A Declarative Approach For Generating Software Frameworks Dedicated To Ubiquitous Computing

Wilfried Jouve

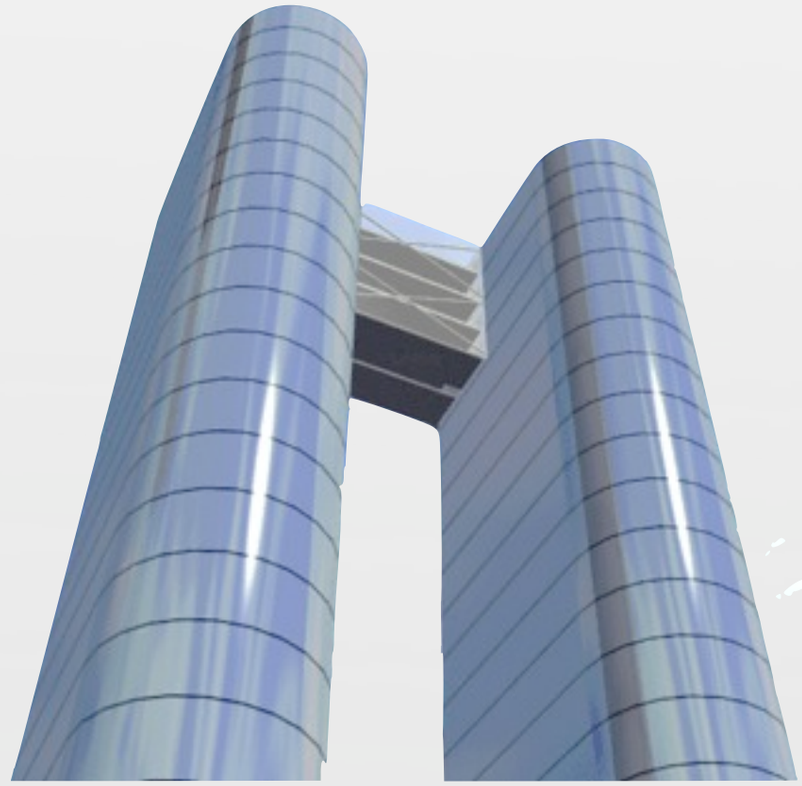
Advisor: Charles Consel

INRIA Bordeaux, Phoenix Research Group
University of Bordeaux 1

8th of April 2009

Ubiquitous Applications

Automation of everyday activities



Requirements

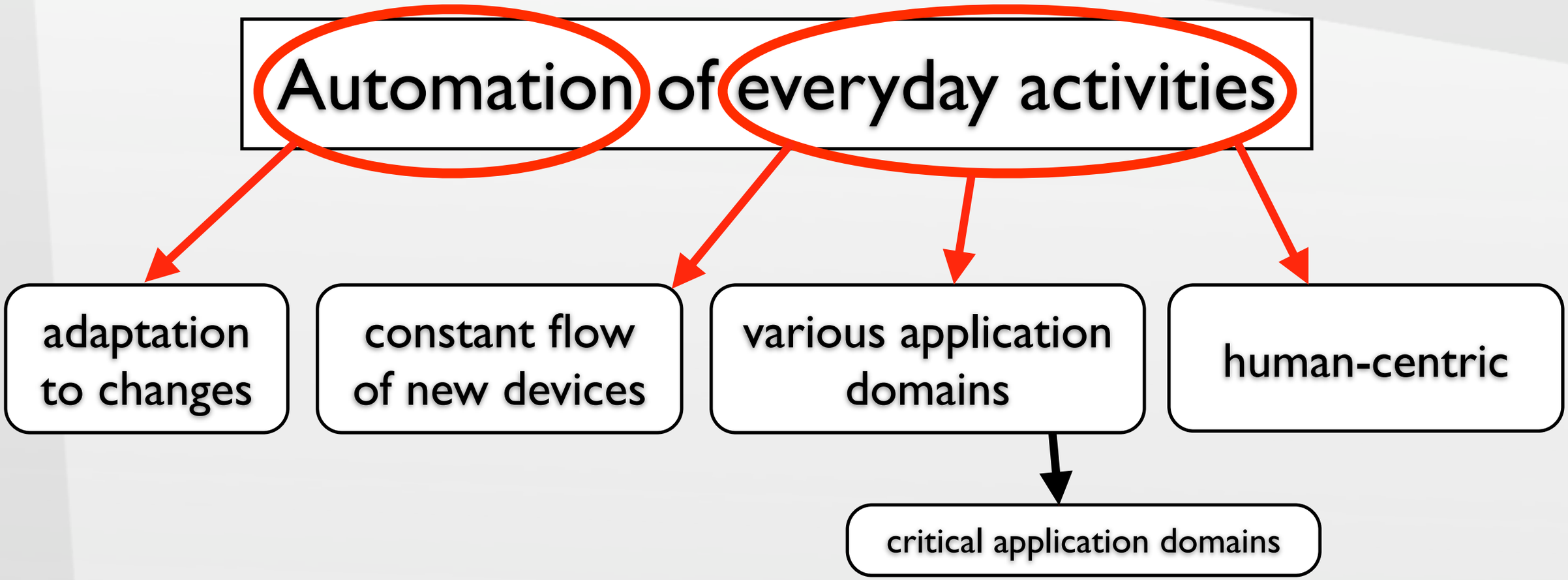
Automation of everyday activities

Requirements

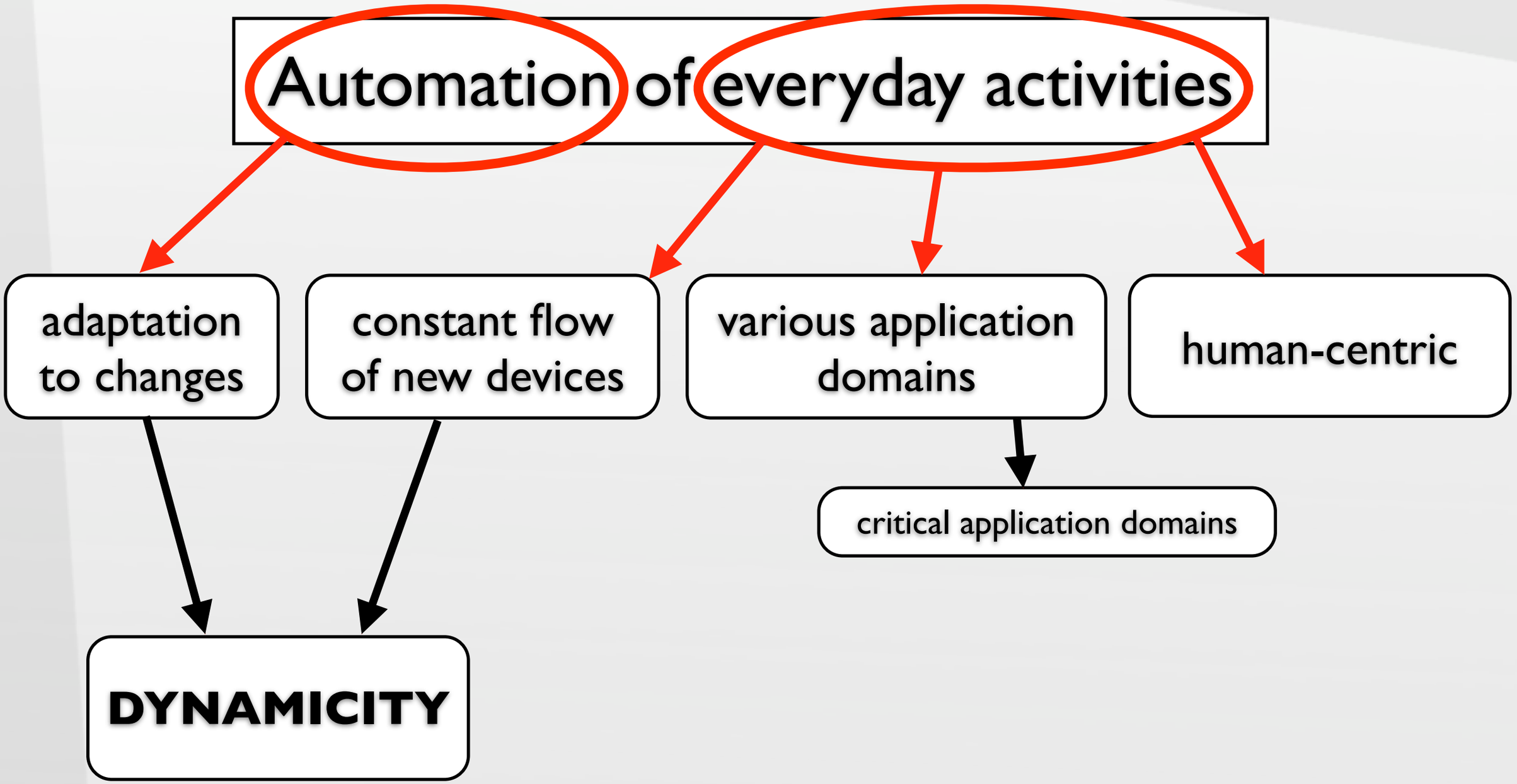
Automation of everyday activities

adaptation to changes

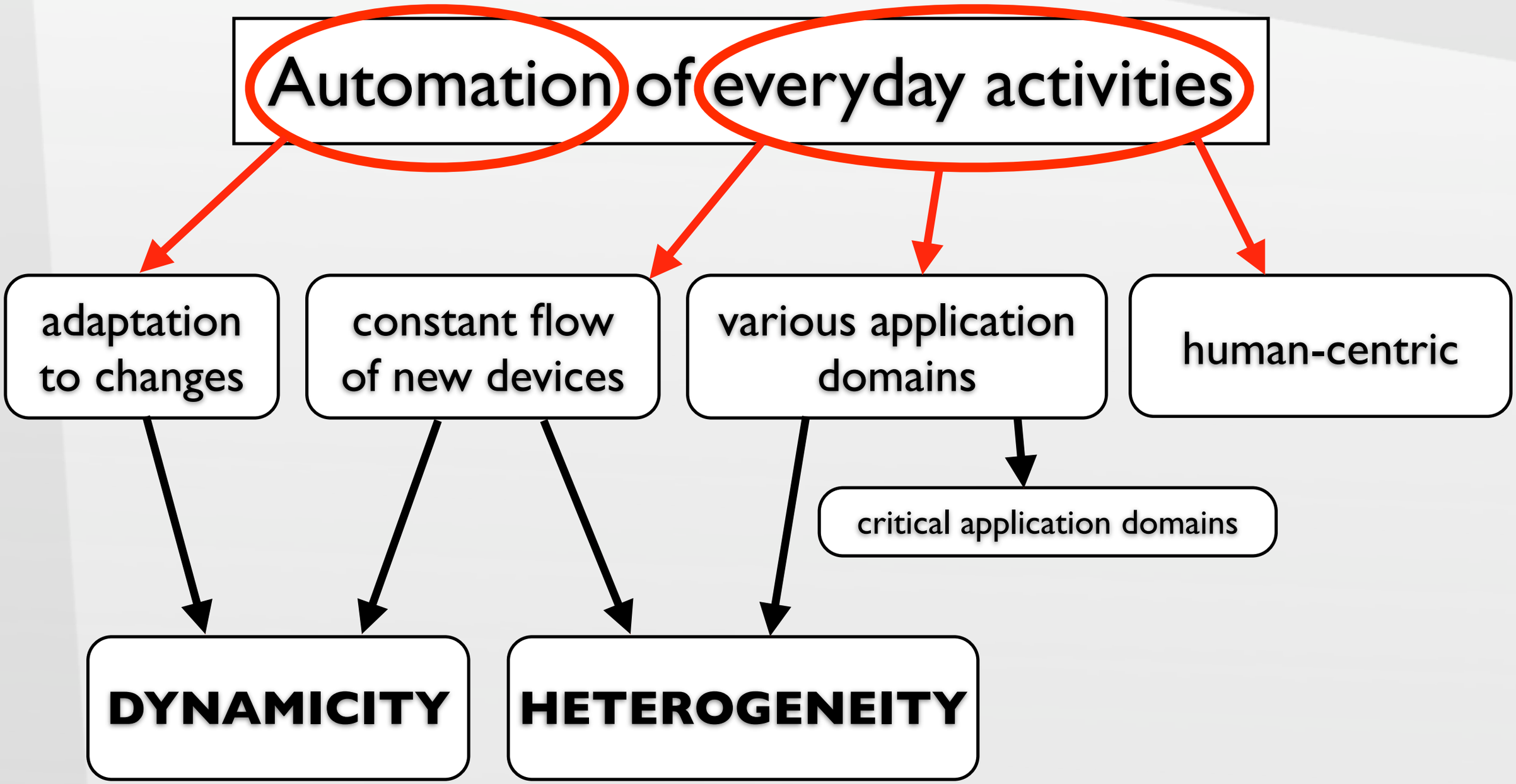
Requirements



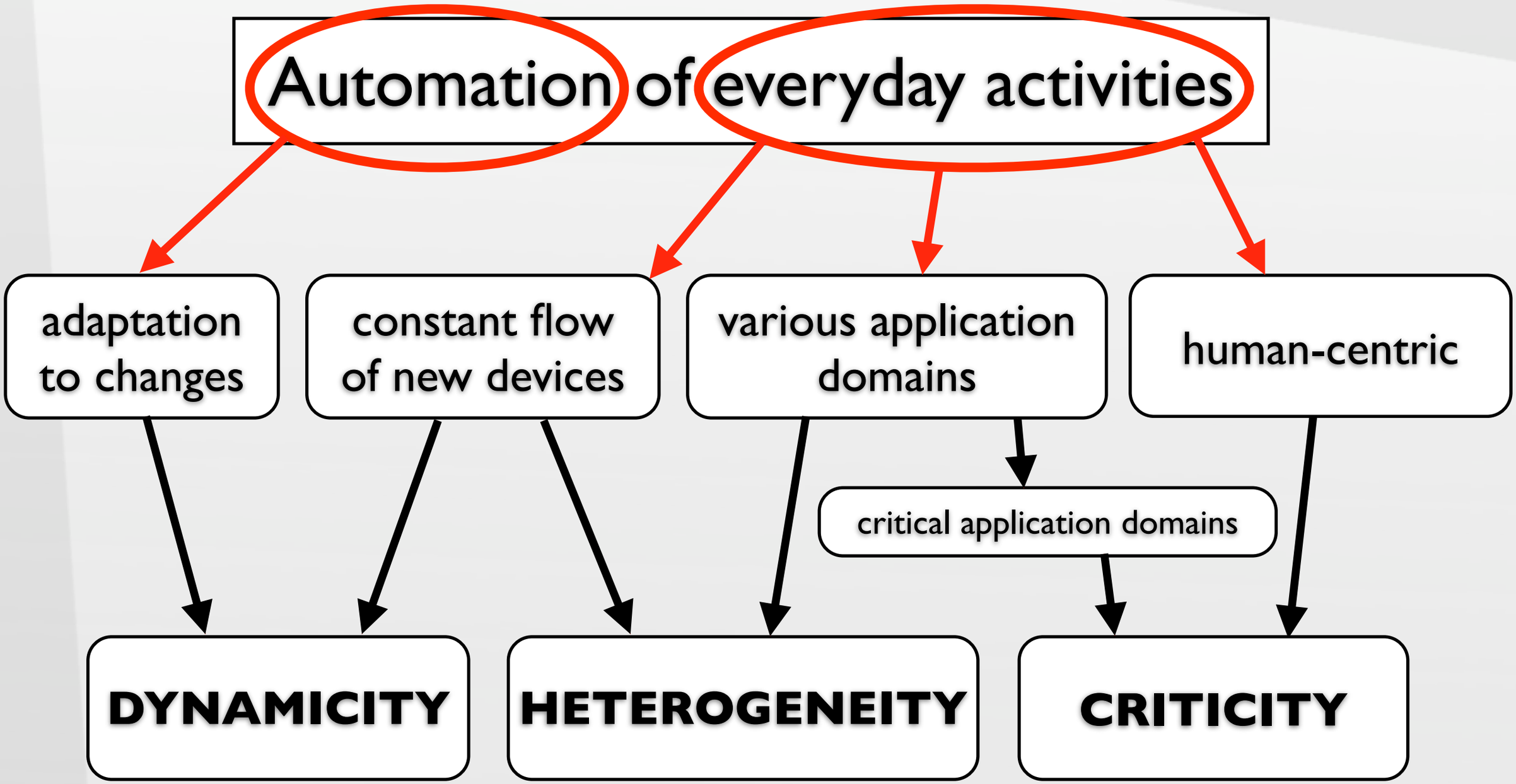
Requirements



Requirements

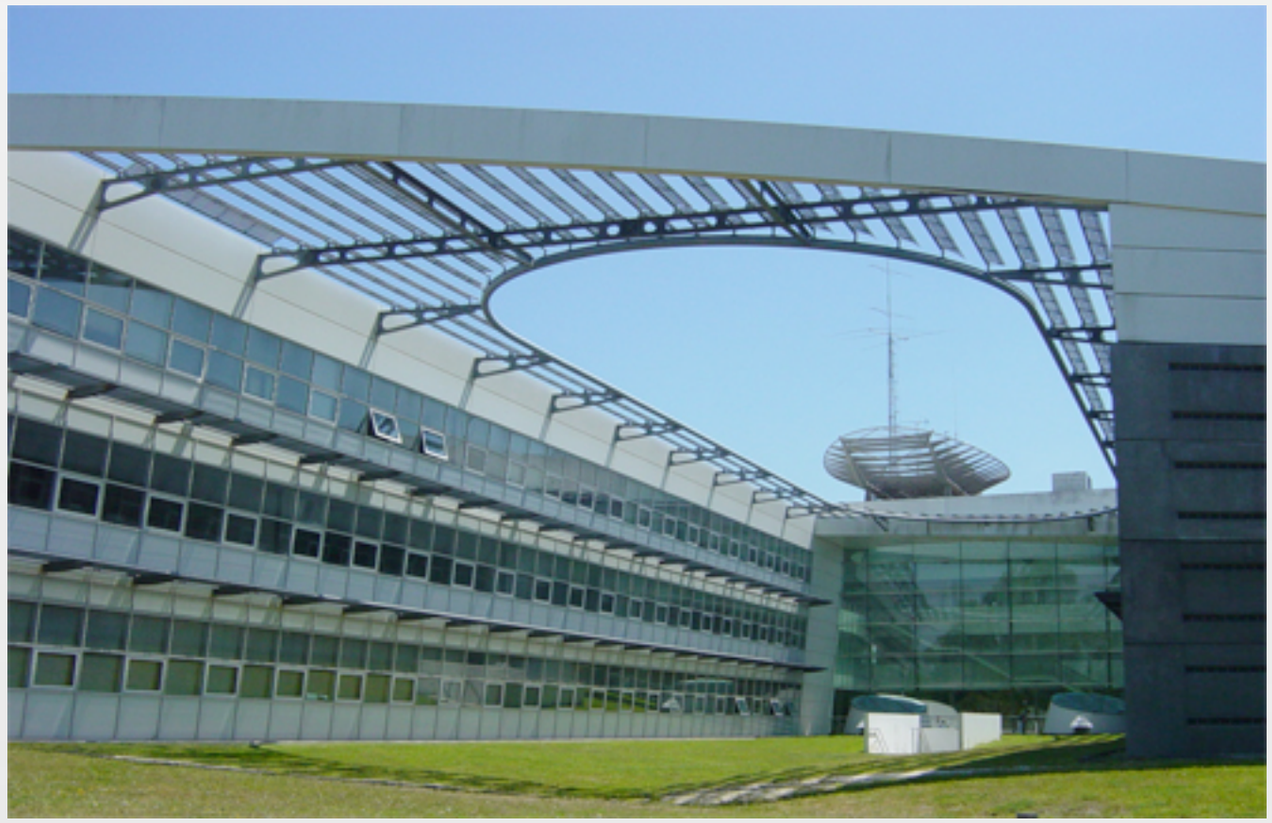


Requirements



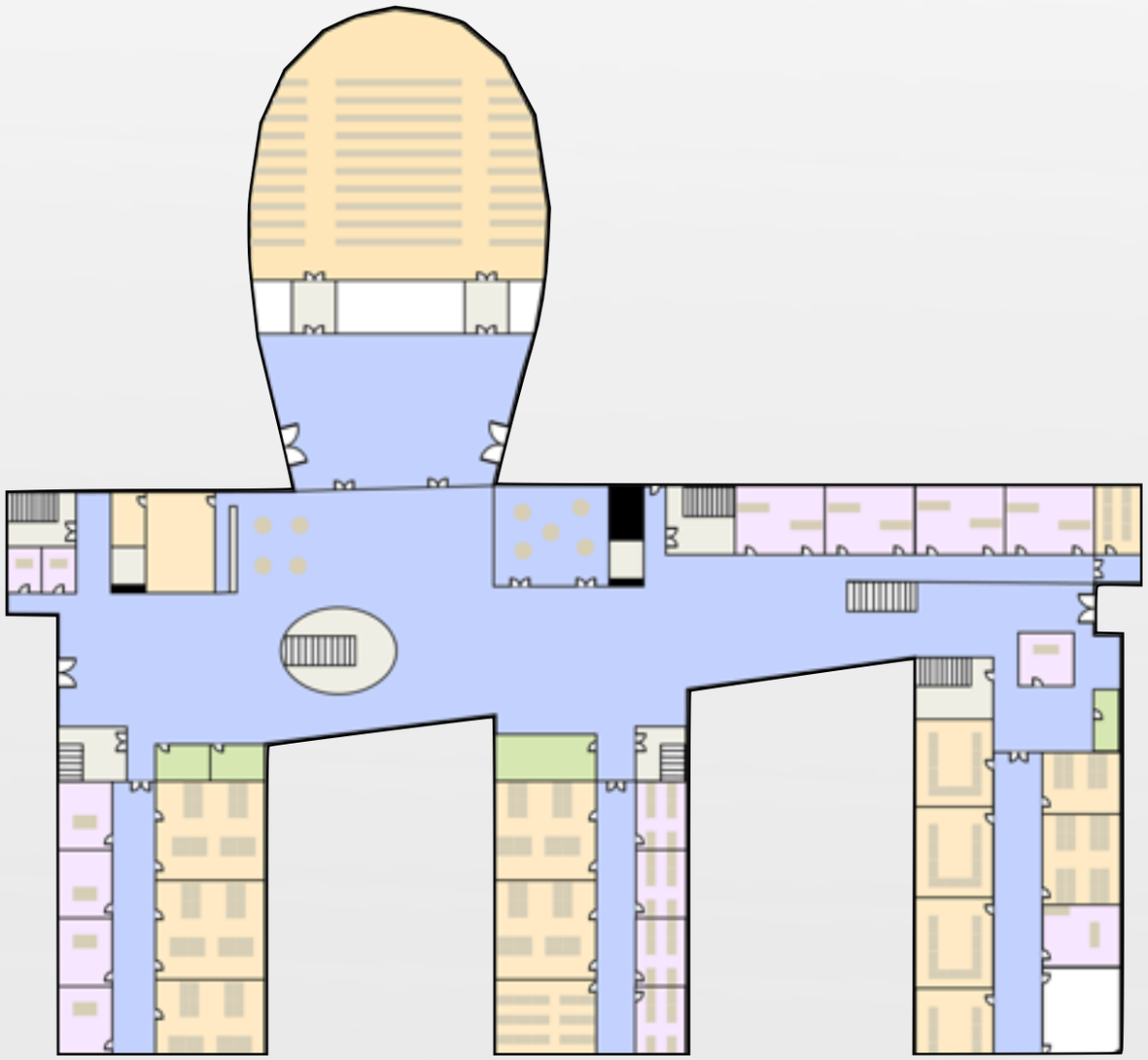
ILLUSTRATION

ENSEIRB, a graduate engineering school



Illustration

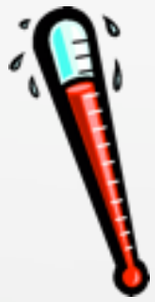
ENSEIRB, a graduate engineering school



INRIA

Situation

Physical parameters

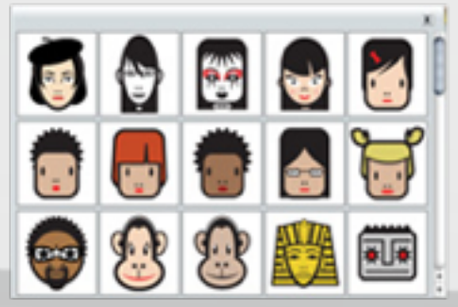


temperature

motion

smoke

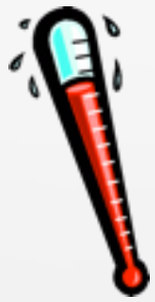
luminosity



profile

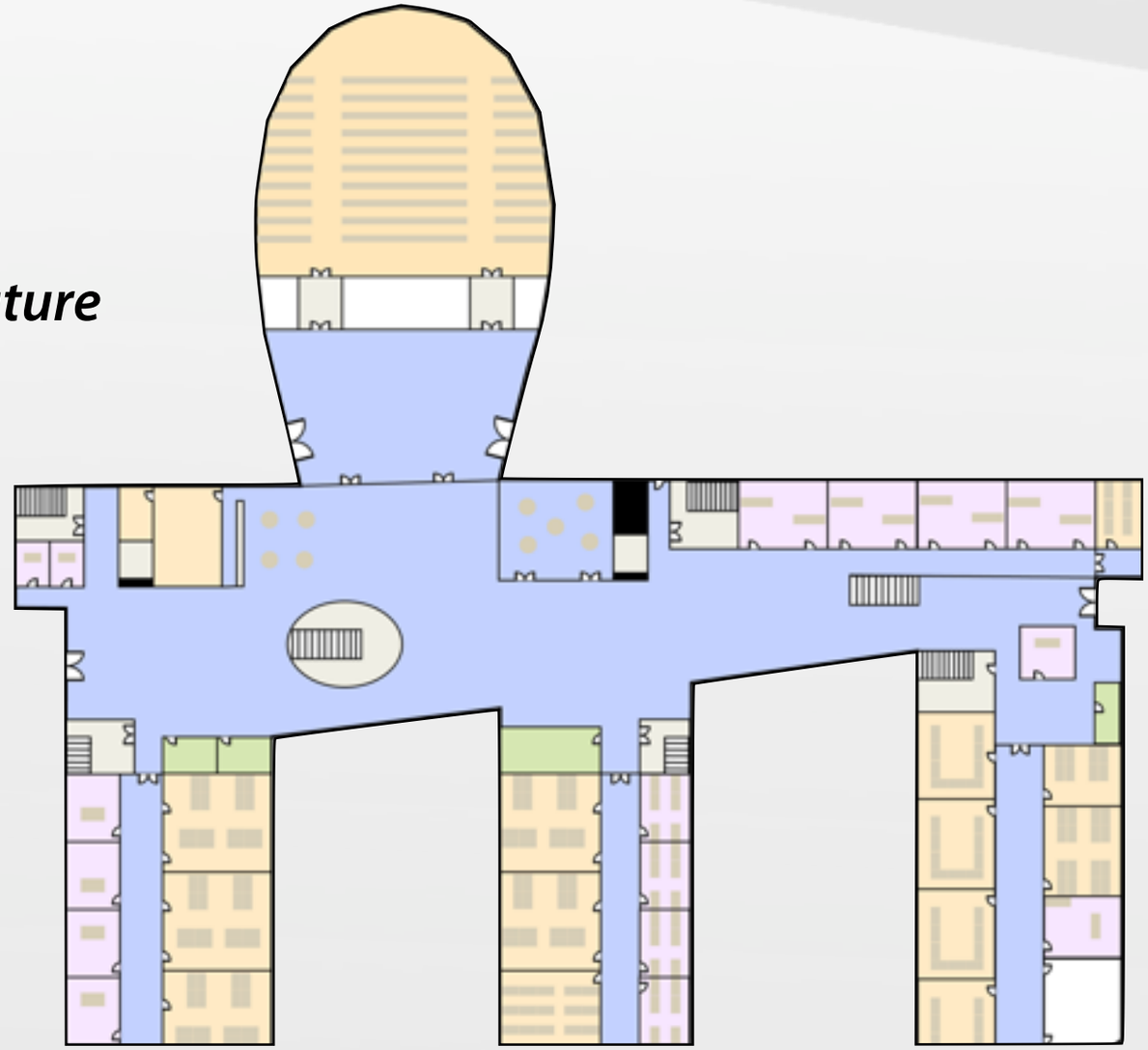
agenda

Situation (1)

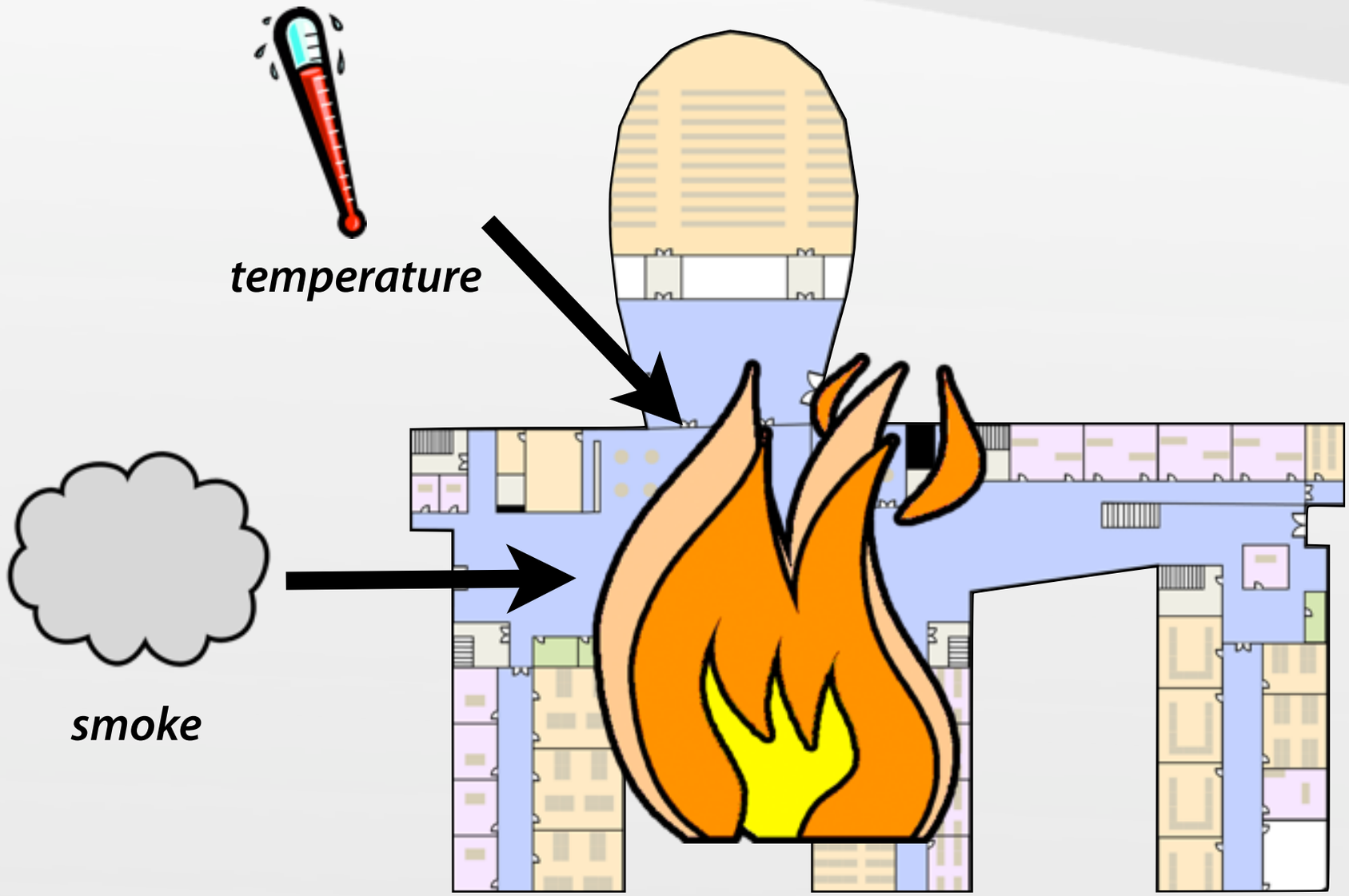


temperature

smoke



Situation (1)

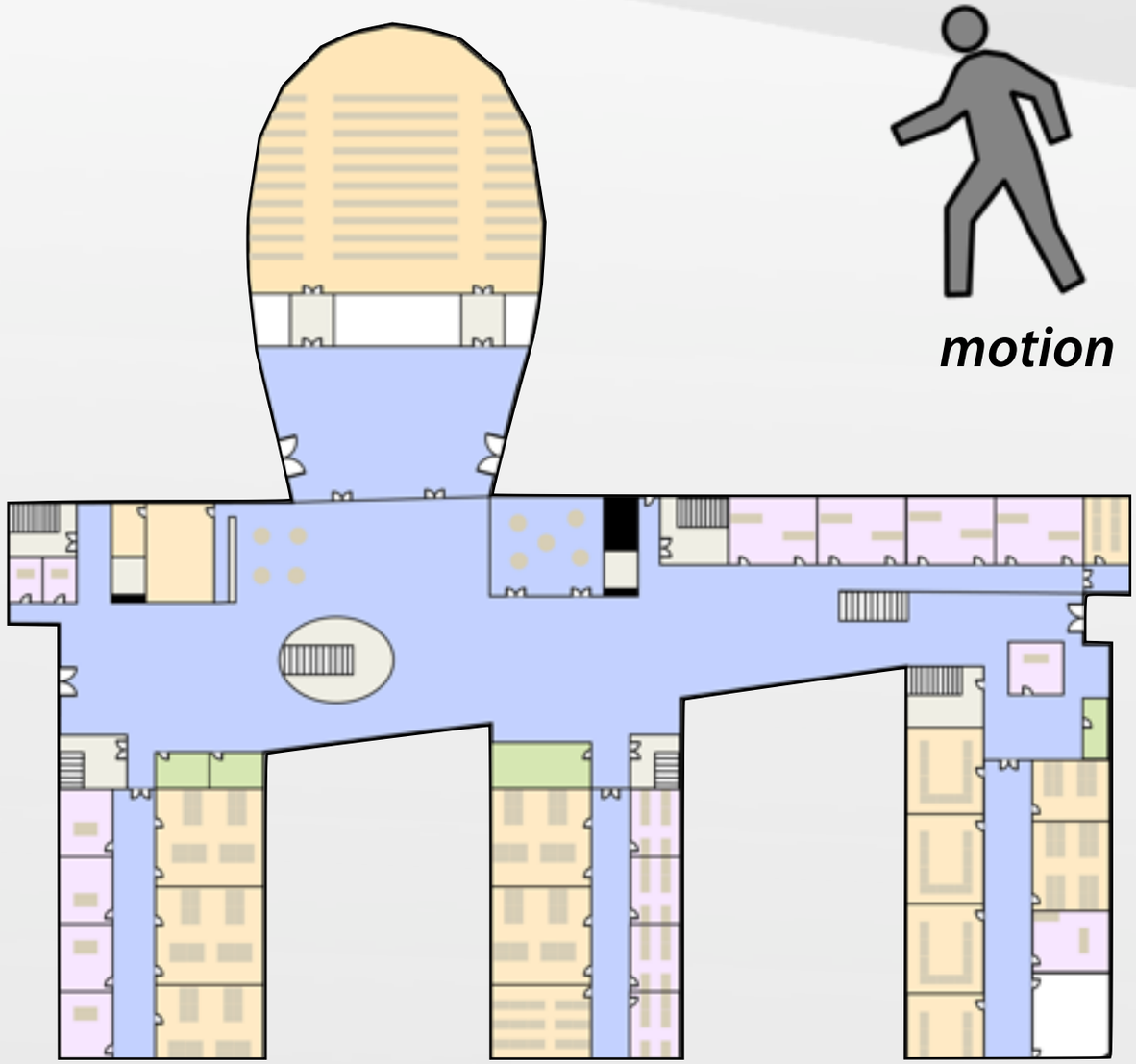


Situation of FIRE

Fire Manager

INRIA

Situation (2)

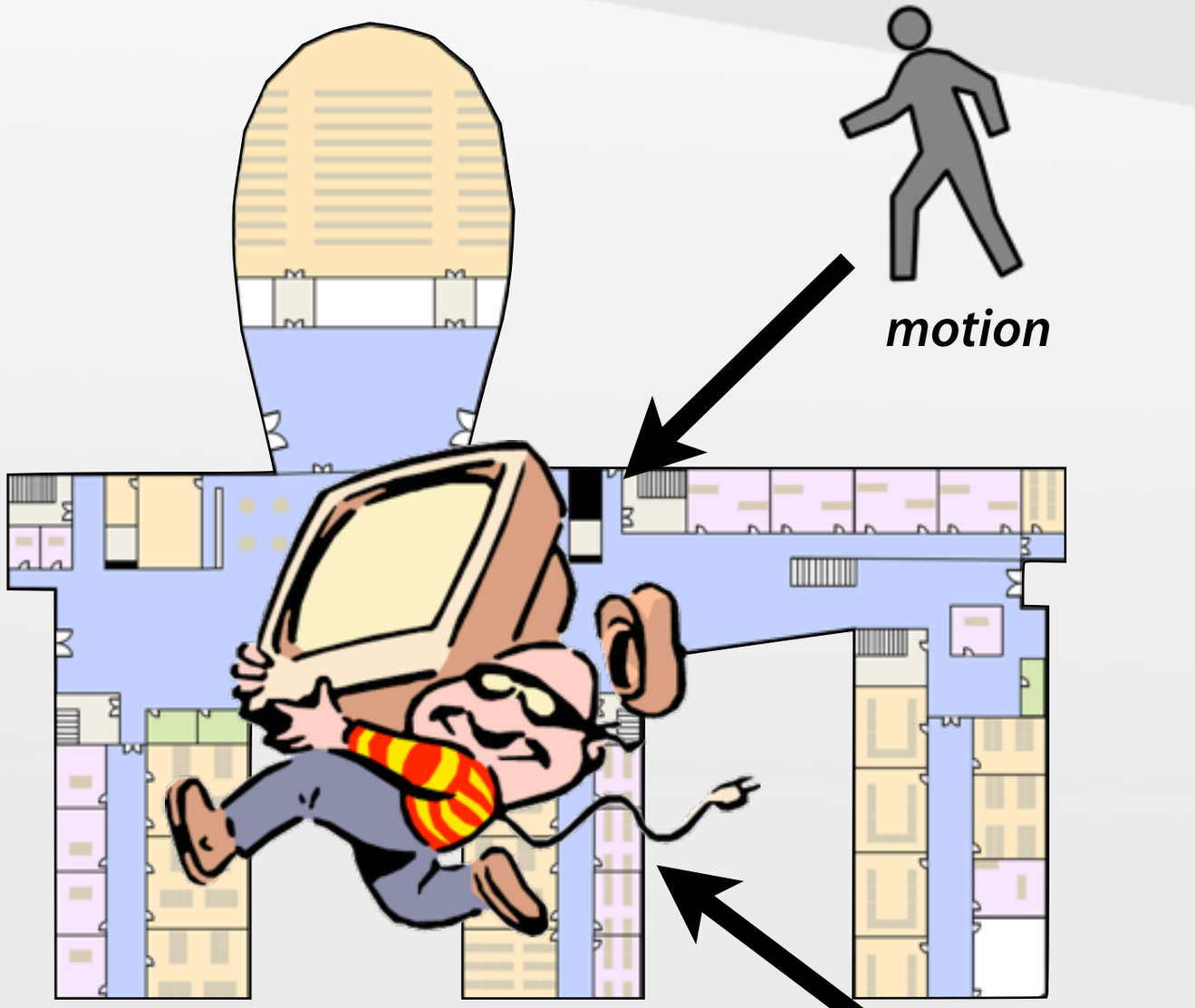


motion

agenda

INRIA

Situation (2)



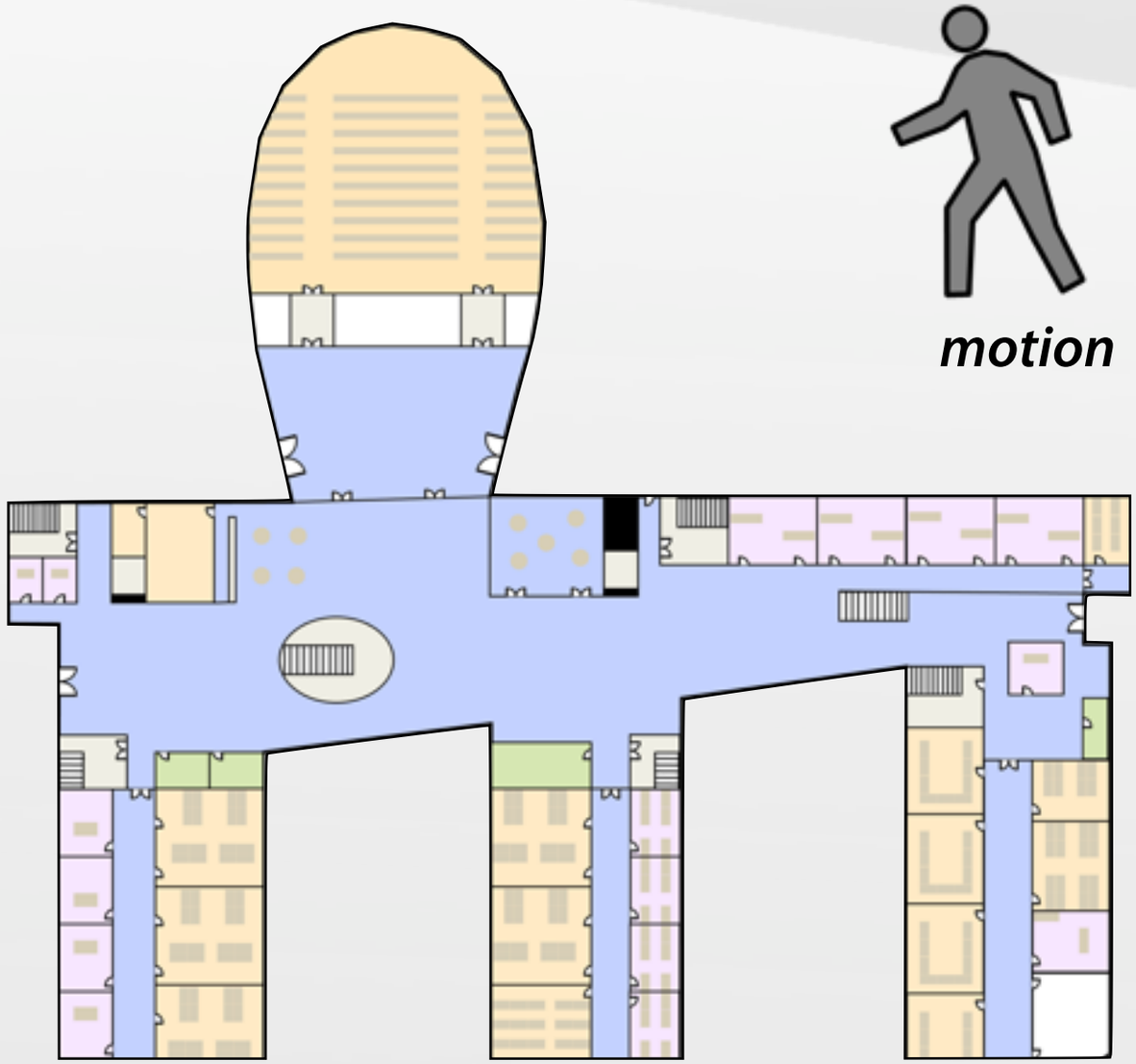
Situation of ROBBERY

agenda

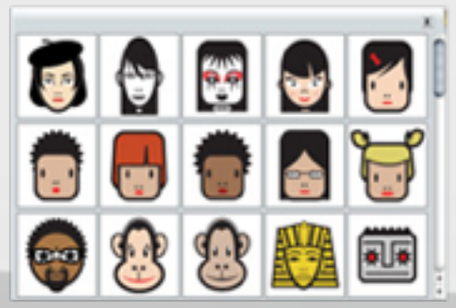
Intrusion Manager

INRIA

Situation (3)

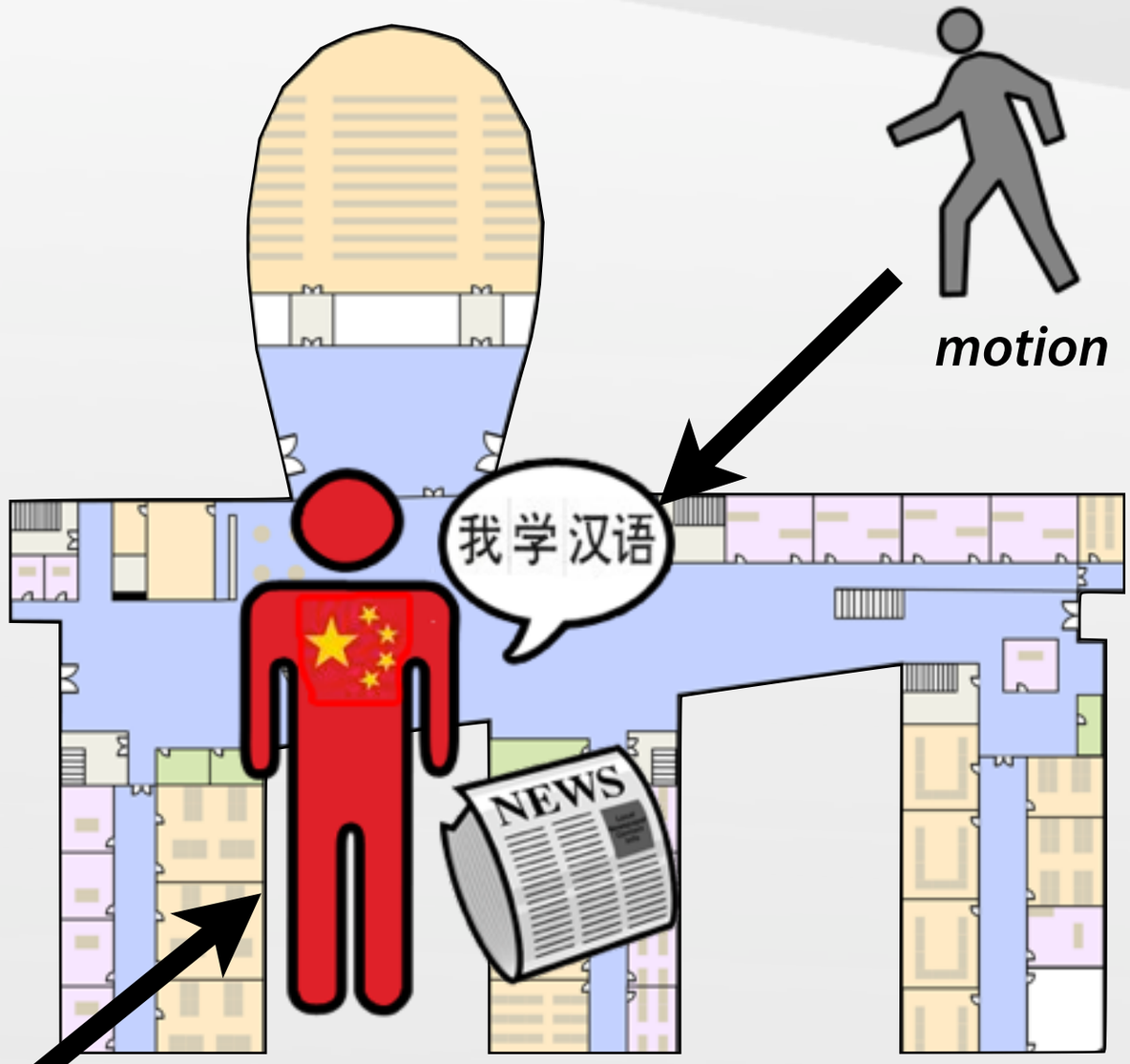


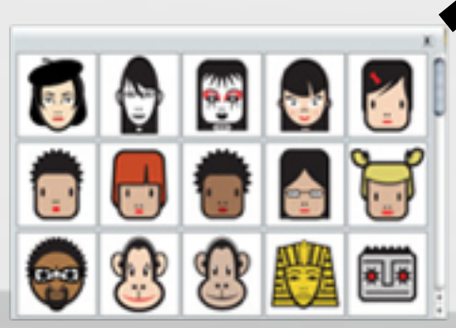
motion



profile

Situation (3)





Situation of PRESENCE

profile

Newscast Manager

INRIA

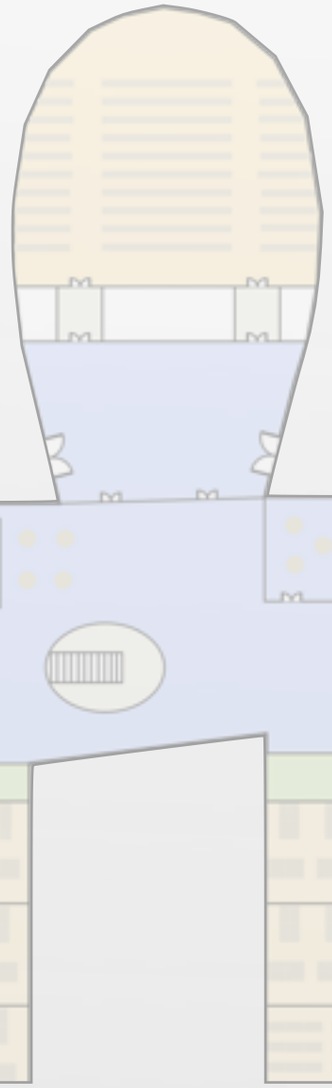
Heterogeneous Devices

light sensor

desktop computer

lamp

tag reader



telephone

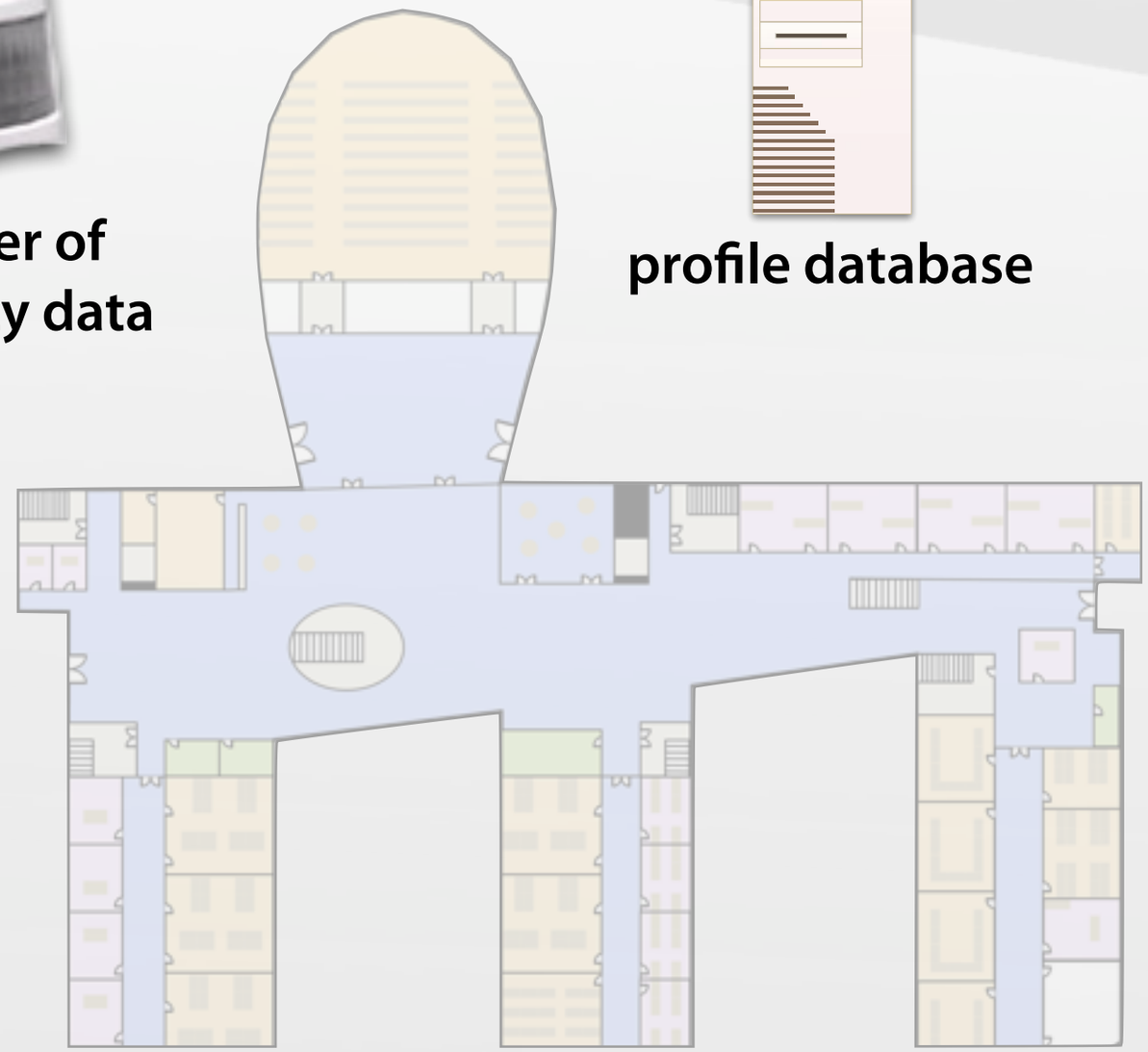
screen

PDA

INRIA

Heterogeneous Services

producer of
luminosity data



profile database

light switch

producer of
position data

audio receiver

news notifier

intrusion notifier

INRIA

Heterogeneous Services

Heterogeneous software buses

producer of luminosity data

profile database

light switch

SIP

SOAP

SMTP

producer of position data

UPnP

X10

HTTP

RTSP

TCP

audio receiver

news notifier

intrusion notifier

INRIA

Heterogeneous Services

producer of
luminosity data

profile database

light switch

producer of
position data

newscast
application

- Detects situations
- Coordinates services

audio receiver

news notifier

intrusion notifier

INRIA

Service Mobility and Availability

producer of
luminosity data

profile database

light switch

producer of
position data

newscast
application

audio receiver

news notifier

intrusion notifier

Service Mobility and Availability

producer of
luminosity data

profile database

light switch

producer of
position data

newscast
application

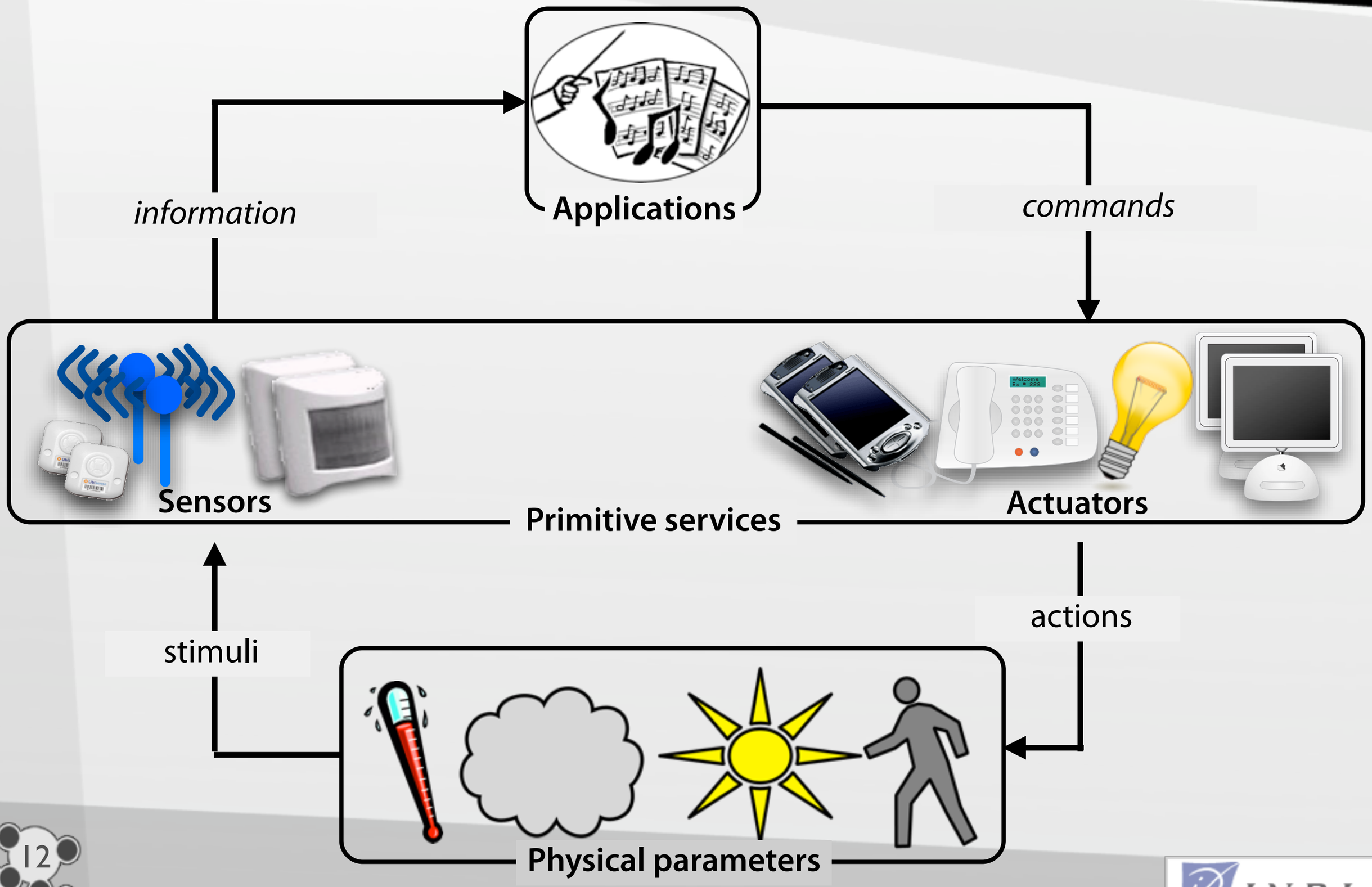
audio receiver

news notifier

intrusion notifier

INRIA

Ubiquitous Computing Environments



INRIA

Issues

Applications

Primitive services

Physical parameters

Issues

Applications

criticity

Primitive services

heterogeneity

dynamicity

Physical parameters

dynamicity

Issues

Applications

criticity

- Verifying applications prior to run time
- Testing applications at run time

Primitive services

heterogeneity

- Integration of existing and future entities
- Common interface model

dynamicity

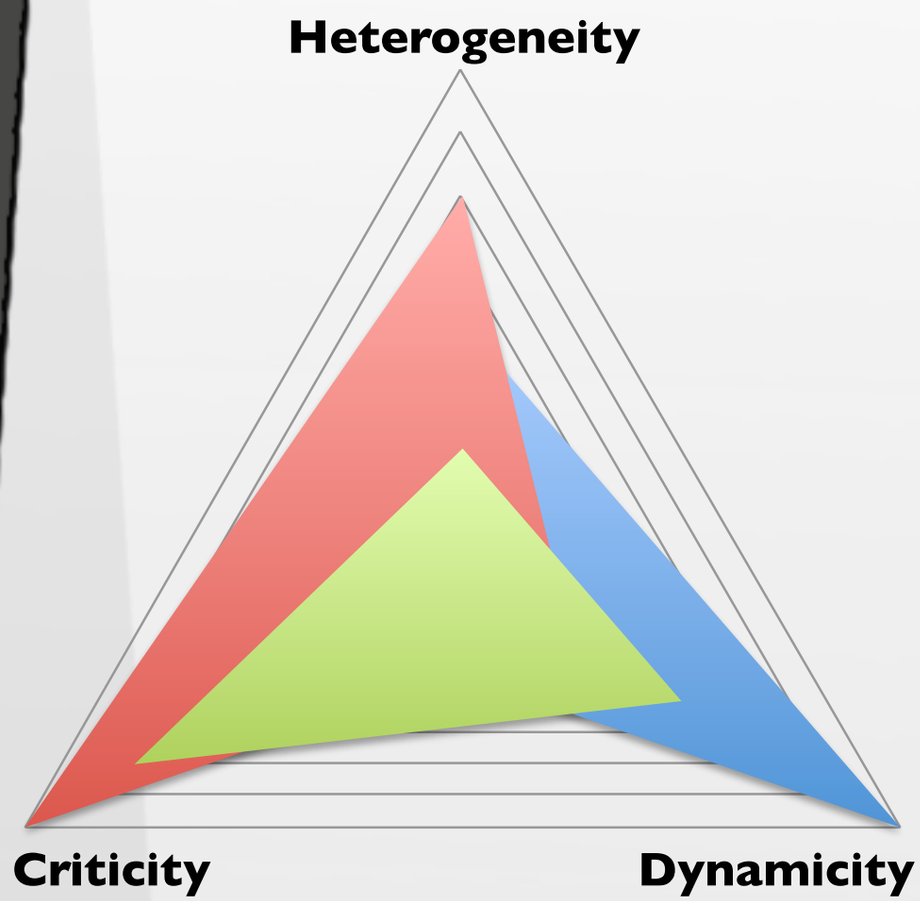
- Late binding

Physical parameters

dynamicity

- Detecting situation

Existing Solutions for Programming Ubiquitous Applications



Component-based systems

+ More verifications

- Incomplete handling of dynamicity

Distributed systems

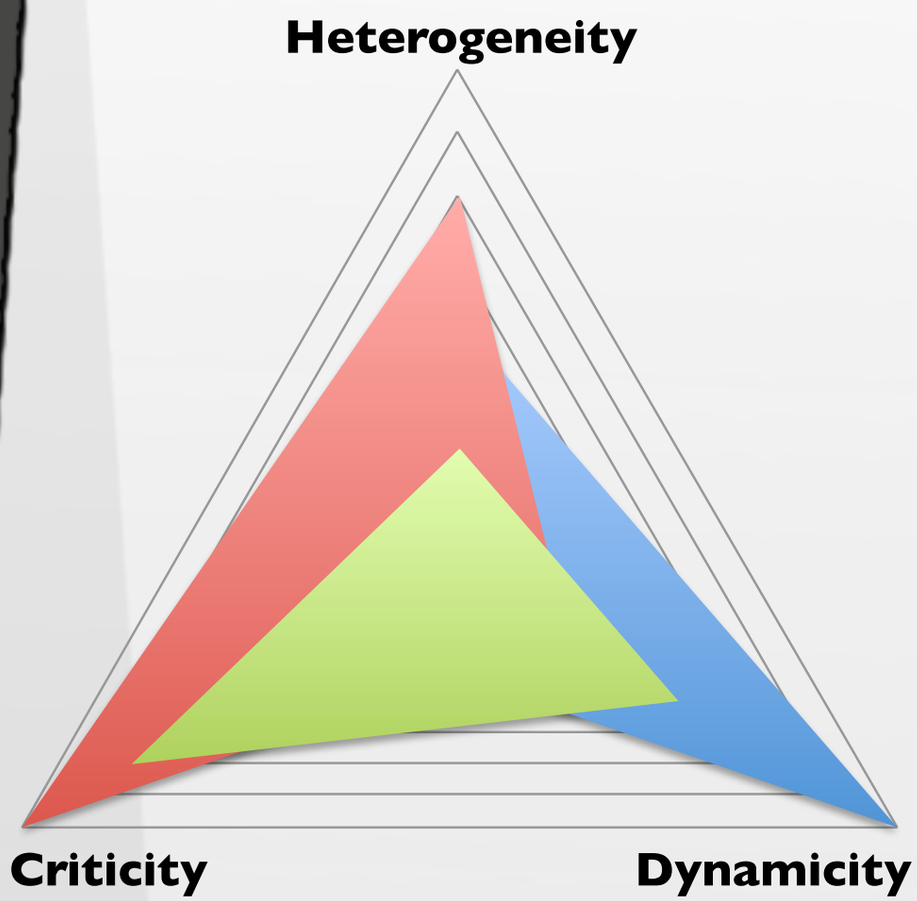
- Few static verifications

Ubiquitous systems

+ Higher-level abstractions for dynamicity

- Few verifications

Existing Solutions for Programming Ubiquitous Applications



Component-based systems

+ More verifications
- Incomplete handling of dynamicity

Distributed systems

- Few static verifications

Ubiquitous systems

+ Higher-level abstractions for dynamicity
- Few verifications

Limits

* No approach handling dynamicity AND verifications
* No approach abstracting underlying technologies
* No integrated approach to test ubiquitous applications

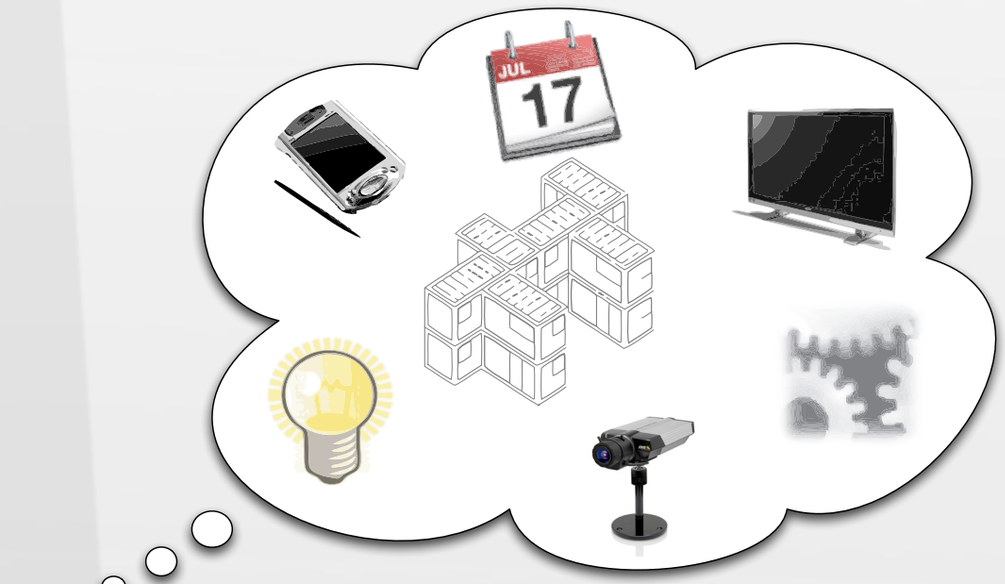
THESIS

THESIS

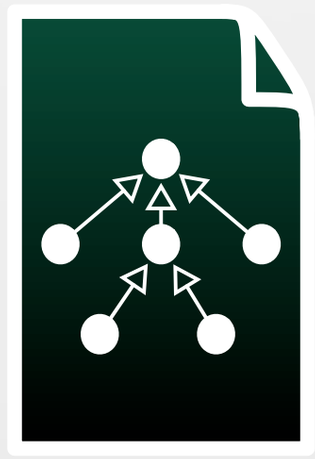
To provide a high-level and integrated approach for developing reliable ubiquitous applications

- *Specifying* ubiquitous environments
  - DIASPEC, a software architecture language
- *Developing* ubiquitous applications
  - Dedicated programming frameworks
  - Programming support to handle dynamicity
  - Static verifications
- *Testing* ubiquitous systems at runtime
  - DIASIM, a simulator
  - Integrated approach

The diagen Approach

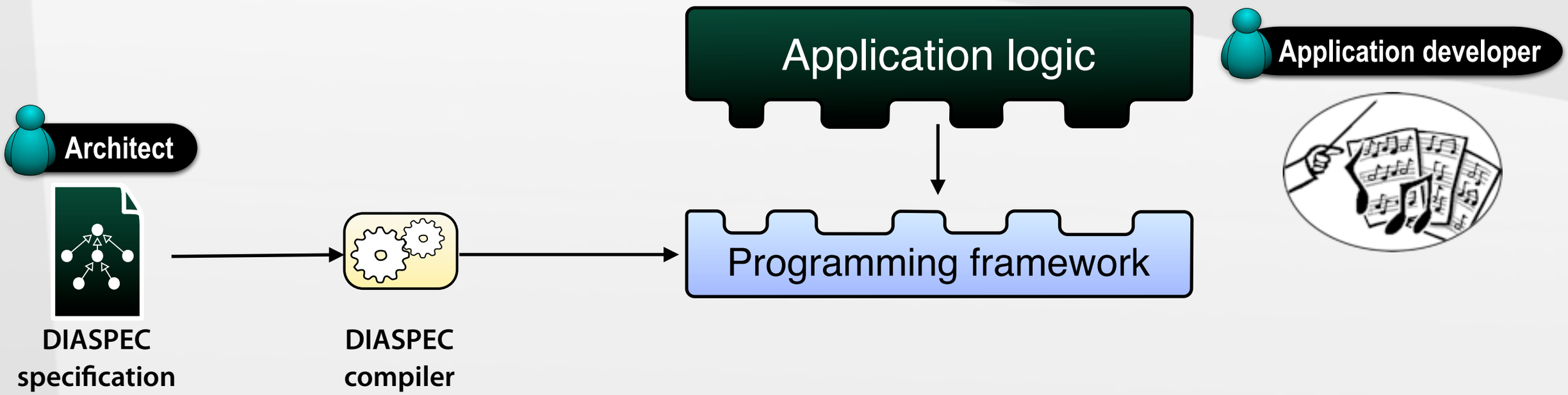


Architect

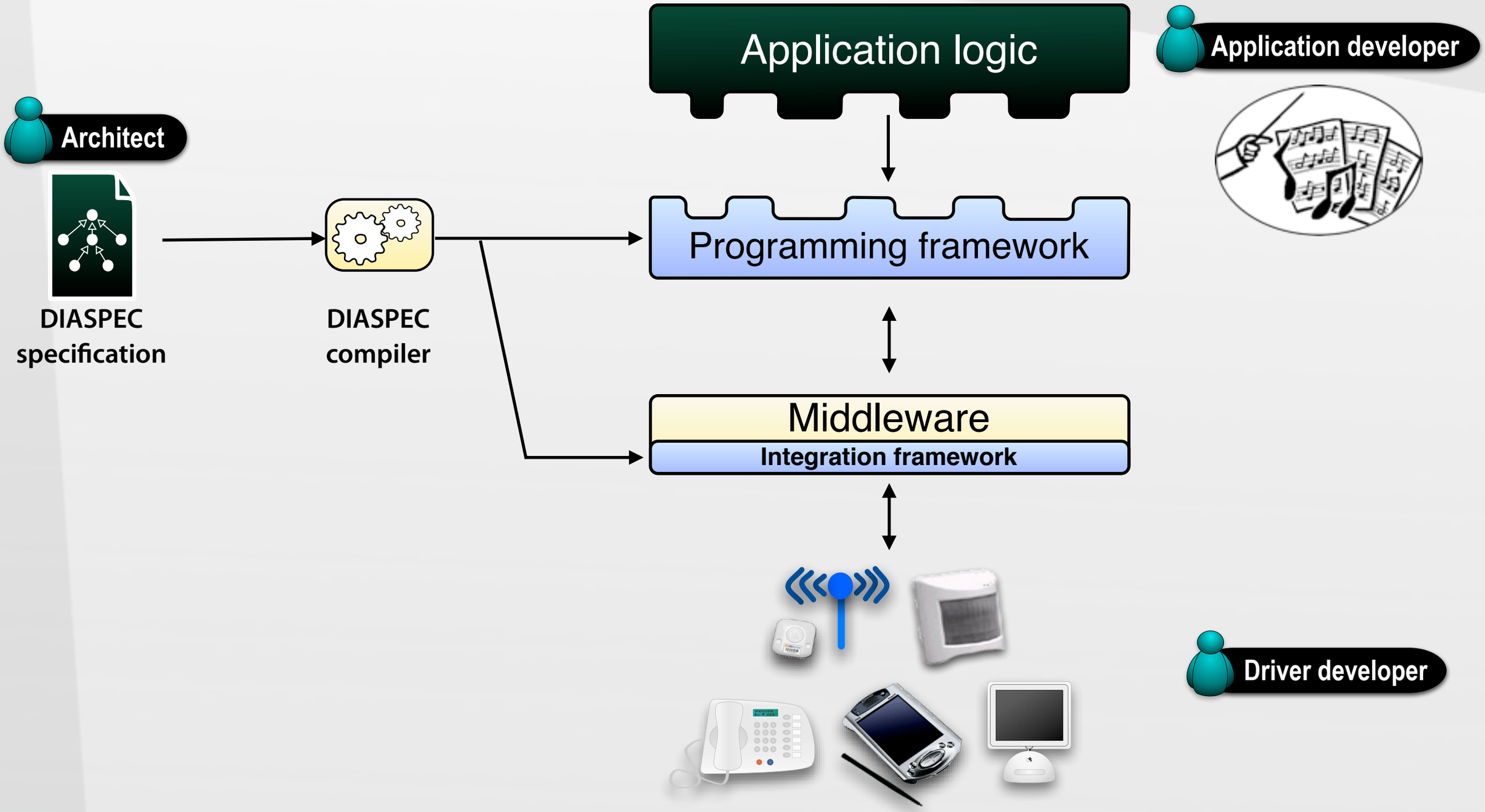


DIASPEC specification

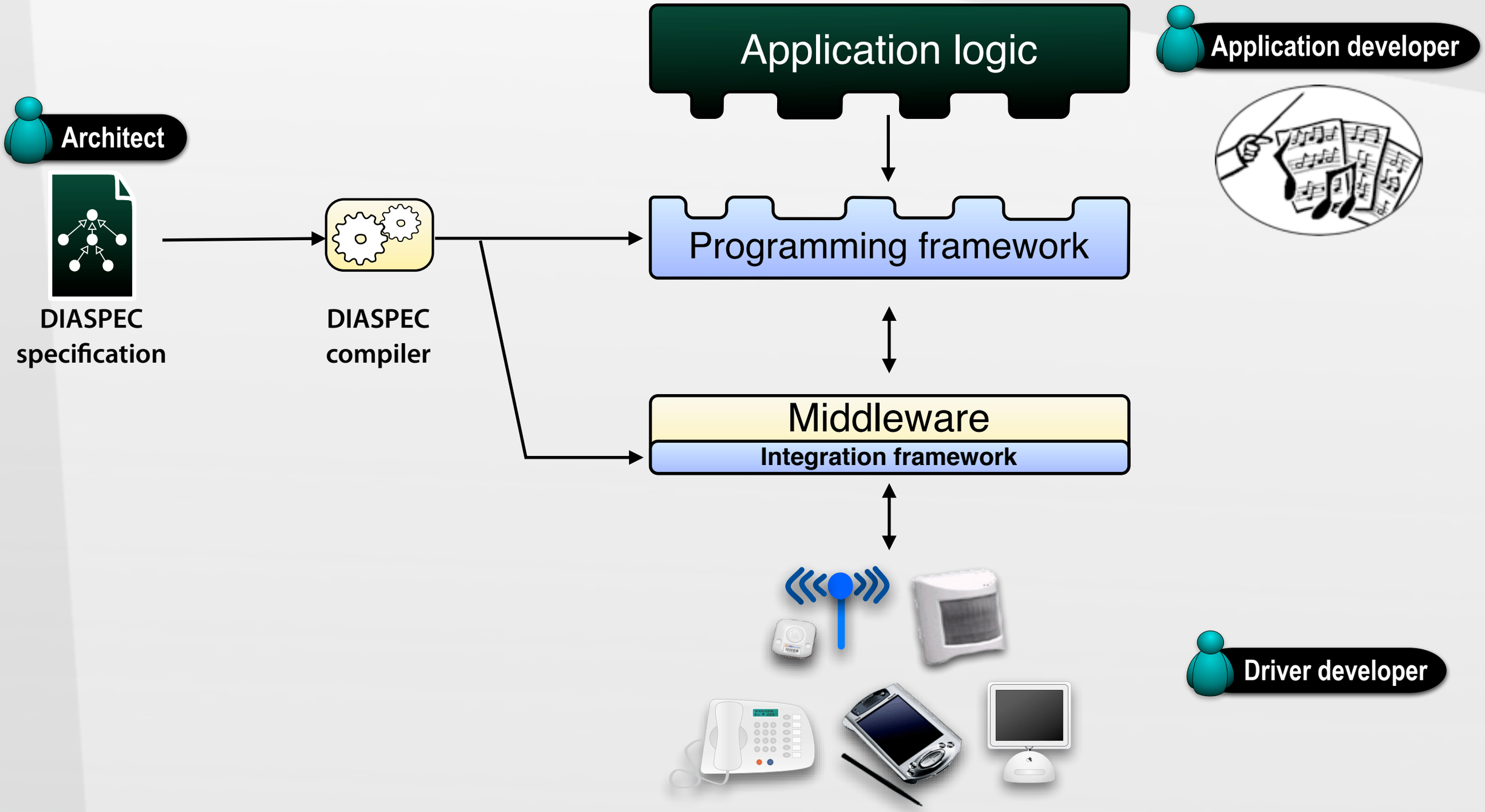
The DiaGen Approach



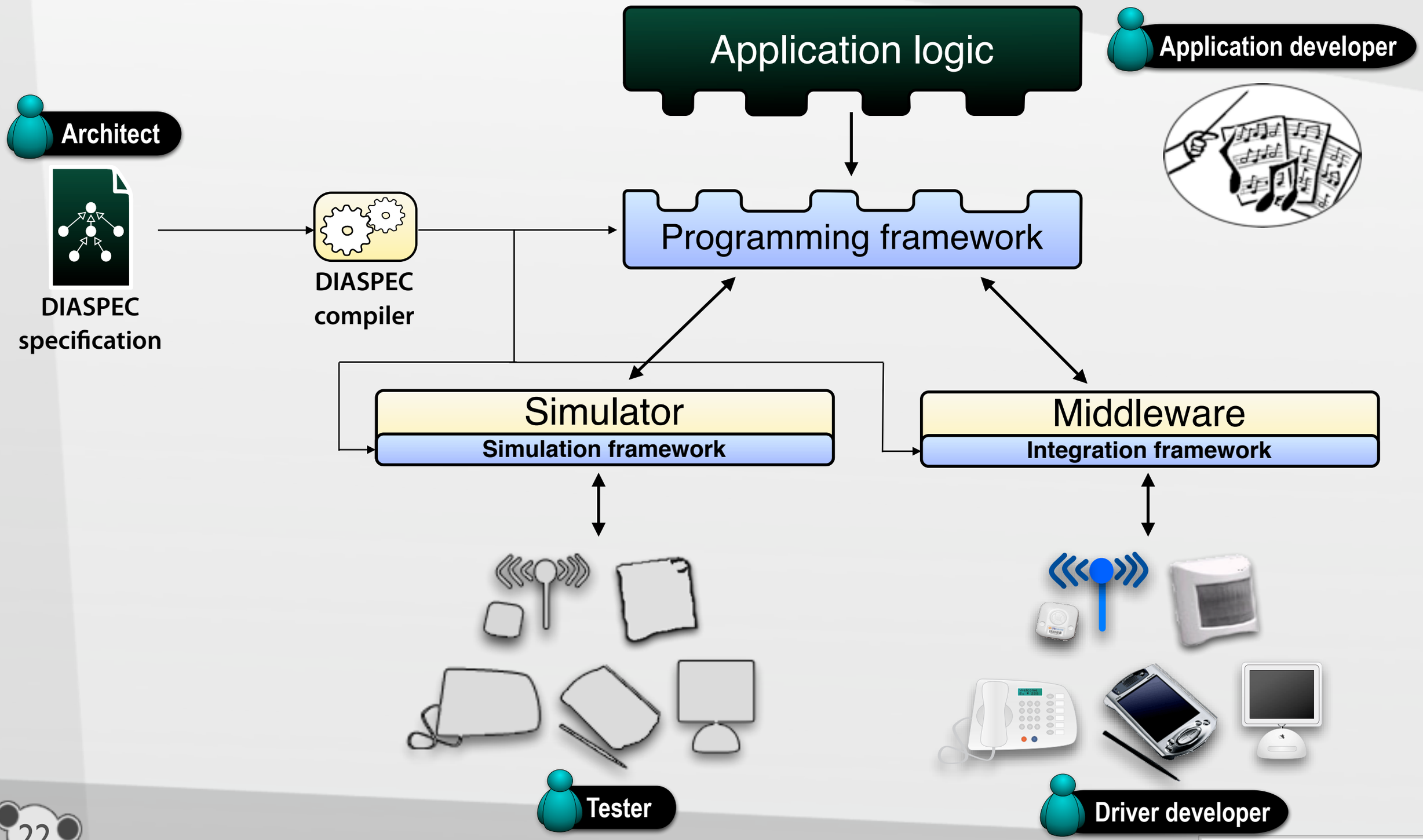
The DiaGen Approach



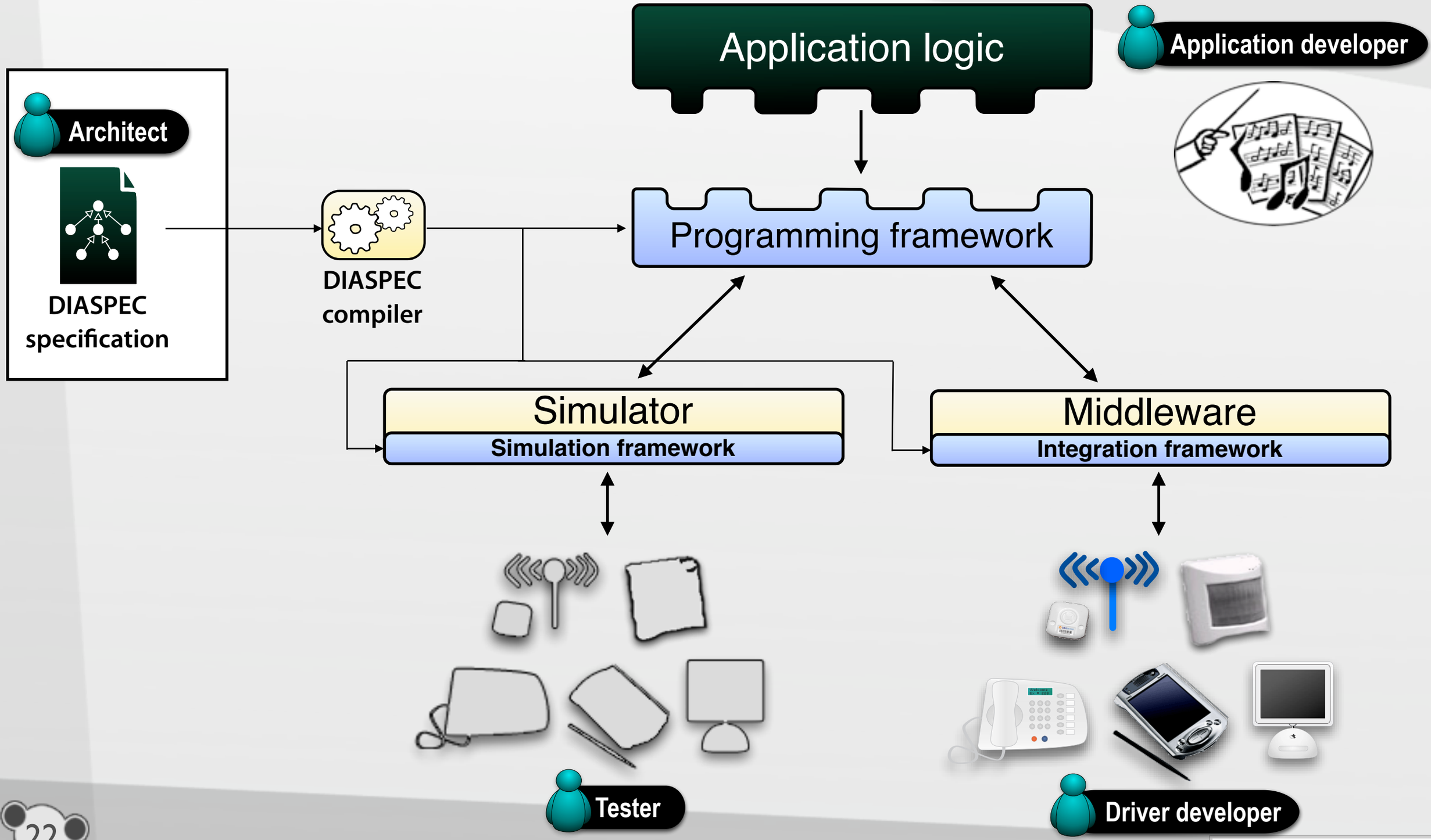
The DiaGen Approach



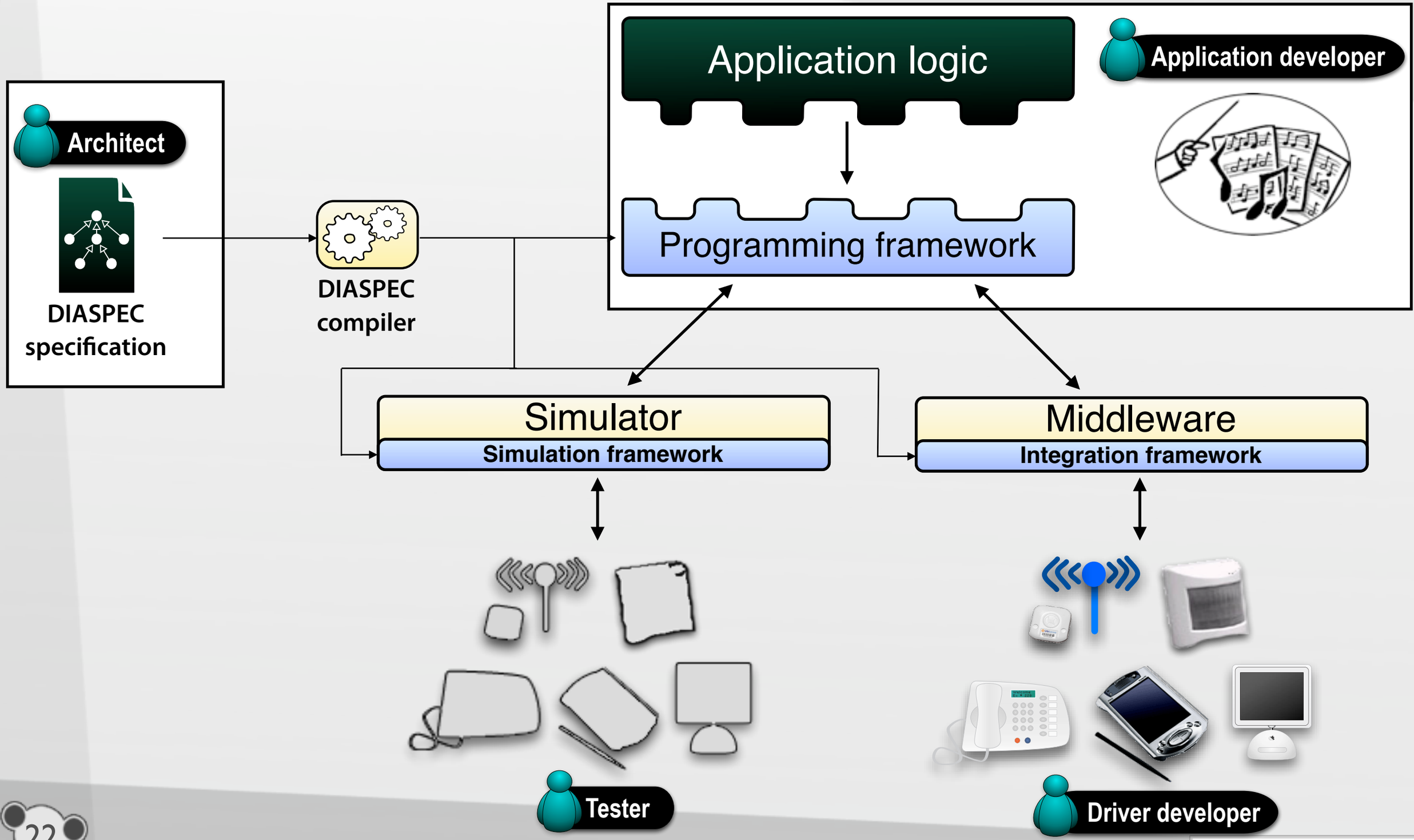
The DIAGEN Approach



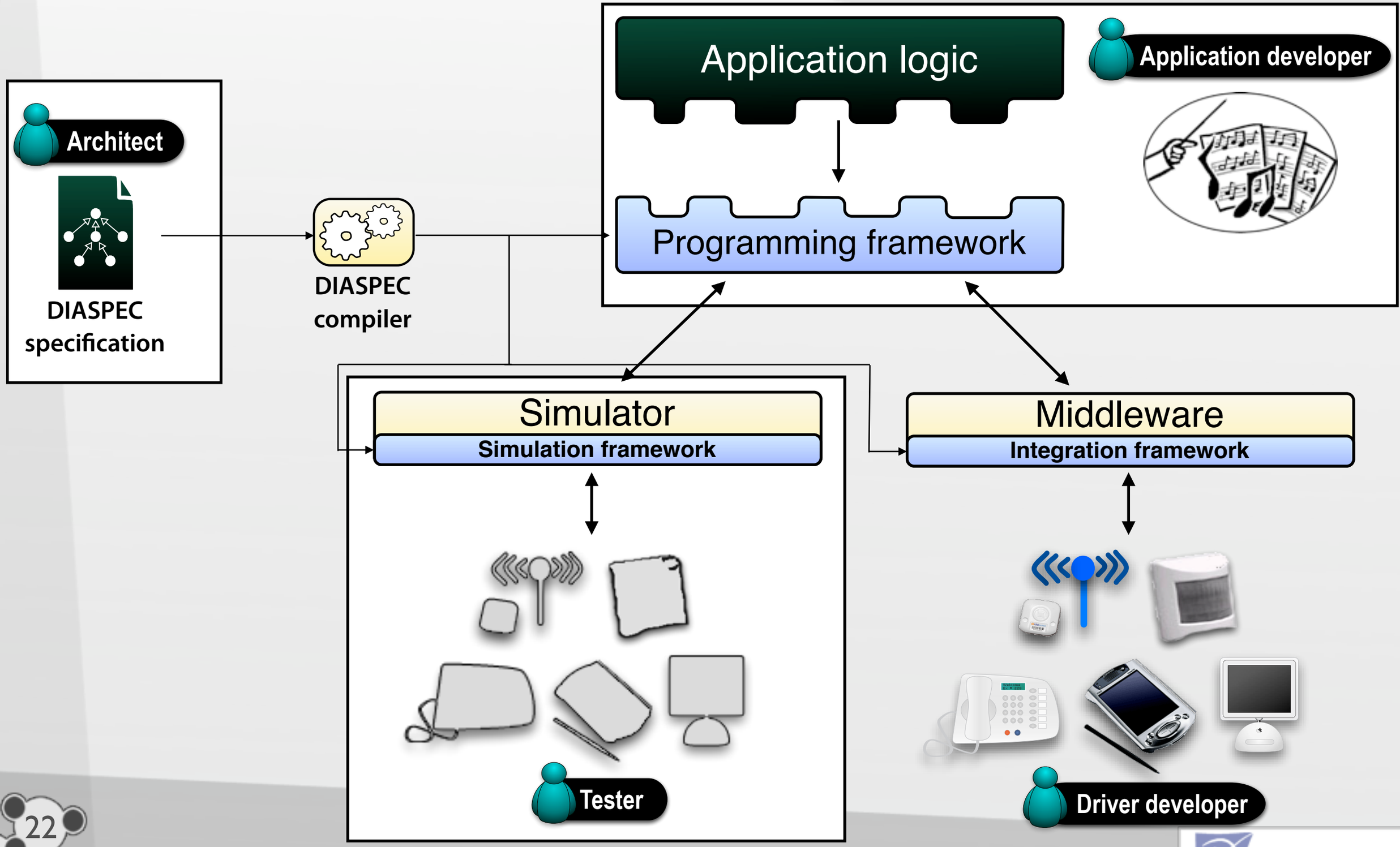
The DIAGEN Approach



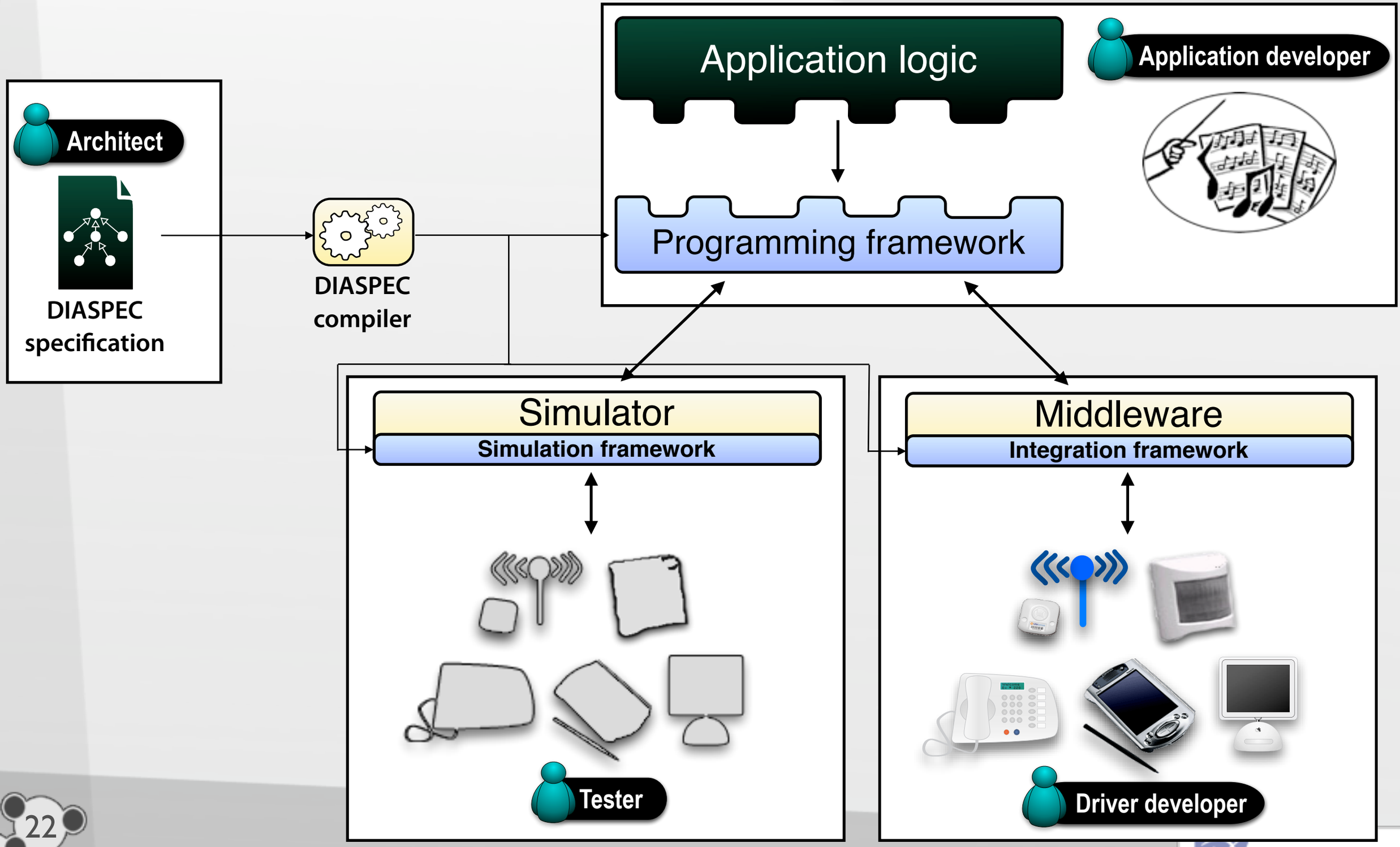
The DIAGEN Approach



The DIAGEN Approach



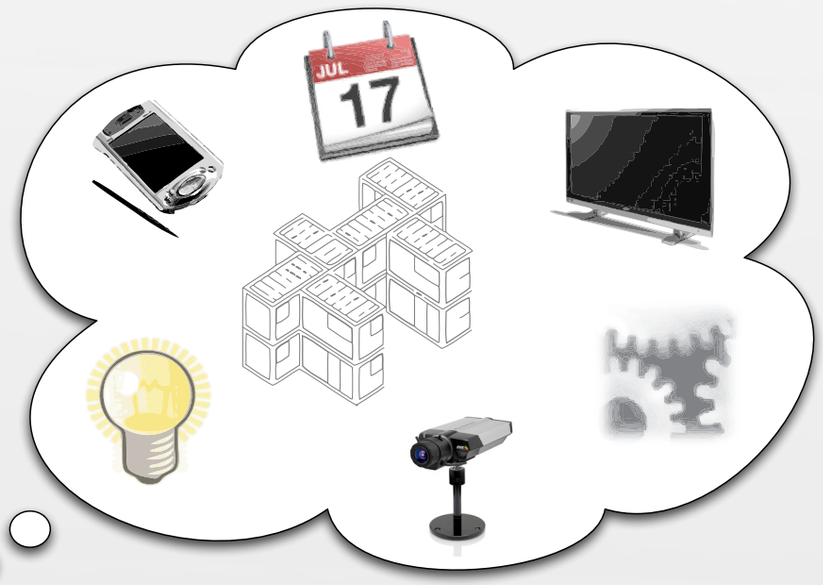
The DIAGEN Approach



SOFTWARE ARCHITECTURE SPECIFICATION WITH DIASPEC

specifying

Specifying



Architect

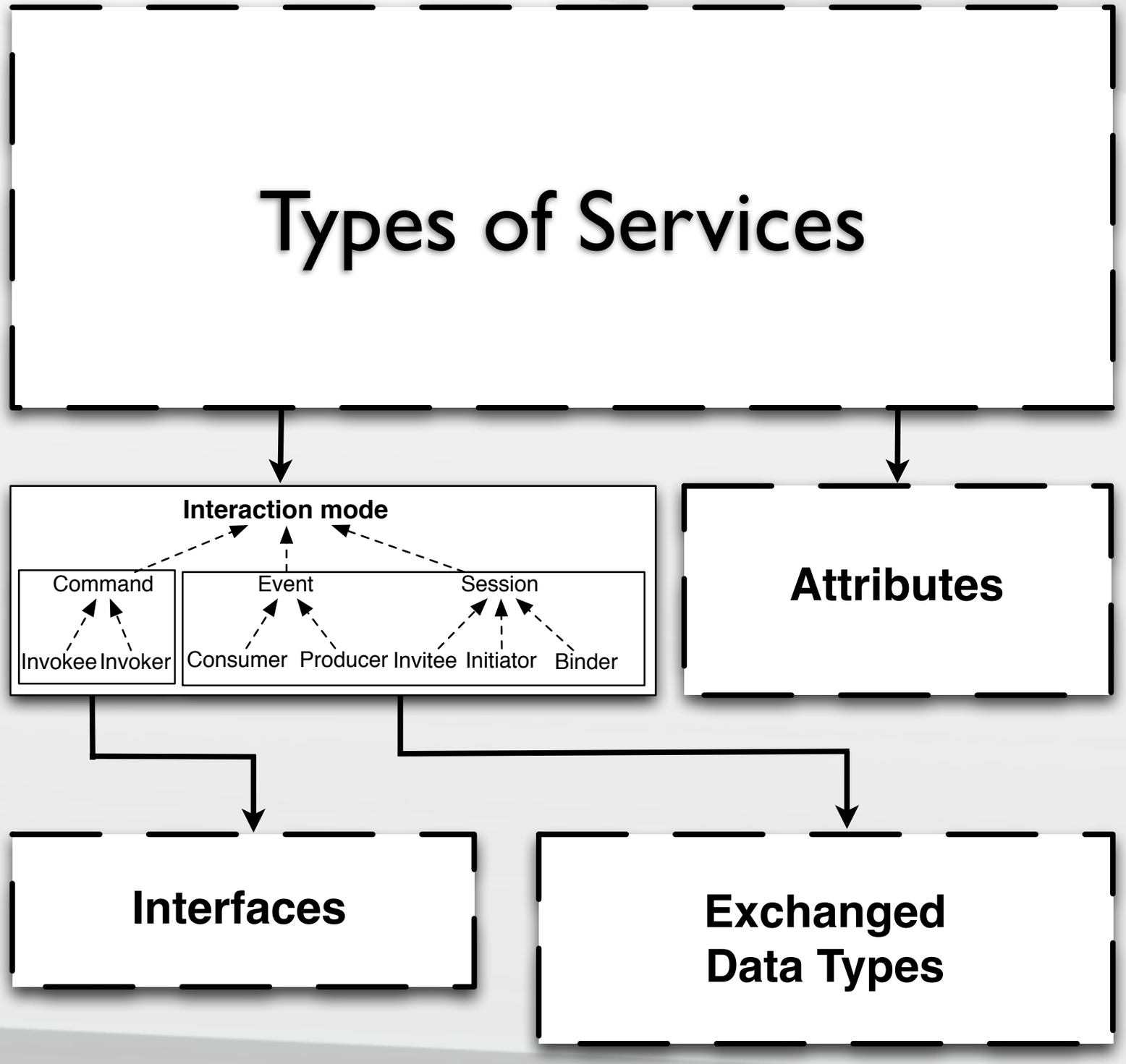
DIASPEC specification

INRIA

Software Architecture Language

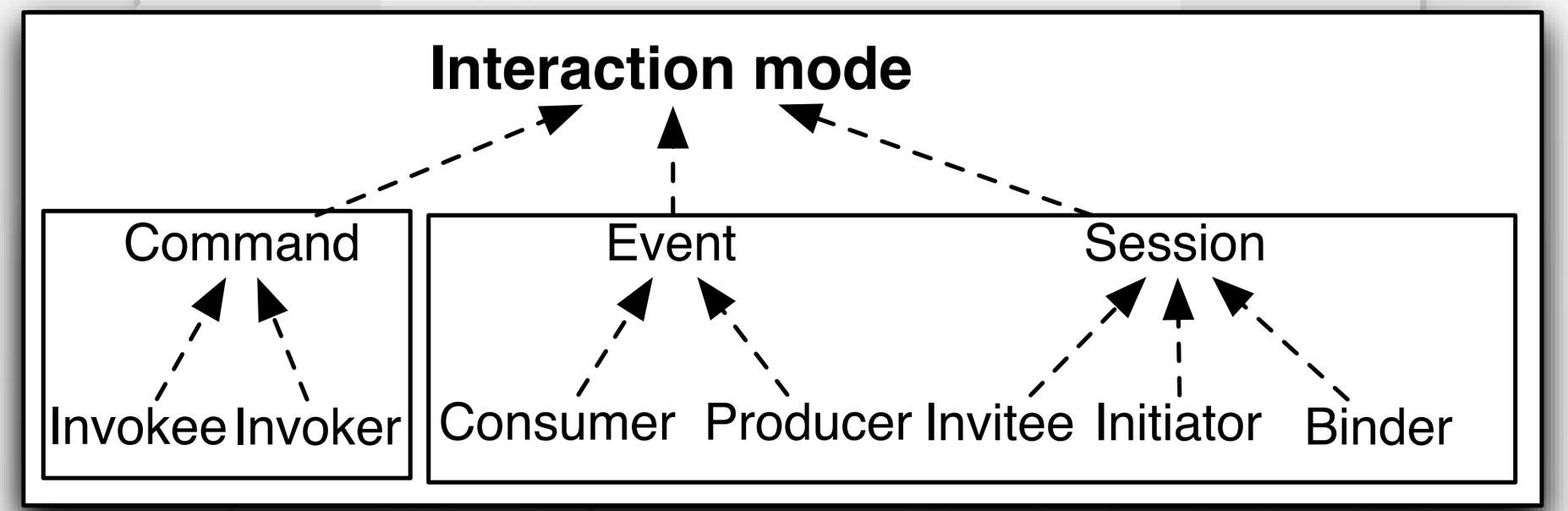
- Motivations
  - Easier to declare needs rather than program them
  - Declaring types of services
  - Declaring functionalities and connections

A DIASPEC Specification



A DIASPEC Specification

Types of Services



Interfaces

Exchanged Data Types

INRIA

Illustration: Building Automation

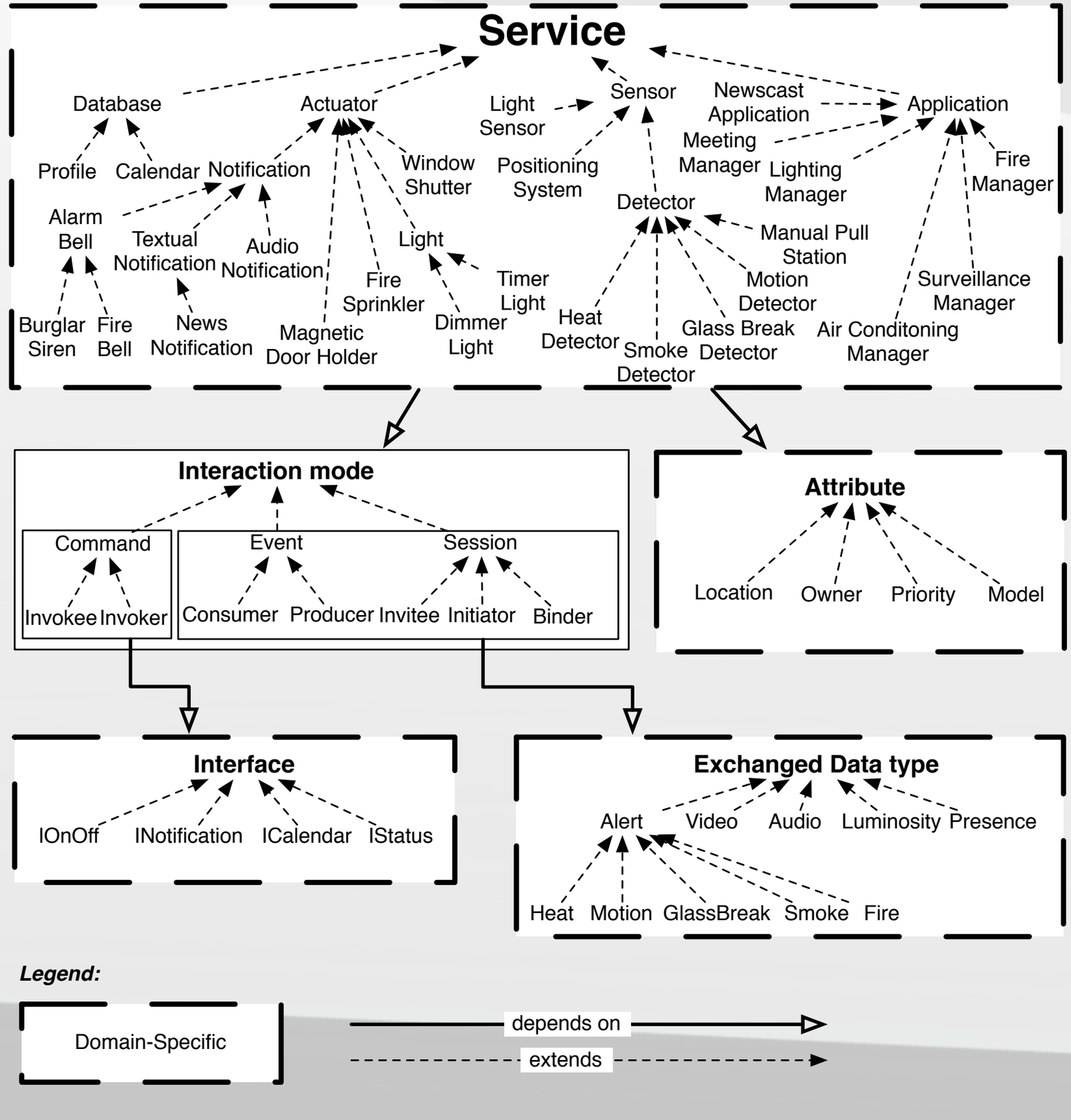
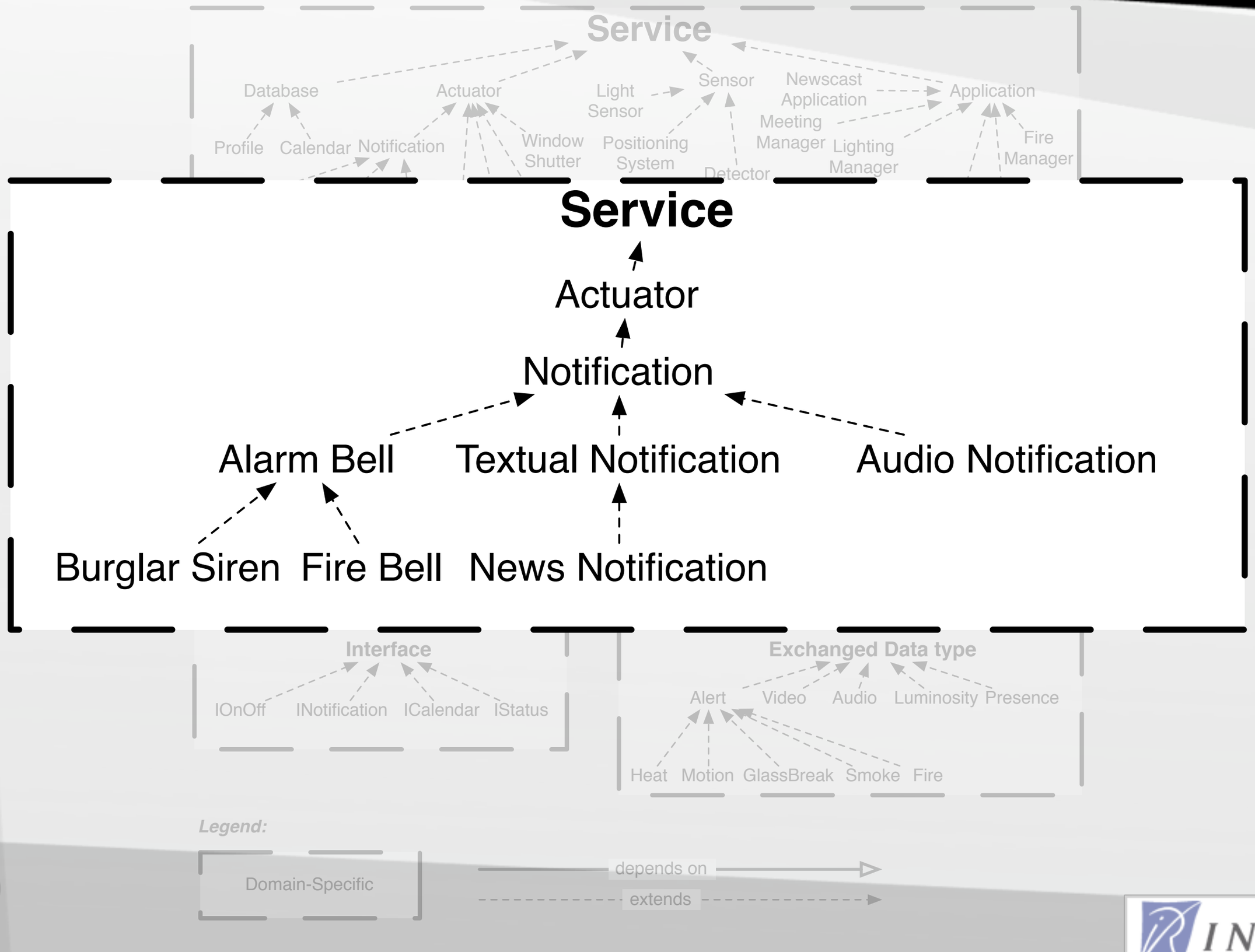


Illustration: Building Automation



Textual Excerpt

```
service Sensor(Location location) extends Service {}

service TagReader() extends Sensor {
    provides event Presence to NewscastManager;
}

service NewscastManager() extends Manager {
    requires event Presence from TagReader;
    requires command DisplayNews from NewsNotification;
    binds session AudioMessage from Speaker, AnnouncementCenter;
}
```

Textual Excerpt

```
service Sensor(Location location) extends Service {}

service TagReader() extends Sensor {
    provides event Presence to NewscastManager;
}

service NewscastManager() extends Manager {
    requires event Presence from TagReader;
    requires command DisplayNews from NewsNotification;
    binds session AudioMessage from Speaker, AnnouncementCenter;
}
```

Textual Excerpt

```
service Sensor(Location location) extends Service {}

service TagReader() extends Sensor {
    provides event Presence to NewscastManager;
}

service NewscastManager() extends Manager {
    requires event Presence from TagReader;
    requires command DisplayNews from NewsNotification;
    binds session AudioMessage from Speaker, AnnouncementCenter;
}
```

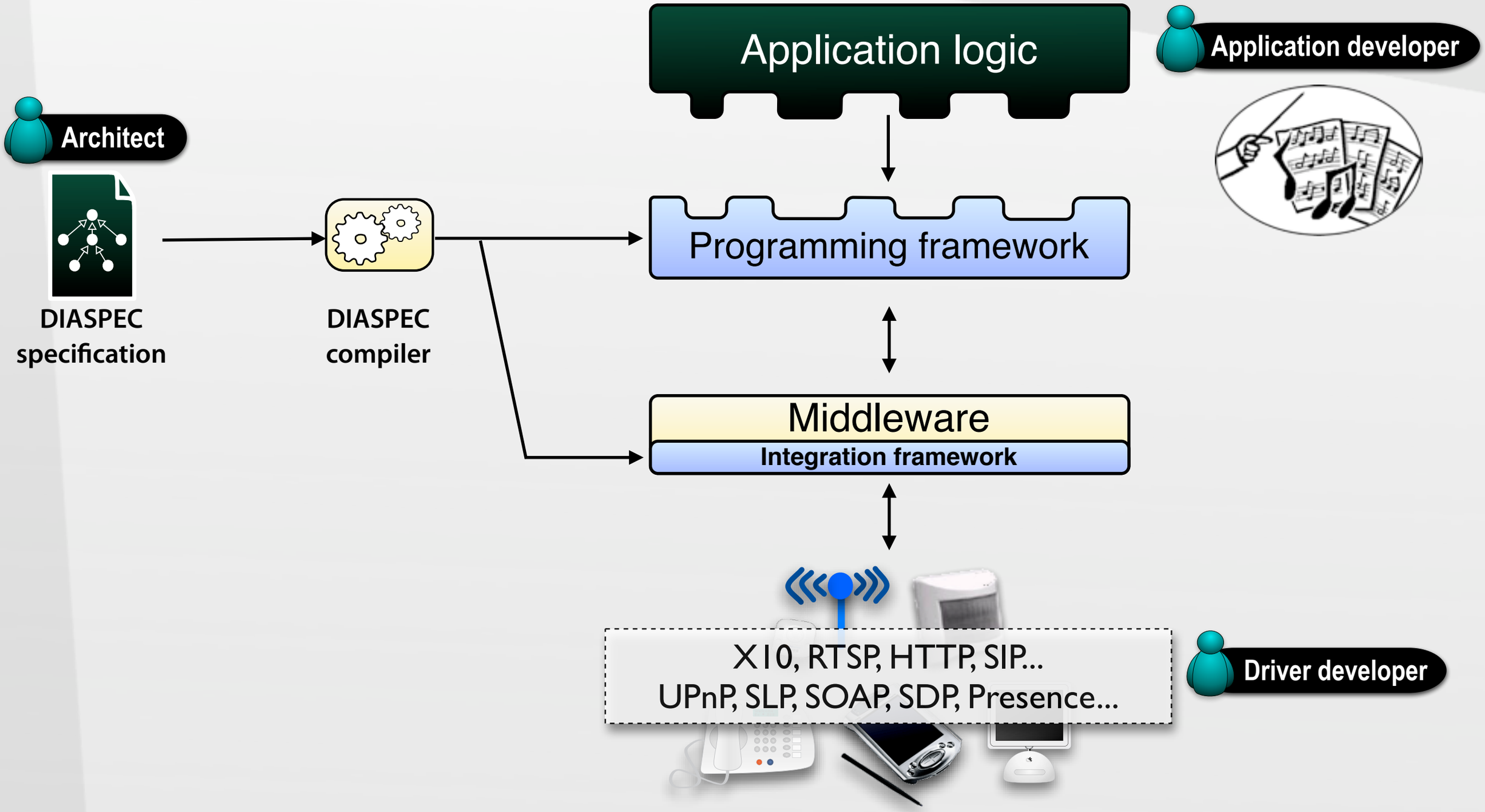
Textual Excerpt

```
service Sensor(Location location) extends Service {}

service TagReader() extends Sensor {
    provides event Presence to NewscastManager;
}

service NewscastManager() extends Manager {
    requires event Presence from TagReader;
    requires command DisplayNews from NewsNotification;
    binds session AudioMessage from Speaker, AnnouncementCenter;
}
```

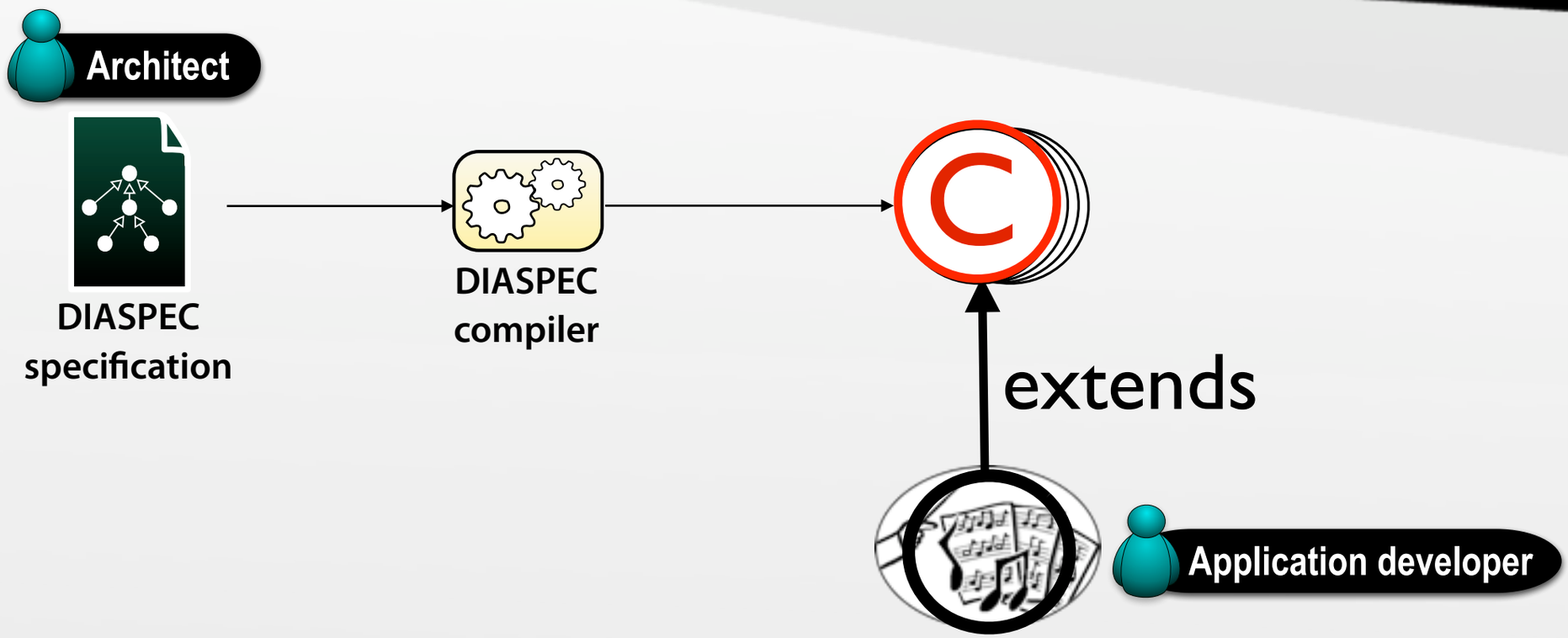
Dedicated Programming Frameworks

Developing

Programming Frameworks

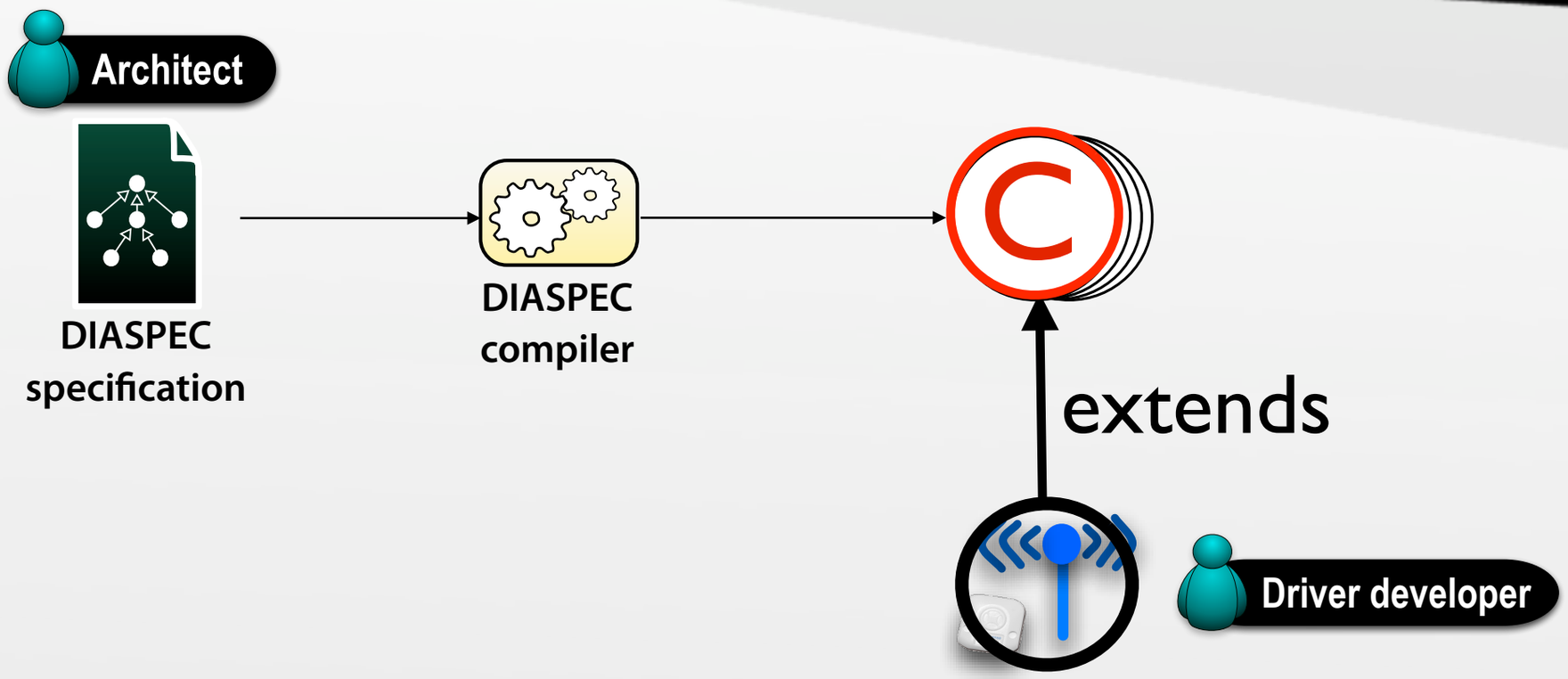


Programming Support




```
class MyNewscastMng extends NewscastManager {
   [...]
}
```

Programming Support




```
class MyTagReader extends TagReader {
   [...]
}
```

Programming Support

- To interact with services
- To abstract underlying technologies
- To statically verify applications

Programming Support

- To interact with services
- To abstract underlying technologies
- To statically verify applications

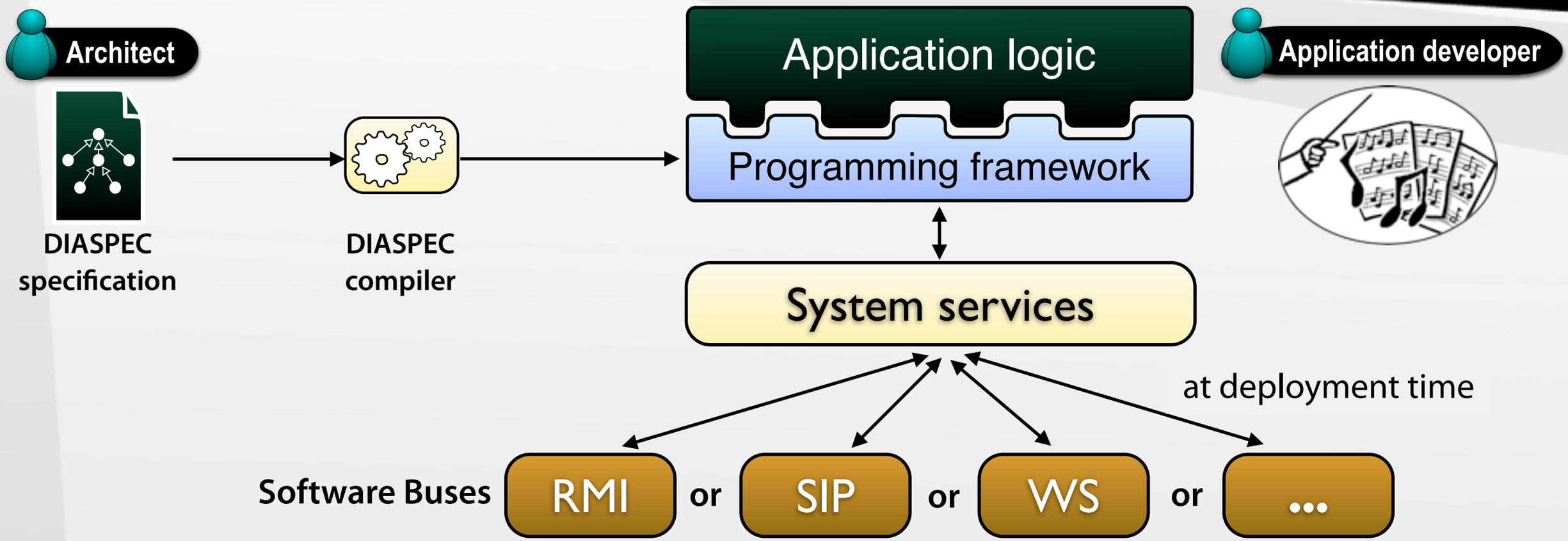
Interact With Services

- Service filtering
  - Service discovery
  - Event subscription / unsubscription
- Data exchange
  - Invoking commands
  - Receiving events

Programming Support

- To interact with services
- To abstract underlying technologies
- To statically verify applications

Abstracting Underlying Technologies



- To simplify the development of applications
- To enable the portability of applications

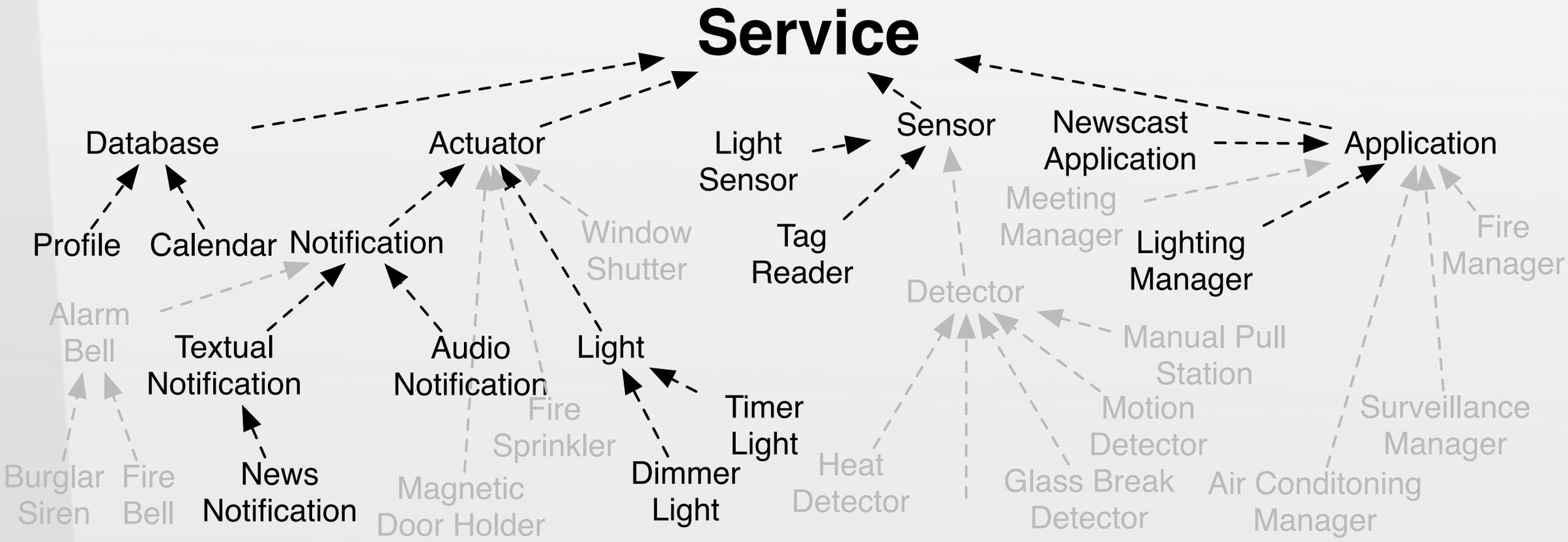
Programming Support

- To interact with services
- To abstract underlying technologies
- To statically verify applications

To Statically Verify Applications

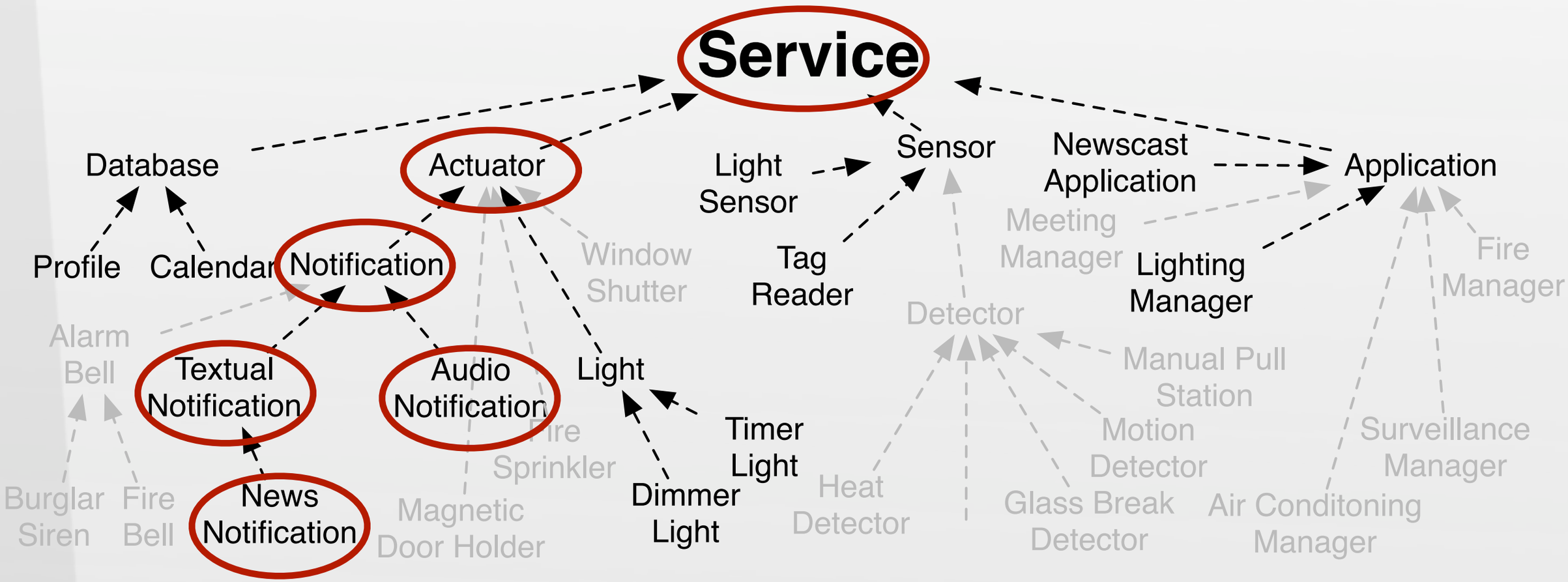
- A *service* may only communicate with the *services* it is connected to in the specification *(communication integrity)*
- How? => Using the Java type system
  - By supporting interaction with declared service types and its descendants
  - By forbidding to communicate with other types of services

Service Filtering

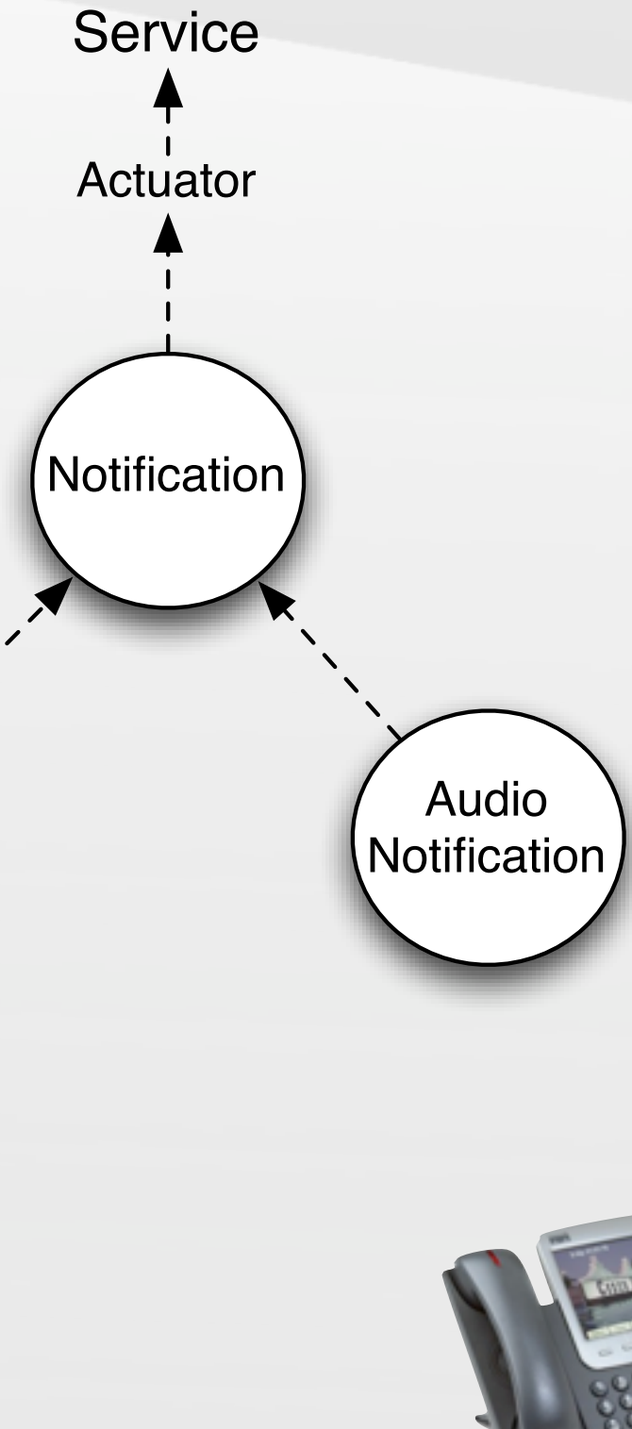


INRIA

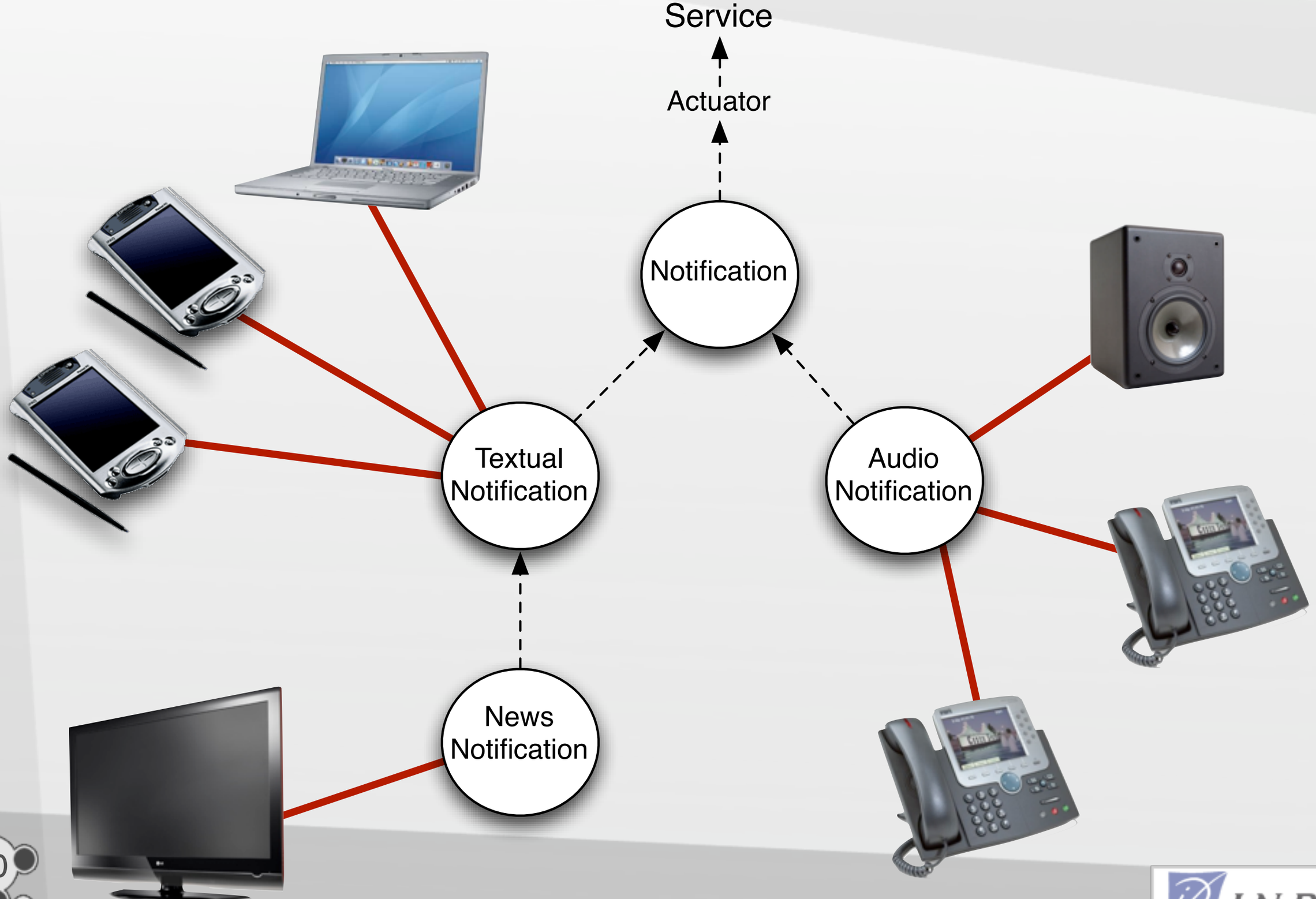
Service Filtering



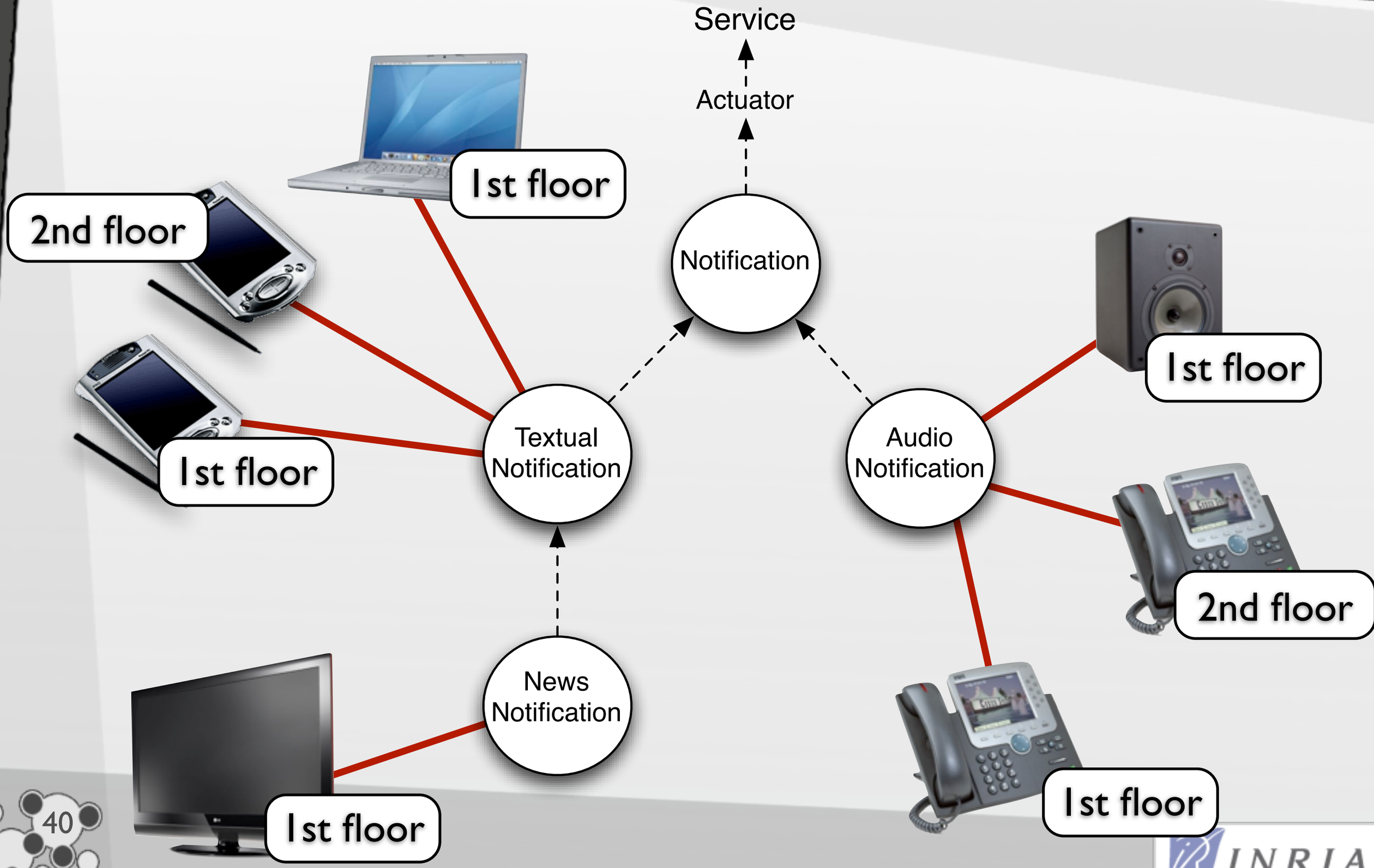
Service Filtering



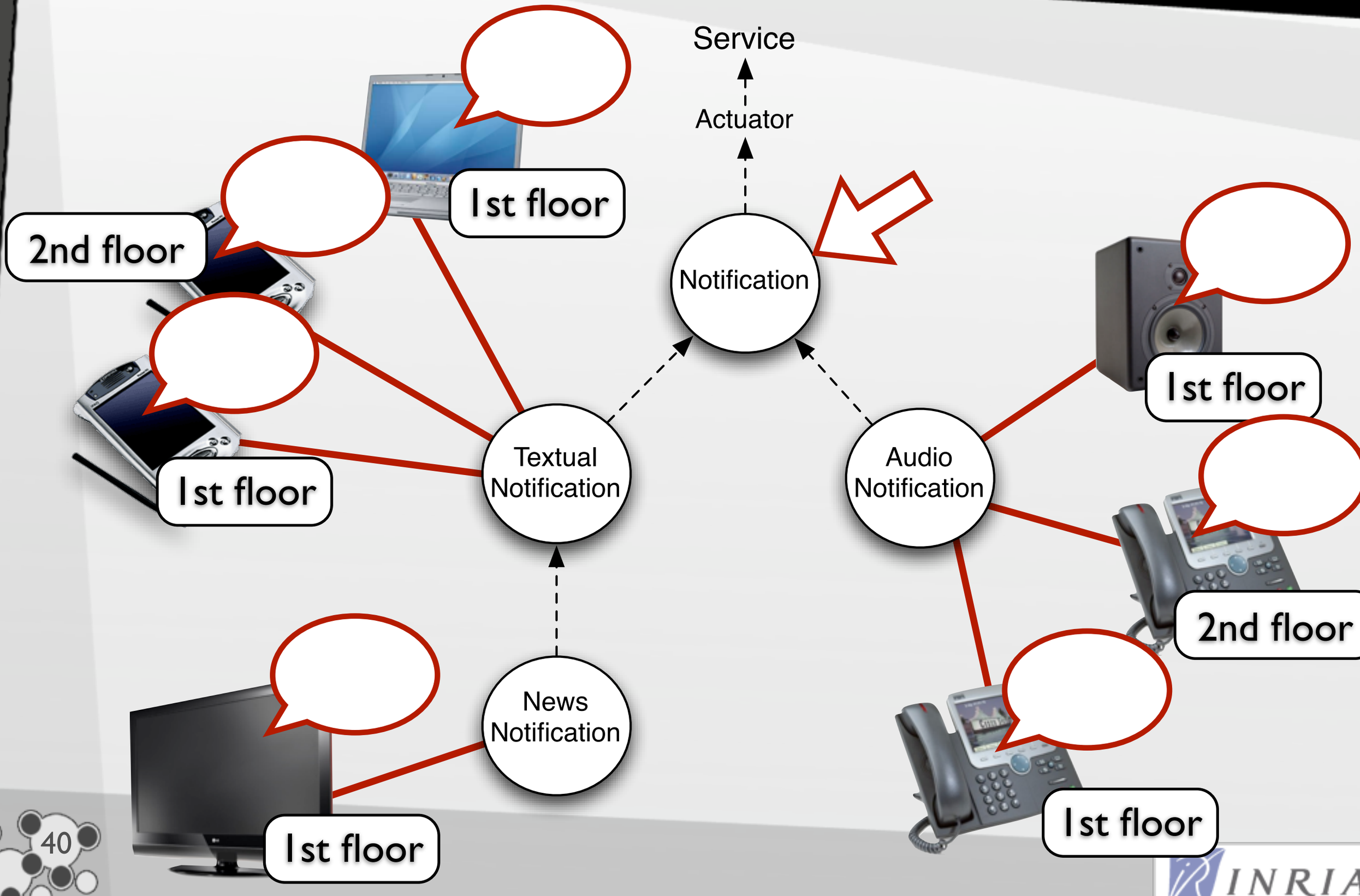
Service Filtering



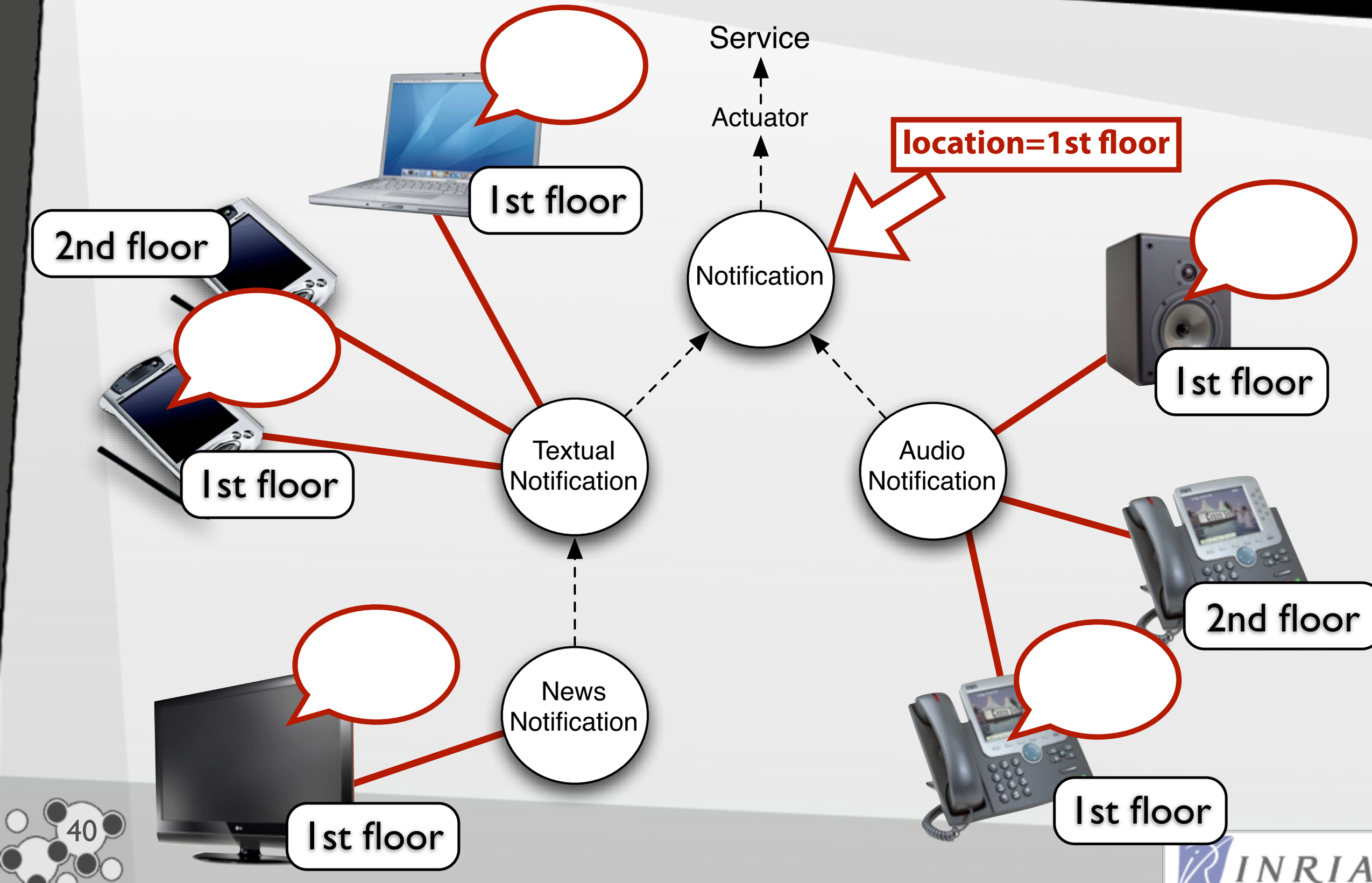
Service Filtering



Service Filtering



Service Filtering



Service Registration

DIASPEC

```
service Service(Owner owner) { }
service Actuator(Location location) extends Service { }
[...]
service NewsNotification() extends TextualNotification { }
```

Framework

```
class MyNewsNotification extends NewsNotification {
   MyNewsNotification() {
      super(Owner.ADMIN, new Hall());
      [...]
   }
   [...]
}
```

Service Registration

DIASPEC

```
service Service(Owner owner) { }
service Actuator(Location location) extends Service { }
[...]
service NewsNotification() extends TextualNotification { }
```

Framework

```
class MyNewsNotification extends NewsNotification {
   MyNewsNotification() {
      super(Owner.ADMIN, new Hall());
      [...]
   }
   [...]
}
```

Service Filtering

DIASPEC

```
service NewsNotification(Location location) extends TextualNotification {
    ...
}
```

Framework

```
NewsNotificationFilter filter = NewsNotification.getFilter();
filter.setLocation(Location.Hall);
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                        getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                        getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                        getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                    getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                    getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                        getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                        getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

  public void display(News news) {
     // TODO Auto-generated by ECLIPSE
  }
}
```

Programming Command

newscast manager

Invoker

news notification

Invokee

DIASPEC

```
service NewscastManager() extends Manager {
    requires command DisplayNews from
NewsNotification;
}
```

```
service NewsNotification() extends
TextualNotification {
    provides command DisplayNews to
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   News news;
   NewsNotification notifier =
                        getDisplayNewsService(filter);
   notifier.display(news);
}
```

```
class MyNewsNotification extends
NewsNotification {

 public void display(News news) {
    // TODO Auto-generated by ECLIPSE
 }
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
   requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
      provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
  requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
    provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
  requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
    provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
  requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
    provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
   subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
  requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
    provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

 public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
 }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
   requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
      provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
   requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
      provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Event

newscast manager

Consumer

tag reader

Producer

DIASPEC

```
service NewscastManager() extends Manager {
   requires event Presence to TagReader;
}
```

```
service TagReader() extends Sensor {
      provides event Presence from
NewscastManager;
}
```

Framework

```
class MyNewscastMng extends NewscastManager
{
    subscribePresence(filter);

  public void receive(Presence presence) {
    // TODO Auto-generated by ECLIPSE
  }
}
```

```
class MyTagReader extends TagReader {
  [...]
    publish(new Presence(123456789,
                         new SchoolHall()));
  [...]
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
      binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
      provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
            = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                  connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
            disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
      binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
      provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
            = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                  connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
            disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
    binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
    provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
          = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
          disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
    binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
    provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
          = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
          disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
      binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
      provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
            = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                  connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
            disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
    binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
    provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
          = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
          disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

Programming Session

newscast manager

Binder

speaker

Invitee

DIASPEC

```
service NewscastManager() extends Manager {
      binds session AudioMessage from Speaker,
AnnouncementCenter
}
```

```
service Speaker(AudioCapacities) extends
Device {
      provides session AudioMessage to
NewscastManager
}
```

Framework

```
class MyNewscastMng extends NewscastManager {
  SpeakerFilter filter1;
  AnnouncementCenter filter2;
  AudioMessage msg;
  AudioMessageSession session
            = bindAudioMessage(filter1, filter2, msg);
}
```

```
class MySpeaker extends Speaker {
  public AudioMessageSession
                  connect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
  public void
            disconnect(AudioMessageSession s)
  { // TODO Auto-generated by ECLIPSE }
}
```

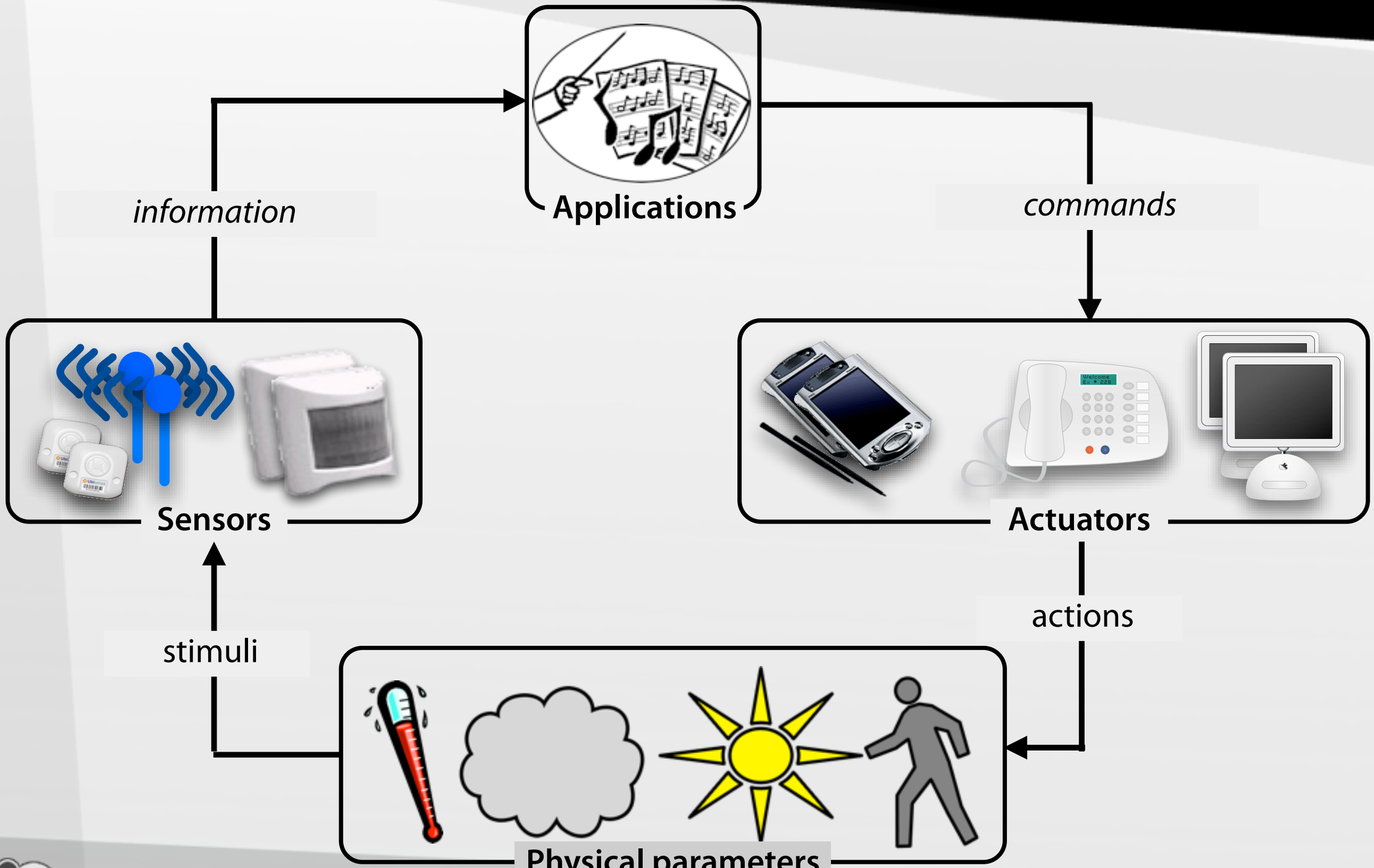
The DIASIM Approach: Applying DIAGEN To Simulation

Testing

Motivations

- Testing applications in actual environments
  - Time-consuming
  - Money-consuming
  - Not always possible
- Simulation
  - Fast if appropriate tools are provided
  - Cheap
  - Any scenarios

Ubiquitous Computing Environments



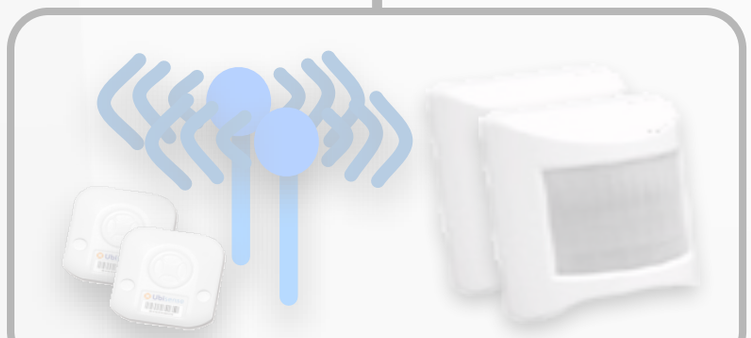
INRIA

Ubiquitous Computing Environments

Applications

information

commands



Sensors

Actuators

To simulate

stimuli

actions

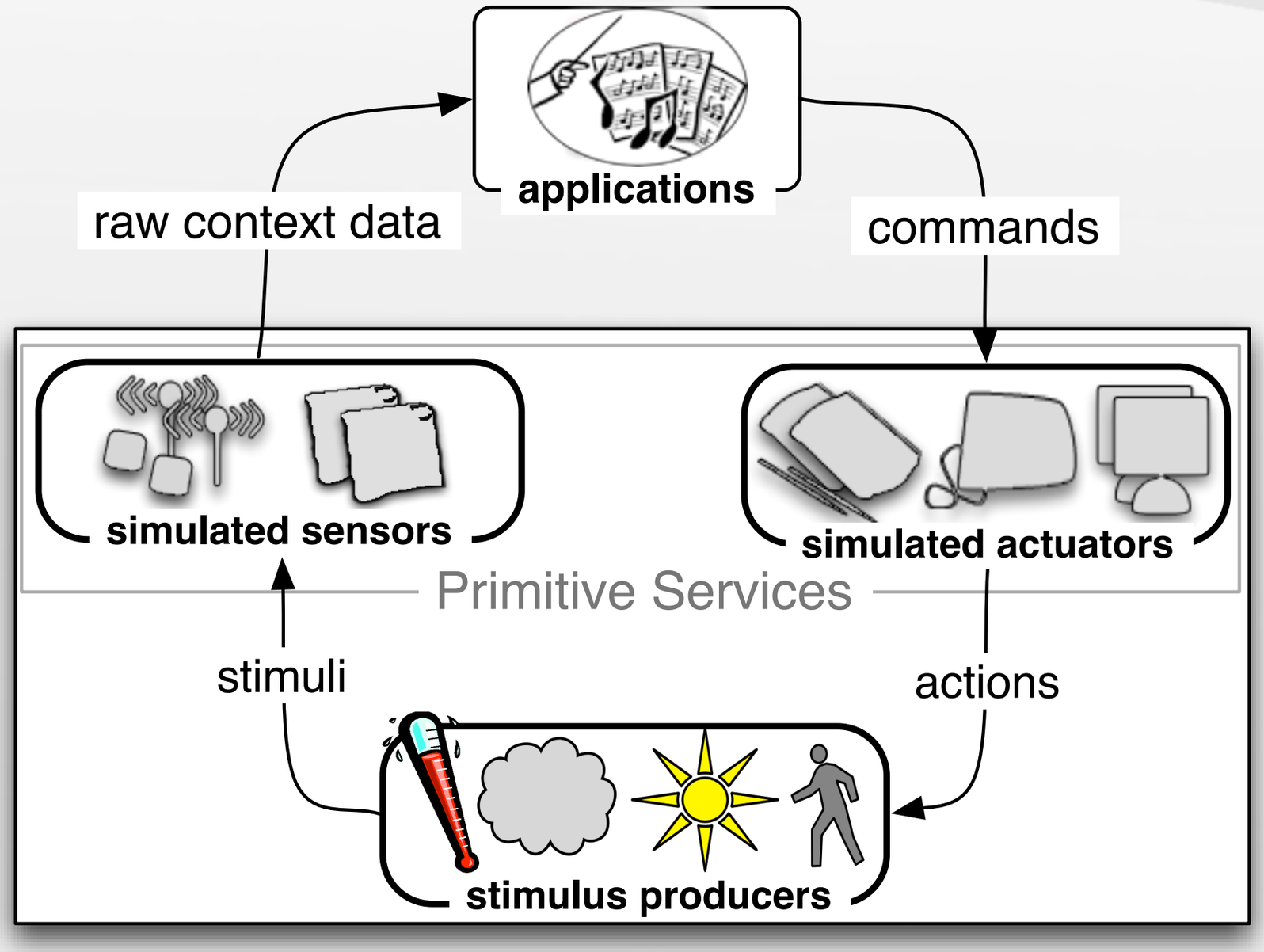
Physical parameters

INRIA

Requirements (1)

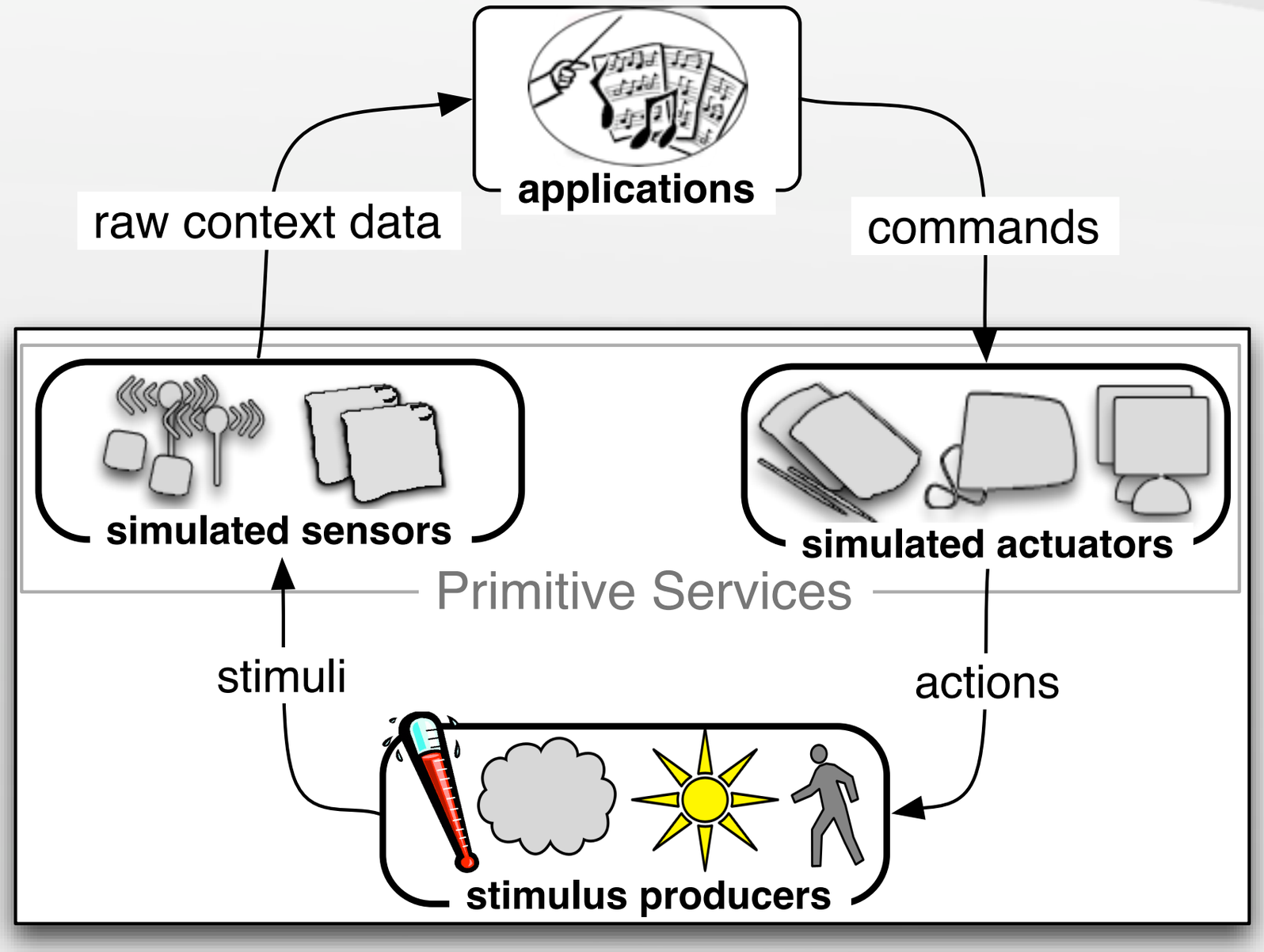
Testing applications against multiple simulated environments

Simulated Environment



Requirements (1)

Testing applications against multiple simulated environments

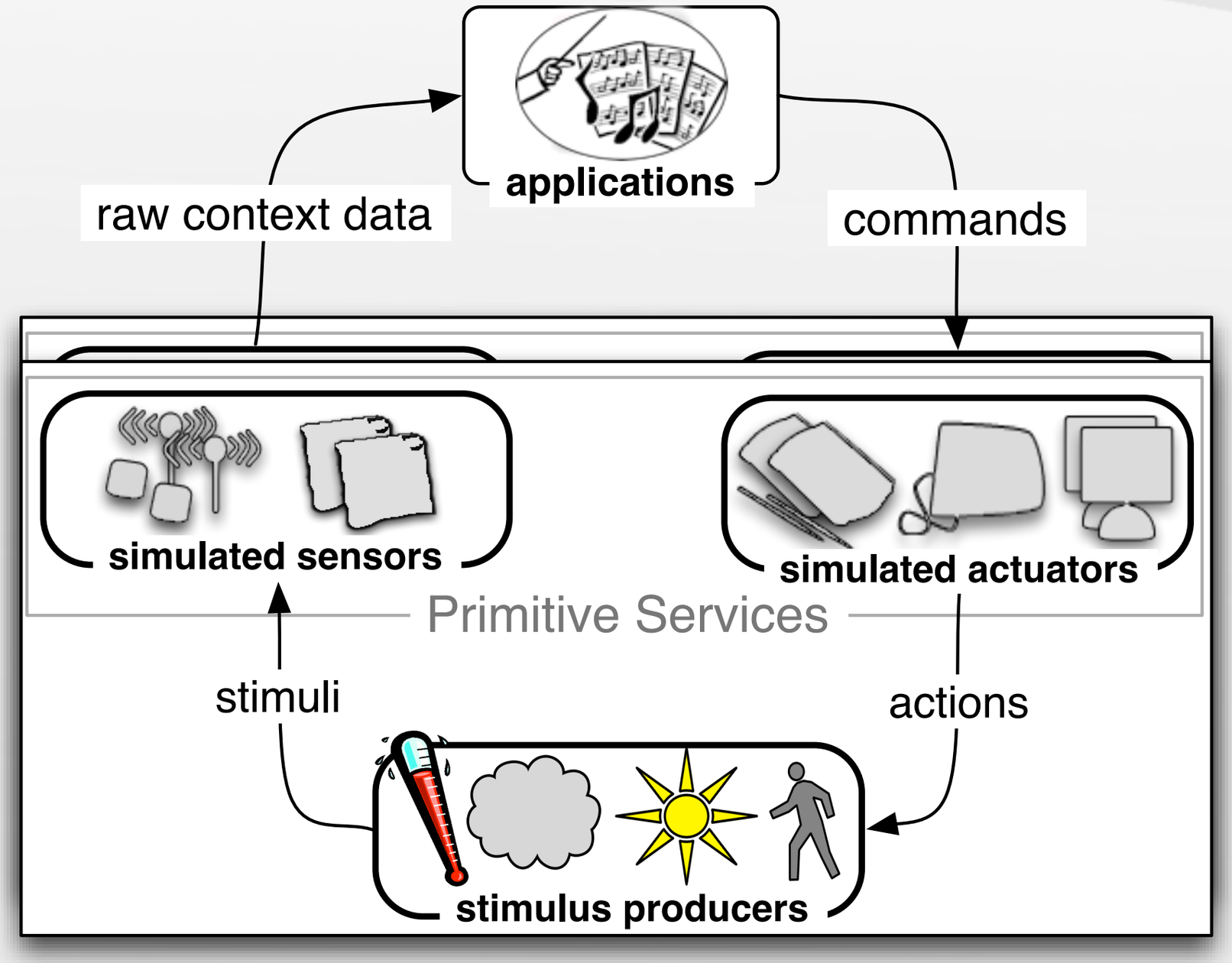


Simulated Environment

- [x] scenario 1

Requirements (1)

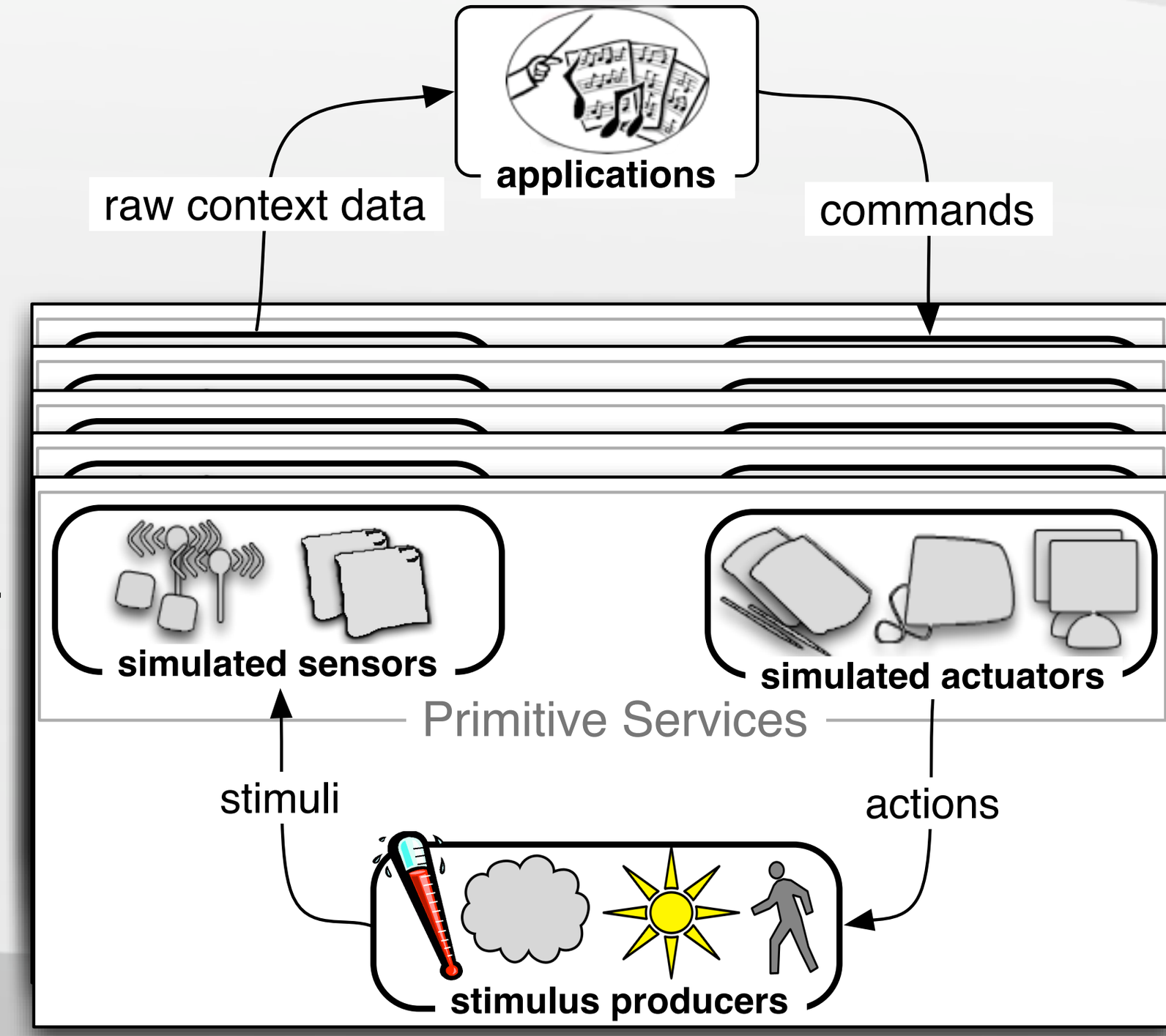
Testing applications against multiple simulated environments



- [x] scenario 1
- [x] scenario 2

Requirements (1)

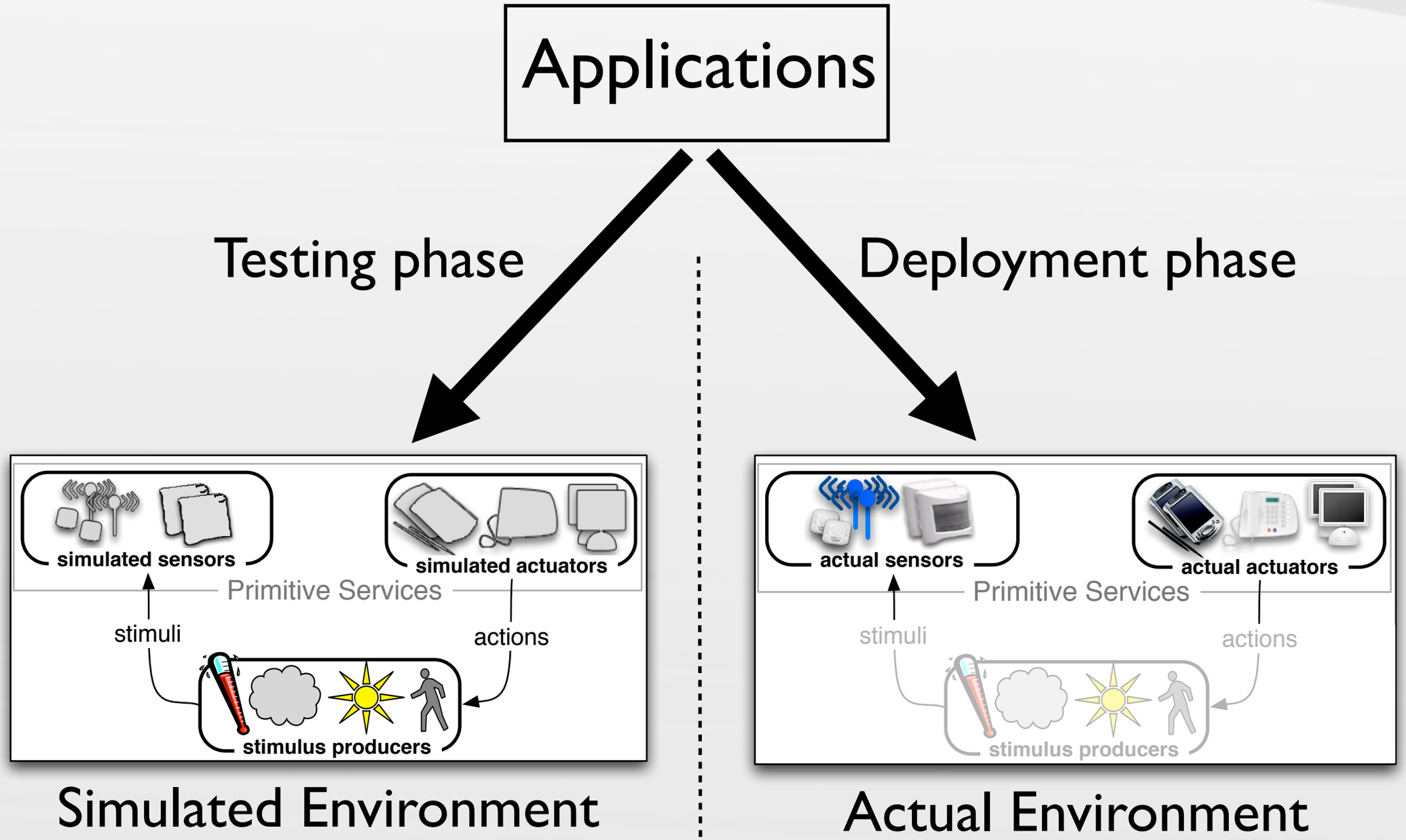
Testing applications against multiple simulated environments



- ☑ scenario 1
- ☑ scenario 2
- ☑ scenario 3
- ☑ scenario 4
- ☑ scenario 5
- ☑ ...

Requirements (2)

Testing applications without code modification



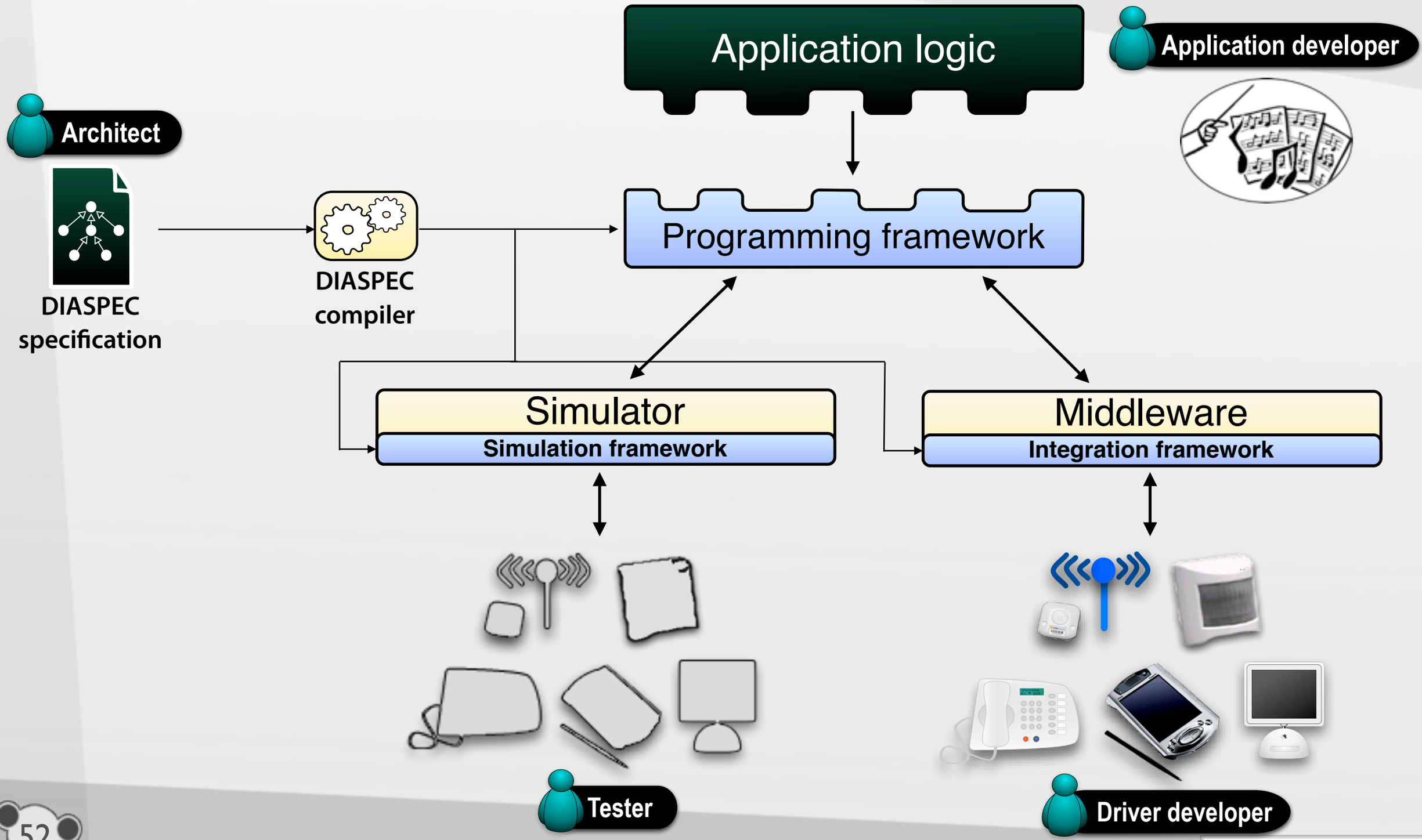
Simulated Environment

Actual Environment

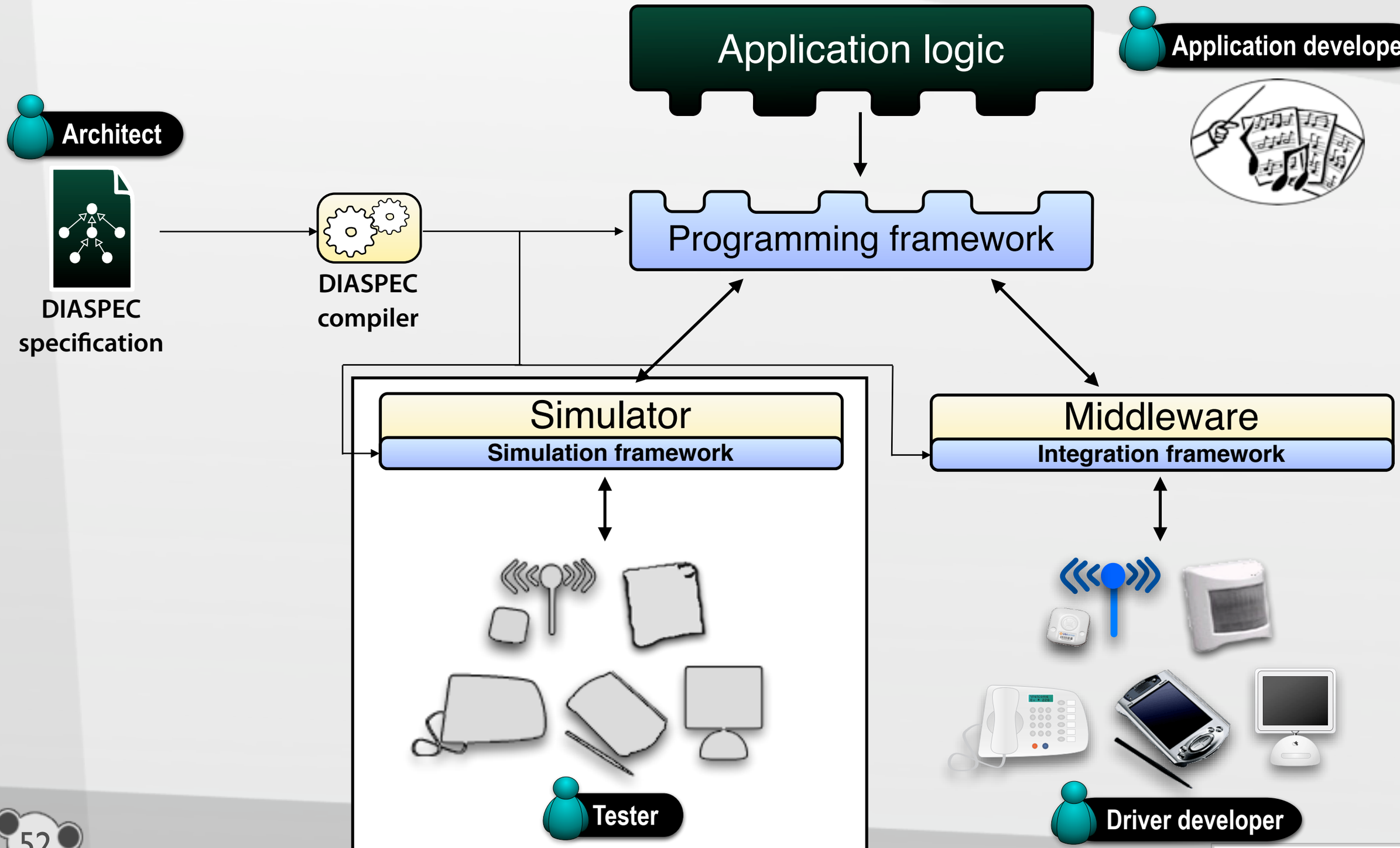
The DIASIM approach

- Testing applications from various application domains
  - Parameterizing the simulator w.r.t. a DIASPEC specification
- Testing applications against multiple simulated environments
  - Supporting the development of simulated services
  - Integration of actual primitive services
- Testing applications without code modification
  - Emulation of simulated environments

Testing

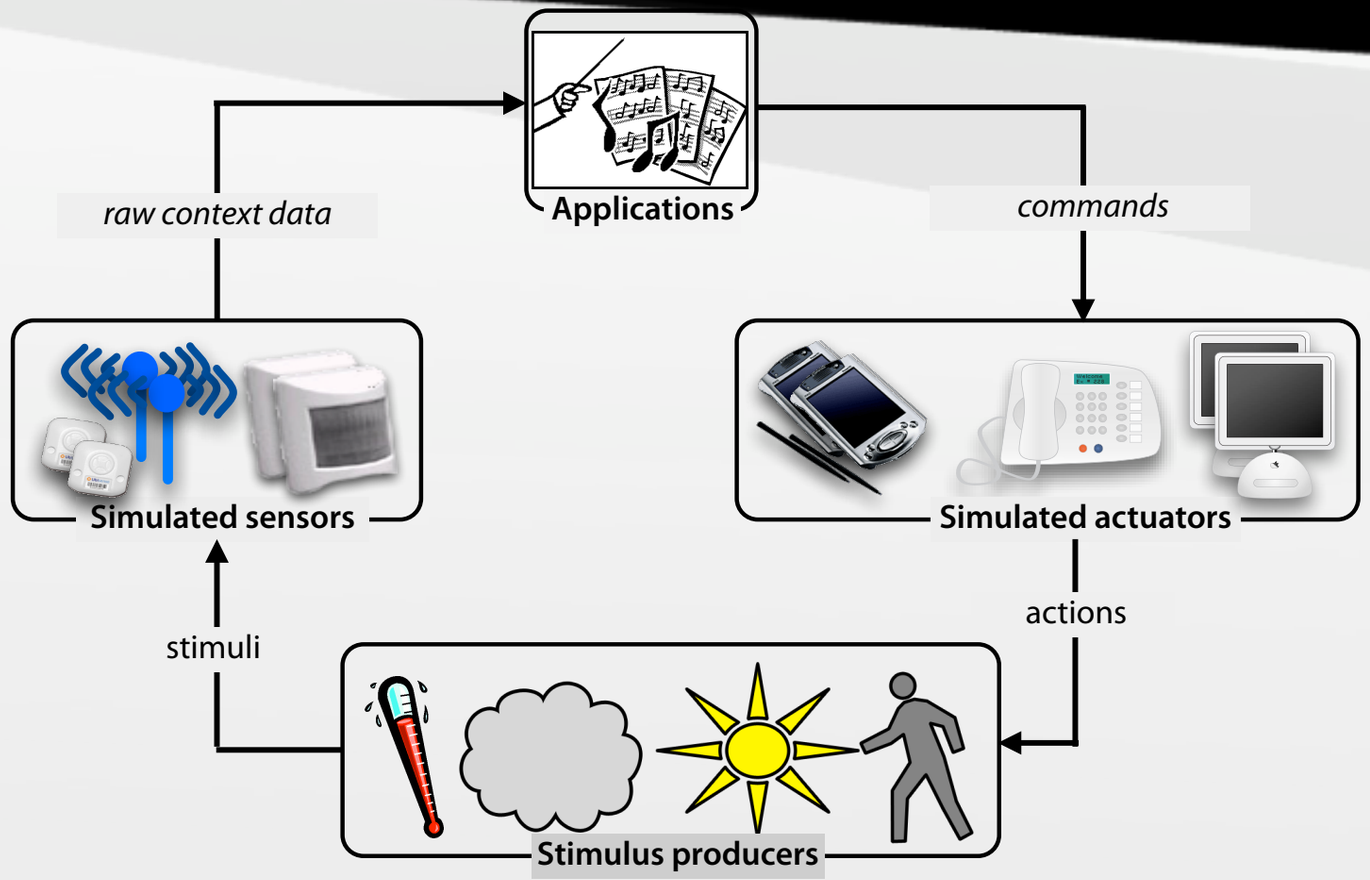


Testing



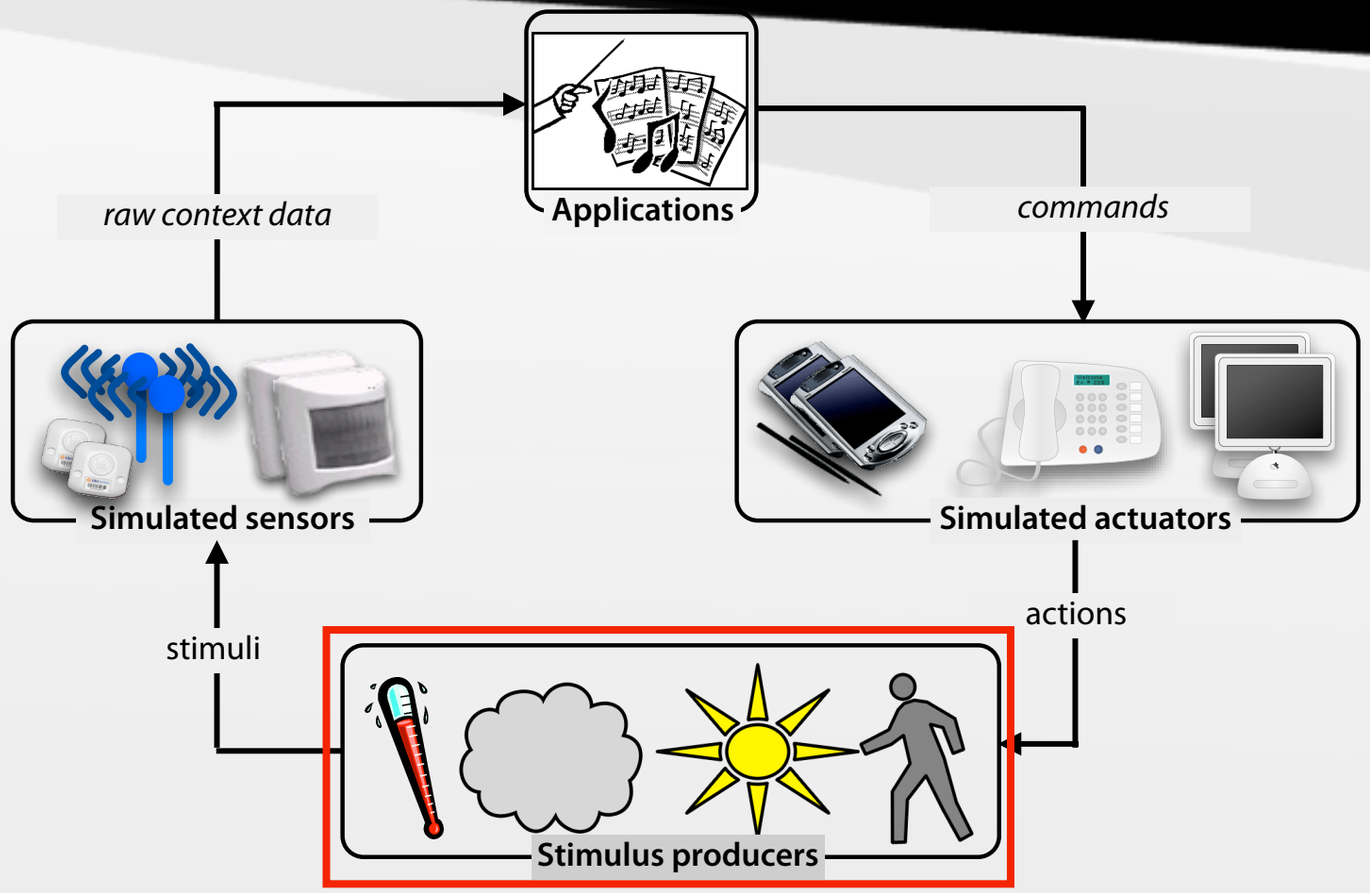
Specifying Simulated Environments

- Stimulus producers
- Simulated sensors
- Simulated actuators



Specifying Simulated Environments

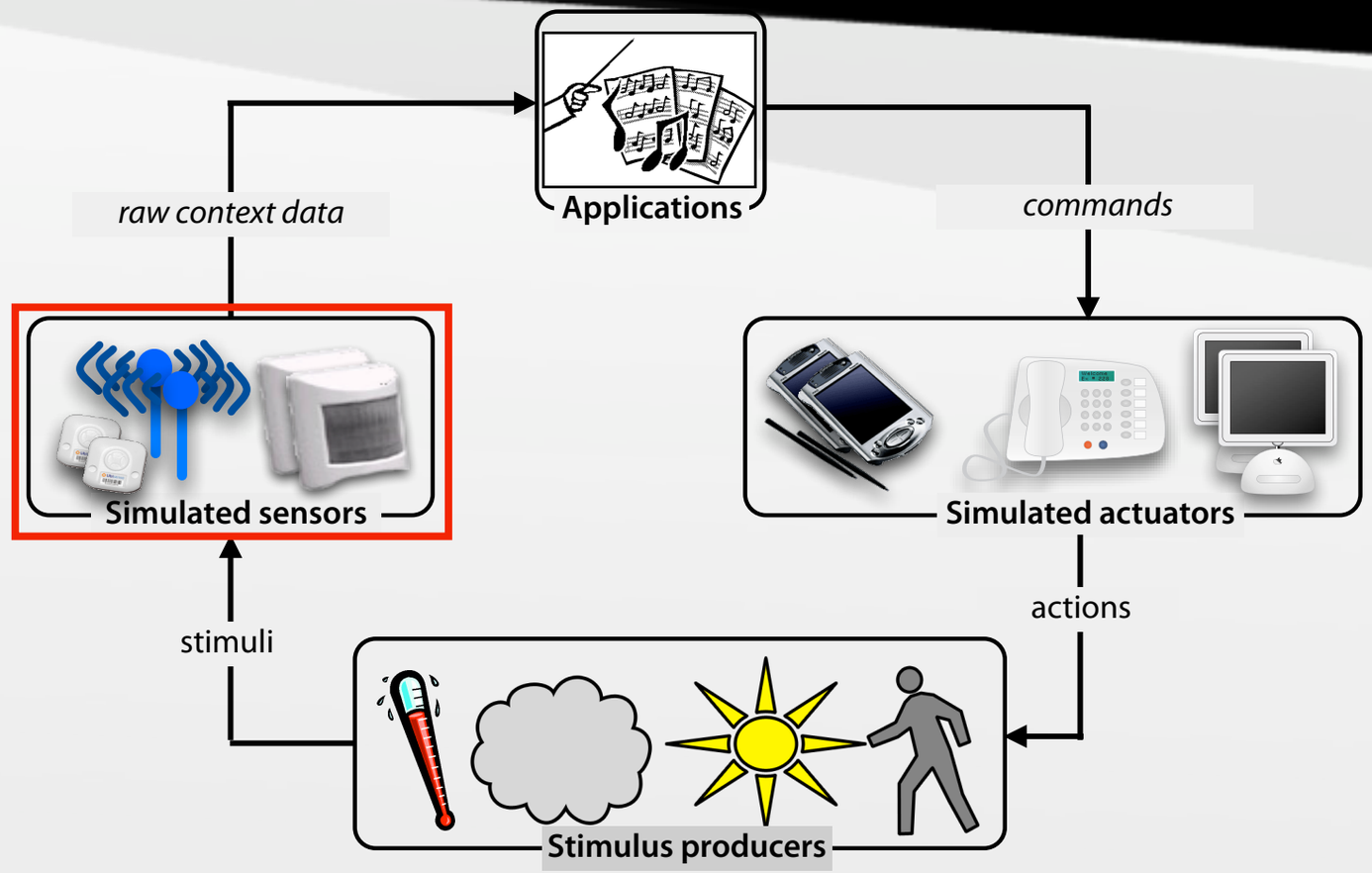
- Stimulus producers
- Simulated sensors
- Simulated actuators




```
service LuminosityStimulusProducer() {
   provides command ILuminosityStimulusProducer to [...];
}

icommand ILuminosityStimulusProducer {
   LuminosityStimulusProducer[] getStimuli(Time time);
}
```

Specifying Simulated Environments

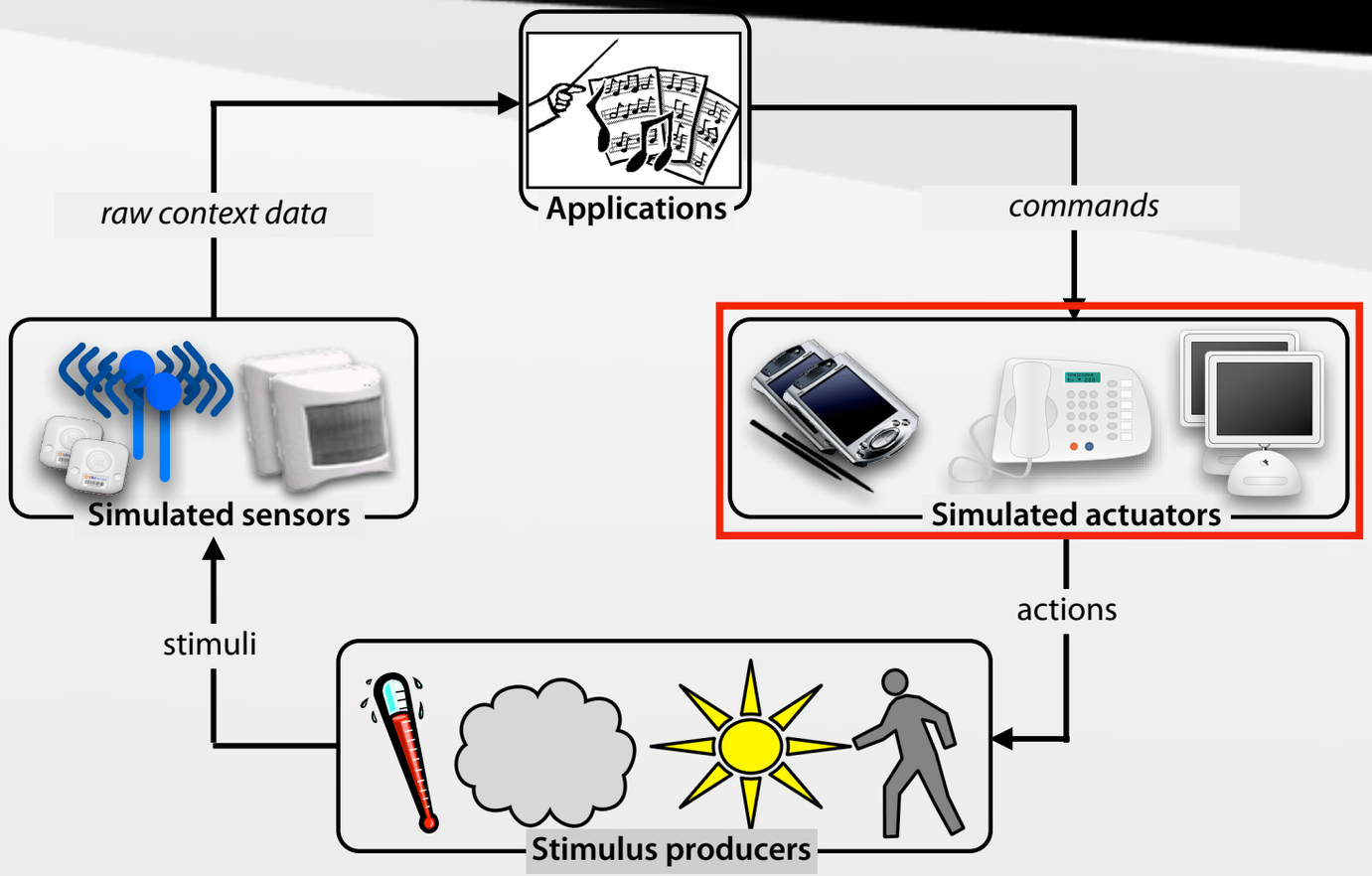
- Stimulus producers
- Simulated sensors
- Simulated actuators




```
service LightSensor() extends [...] {
   provides event Luminosity to [...];
}

service SimulatedLightSensor() extends LightSensor {
   requires event LuminosityStimulus from [...];
}
```

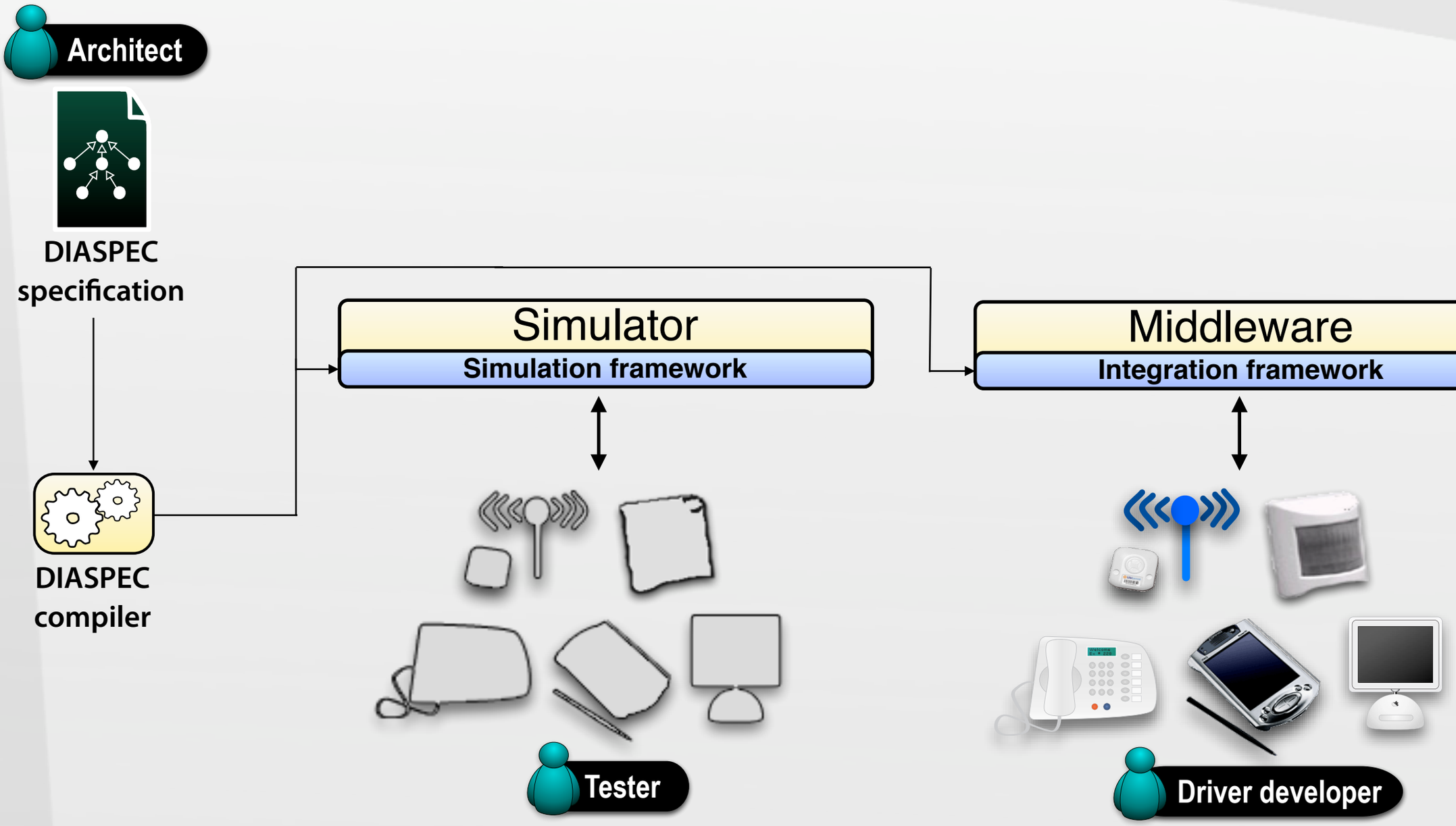
Specifying Simulated Environments

- Stimulus producers
- Simulated sensors
- Simulated actuators



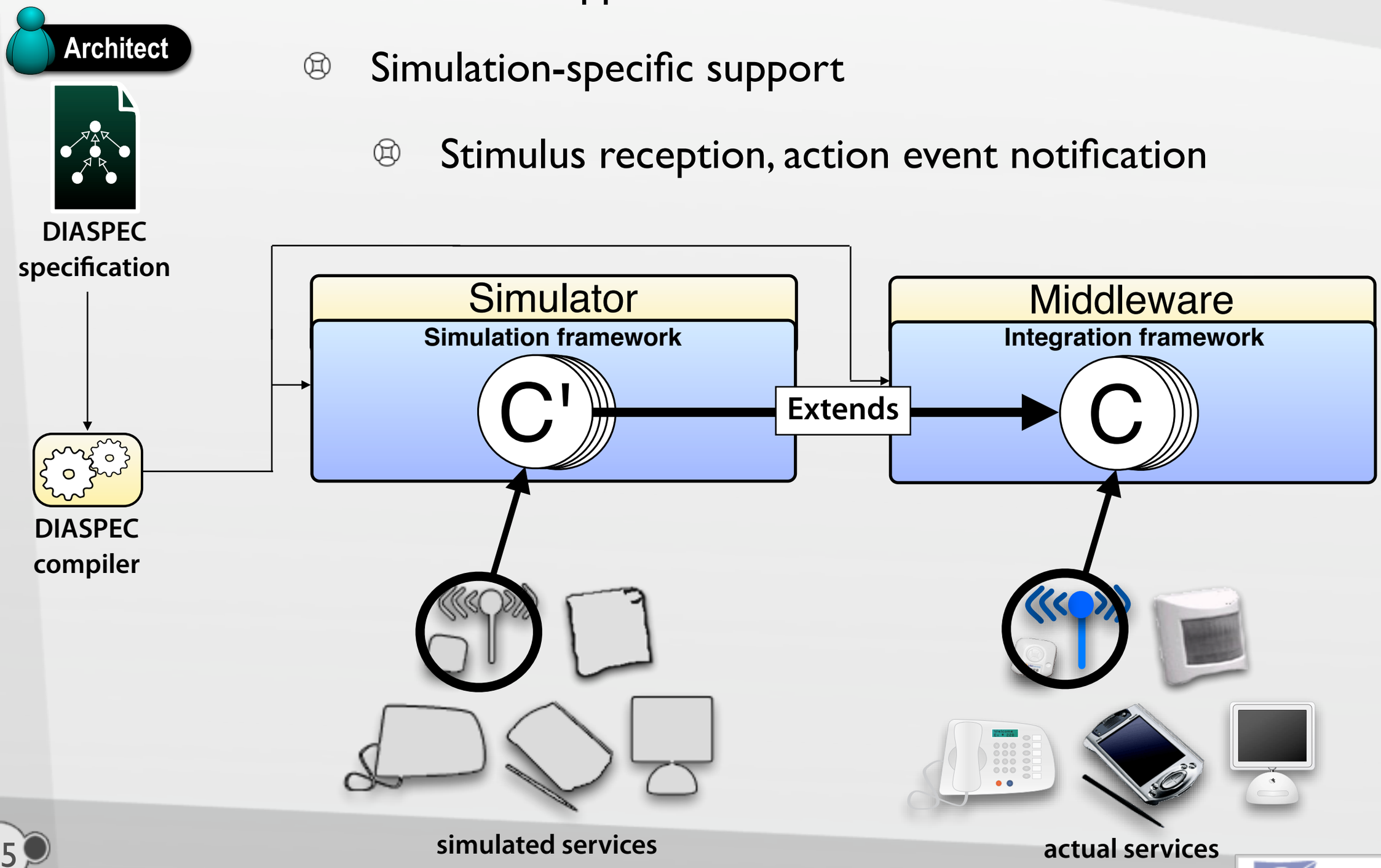

```
service Light() extends [...] {
   provides command ILight to [...];
}

service SimulatedLight() extends Light {
   provides event Action to [...];
}
```

Developing Simulated Environments



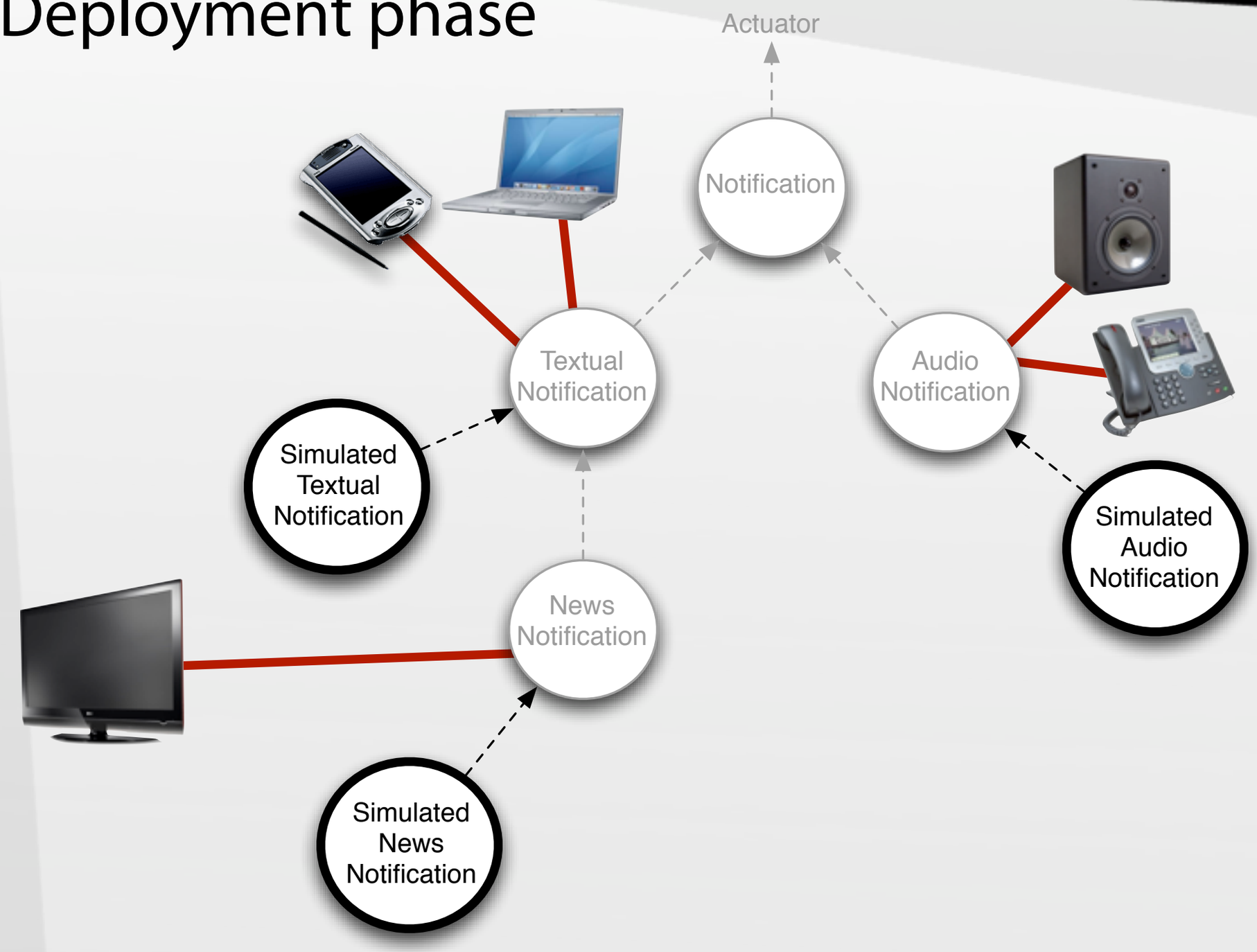
Developing Simulated Environments

- Inherited support from classes of actual services
- Simulation-specific support
  - Stimulus reception, action event notification



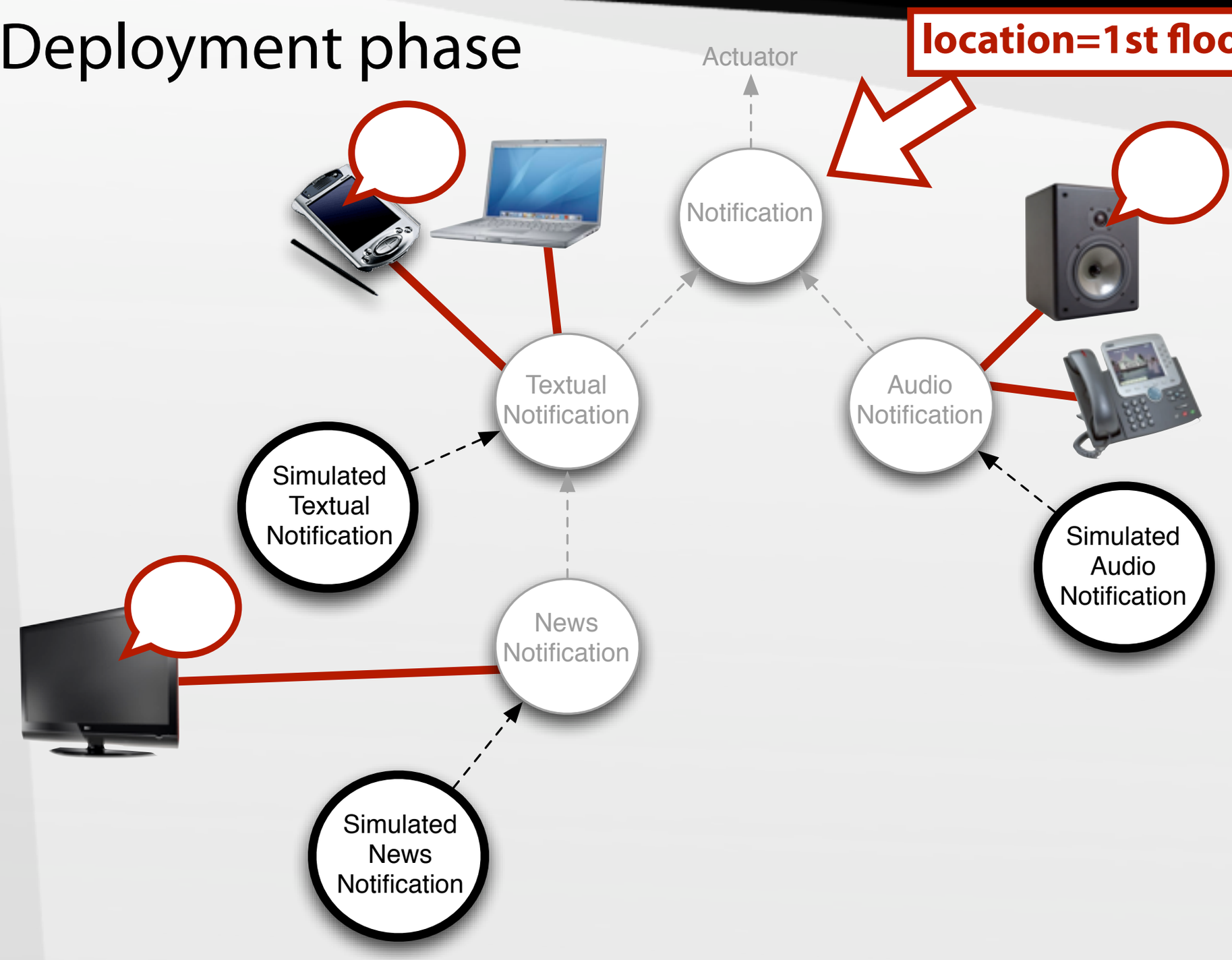
Testing Applications Without Code Modification

Deployment phase



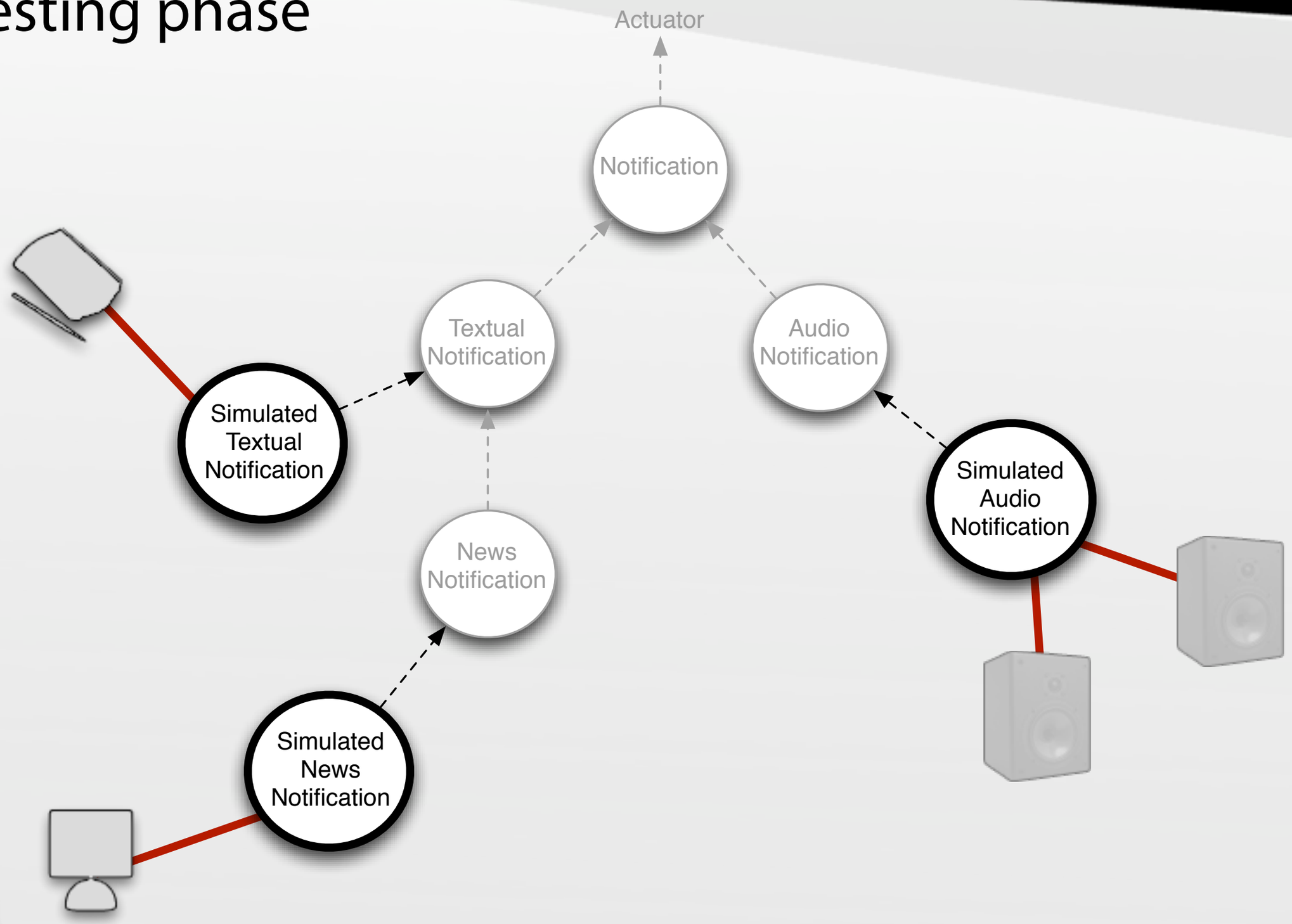
Testing Applications Without Code Modification

Deployment phase



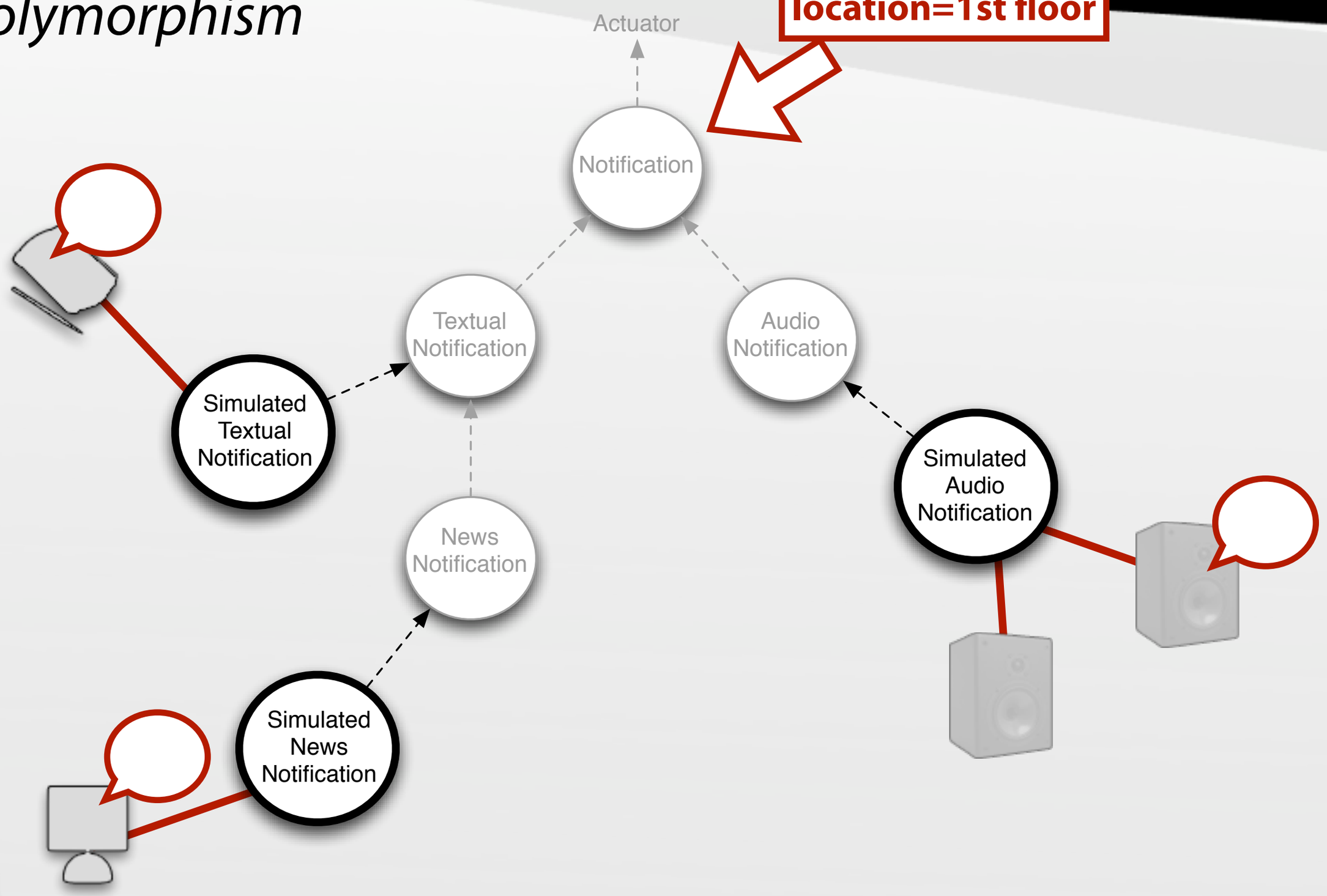
Testing Applications Without Code Modification

Testing phase



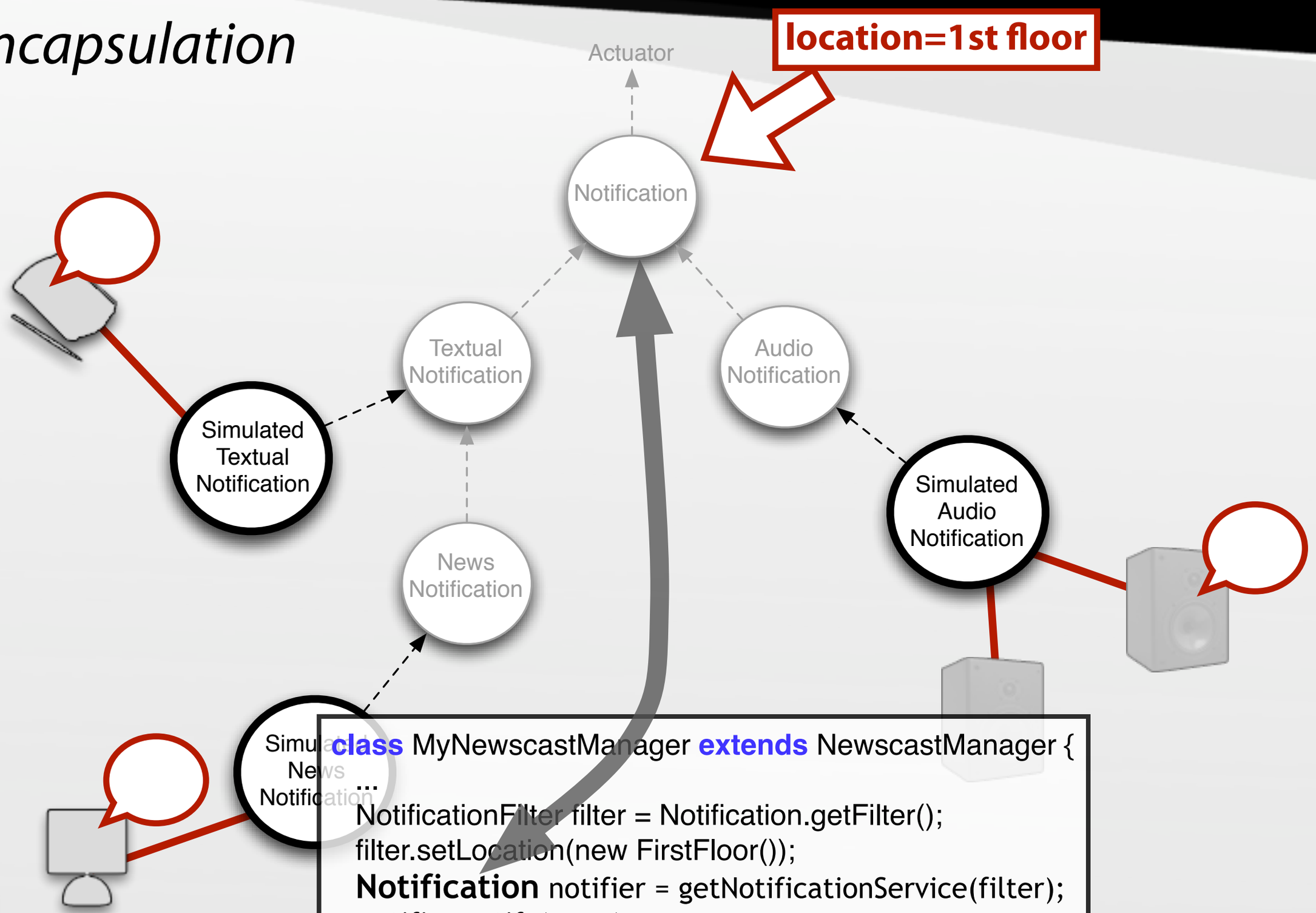
Testing Applications Without Code Modification

Polymorphism



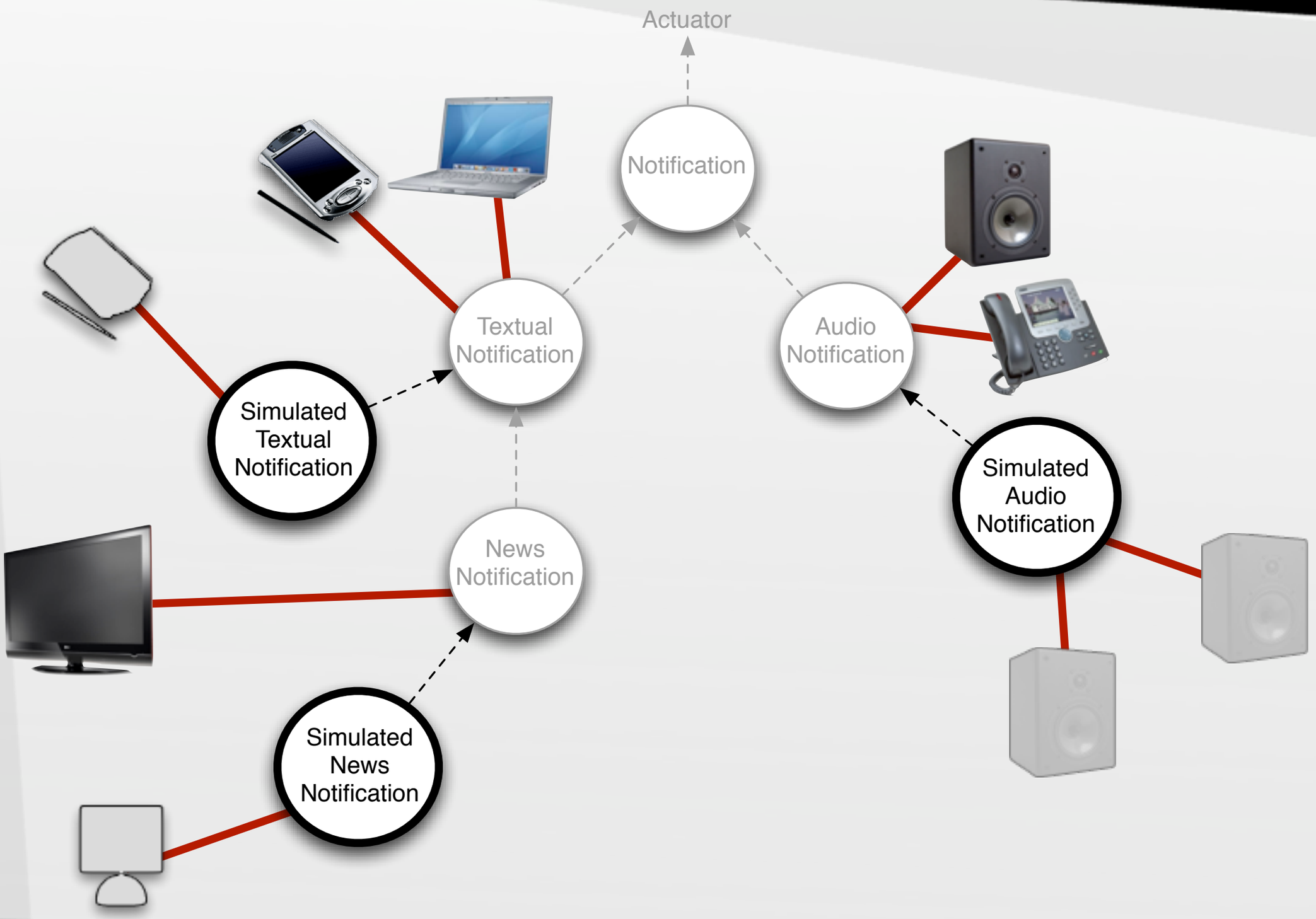
Testing Applications Without Code Modification

Encapsulation

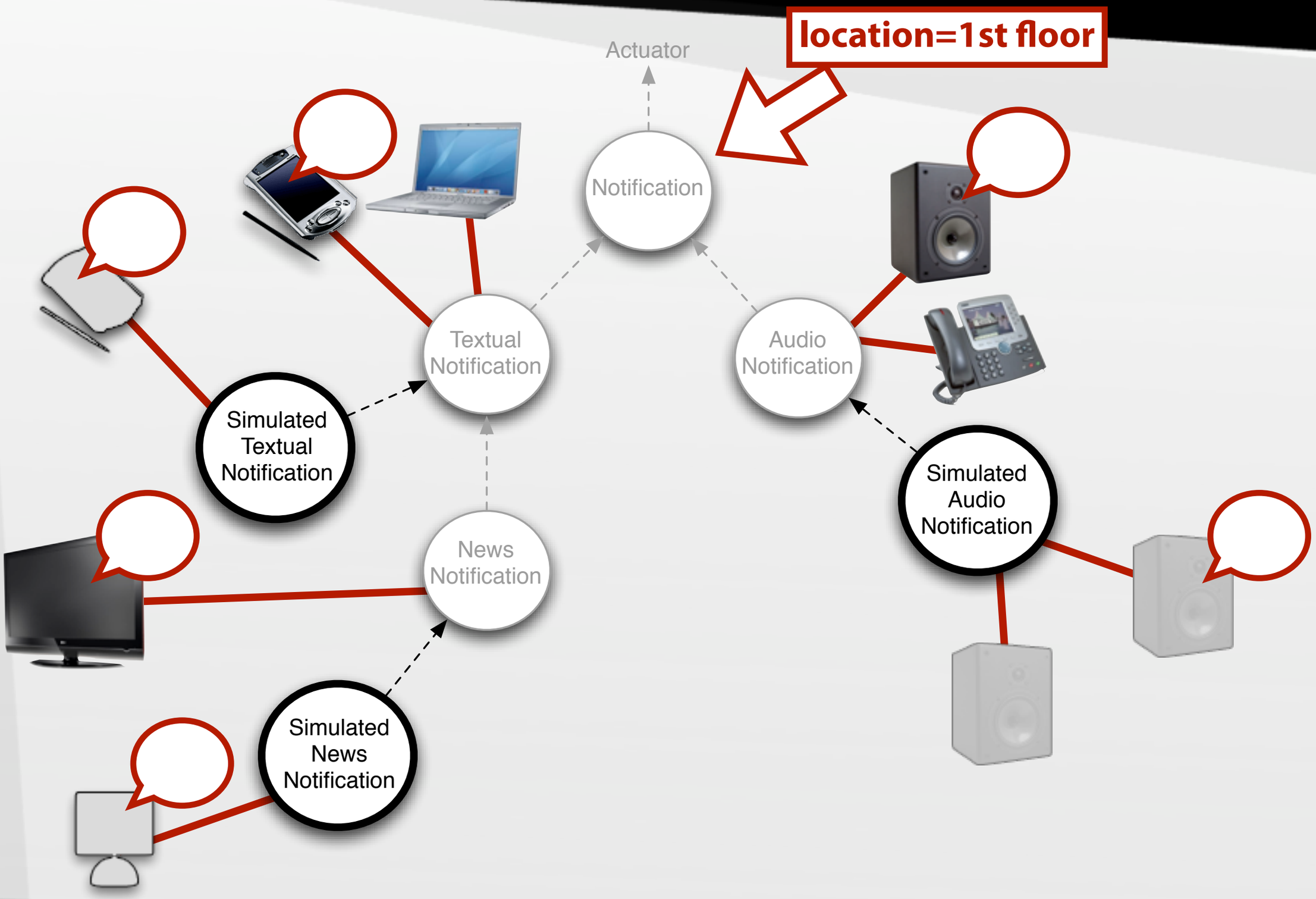



```
class MyNewscastManager extends NewscastManager {
    ...
    NotificationFilter filter = Notification.getFilter();
    filter.setLocation(new FirstFloor());
    Notification notifier = getNotificationService(filter);
    notifier.notify(news);
    ...
}
```

Testing Applications In Hybrid Environments



Testing Applications In Hybrid Environments



Testing Applications In Hybrid Environments

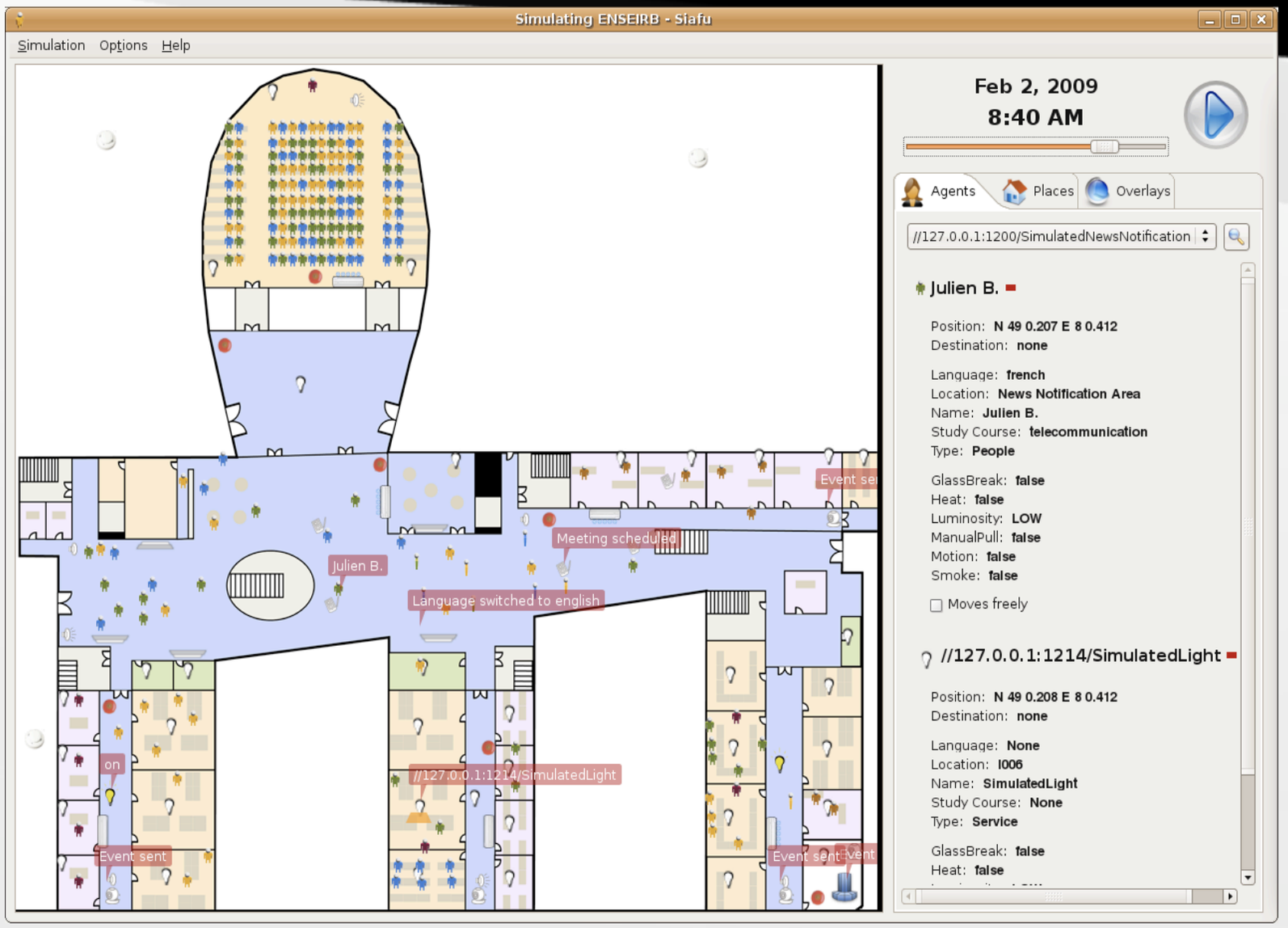
Why integrating actual entities?

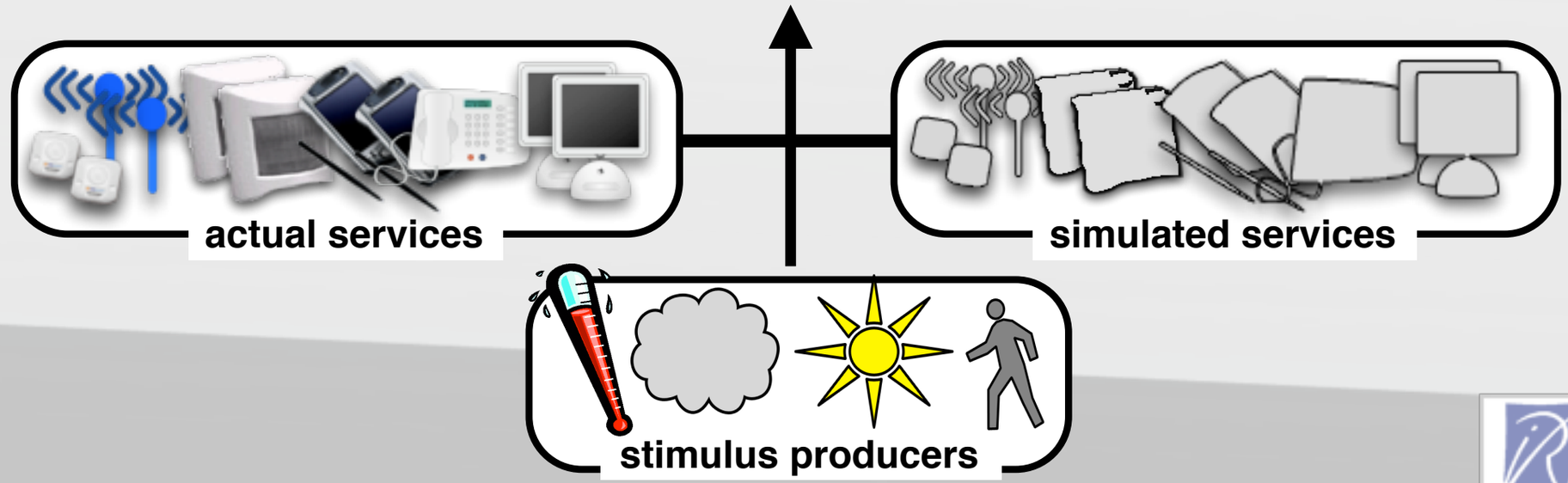
Reuse

Test

INRIA

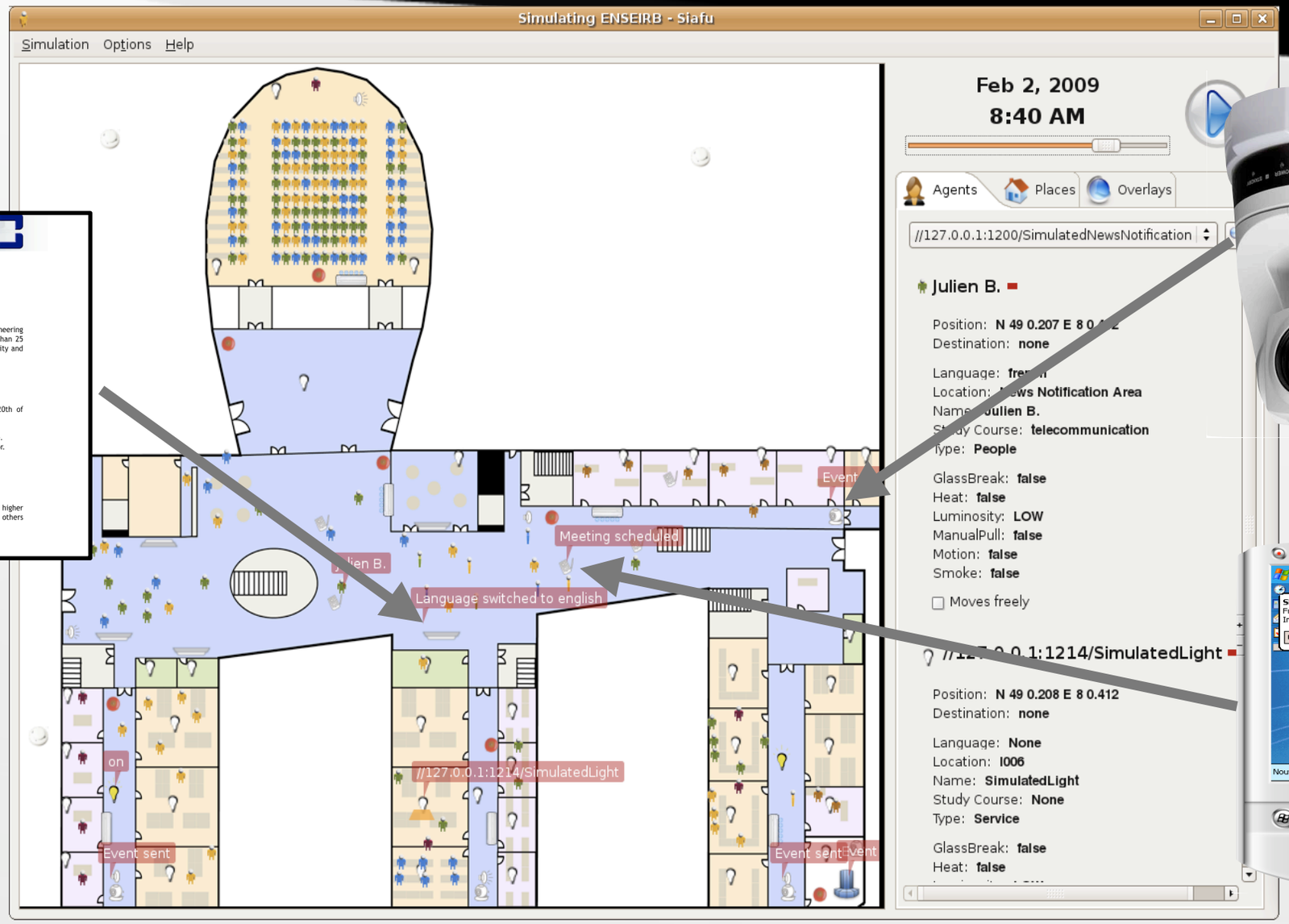
Simulation Renderer

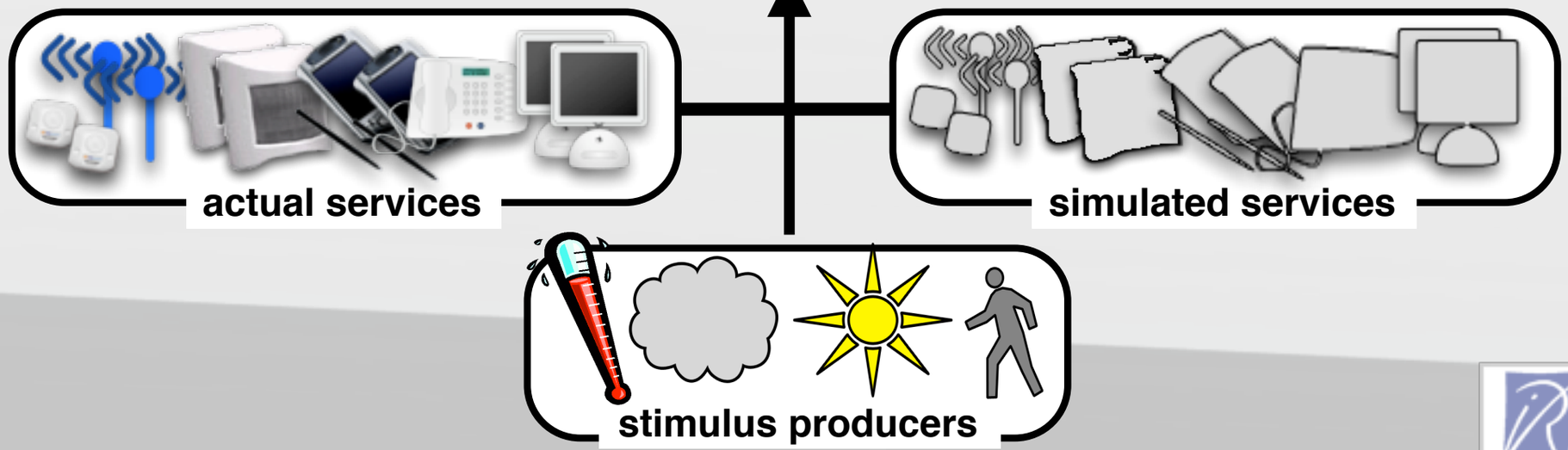




INRIA

Simulation Renderer





INRIA

Conclusion

Conclusion

- Integrated approach for specifying, developing and testing
- Parameterized approach
- Programming frameworks
  - Dynamicity and communication integrity
  - Abstracting underlying technologies
- Test of applications
  - Without code modification
  - In hybrid environments

Related Projects

- Pantaxou approach - *Julien Mercadal, Nicolas Palix (GPCE'08)*
  - Domain Specific Language
- Pantagruel approach - *Zoé Drey, Julien Mercadal (DSL'09)*
  - Visual language
- HomeSIP project - *Orange Labs (IPTComm'08)*

Perspectives

- Ubiquitous computing concerns
  - Context
  - Dynamic reconfiguration
- Non-functional concerns
  - Security
  - Quality of Service
  - Concurrence
- Deployment framework

Publications

DIAGEN approach

- **W. Jouve**, N. Palix, C. Consel and P. Kadionik. « A SIP-based Programming Framework for Advanced Telephony Applications ». In proceedings of IPTComm'08, Heidelberg, Germany, July 2008. Awarded Best Student Paper Award.
- **W. Jouve**, J. Lancia, N. Palix, C. Consel, and J. Lawall. « High-level Programming Support for Robust Pervasive Computing Applications ». In proceedings of PerCom'08, Hong Kong, China, March 2008 (WiP Session).
- C. Consel, **W. Jouve**, J. Lancia, and N. Palix. « Ontology-Directed Generation of Frameworks For Pervasive Service Development ». In proceedings of PerWare'07, White Plains, New York, USA, March 2007 (Workshop).

DIASIM approach

- **W. Jouve**, J. Bruneau, and C. Consel. « DiaSim : A Parameterized Simulator for Pervasive Computing Applications ». In proceedings of PerCom'09, Galveston, Texas, March 2009 (Demo).
- **W. Jouve**, J. Bruneau, and C. Consel. « DiaSim : A Parameterized Simulator for Pervasive Computing Applications ». Submitted to Mobiquitous'09.

Applications

- **W. Jouve**, N. Ibrahim, L. Réveillère, F. Le Mouel, et C. Consel. « Building Home Monitoring Applications : From Design to Implementation into The Amigo Middleware ». In proceedings of ICPCA'07, Birmingham, UK, July 2007.
- Y.-D. Bromberg, C. Consel, **W. Jouve**, S. Ben Mokhtar, N. Georgantas, V. Issarny, and P.-G. Raverdy. « Middleware for ubiquitous computing ». In « ARAGO 31 : Ubiquitous Computing » OFTA report. May 2007.

Session interaction mode in Web Services

- **W. Jouve**, J. Lancia, C. Consel, and C. Pu. « A Multimedia-Specific Approach to WS-Agreement ». In Proceedings of ECOWS'06, Zurich, Switzerland, December 2006.