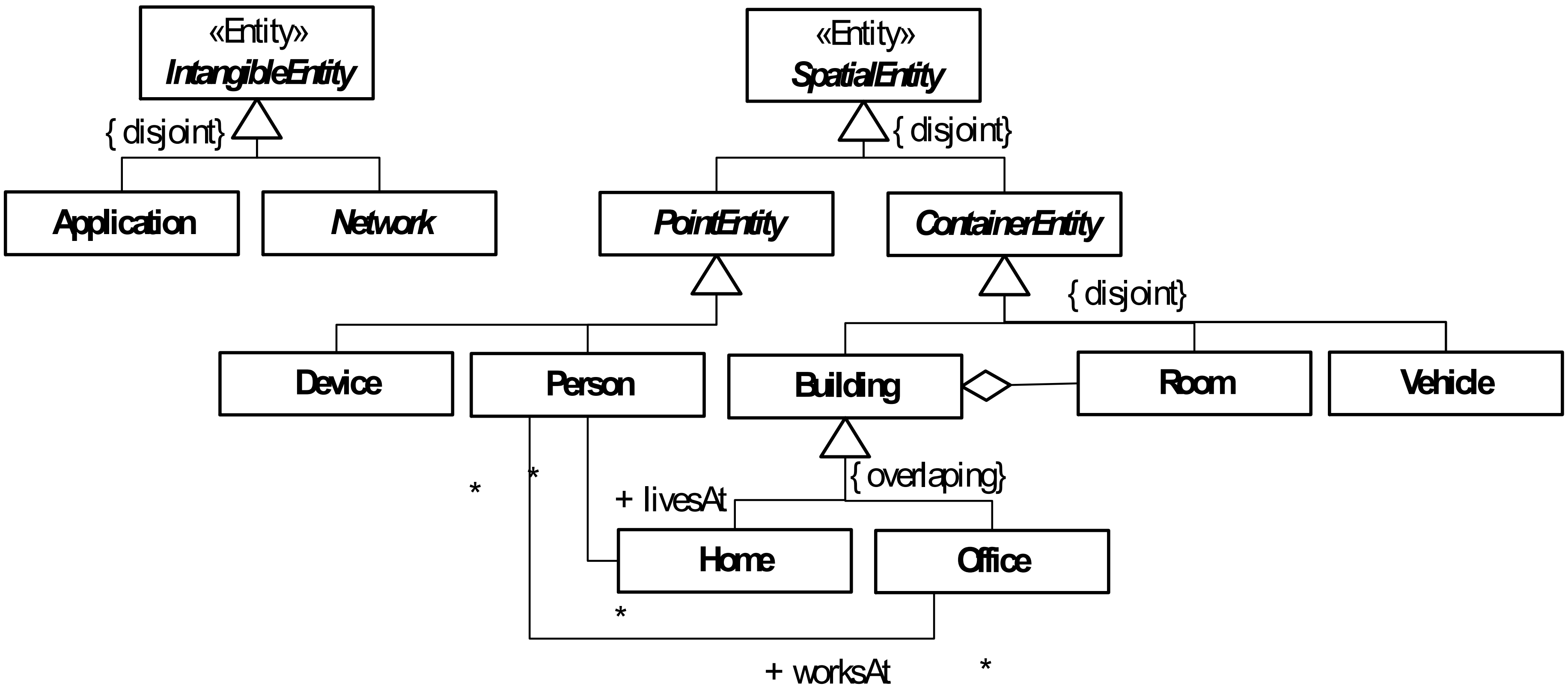
# Foundational Context Concepts (UML profile)

- Artifacts for specification
  - Context profile: <<Entity>>, <<RelationalContext>>, <<IntrinsicContext>>,...
  - Situation profile: <<IntrinsicSituation>>, <<FormalRelationSituation, <<RelationalSituation>>, ...
- Products of specification
  - Context Models: person, Temperature, GeoLocation, GeoLocationCoordinates, Device, etc
  - Situation Models: SituationFever, SituationConnected, SituationPresentation, etc

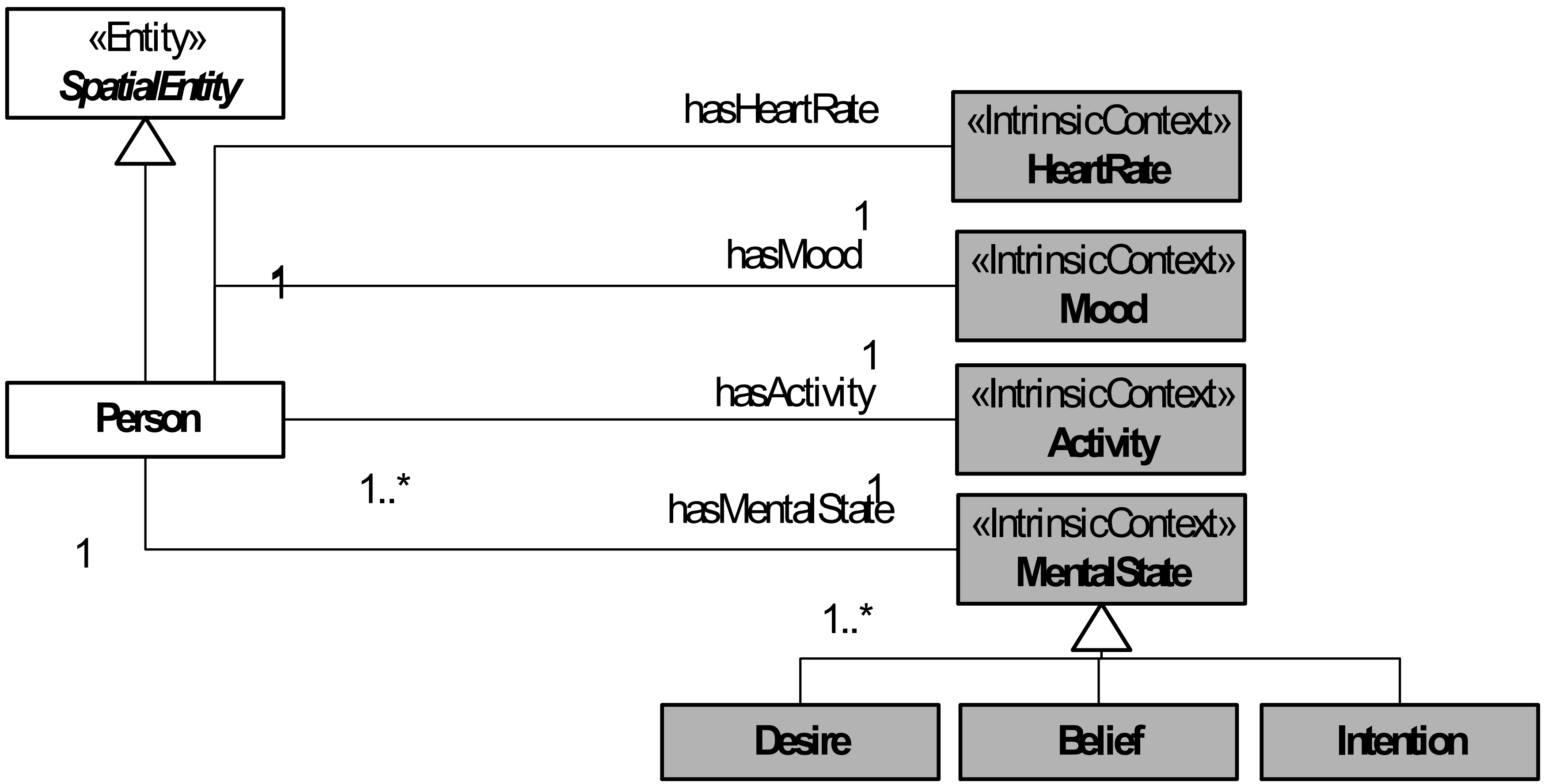
# Foundational Context Concepts: summarized

| **Foundational context concepts** | **Description** |
|---|---|
| *Entity* | an object that beares context |
| *Context* | a particular condition that inheres in an entity |
| *Intrinsic Context* | a particular type of context that belongs to the essential nature of a single entity |
| *Relational Context* | a particular type of context that depends on the relation between distinct entities |
| *Contextual Formal Relation* | a relation that holds directly between two or more entities' intrinsic values (qualities) |
| *Context Situation* | a composite concept that defines particular application's state-of-affairs. It can be composed of entities, contexts, and other situations |
| *Intrinsic Situation* | a context situation composed of a single entity and one of its intrinsic contexts |
| *Relational Situation* | a context situation composed of at least two entities and their pertinent relational contexts |
| *Formal Relation Situation* | a context situation composed of a single entity type and two or more of its intrinsic contexts |
| *Situation of Situations* | a context situation composed of other context situations |

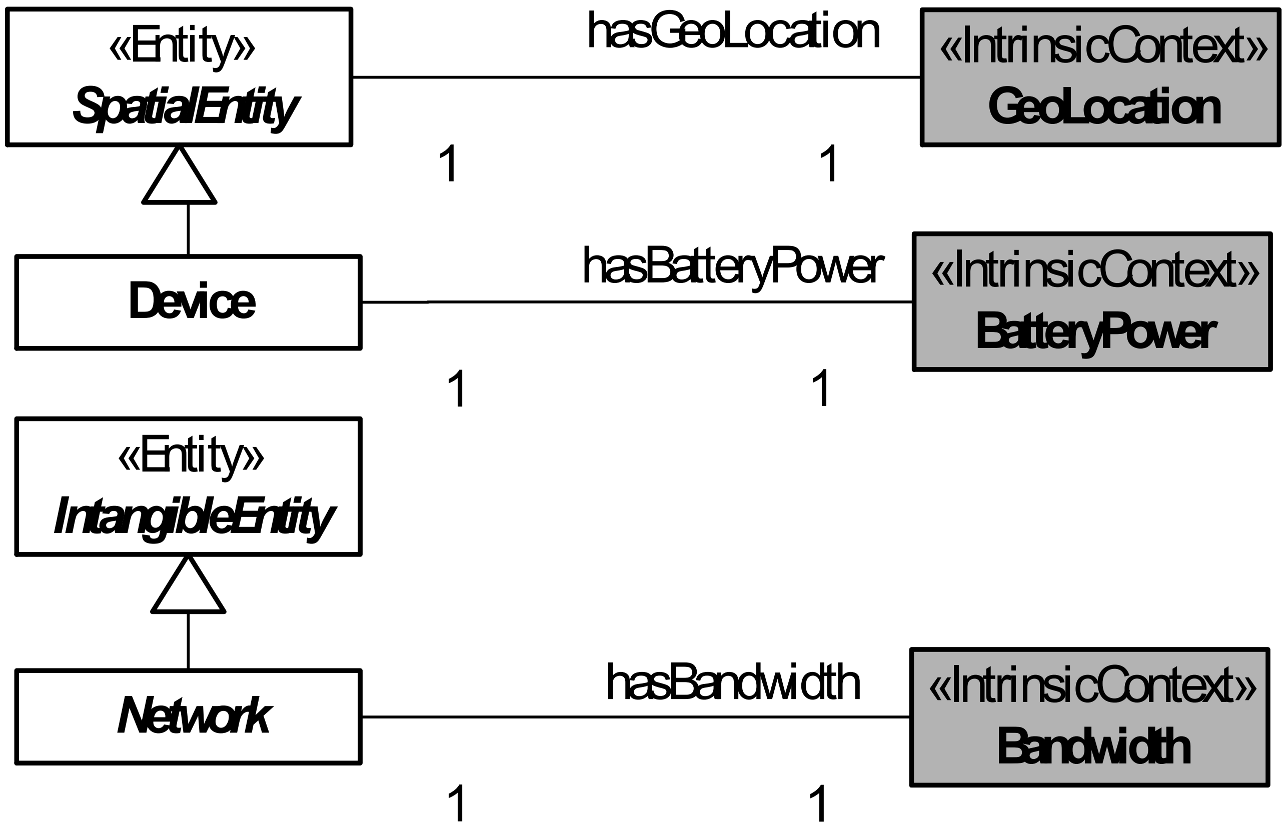
# Context Models: Entity Types



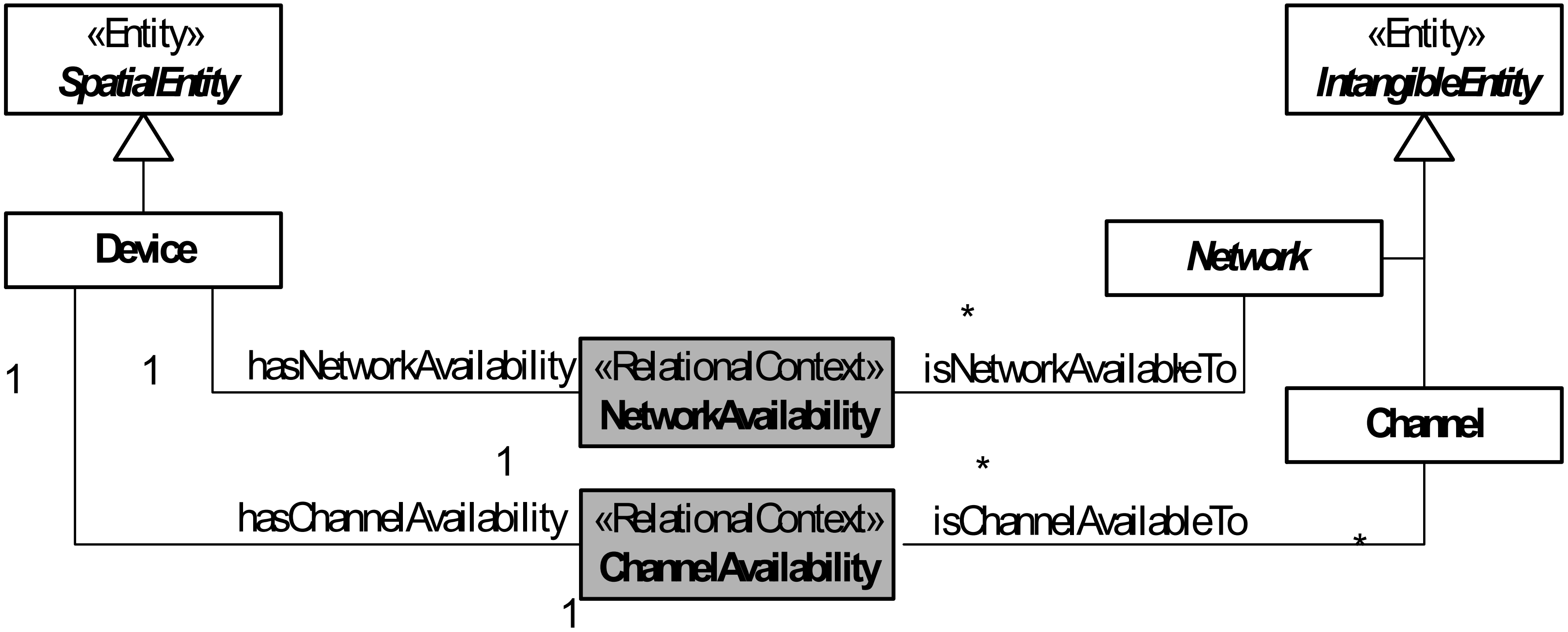
# Intrinsic Context Types



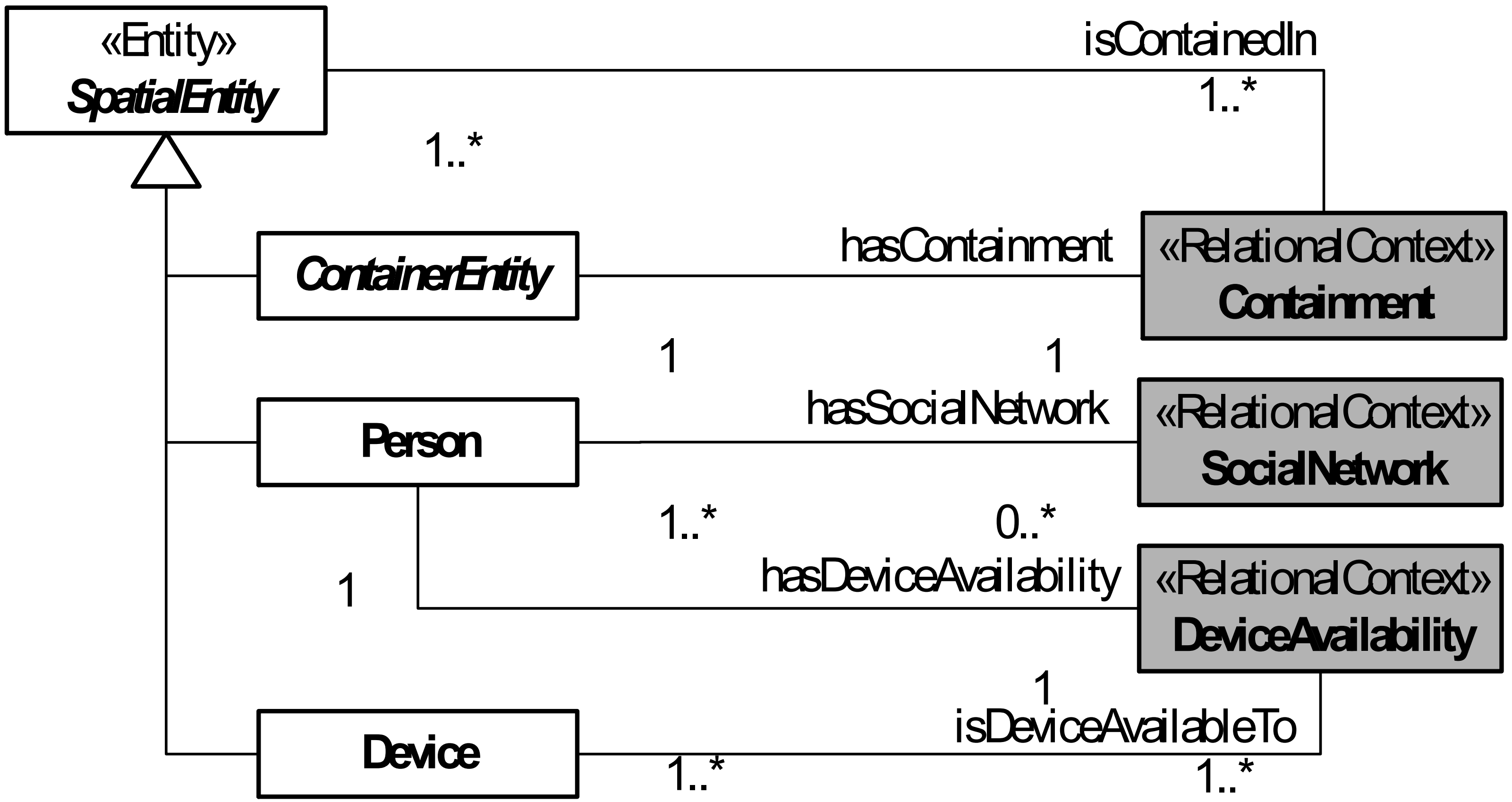
# Intrinsic Context Types



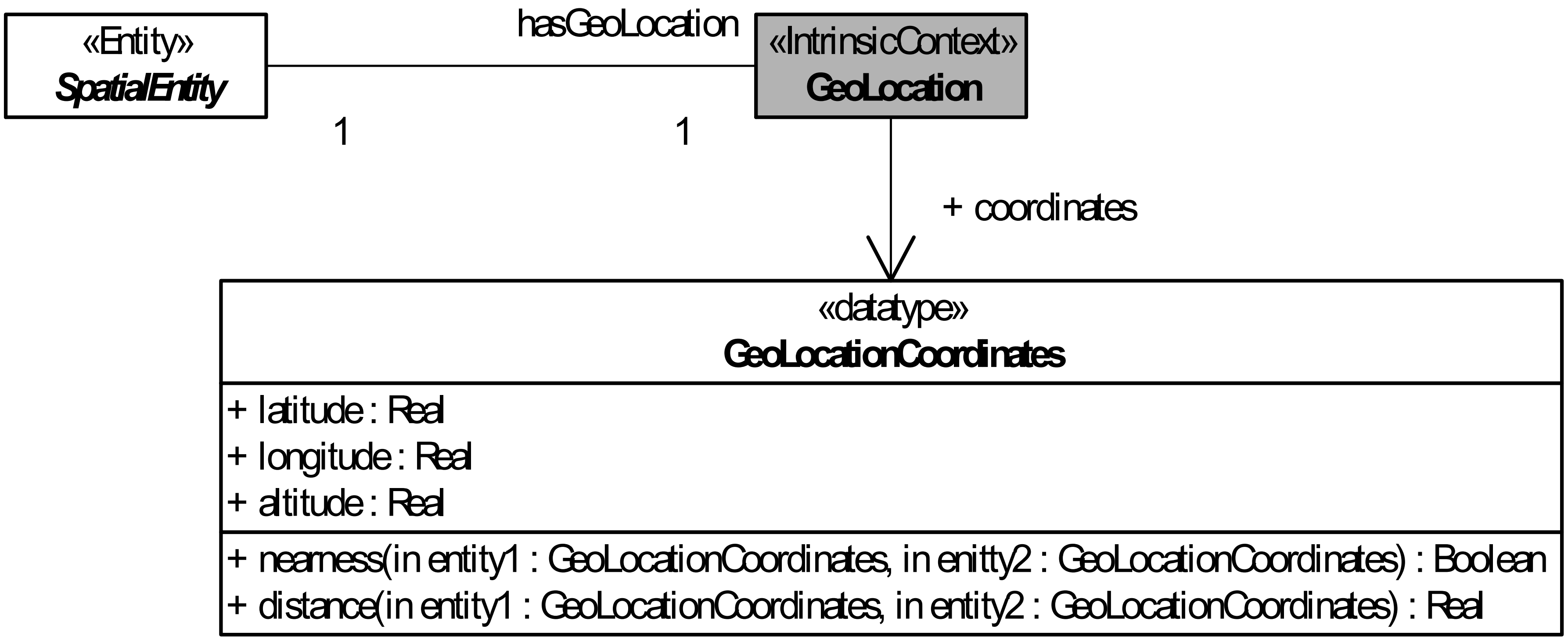
# Relational Context Types



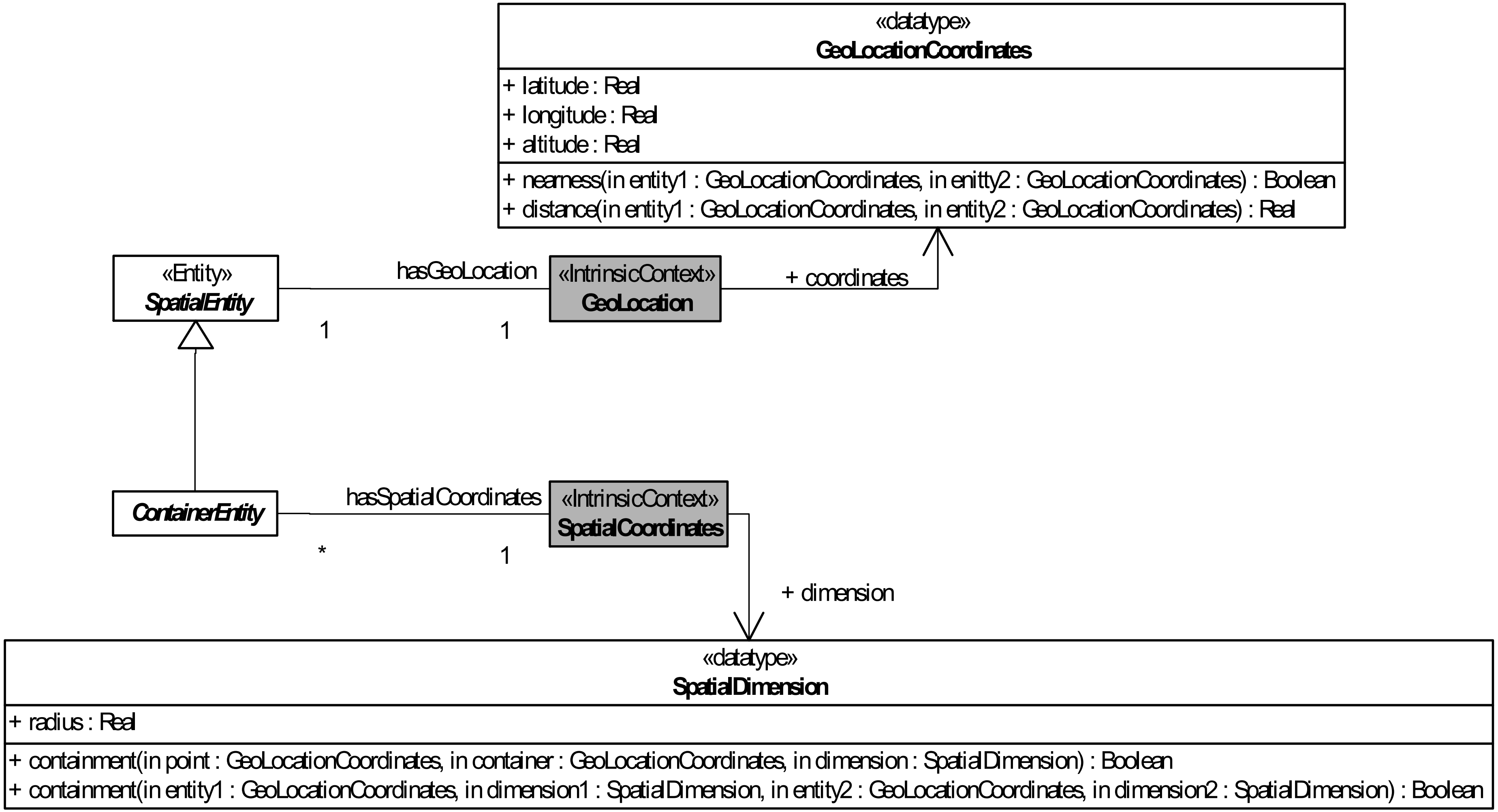
# Relational Context Types



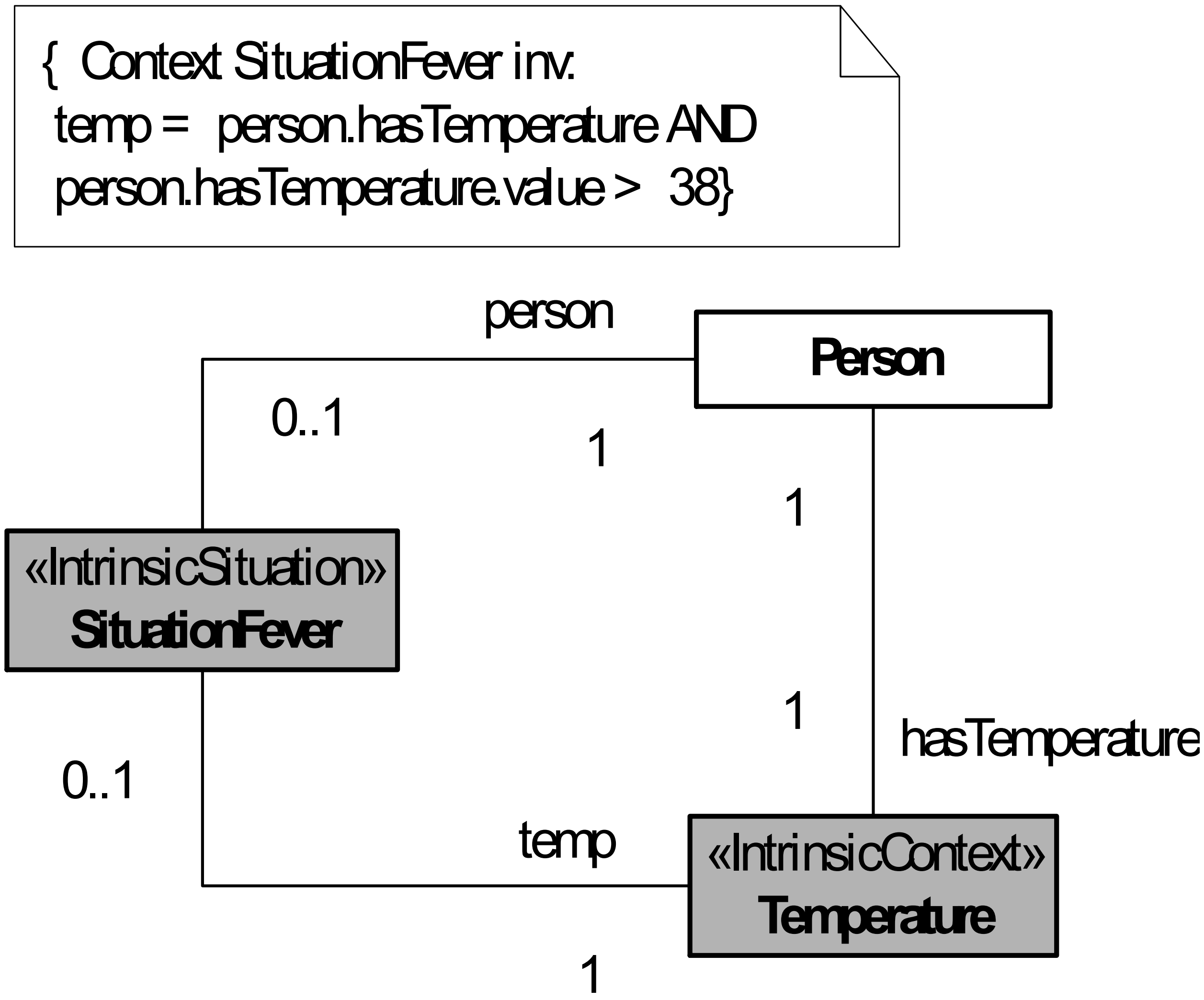
# Contextual Formal Relations



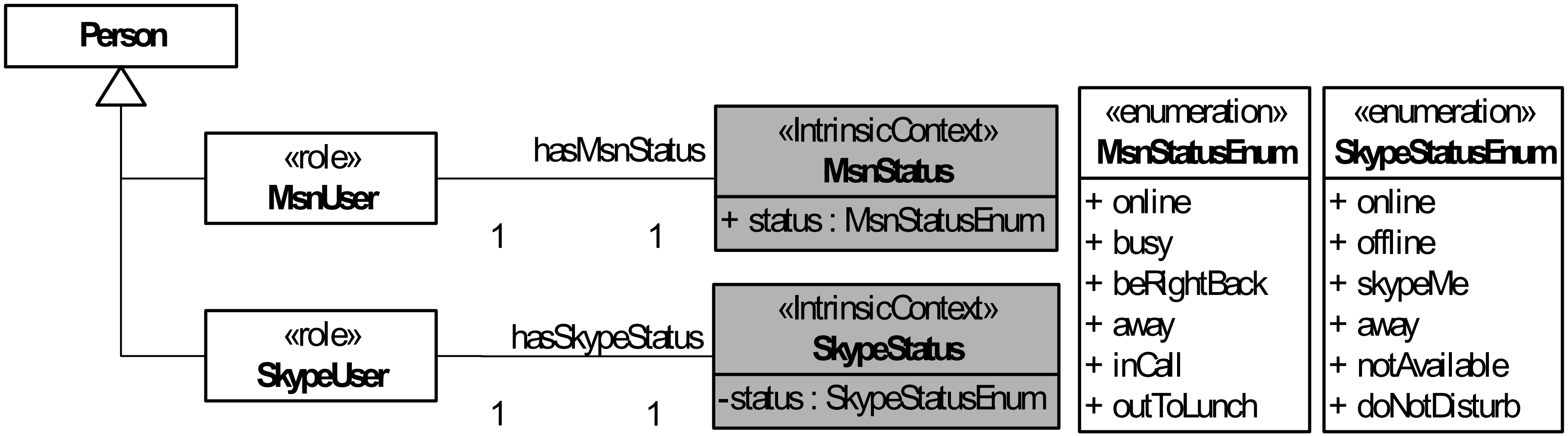
# Contextual Formal Relations



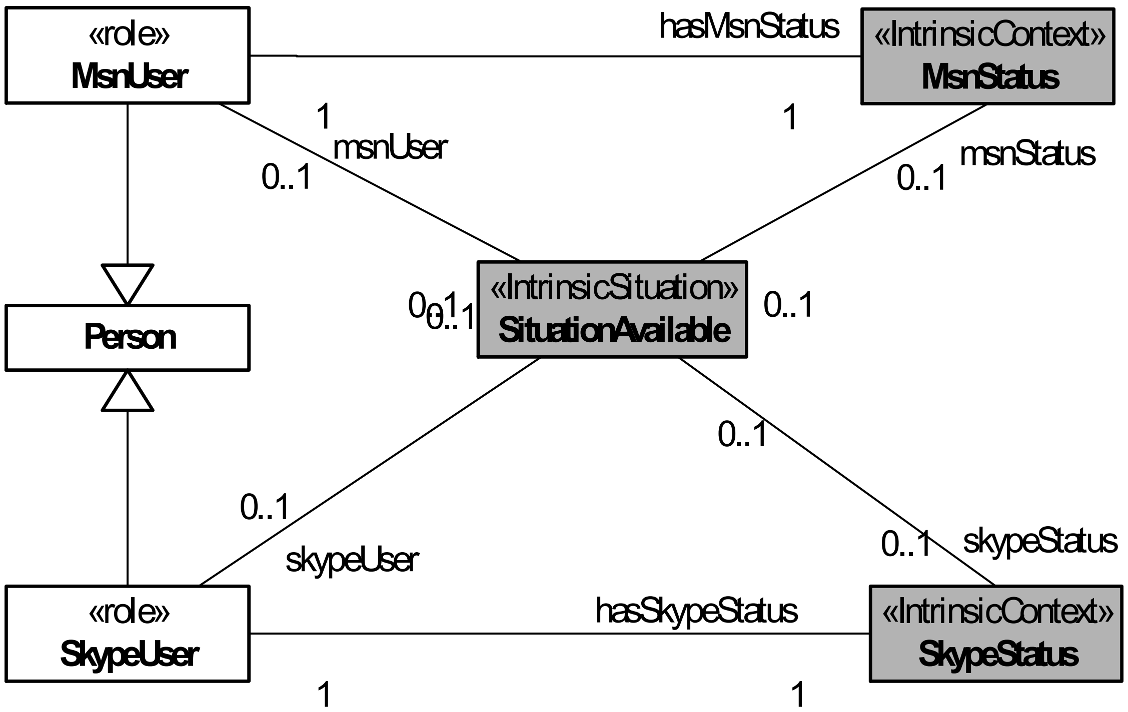
# Context Situation Types: Situation Fever



# Context Situation Types: Context Types



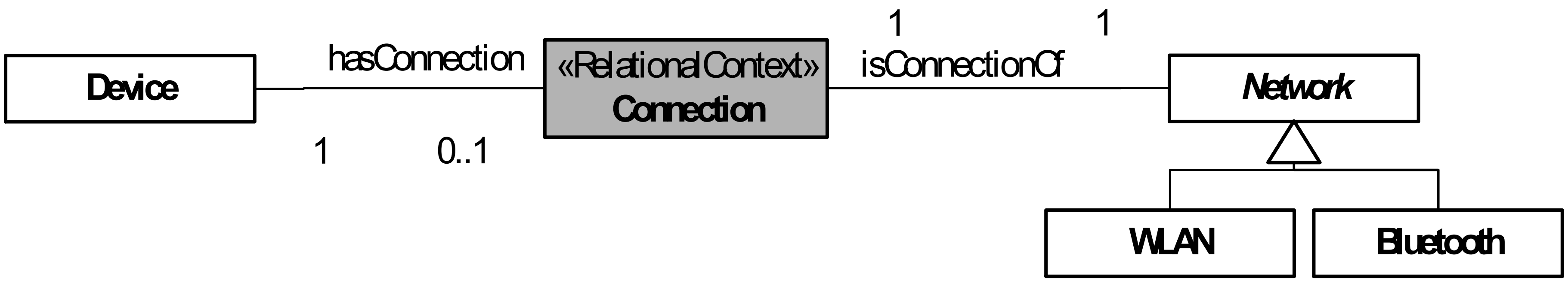
# Intrinsic Situation Types: SituationAvailable

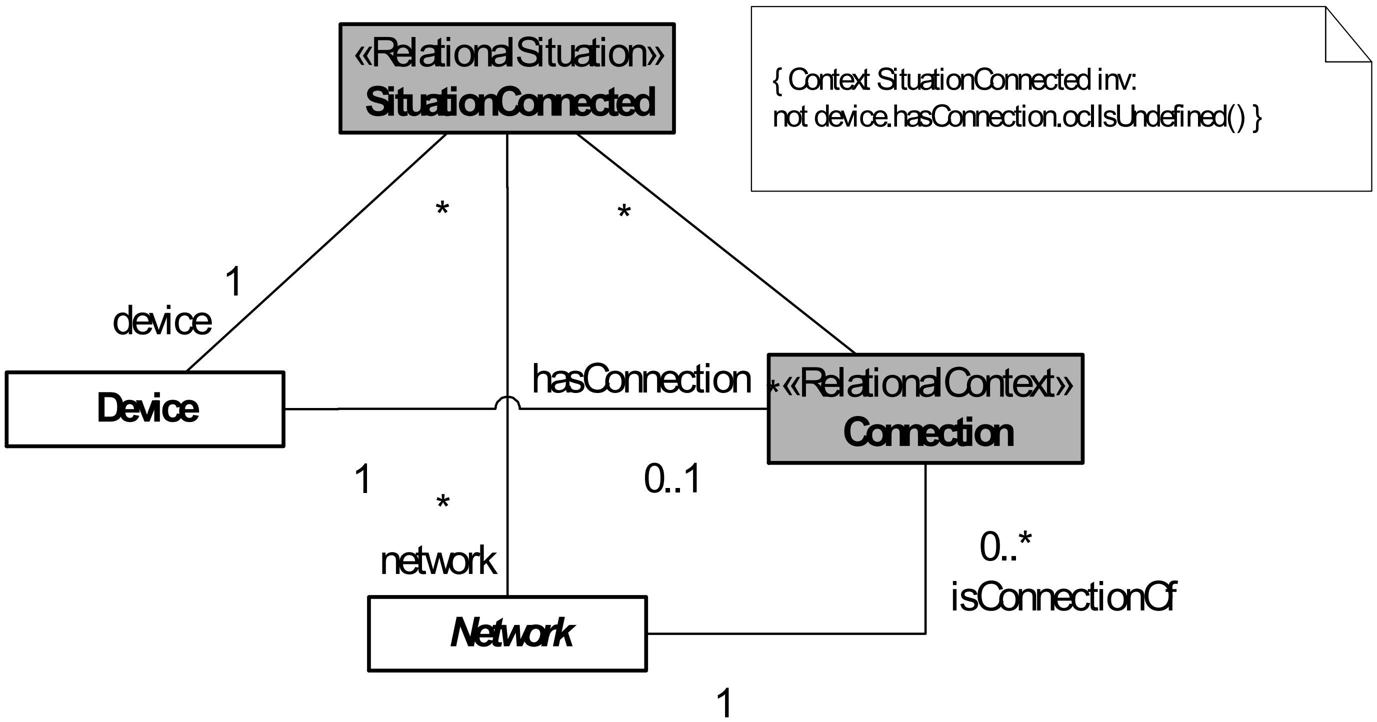



{ Context SituationAvailable inv:

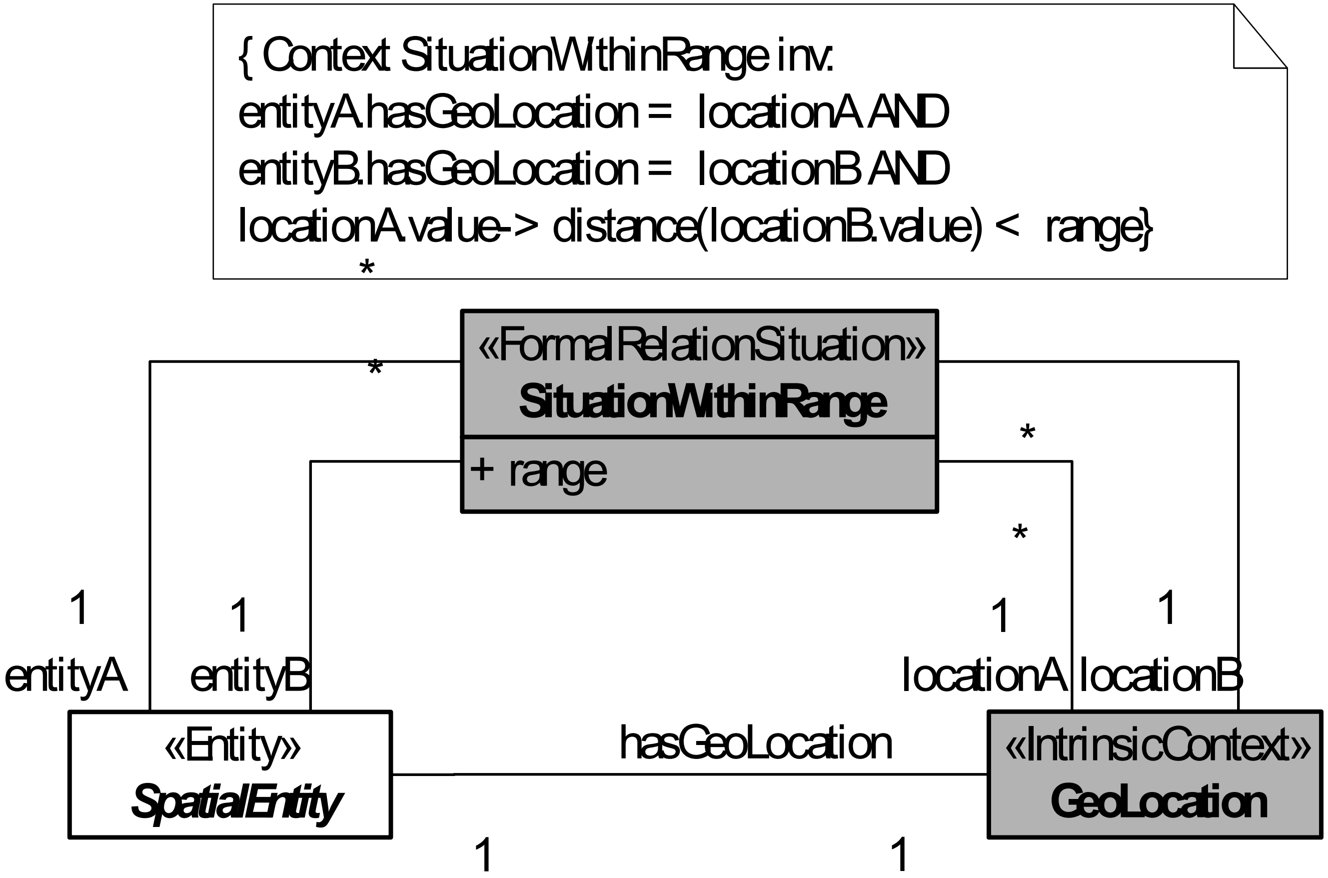
(skypeUser = msnUser) AND
((not skypeUser.oclIsUndefined()) AND
(skypeUser.skypeStatus = skypeStatus) AND
((skypeStatus.value = "Online")
OR (skypeStatus.value = "SkypeMe")))
OR
((not msnUser.oclIsUndefined()) AND
(msnUser.msnStatus = msnStatus) AND
((msnStatus.value = "Online")
OR (msnStatus.value = "BeRightBack"))))}

# Relational Situation Types: SituationConnected

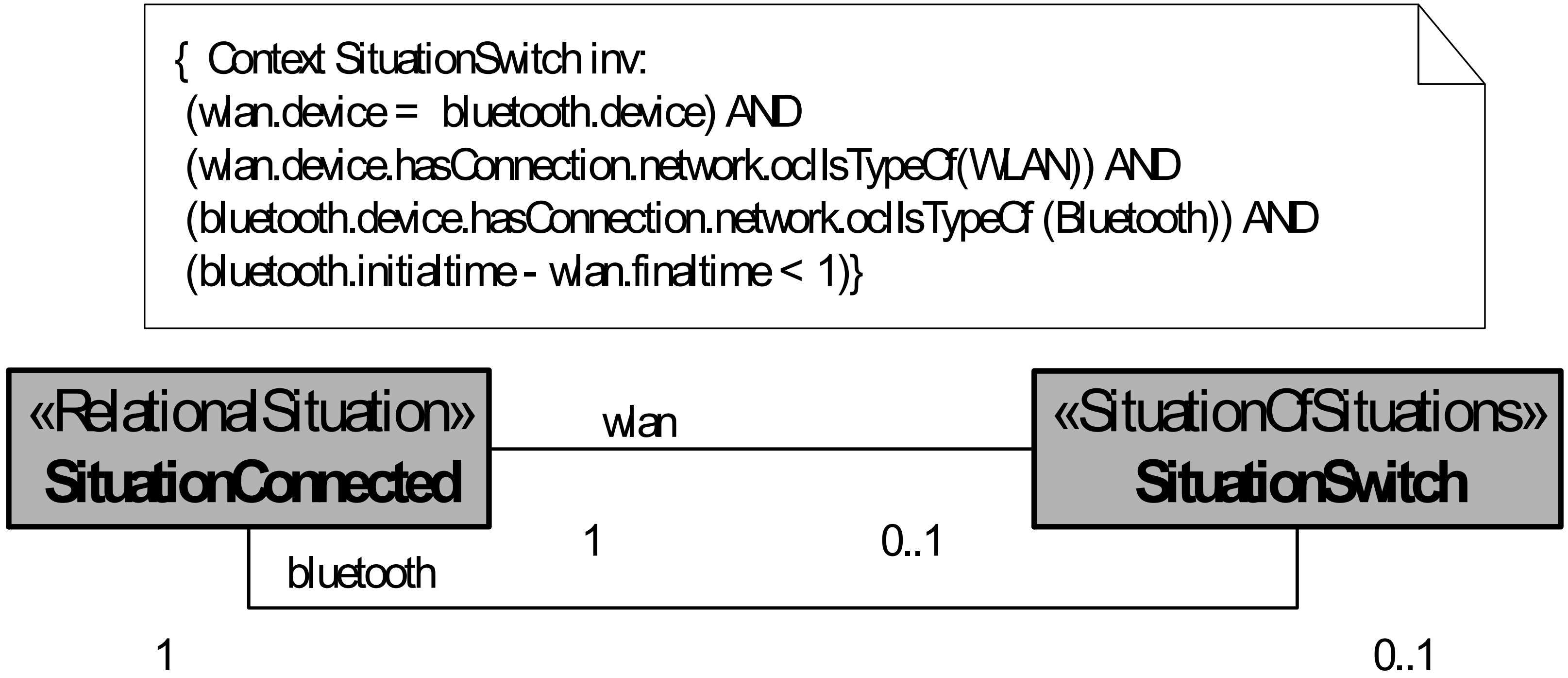


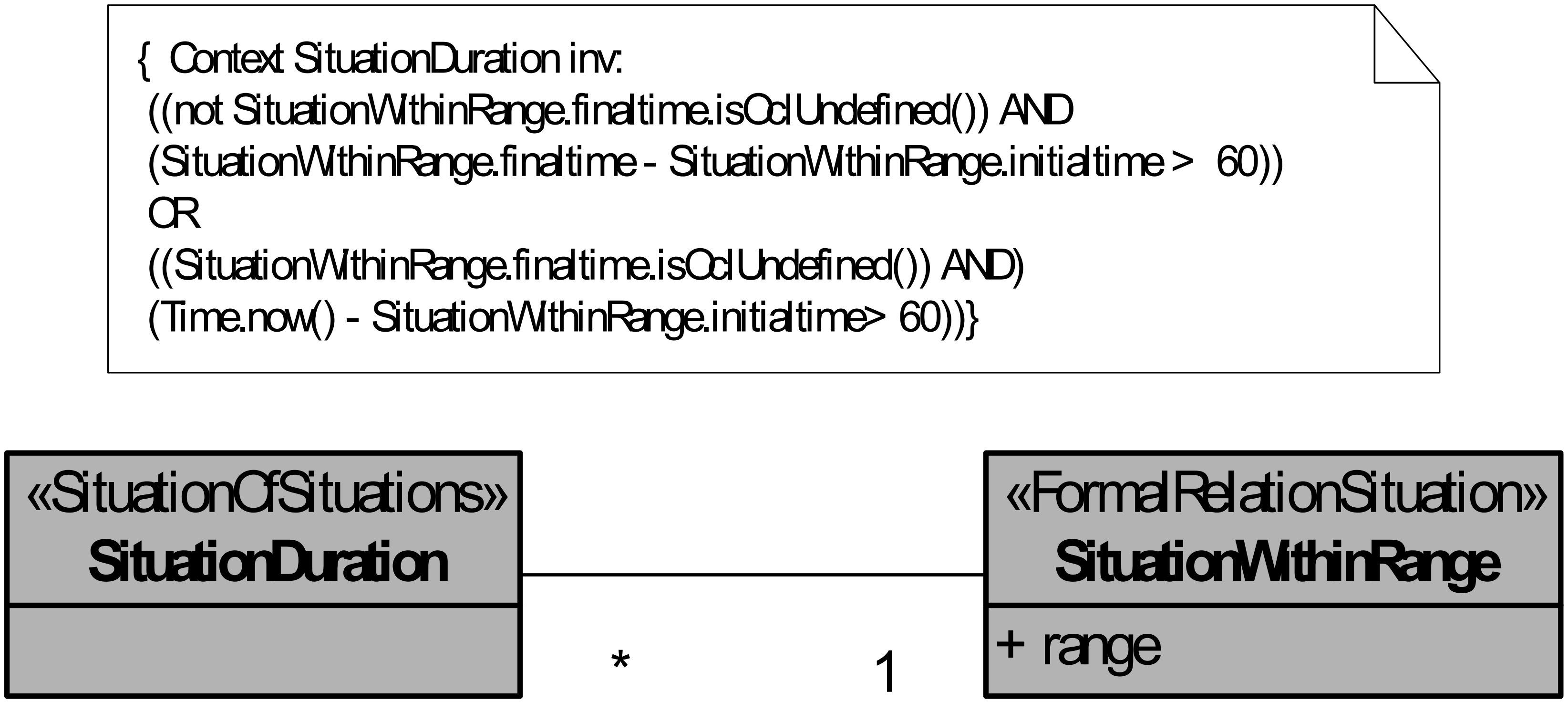
# Formal Relation Situation Types: SituationWithinRange



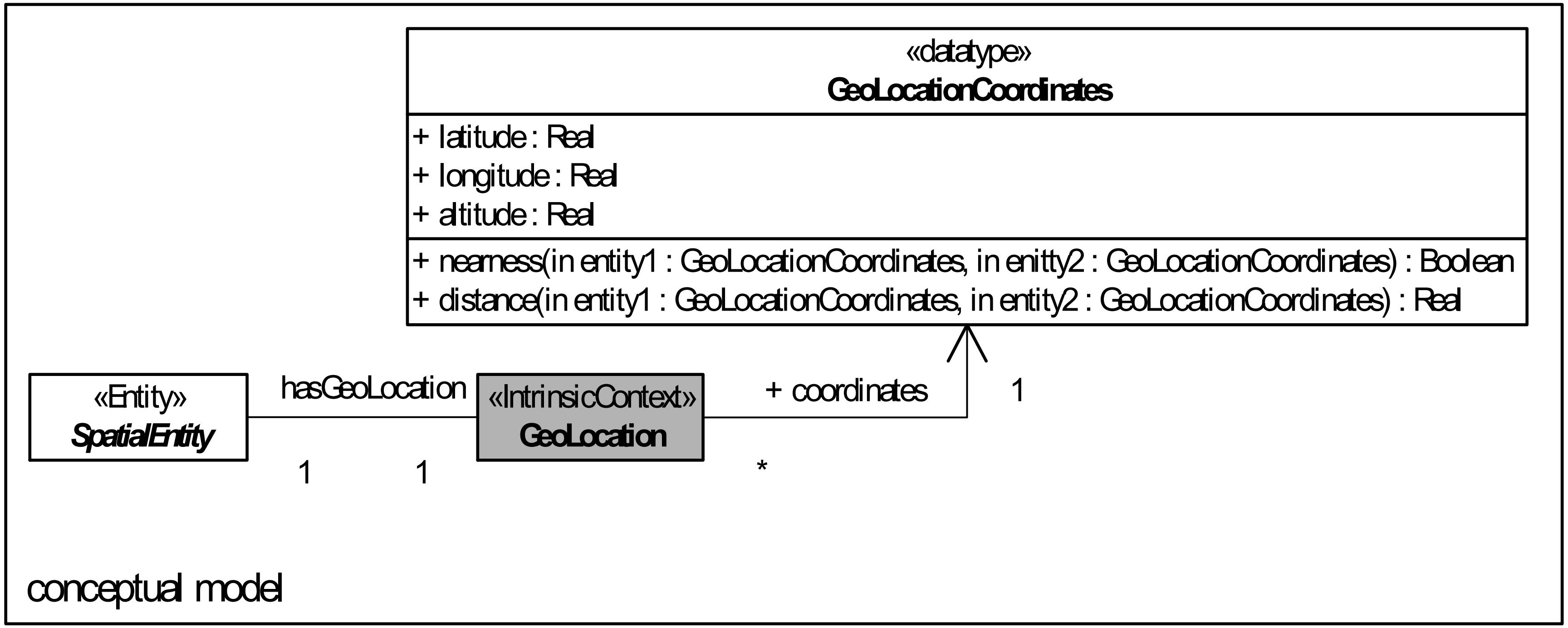
# Situation of Situations Types: SituationSwitch

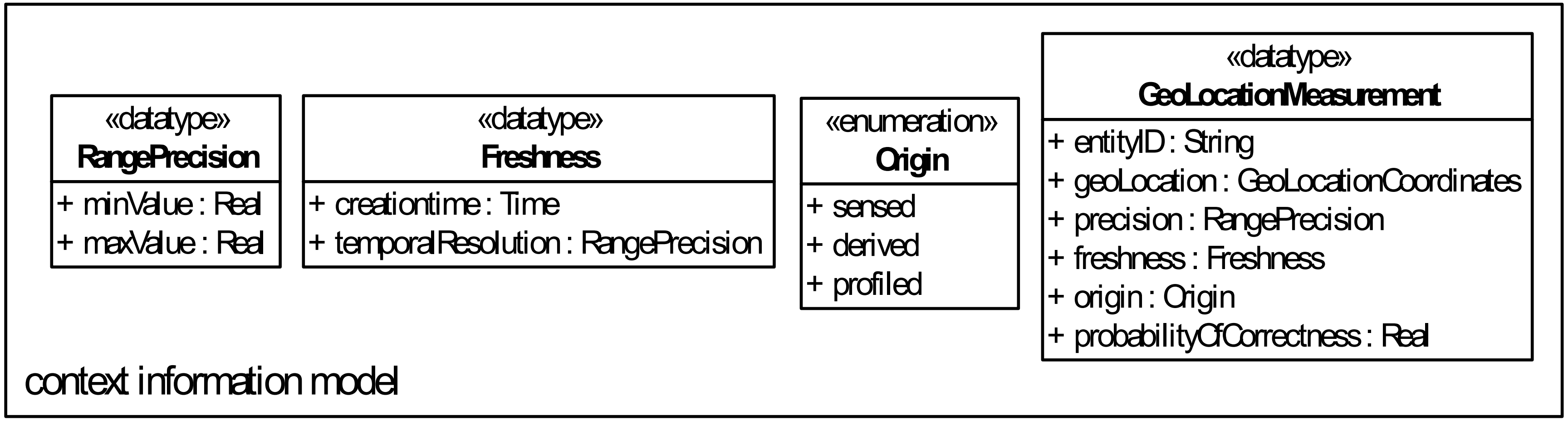


# Situation of Situations Types: SituationSwitch

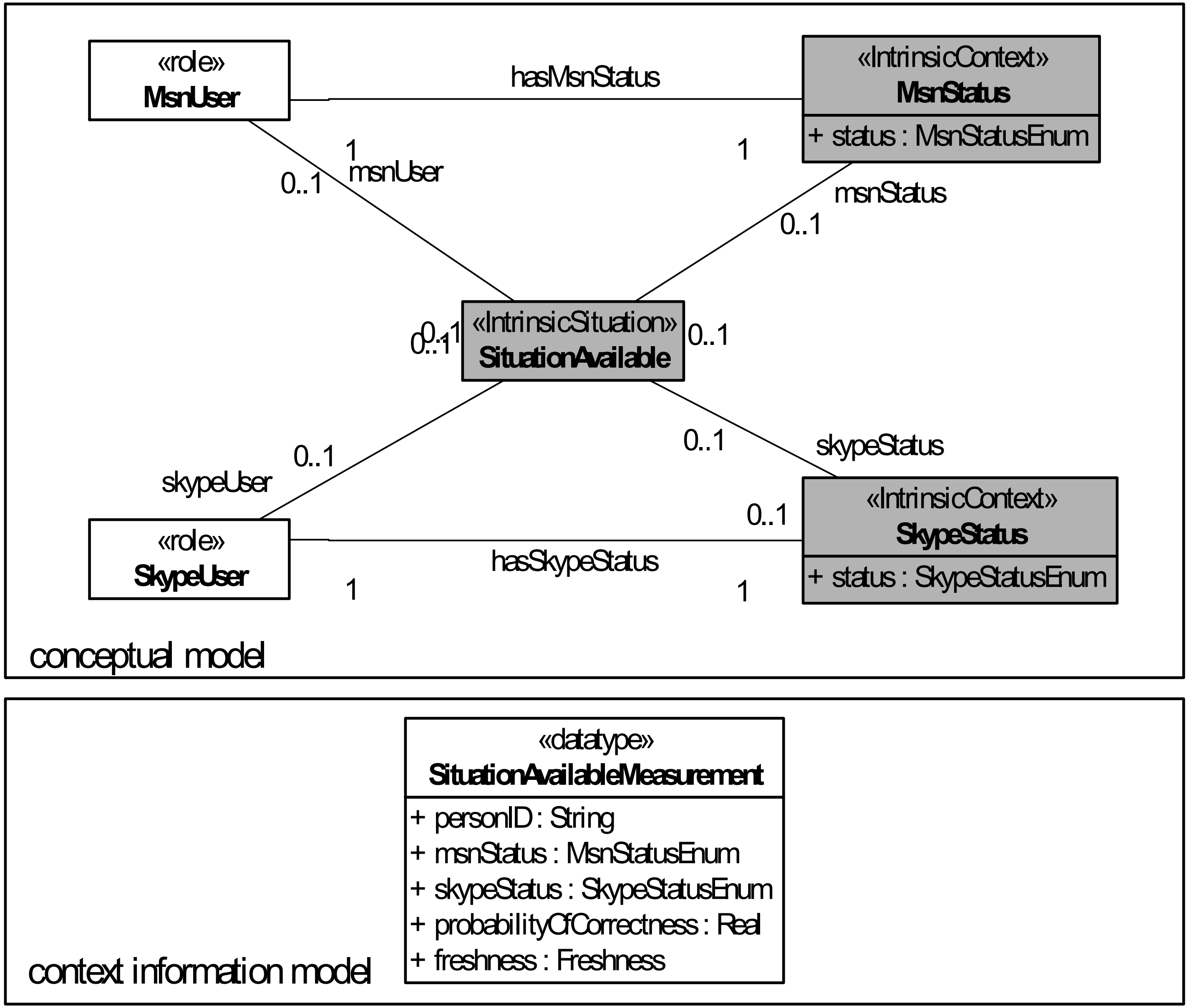


# Context Information Models

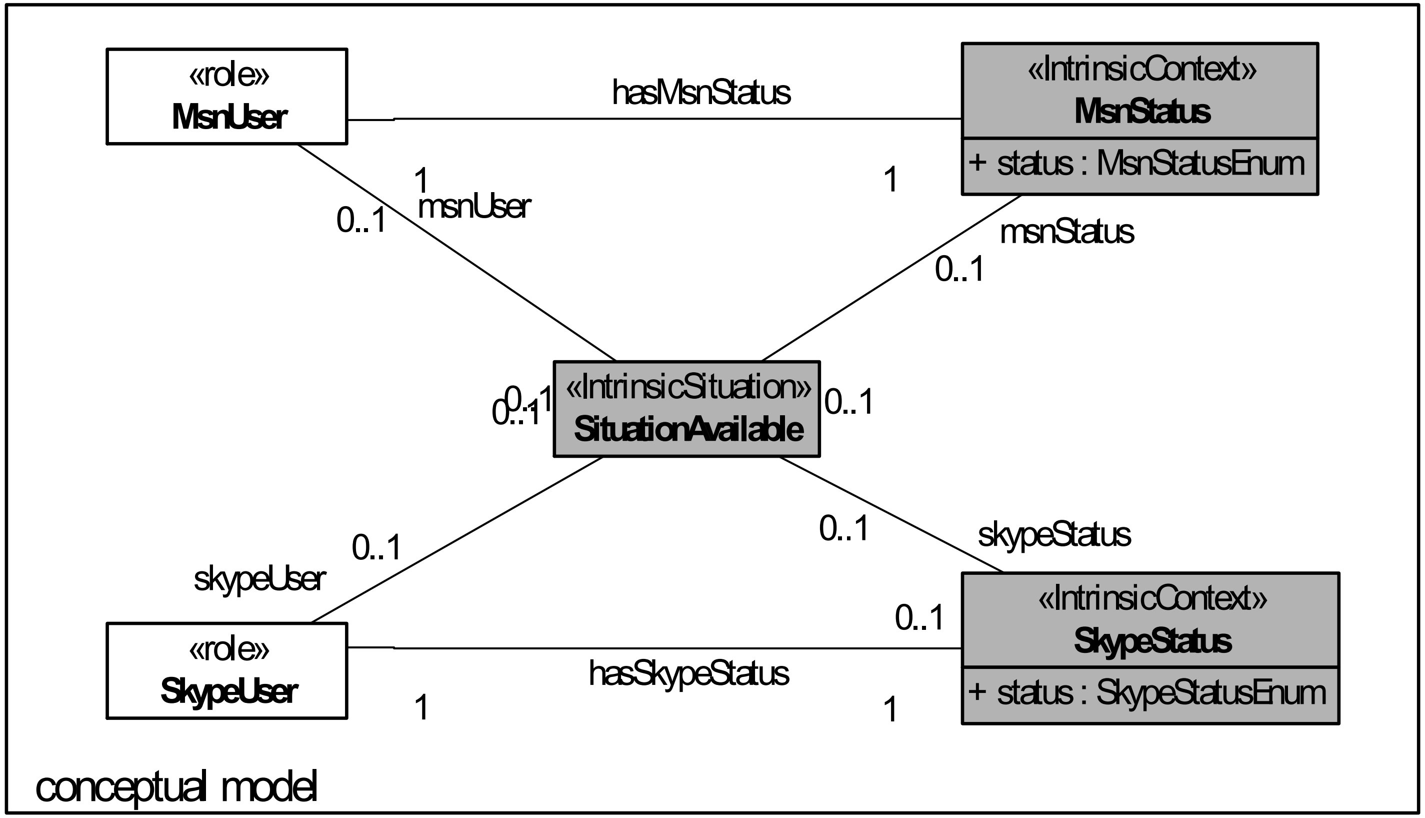


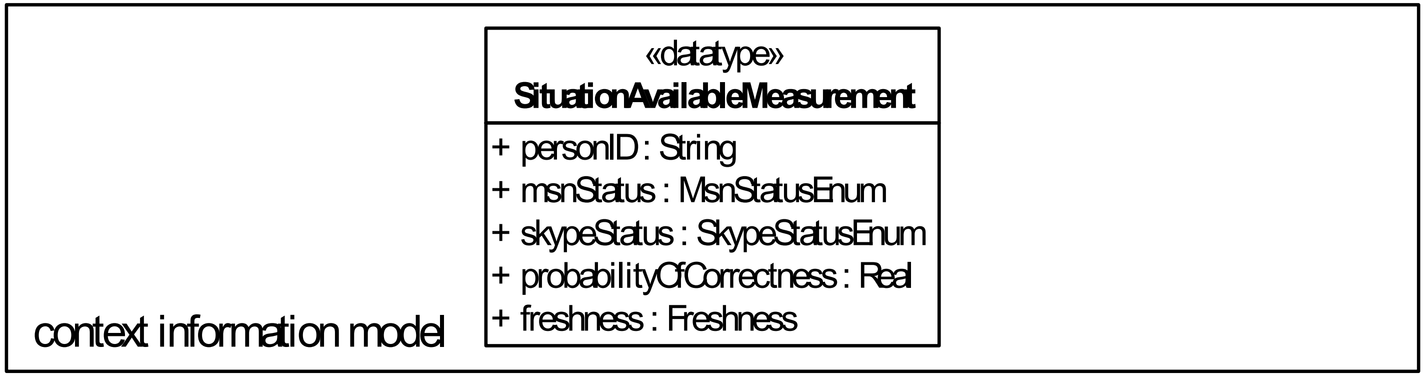



# Context Information Models



# Context Information Models

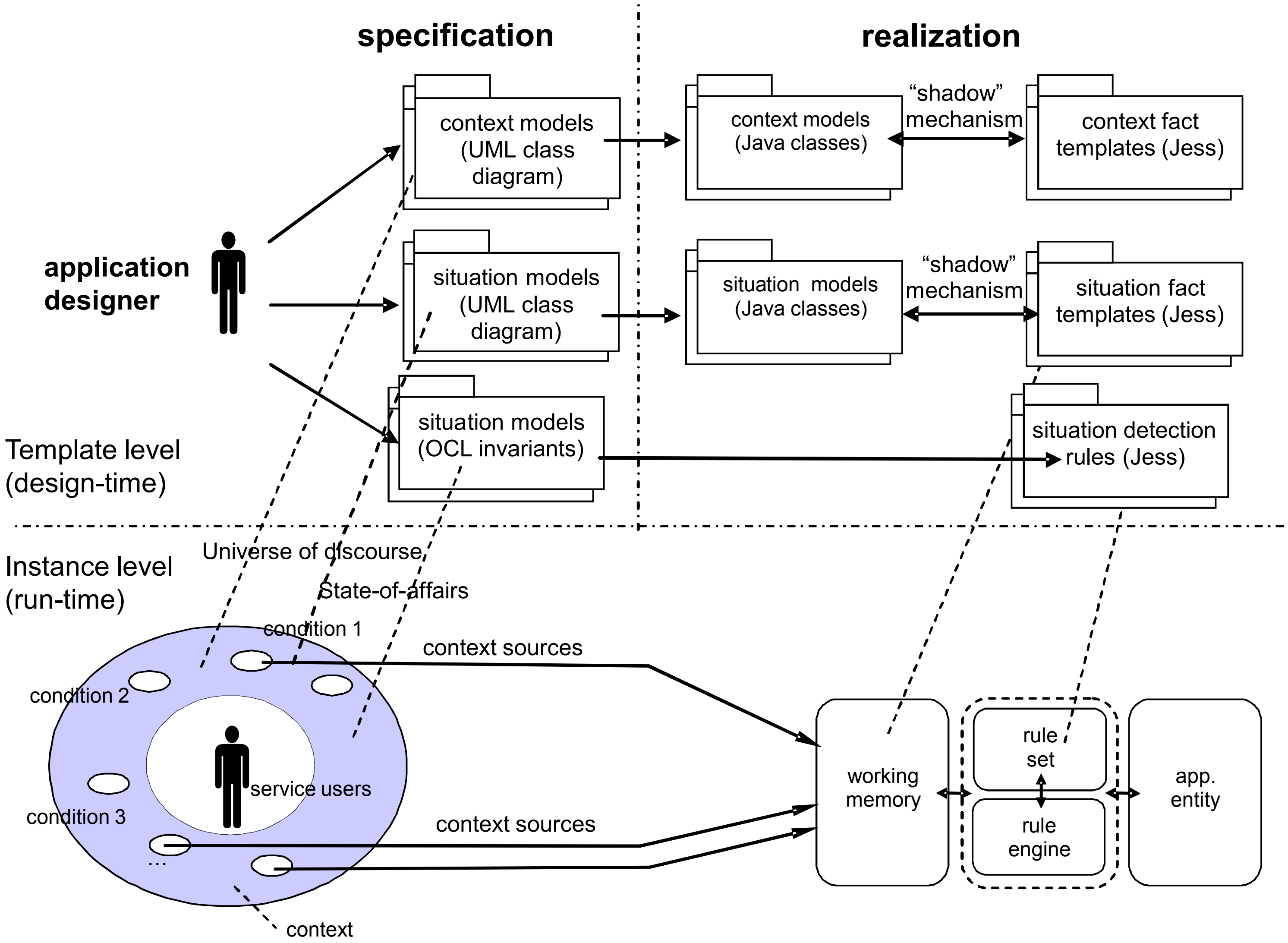


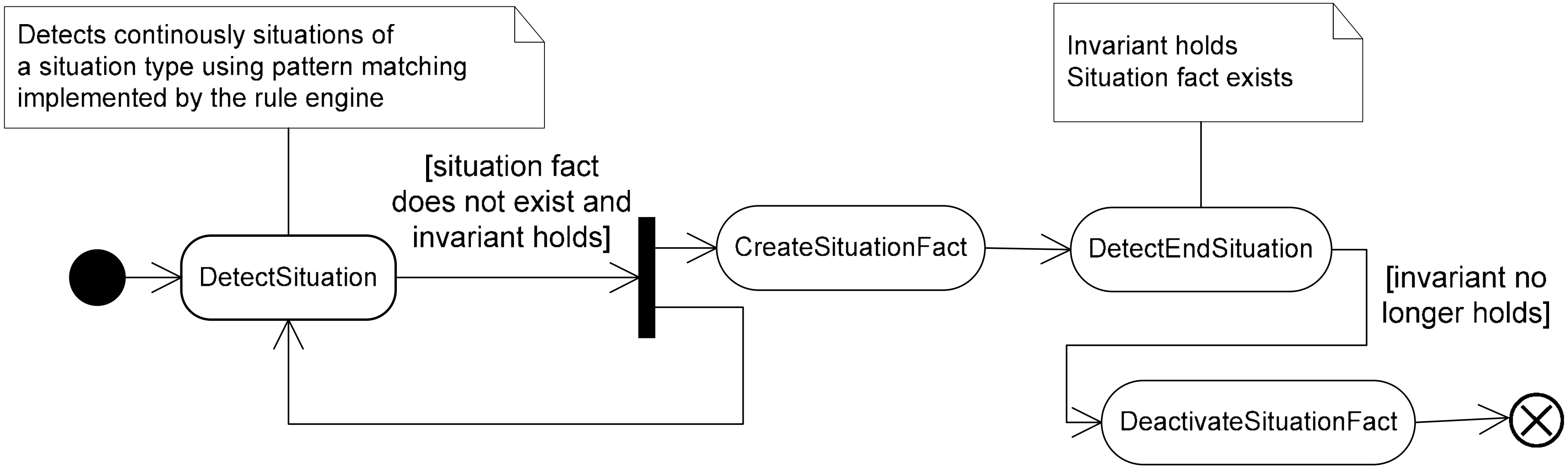
# Situation Realization

- Rule-based approach
  - Fits nicely the nature of situation detection
  - Rules (OCL invariants) are repeatedly applied to a collection of facts (context information)
- Jess
  - Shadow facts
  - Main components: working memory and rule-base

# Situation Realization (overview)



# Situation Lifecycle



# Situation Detection

| Creation Rule | Deactivation Rule |
| --- | --- |
| `(situation type invariant)`<br>`(not (situation exists))`<br>`=>`<br>`create (situation)`<br>`[RaiseEvent()]` | `(not (situation type`<br>`  invariant))`<br>`(situation exists)`<br>`=>`<br>`deactivate (situation)`<br>`[RaiseEvent()]` |

# Mappings

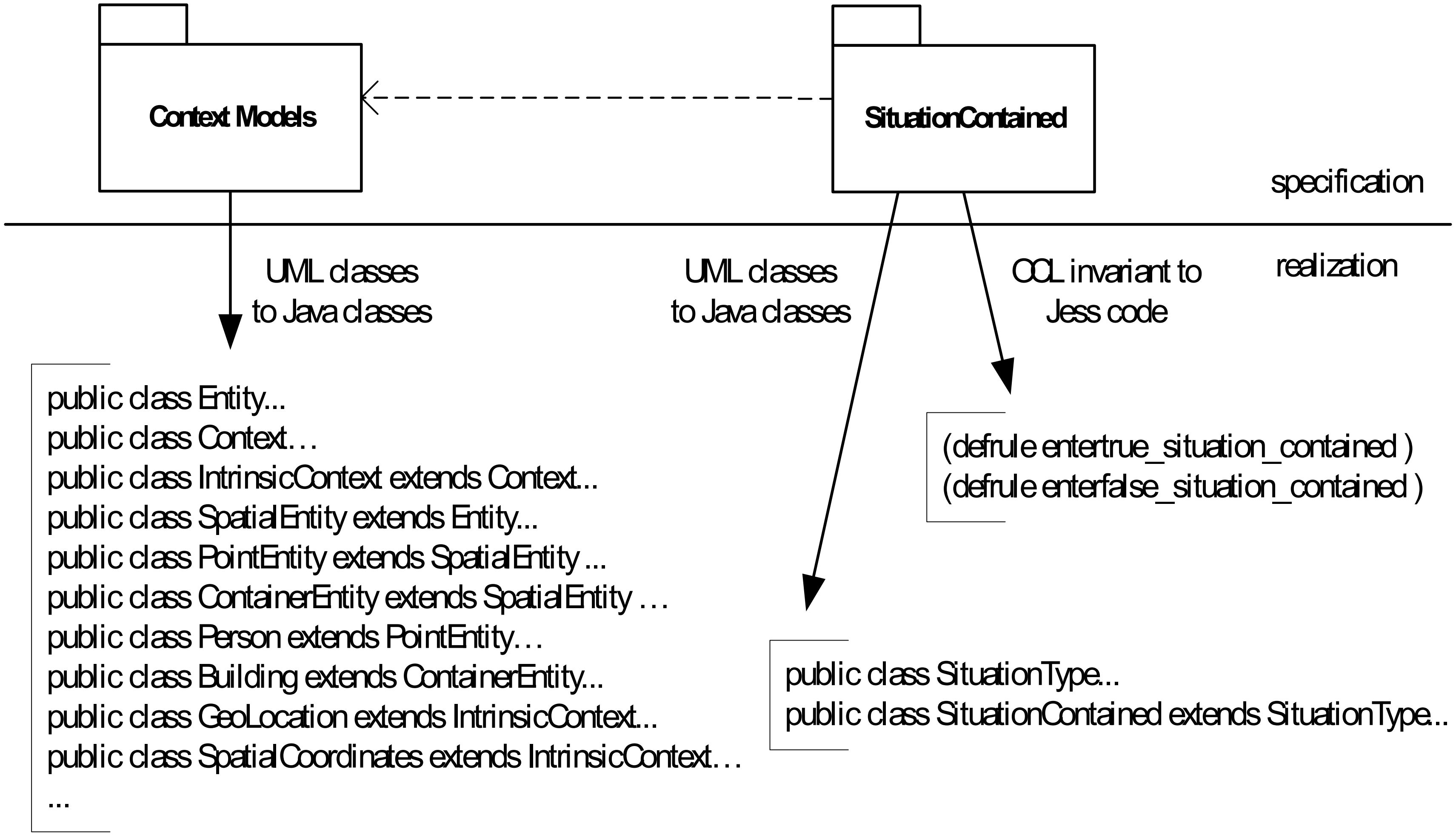
# Context Models => Java

- Octopus (www.klasse.nl/octopus/index.html)
  - Generates java code from UML classes, and statically checks OCL constraints
  - UML classes to Java classes
  - Associations to class attributes
    - One-to-one (one attribute in each class)
    - One-to-many (one of the attributes is a collection)
    - Many-to-many (both attributes are collections)
  - Subsets association

# Situation Models => Java + Jess

| OCL language | Jess language |
|---|---|
| object | (ObjectType (OBJECT ?object)) |
| object.pdatatype | (ObjectType (OBJECT ?object) (pdatatype ?pdatatype)) |
| $object^1.object^2$ | ($ObjectType^1$ (OBJECT $?object^1$) ($object^2$ $?object^2$)) |
| $object^1.object^2.object^3$ | ($ObjectType^1$ (OBJECT $?object^1$) ($object^2$ $?object^2$))<br>($ObjectType^2$ (OBJECT $?object^2$) ($object^3$ $?object^3$))<br>($ObjectType^3$ (OBJECT $?object^3$)) |
| $object^1$.datatype | ($ObjectType^1$ (OBJECT $?object^1$) (datatype ?pdatatype)) |
| $object^1$.datatype.pdatatype | ($ObjectType^1$ (OBJECT $?object^1$) (datatype ?datatype))<br>(DataType (OBJECT ? datatype) (pdatatype ?pdatatype)) |
| $object^1.object^2$. datatype. pdatatype | ($ObjectType^1$ (OBJECT $?object^1$) ($object^2$ $?object^2$))<br>($ObjectType^2$ (OBJECT $?object^2$) (datatype ?datatype))<br>(DataType (OBJECT ?datatype) (pdatatype ?pdatatype)) |
| Object->collection | (ObjectType (OBJECT ?object) (collection ?collection) |

# Context Models => Java



# EnterTrue Rule SituationContained

```
(defrule entertrue_situation_contained
  (Person (OBJECT ?person)( hasGeoLocation ? person_hasGeoLocation))
  (GeoLocation (OBJECT ?locationPerson&:(eq ?locationPerson ? person_hasGeoLocation)))
  (Building (OBJECT ?building)(geoLocation ?building_hasGeoLocation))
  (GeoLocation (OBJECT ?locationBuilding&:(eq ?locationBuilding ?building_hasGeoLocation)))
  (Building (OBJECT ?building)(spatialCoordinates ?building_hasSpatialCoordinates))
  (SpatialCoordinates (OBJECT ?spatialCoord&:(eq ?spatialCoord ?building_hasSpatialCoordinates)))
  (GeoLocation (OBJECT ?locationPerson) (location ?locationPerson_coordinates))
  (GeoLocation (OBJECT ?locationBuilding) (location ?locationBuilding_coordinates))
  (SpatialCoordinates (OBJECT ?spatialCoord) (dimension ?spatialCoord_dimension))
  (test (call context_control.SpatialDimension Containment ?locationPerson_coordinates
                                  ?locationBuilding_coordinates ?spatialCoord_dimension))
   (not (SituationContained (OBJECT ?st)(person ?person) (building ?building) (finaltime nil)))
  = >
  (bind ?SituationContained (new situation_control.SituationContained ?person ?building))
  (definstance SituationContained ?SituationContained)
)
```

# EnterFalse Rule SituationContained

```
(defrule enterfalse_situation_contained
  (not (and (Person (OBJECT ?person)( hasGeoLocation ? person_hasGeoLocation))
        (GeoLocation (OBJECT ?locationPerson&:(eq ?locationPerson ? person_hasGeoLocation)))
        (Building (OBJECT ?building)(geoLocation ?building_hasGeoLocation))
        (GeoLocation (OBJECT ?locationBuilding&:(eq ?locationBuilding ?building_hasGeoLocation)))
        (Building (OBJECT ?building)(spatialCoordinates ?building_hasSpatialCoordinates))
        (SpatialCoordinates (OBJECT ?spatialCoord&:(eq ?spatialCoord ?building_hasSpatialCoordinates)))
        (GeoLocation (OBJECT ?locationPerson) (location ?locationPerson_coordinates))
        (GeoLocation (OBJECT ?locationBuilding) (location ?locationBuilding_coordinates))
        (SpatialCoordinates (OBJECT ?spatialCoord) (dimension ?spatialCoord_dimension))
        (test (call context_control.SpatialDimension Containment ?locationPerson_coordinates
                                    ?locationBuilding_coordinates ?spatialCoord_dimension))
   (SituationContained (OBJECT ?st)(person ?person) (building ?building) (finaltime nil))
  = >
  (call ?SituationContained deactivate)
)
```

# Distribution

- Service-oriented approach: components encapsulate Jess engines, and situation information is exchange by means of the component services
- DJess: separate engines virtually share working memory. Rule engines running on different nodes can apply rules on shared facts

# Conclusions

- Context models help understanding context concepts and how they relate to each other
- Context models are static
- Situations allow one to define state-of-affairs of concern for context-aware applications
- Behaviours can be defined in terms of how the system evolves from situation to situation!
- Situations can be used to define conditions that trigger a rule system, as, e.g., in ECA rules
- Situations can be composed of situations themselves
  - modularization of the situation models, improving organization and reuse of situation specifications

# Conclusions (2)

- Situation realization is rule-based
  - Allows attentive situation detection as opposed to query-based solutions
- Model-driven approach
  - Specification elements are systematically mapped to realization elements
  - UML as a mature technology in model-driven developments