Mapping Context-Dependent Requirements to Event-Based Context-Oriented Programs for Modularity

Tetsuo Kamina (UTokyo)
Tomoyuki Aotani (Tokyo Tech)
Hidehiko Masuhara (Tokyo Tech)

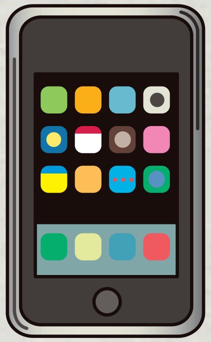
Purpose

- Methodology for context-aware systems
  - from requirements to implementation
  - Context-dependent behavior
    - well-studied in implementation
    - identification of contexts and behavioral variations is not trivial

Requirements model and systematic implementation using event-based COP language EventCJ

Context-awareness

- Capability of a system to behave w.r.t. surrounding contexts (outdoors, indoors)



Map : City map, Floor plan
Positioning : GPS, RFID

- Multiple parts of behavior simultaneously change at runtime

Context-awareness

- Capability of a system to behave w.r.t. surrounding contexts (outdoors, indoors)

Outdoors

Map : City map, Floor plan

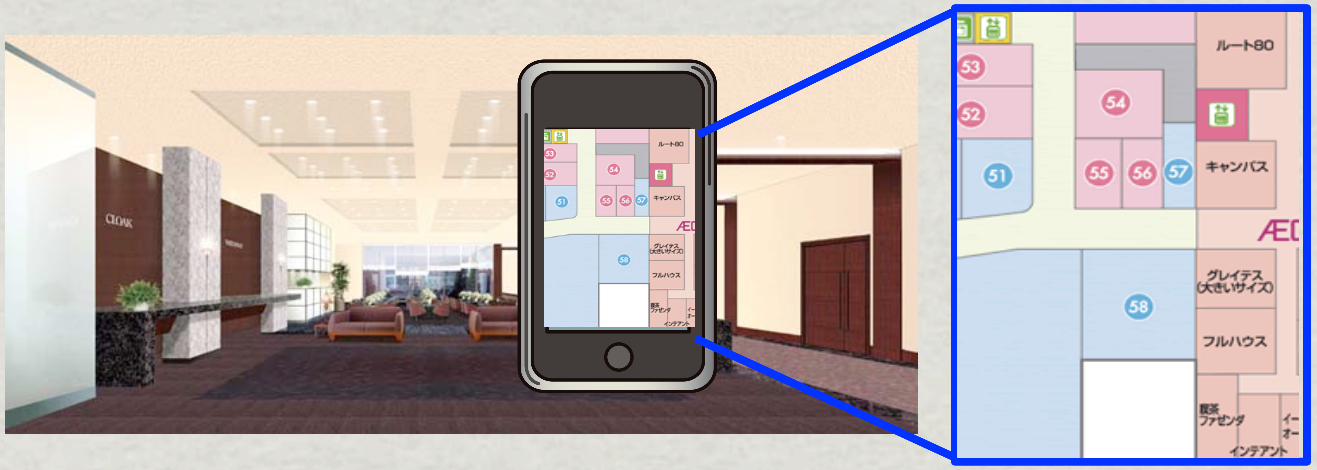
Positioning : GPS, RFID

- Multiple parts of behavior simultaneously change at runtime

Context-awareness

- Capability of a system to behave w.r.t. surrounding contexts (outdoors, indoors)

Indoors

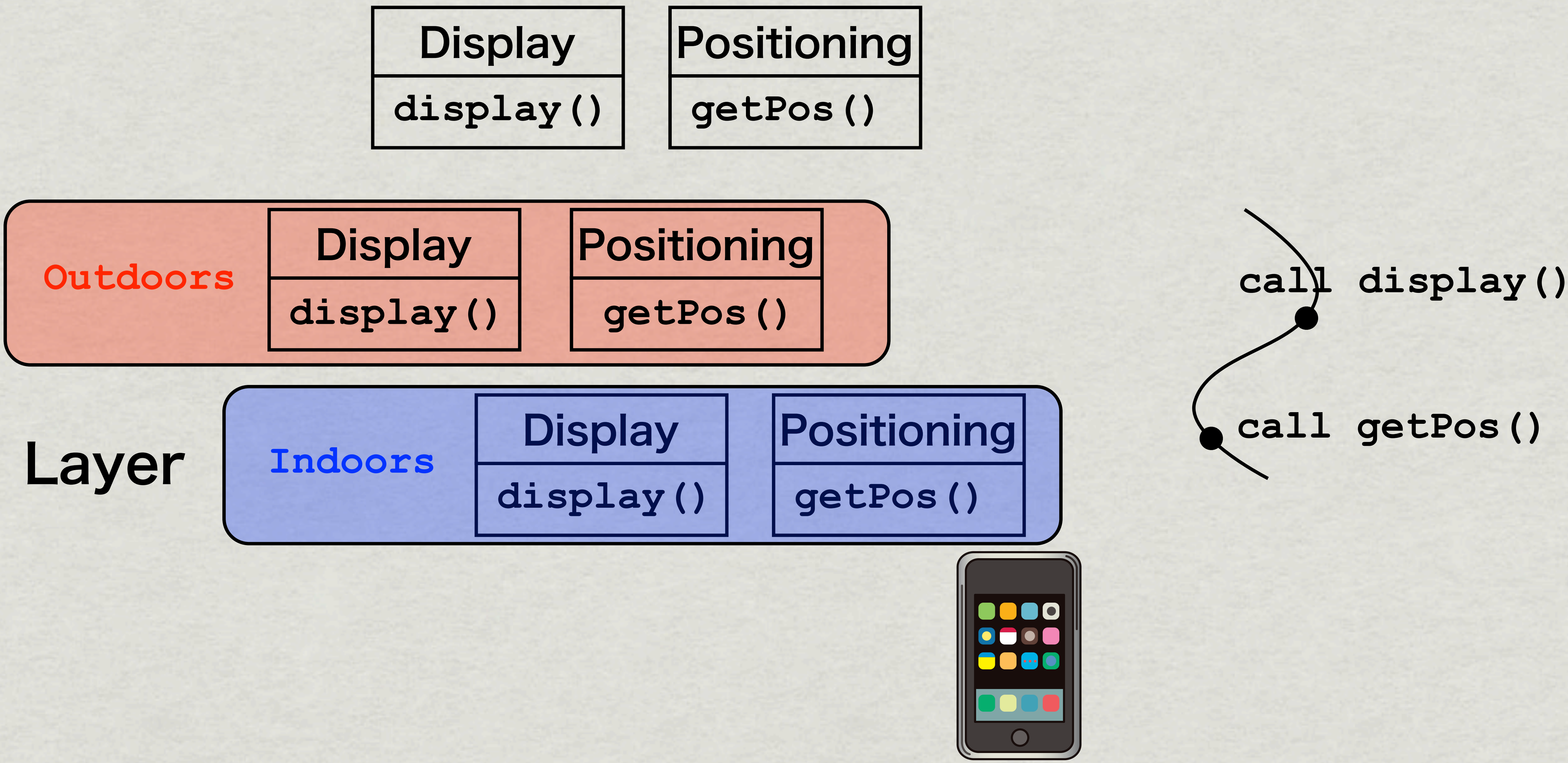


Map : City map, Floor plan
Positioning : GPS, RFID

- Multiple parts of behavior simultaneously change at runtime

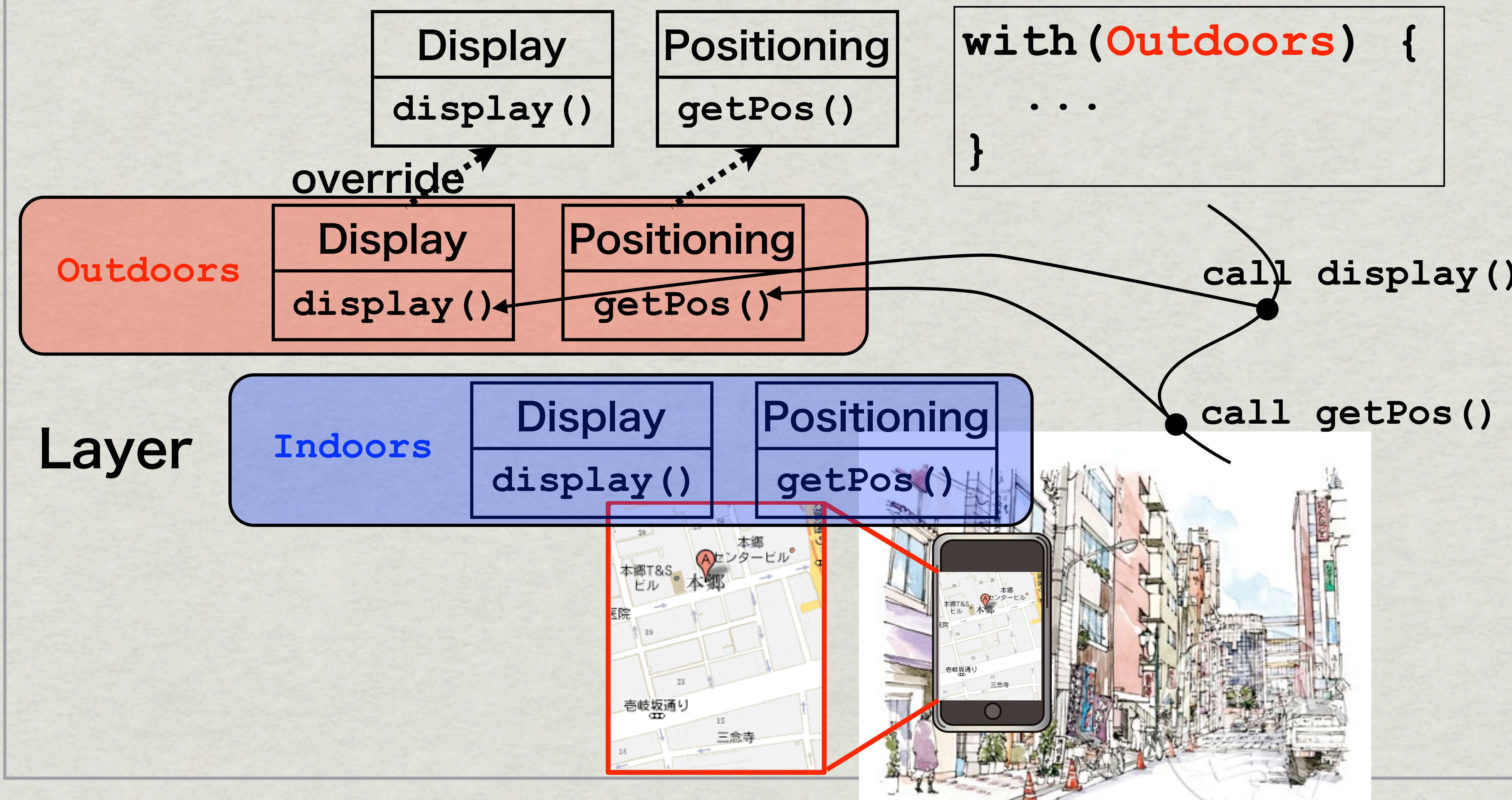
Context-Oriented Programming (COP) [Hirschfeld08]

- modularization of context dep. behavior: **layer**
- disciplined activation of layers



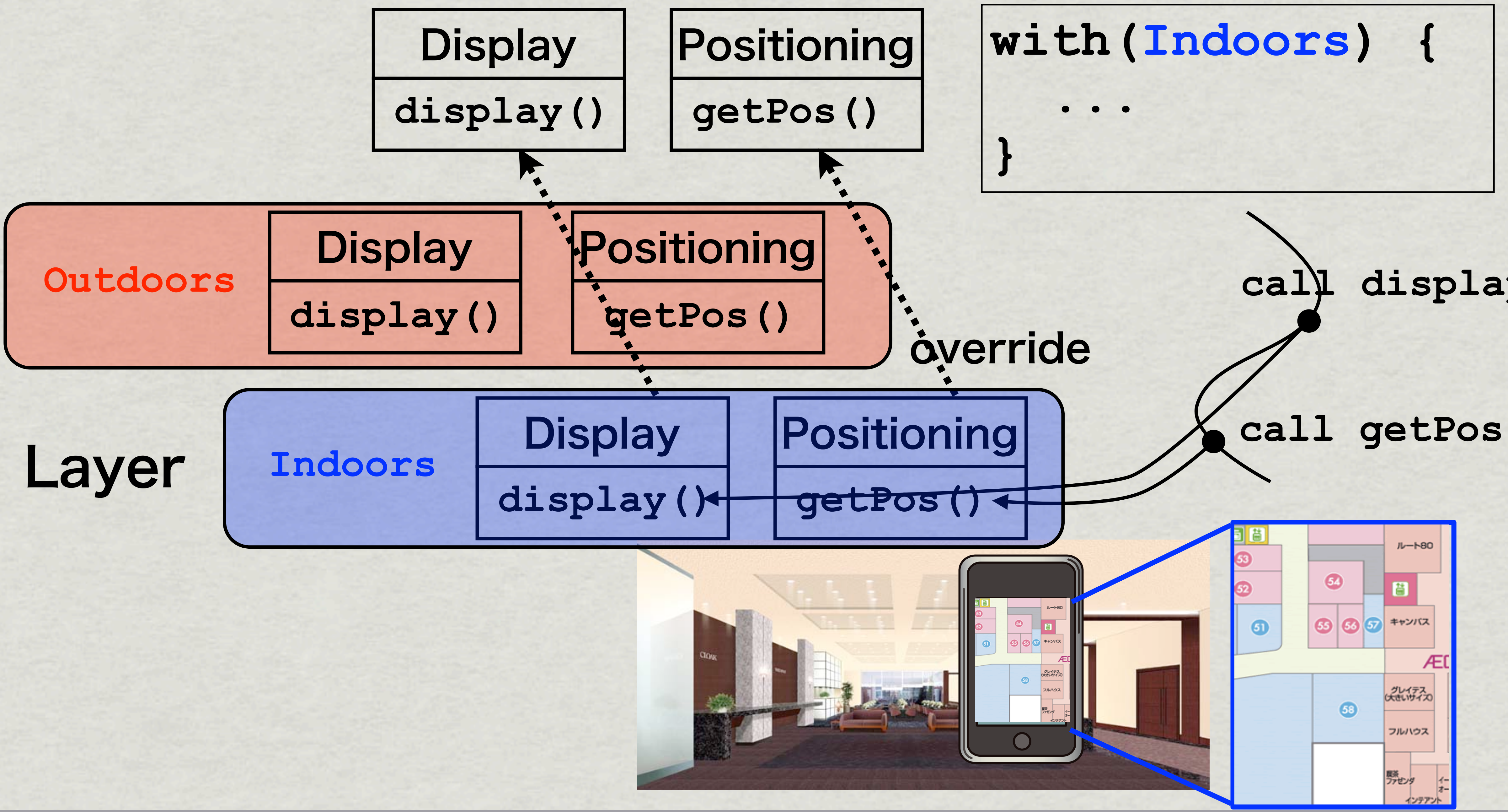
Context-Oriented Programming (COP) [Hirschfeld08]

- modularization of context dep. behavior: **layer**
- disciplined activation of layers



Context-Oriented Programming (COP) [Hirschfeld08]

- modularization of context dep. behavior: **layer**
- disciplined activation of layers



We need to identify:

- Variations of behavior that should be implemented using a layer
- Context that changes behavior
  - A layer assumes a context

    `Outdoors` is active when *the situation is outdoors*

    Layer Context

- Timing when contexts/layers change

We need to identify:

- Variations of behavior that should be implemented using a layer
- Context that changes behavior
  - A layer assumes a context

  `Outdoors` is active when *the situation is outdoors*

  Layer Context
- Timing when contexts/layers change

Do we really know them?

Questions

- When to use layers?
  - the ways (layers, design patterns, **`if`**) affect modularity
- What are contexts?
  - Can a layer always assume only one single context?
  - How relations b/w contexts and layers are defined?
- How can precisely specify when context changes?

Questions

- When to use layers?
  - the ways (layers, design patterns, `if`) affect modularity
- What are contexts?
  - Can a layer always assume only one single context?
  - How relations b/w contexts and layers are defined?
- How can precisely specify when context changes?

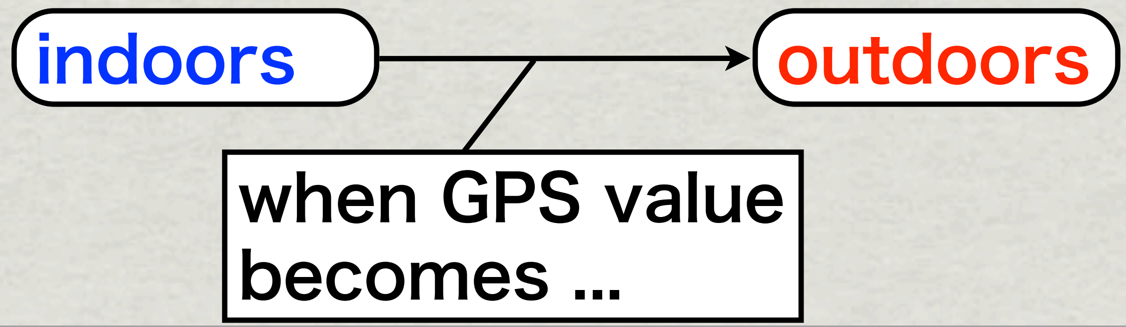
Methodology is required

Overview of methodology

- Identifying **contexts** and **context-dependent use cases**

- Identifying **layers** by grouping use cases

- Identifying **events** that trigger context changes



Example use cases

Pedestrian Navigation System:

- If the user is outdoors, it displays a city map using GPS based positioning
- If the user is indoors, it displays a floor plan using Wi-Fi based positioning
- If the floor plan is not available, it displays a city map
- If no positioning is available, it displays a static map and showing an alert message

Identifying contexts

- We identify contexts from behavior
  - Documents describing system-to-be (e.g. use cases)
  - Prototypes

- Conditions are candidates for contexts
  - If the use is outdoors, the system displays a city map
  - If the use is indoors, the system displays a floor plan
  - If the floor plan is not available, the system displays a city map
  - If no positioning is available, the system displays a static map

※conditions affecting a number of parts
(e.g., external environmental conditions)

Identifying contexts

- We identify contexts from behavior
  - Documents describing system-to-be (e.g. use cases)
  - Prototypes

- Conditions are candidates for contexts
  - If the use is outdoors, the system displays a city map
  - If the use is indoors, the system displays a floor plan
  - If the floor plan is not available, the system displays a city map
  - If no positioning is available, the system displays a static map

candidates

※conditions affecting a number of parts (e.g., external environmental conditions)

Defining contexts

- We define a context in terms of variables
  - outdoors/indoors situations are merged

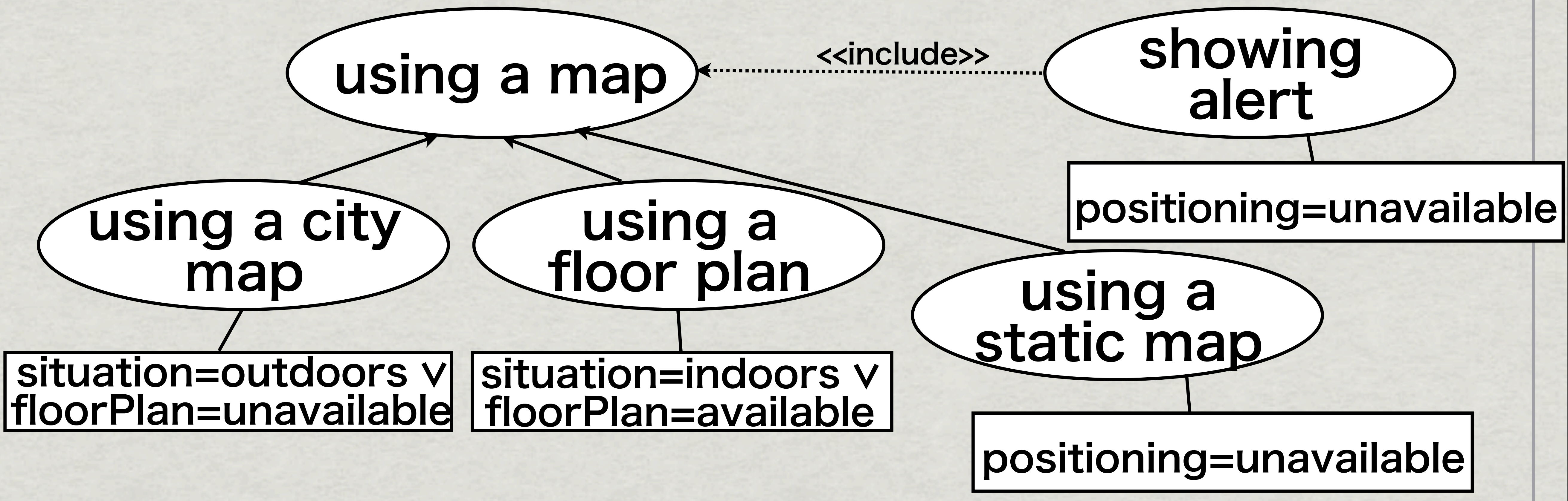
name	value
situation	outdoors, indoors
floorPlan	available, unavailable
positioning	available, unavailable

  - A context is a specific setting of value to a variable (a Boolean term)

    e.g. situation=outdoors

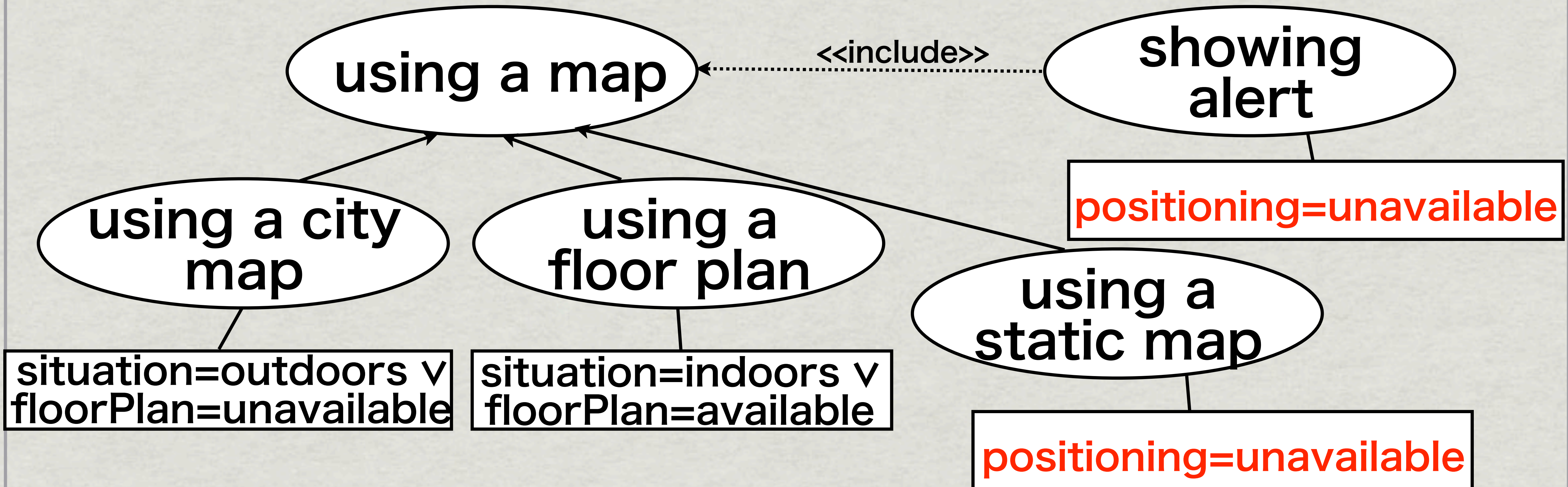
Context-dependent use cases

- Defining context-dependent use cases
  - a specialization of use case applicable in specific contexts
  - Annotated with proposition of contexts



Identifying layers

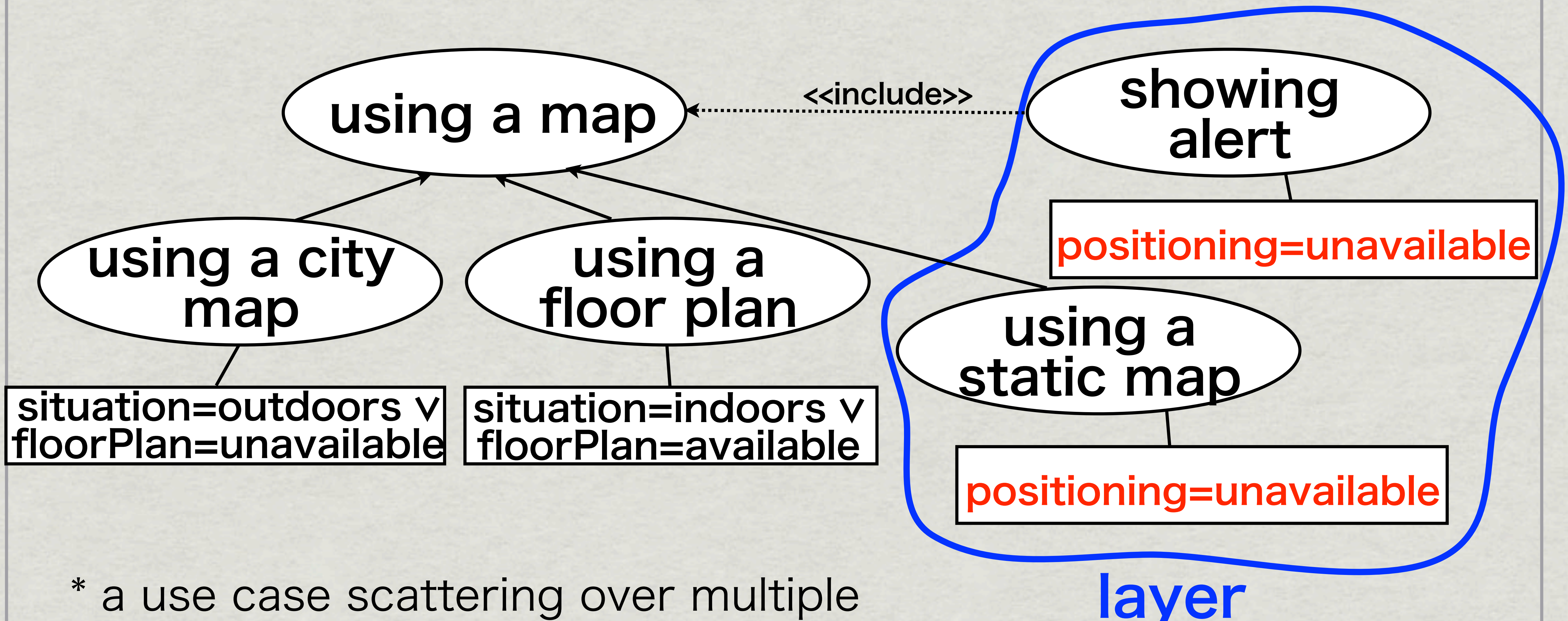
- Layer: a set of use cases with the same proposition



* a use case scattering over multiple objects may also be identified as a layer (cf. Jacobson, 2005)

Identifying layers

- Layer: a set of use cases with the same proposition



* a use case scattering over multiple objects may also be identified as a layer (cf. Jacobson, 2005)

To identify events...

Contexts are abstract in use cases

- We need to decompose context into more specific states of the machine (sensors)
  - State changes are identified as events

Decomposing contexts

- Detailed specification consists of sensors (GPS, Wi-Fi) and external entities (floor plan)
- Some contexts depend on multiple sensors

context	detailed context specification
situation=outdoors	GPS=over the criterion value
situation=indoors	GPS=under the criterion value
floorPlan=available	The floor plan service exists
floorPlan=unavailable	The floor plan service does not exist
positioning=available	GPS=on or Wi-Fi=connected
positioning=unavailable	GPS=off and Wi-Fi=disconnected

Identifying events

- Specifying how/when the status of detailed context specification changes

event	how	when
StrongGPS	GPS=under the criterion → GPS=over the criterion	the GPS signal value becomes over XXX
GPSEvent	GPS=off → GPS=on	the GPS device is becoming on
WifiEvent	Wi-Fi=disconnected → Wi-Fi=connected	the Wi-Fi device is connected …

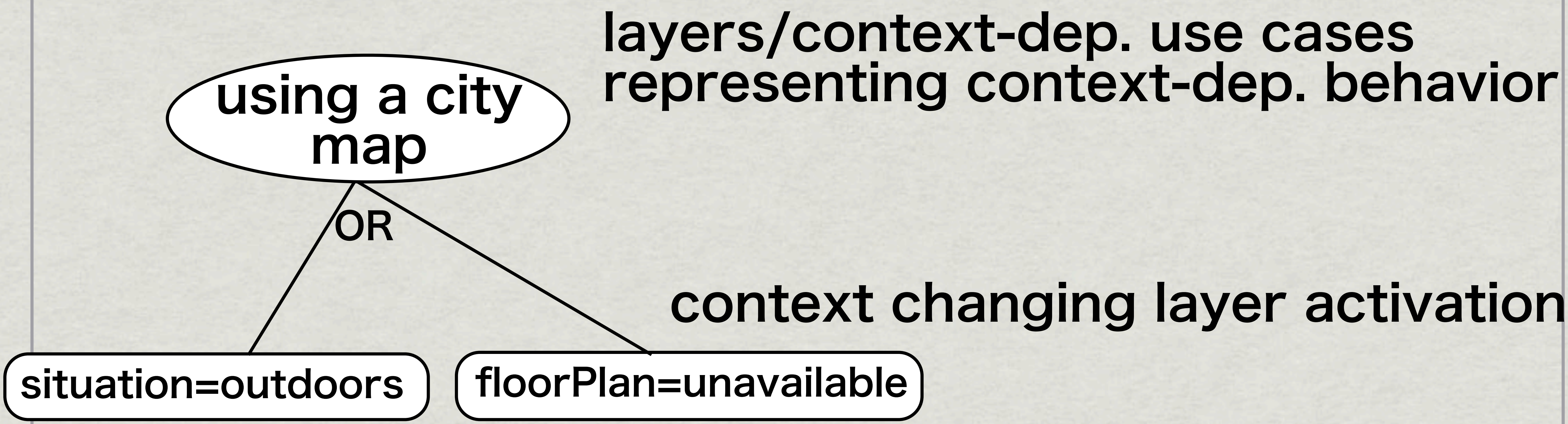
We have obtained so far..

We have obtained so far..

using a city map

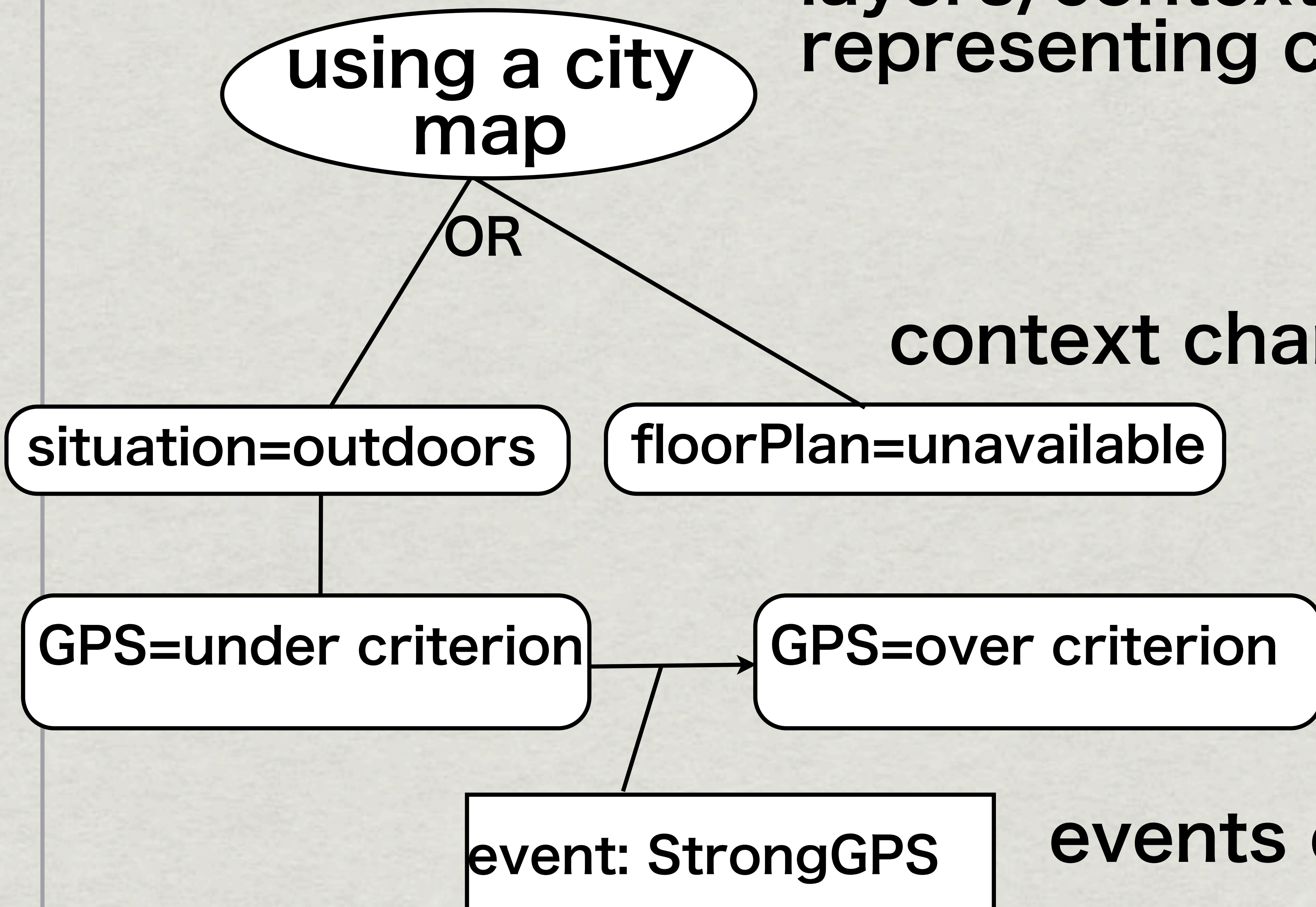
layers/context-dep. use cases
representing context-dep. behavior

We have obtained so far..



We have obtained so far..

layers/context-dep. use cases representing context-dep. behavior

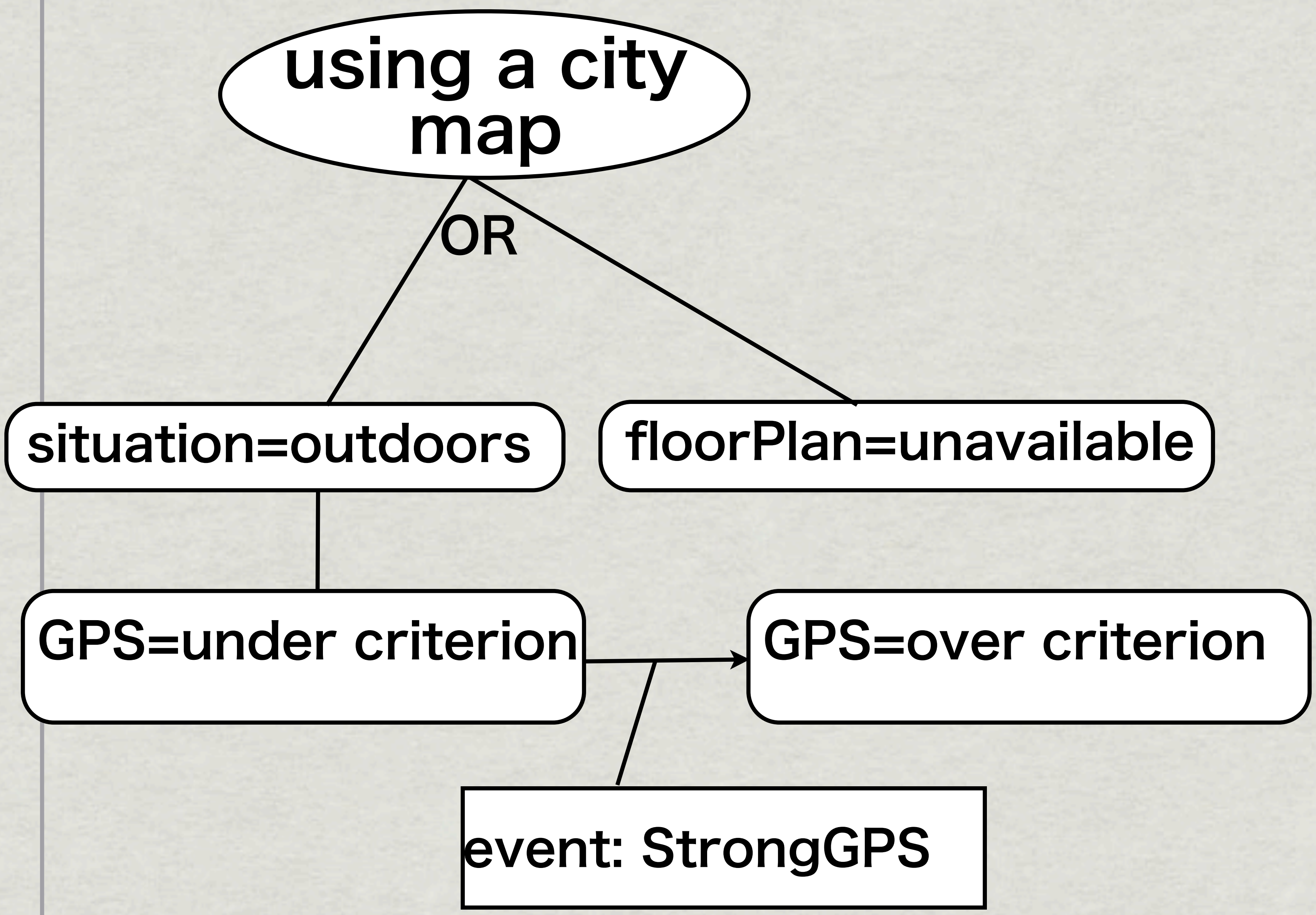


context changing layer activation

events changing contexts

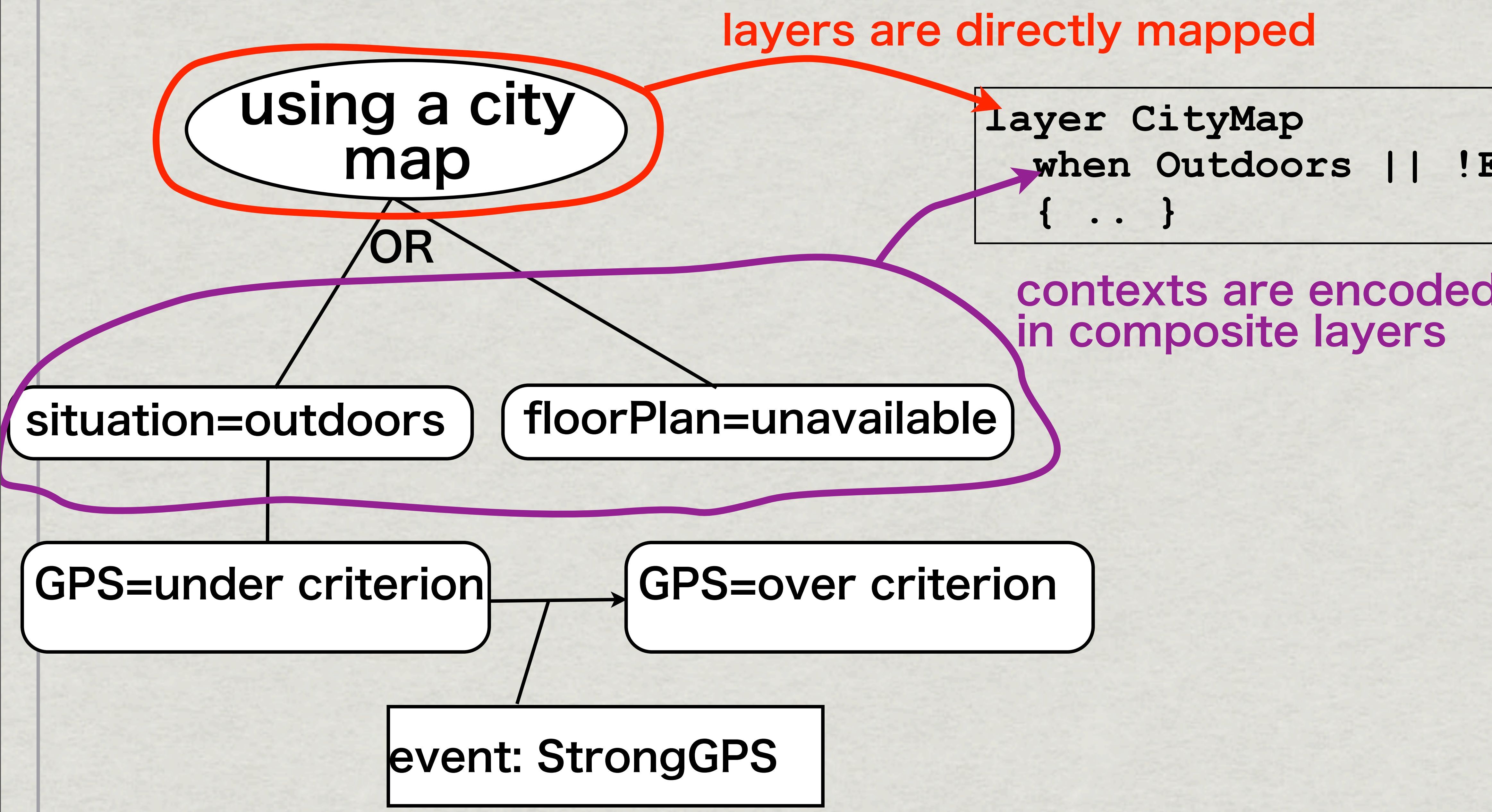
Translating to implementation

Translating specifications to corresponding constructs in EventCJ[Kamina11]



Translating to implementation

Translating specifications to corresponding constructs in EventCJ[Kamina11]

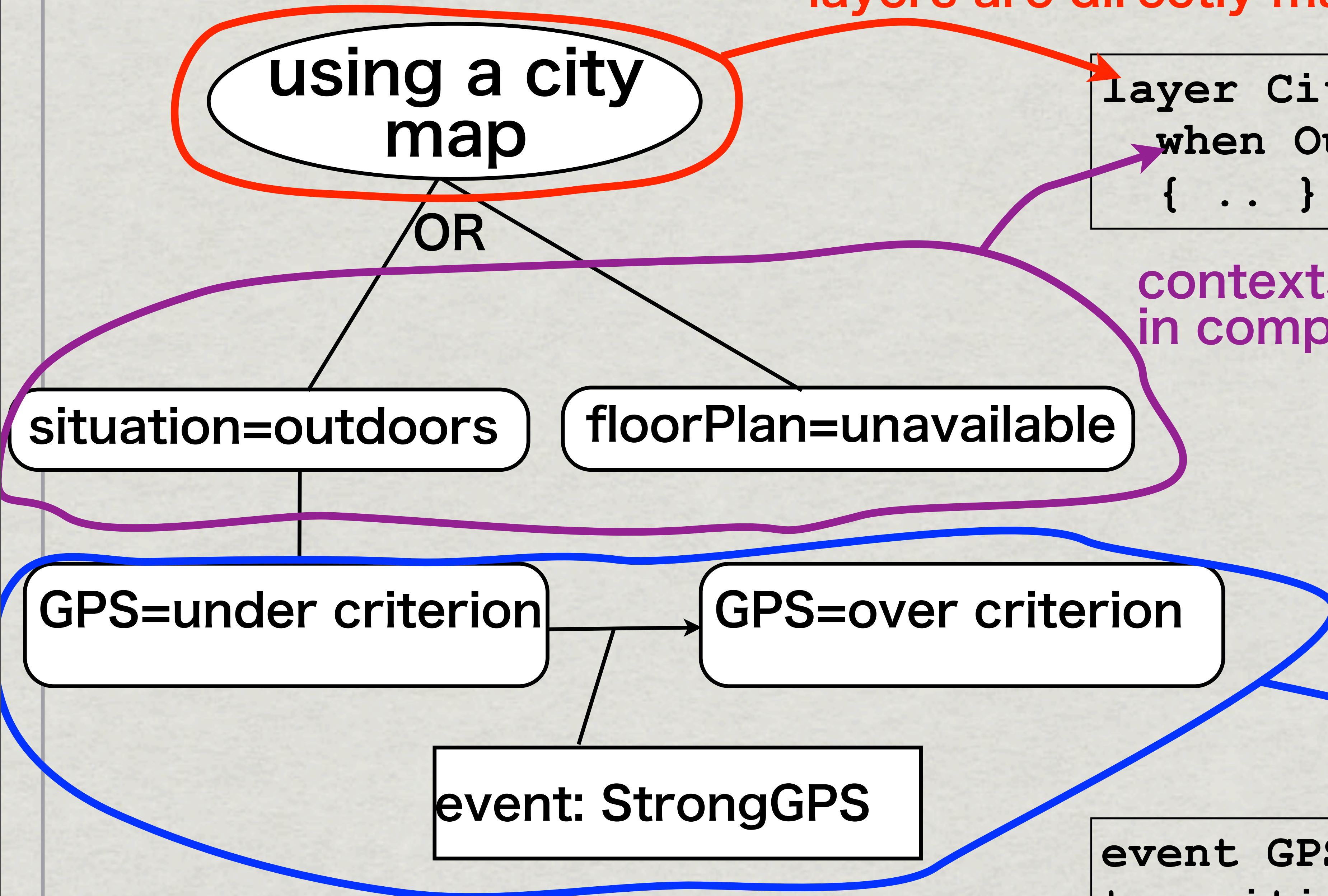



```
layer CityMap
  when Outdoors || !FPExists
  { .. }
```

Translating to implementation

Translating specifications to corresponding constructs in EventCJ[Kamina11]

layers are directly mapped




```
layer CityMap
  when Outdoors || !FPExists
  { .. }
```

contexts are encoded in composite layers

events are encoded in layer transition rules

```
event GPSEvent ...
transition StrongGPS:
  !Outdoors ? -> Outdoors;
```

EventCJ: event-based layer transition

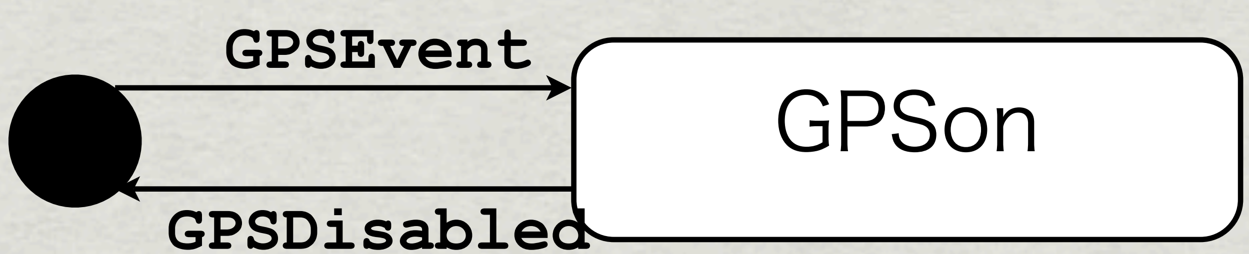
- Layer switching is triggered by events

```
event GPSEvent(Navigation n)
   :after call(void *.onStatusChanged())
     && target(n) && if(GPS.isAvailable())
   :sendTo(n);
```

  - Specifying when to generate events using AspectJ-like pointcut language

- Layer switching is specified by rules

```
transition GPSEvent: !GPSon ? -> GPSon
```

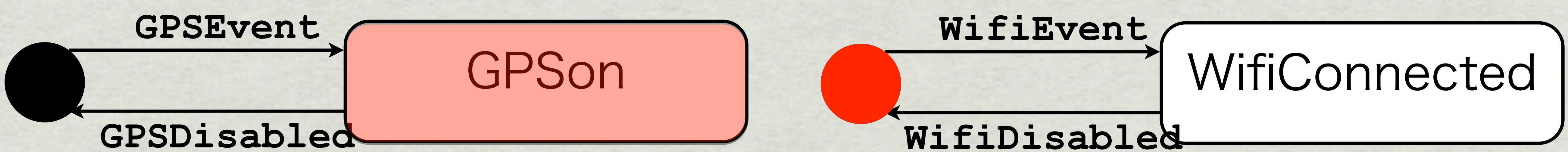


EventCJ: composite layers

[Kamina13]

- Composite layers are implicitly activated when the condition on other layers holds

```
layer StaticMap when !GPSon && !WifiConnected {
  /* static map functions */
}
```



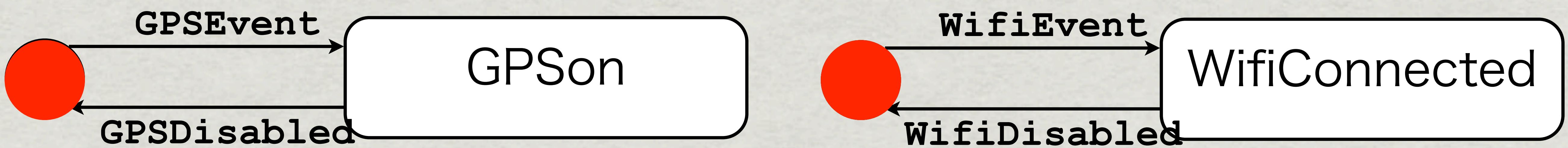
`StaticMap` is inactive

EventCJ: composite layers

[Kamina13]

- Composite layers are implicitly activated when the condition on other layers holds

```
layer StaticMap when !GPSon && !WifiConnected {
  /* static map functions */
}
```



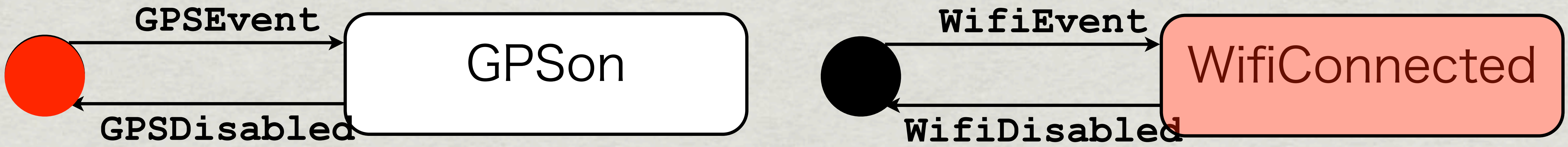
`StaticMap` is active

EventCJ: composite layers

[Kamina13]

- Composite layers are implicitly activated when the condition on other layers holds

```
layer StaticMap when !GPSon && !WifiConnected {
  /* static map functions */
}
```



`StaticMap` is inactive

Discussion

- Systematic identification of context-related requirements
  - Use cases: useful tool to find contexts
  - Identification of layers
  - Stepwise elicitation of events
- Translation preserves separation of concerns
- More sophisticated case studies are in paper
  - Conference guide system
  - Program editor

Related work

- Jacobson's AOSD (2005)
  - Use case driven methodology
  - A use case scattering multiple classes is implemented by an aspect
  - Mapping "extension points" in use cases to pointcuts in AspectJ
  - Dynamic deployment of behavior is not discussed
- Requirements engineering [Salifu07, Sutcliffe06, Lapouchnian09]
  - Focusing only on requirements variability
  - Lacks viewpoint of detailed context specification
  - Lacks viewpoint of modular implementation

Conclusions

- Use case driven methodology for developing context-aware systems
- Organizing requirements specifications
  - Identifying contexts from behavior
  - Classifying variations of behavior
    - Identification of layers in use cases
  - Stepwise elicitation of details of contexts
- Systematic implementation preserving SoC