8th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2014)

Features on Demand

Nicolás Cardozo[2], Kim Mens[1], Sebastián González[1],
Pierre-Yves Orban[1], Wolfgang De Meuter[2]

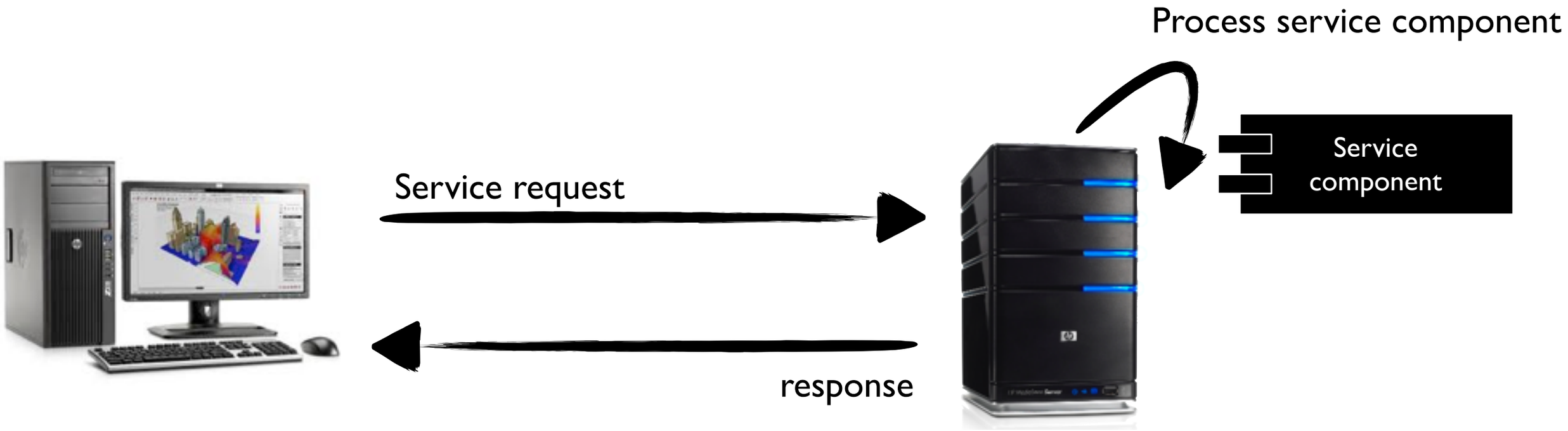
23 – 01 – 2014

[1] Université catholique de Louvain (kim.mens@uclouvain.be)
[2] Vrije Universiteit Brussel (ncardozo@vub.ac.be)

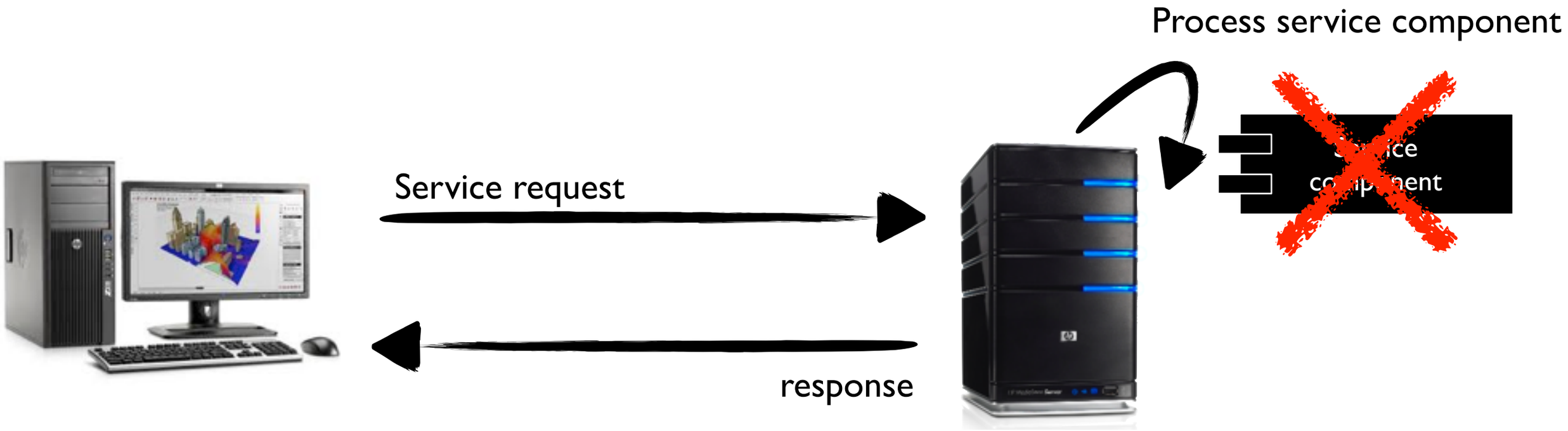
RELEASED

Software Languages Lab

Service composition



Service composition



Feature clouds vision

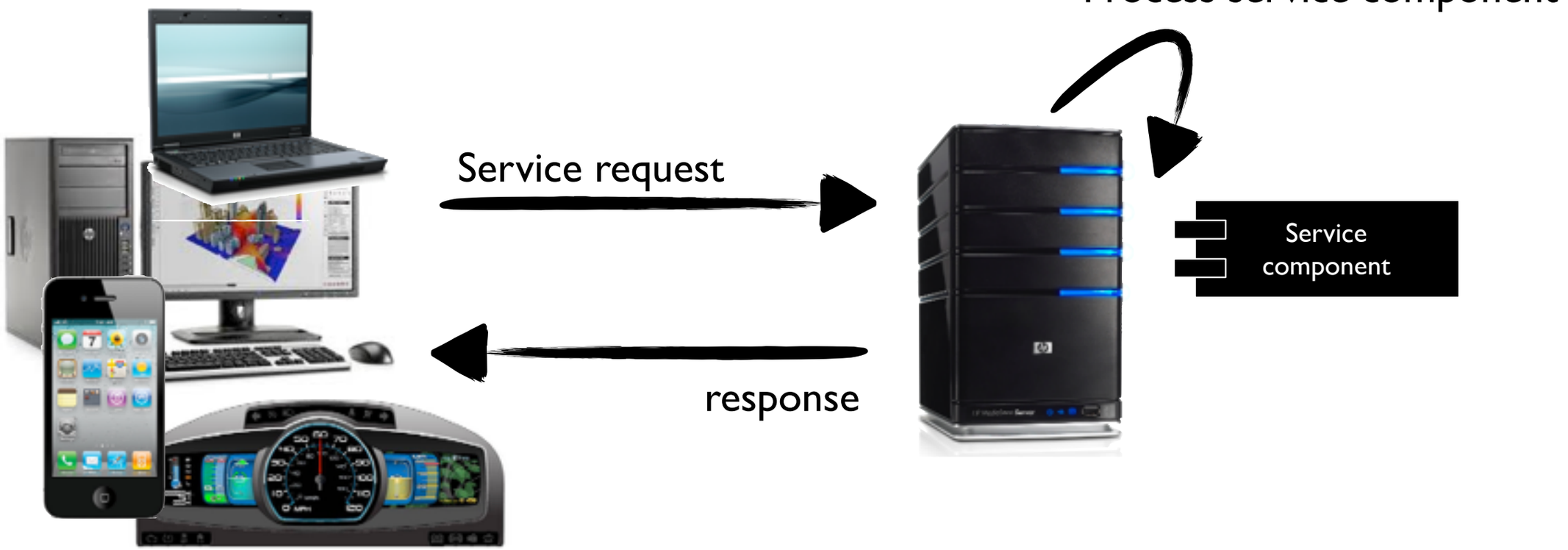
Service request

response

Process service component

Service component

Feature clouds vision



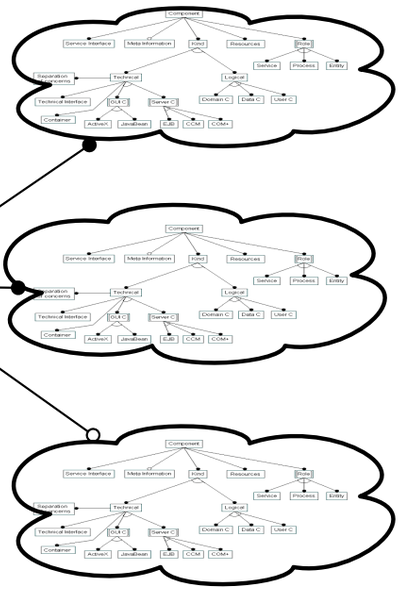
Feature clouds vision

Service request

response

Process service component

Service component

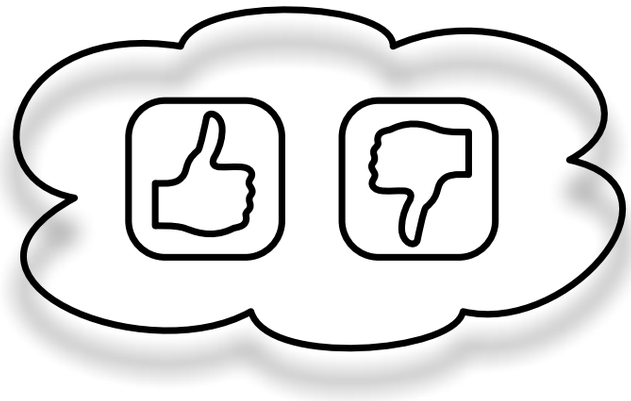


Features on demand

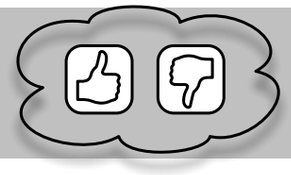
Motivating Example

Requirements

Feature Clouds Model



Conclusion



Motivating Example

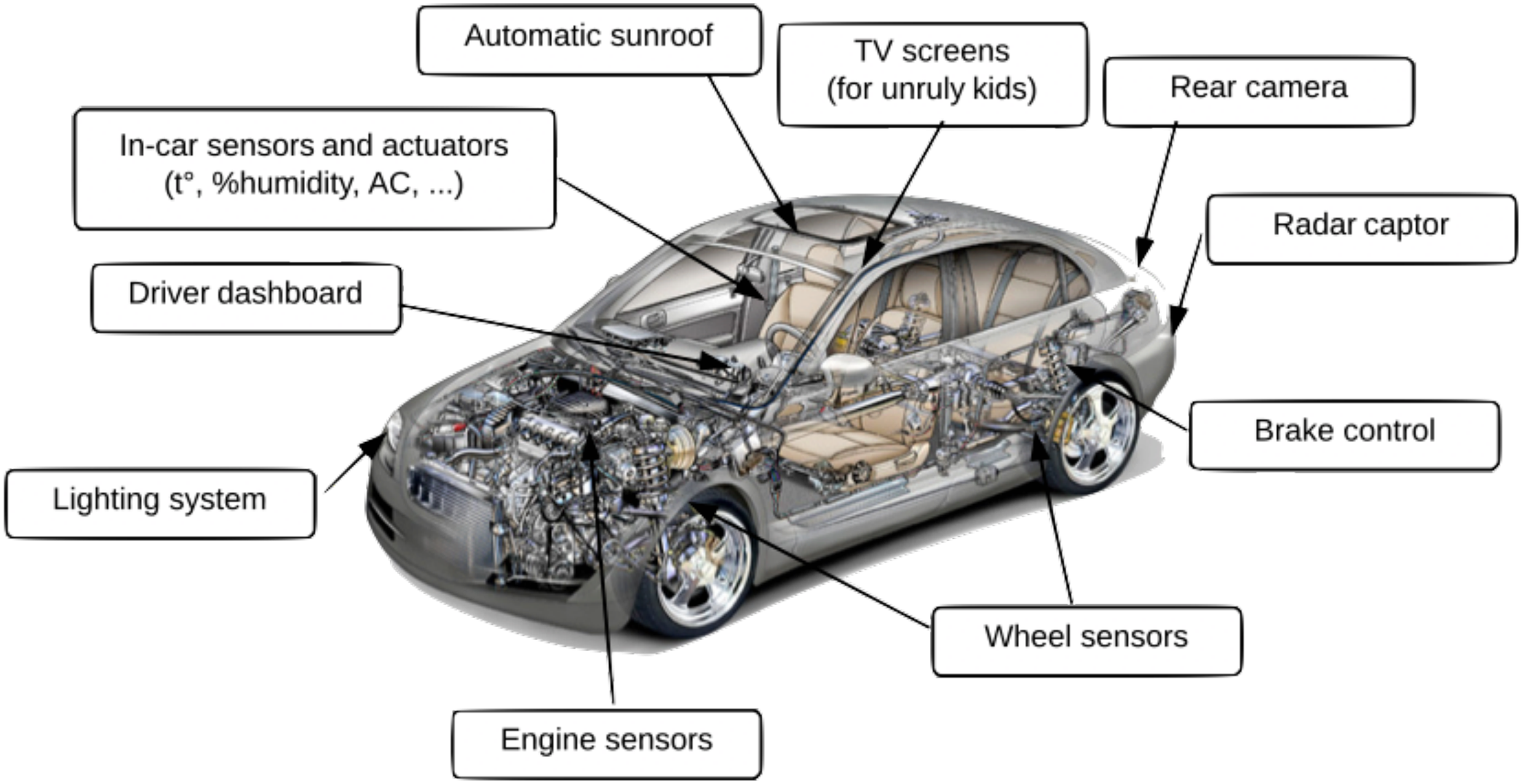
On-board car system

OBD Bluetooth adapter

Android device
(Samsung Galaxy Nexus i9250,
Android 4.3 Jelly Bean)

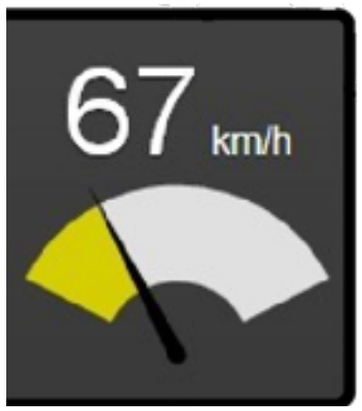
Feature clouds model



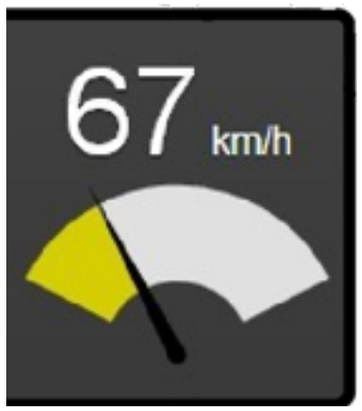



On-board car system

On-board car system



On-board car system



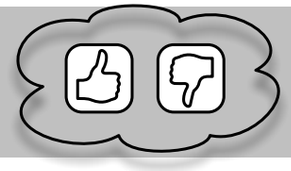
Feature store

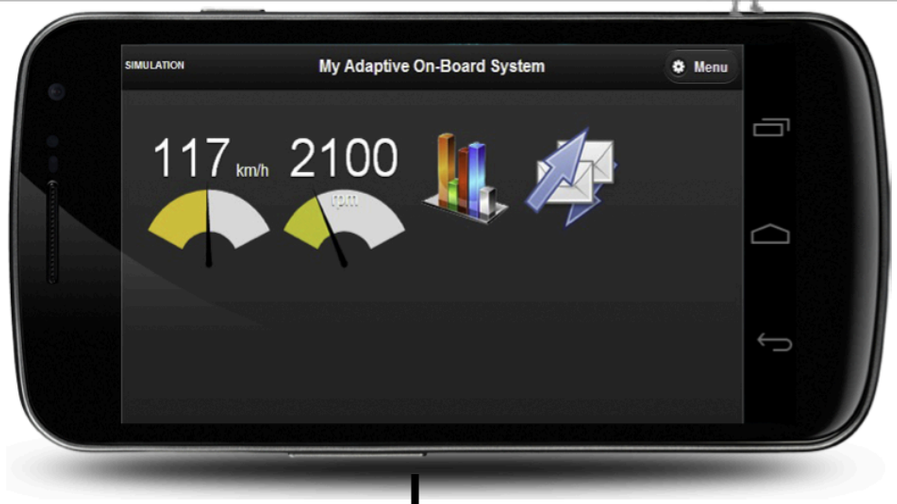
Features Store

 2800 rpm RPM Gauge ★★★★★ Free	 96 km/h Speed Gauge ★★★★★ Free	 41 km/h 1900 rpm Speed&RPM Gauge ★★★★★ Free
 Statistics ★★★★★ Free	 SMS Panel ★★★★★ Free	 SMS Sending ★★★★★ Free
 Notification Center	 SMS Notification	 Discreet SMS Notification

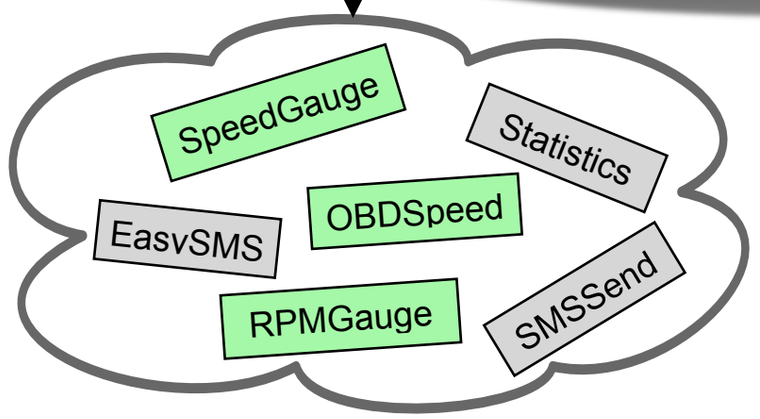
Back to the dashboard

Feature interaction

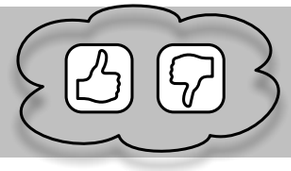


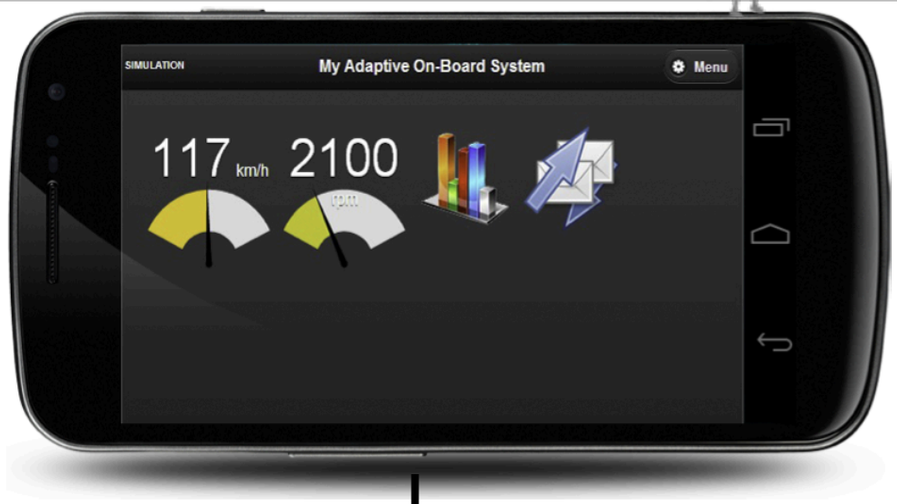


Assistent driving service request

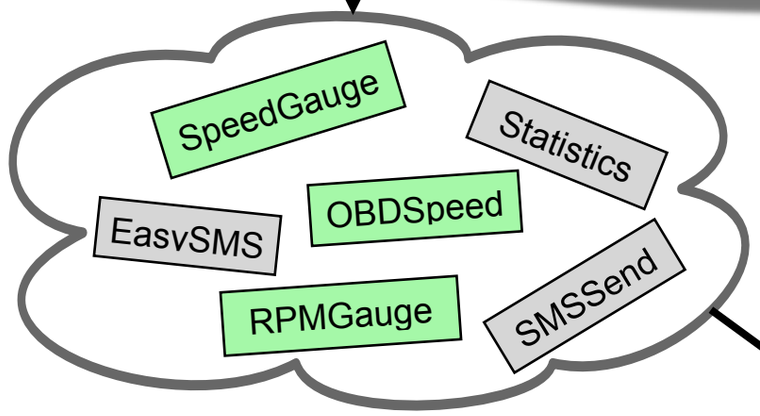


Feature interaction

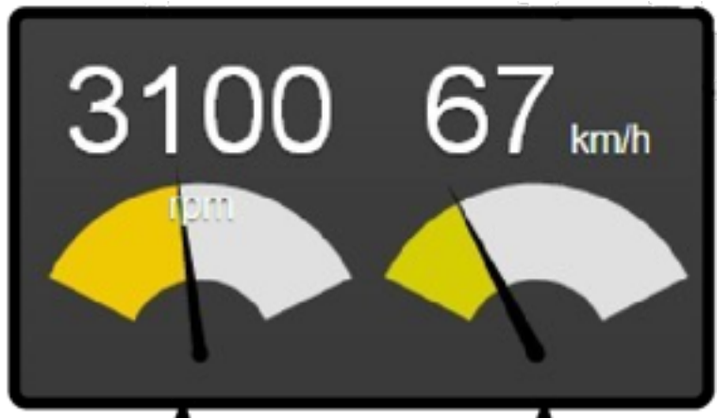




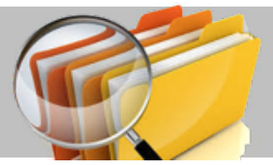
Assistent driving service request

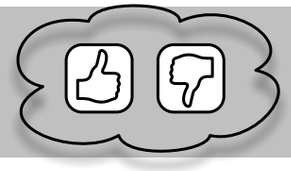


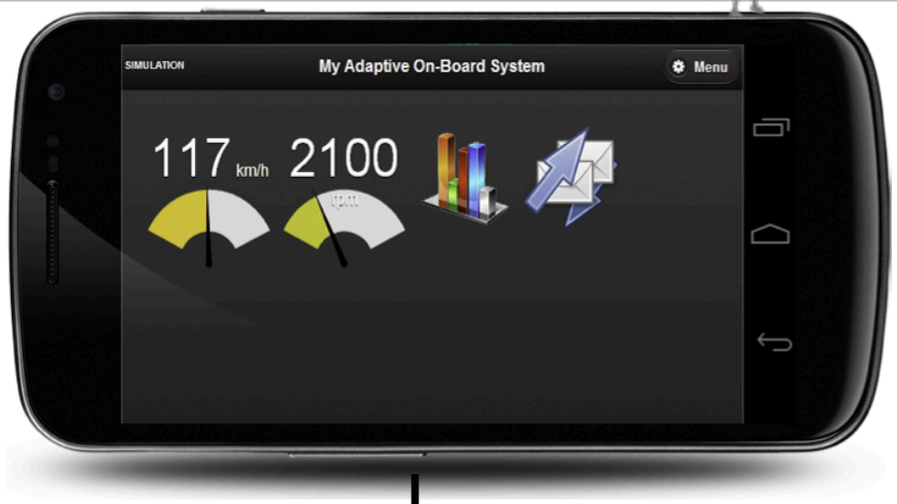
no relation



Feature interaction





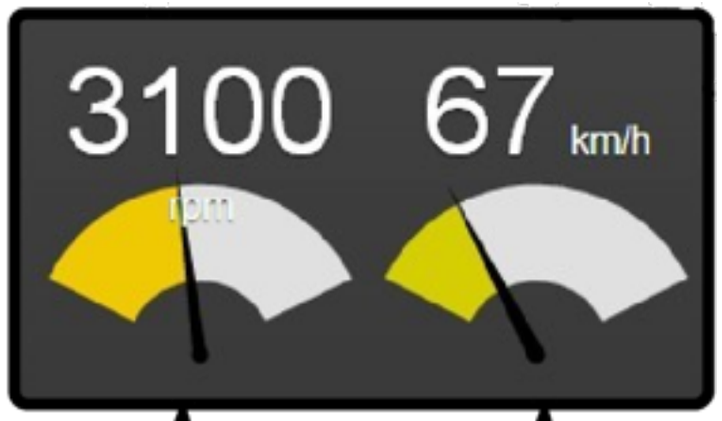


Assistent driving service request

subsumption relation

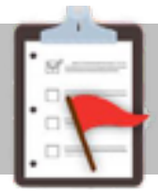
no relation

SpeedRPMGauge



RPM Gauge

SpeedGauge



Requirements

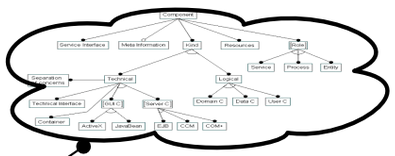
Feature clouds requirements

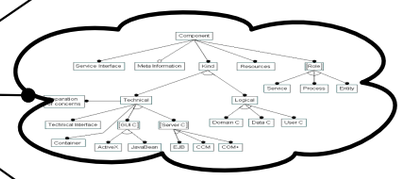
Service request

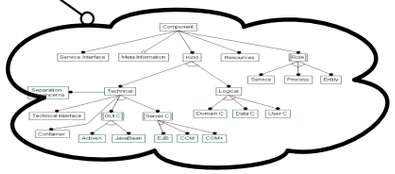
response

Process service component

Service component







Feature clouds requirements

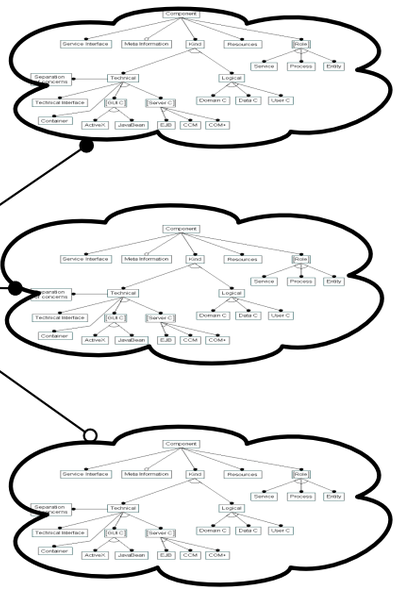
Dynamic behavior composition

Service request

response

Process service component

Service component



Feature clouds requirements

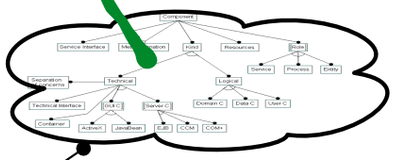
Dynamic behavior composition

Fine-grained features

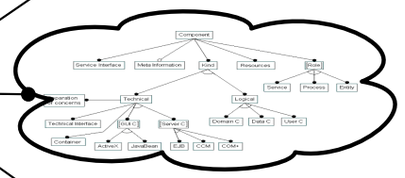
Service request

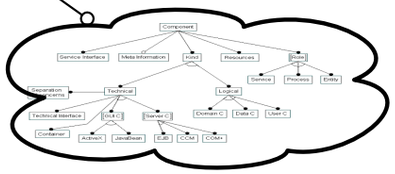
response

Process service component



Service component





Feature clouds requirements

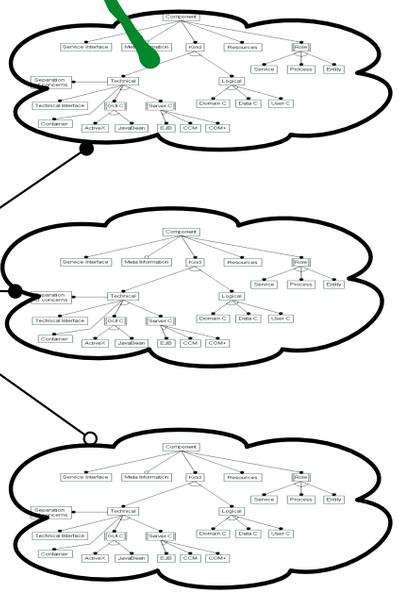
Dynamic behavior composition

Fine-grained features

Service request

response

Process service component



Service component

Manage feature interaction

Feature clouds requirements

Dynamic behavior composition

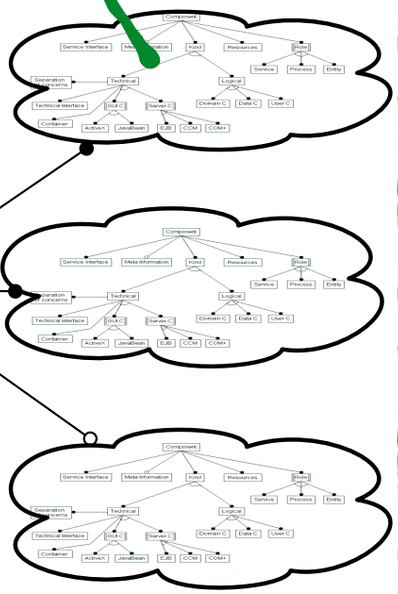
Fine-grained features

Process service component

Service request

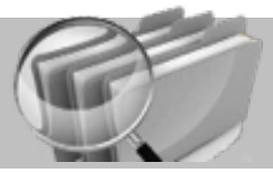
response

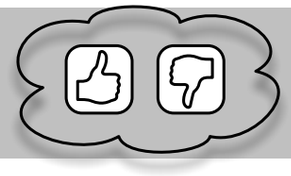
Service component



Appropriate infrastructure

Manage feature interaction





Feature Clouds Model

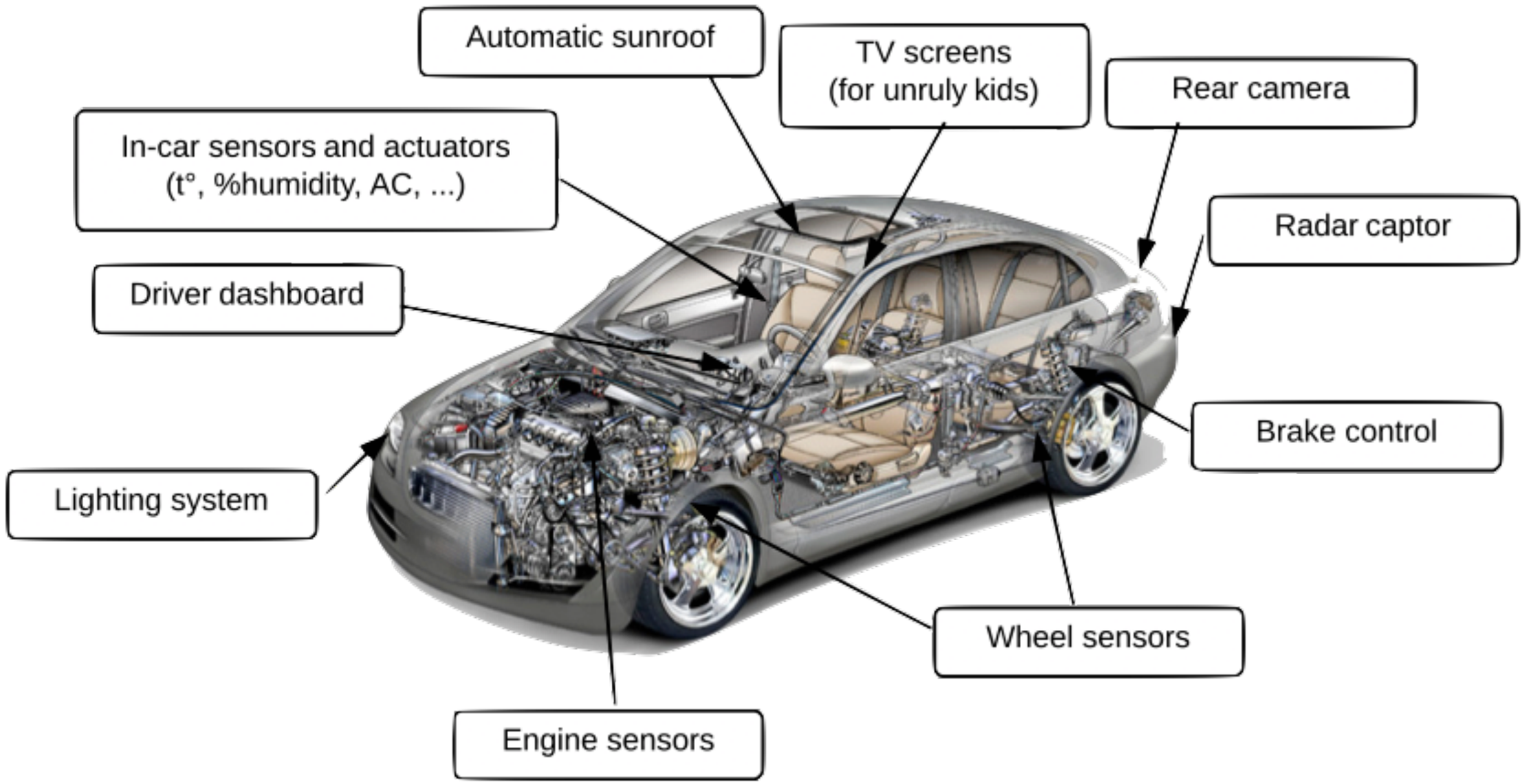
Feature clouds model




Feature-oriented programming

+

Context-oriented programming



Feature clouds model




Feature-oriented programming

+

Context-oriented programming

Feature clouds model




Feature-oriented programming

+

Context-oriented programming

location = UK

Display speed reading using the imperial system units

Feature clouds model




Feature-oriented programming

+

Context-oriented programming

location = UK

Display speed reading using the imperial system units

location = EC

Display speed reading using the metric system units

Feature clouds model




Feature-oriented programming

+

Context-oriented programming

location = UK

Display speed reading using the imperial system units

location = EC

Display speed reading using the metric system units

Feature clouds model




Feature-oriented programming

+

Context-oriented programming

location = UK

Display speed reading using the imperial system units

location = EC

Display speed reading using the metric system units

Feature clouds model



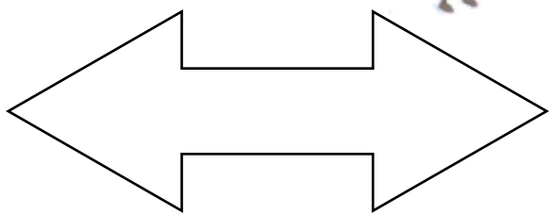

contexts

```
UKSpeedGauge = new  cop.Context ({
          name: ‘uk_speed_gauge’ });

gps.EventListener('gps_reading',
  function(location) {
      if(location is ‘UK’)
          UKSpeedGauge.activate();
      else UKSpeedGauge.deactivate(); })

ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });

 UKSpeedGauge.adapt(SpeedGauge, ImperialSystem);
```

[Context Traits: Dynamic Behaviour Adaptation Through Run-Time Trait Recomposition. *Modularity*’13]

Feature clouds model




contexts  **features**

```
UKSpeedGauge = new Feature ({
          name: ‘uk_speed_gauge’ });

gps.EventListener('gps_reading',
  function(location) {
      if(location is ‘UK’)
          UKSpeedGauge.activate();
      else UKSpeedGauge.deactivate(); })

ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });

 UKSpeedGauge.adapt(SpeedGauge, ImperialSystem);
```

[Context Traits: Dynamic Behaviour Adaptation Through Run-Time Trait Recomposition. *Modularity*’13]

Feature clouds model




contexts ⟷ **features**

```
UKSpeedGauge = new Feature ({
        name: ‘uk_speed_gauge’ });
```

→ Feature definition

```
gps.EventListener('gps_reading',
  function(location) {
      if(location is ‘UK’)
          UKSpeedGauge.activate();
      else UKSpeedGauge.deactivate(); })

ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });

 UKSpeedGauge.adapt(SpeedGauge, ImperialSystem);
```

[Context Traits: Dynamic Behaviour Adaptation Through Run-Time Trait Recomposition. *Modularity*’13]

Feature clouds model




contexts ⟷ **features**

```
UKSpeedGauge = new Feature ({
          name: ‘uk_speed_gauge’ });
```

Feature definition

```
gps.EventListener('gps_reading',
  function(location) {
      if(location is ‘UK’)
          UKSpeedGauge.activate();
      else UKSpeedGauge.deactivate(); })
```

Feature selection

```
ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });

 UKSpeedGauge.adapt(SpeedGauge, ImperialSystem);
```

[Context Traits: Dynamic Behaviour Adaptation Through Run-Time Trait Recomposition. *Modularity*'13]

Feature clouds model




contexts ⟷ **features**

```
UKSpeedGauge = new Feature ({
          name: ‘uk_speed_gauge’ });
```

→ Feature definition

```
gps.EventListener('gps_reading',
  function(location) {
      if(location is ‘UK’)
          UKSpeedGauge.activate();
      else UKSpeedGauge.deactivate(); })
```

→ Feature selection

```
ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });

 UKSpeedGauge.adapt(SpeedGauge, ImperialSystem);
```

→ Behavior definition

Feature clouds model




contexts ⟷ **features**

```
UKSpeedGauge = new Feature ({
          name: ‘uk_speed_gauge’ });
```

→ Feature definition

```
gps.EventListener('gps_reading',
  function(location) {
      if(location is ‘UK’)
          UKSpeedGauge.activate();
      else UKSpeedGauge.deactivate(); })
```

→ Feature selection

```
ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });
```

→ Behavior definition

```
UKSpeedGauge.adapt(SpeedGauge, ImperialSystem);
```

→ Feature-behavior association

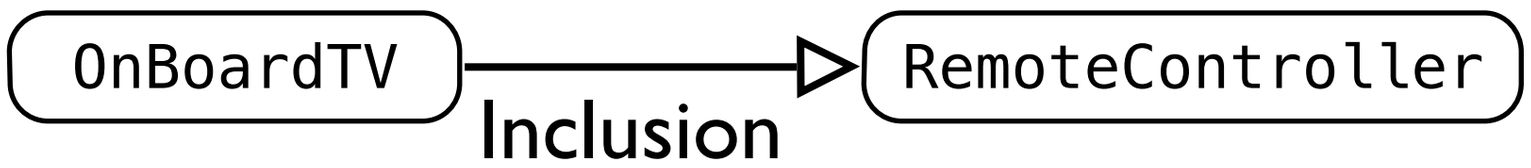
[Context Traits: Dynamic Behaviour Adaptation Through Run-Time Trait Recomposition. *Modularity*'13]

Dependency relations

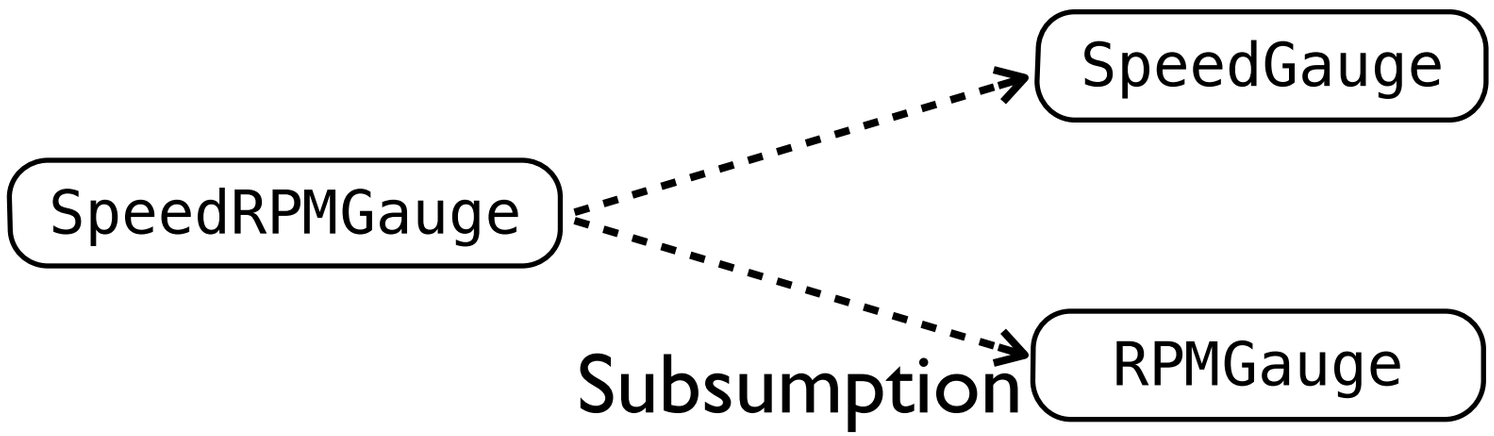



1-1 — Alternative

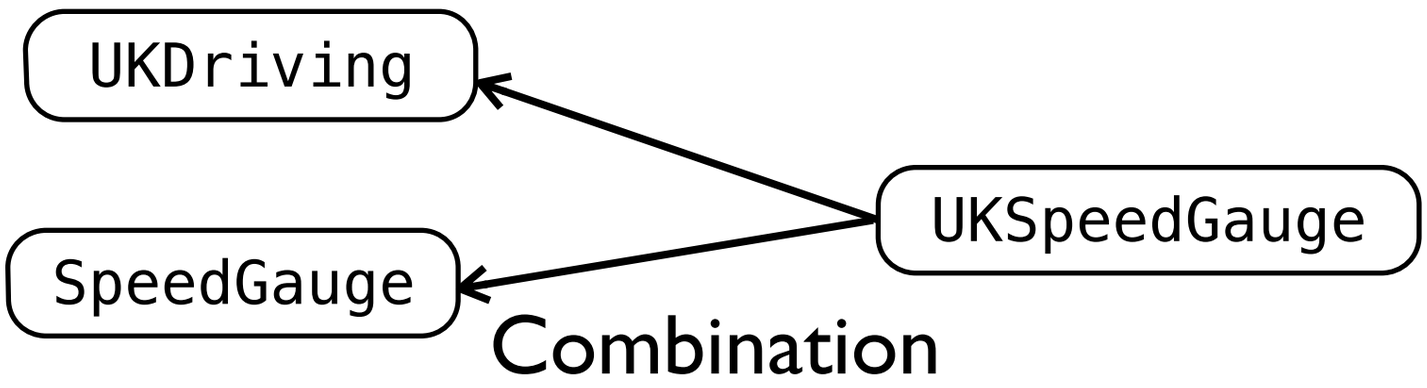
1-1 — Specialize



1-* — Subsume



*-1 — Combine

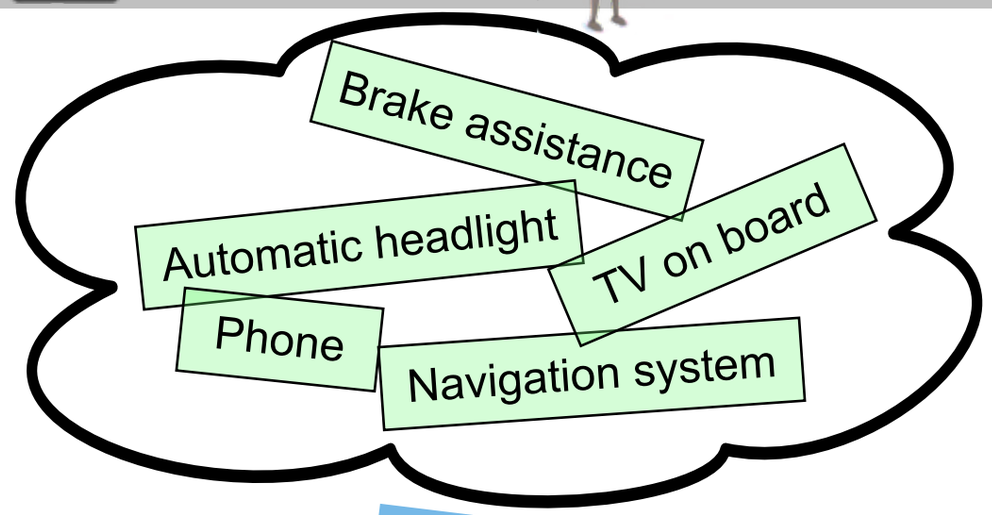


[Subjective-C: Bringing Context to Mobile Platform Programming. *SLE*'10]

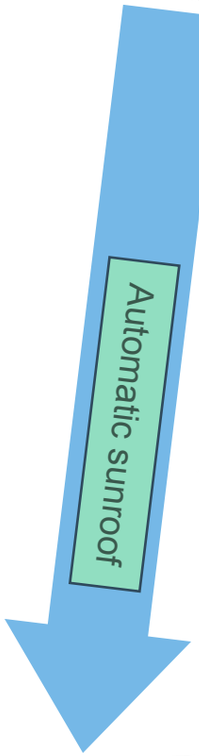
[Modeling and Analyzing Self-Adaptive Systems with Context Petri Nets. *TASE*'13]

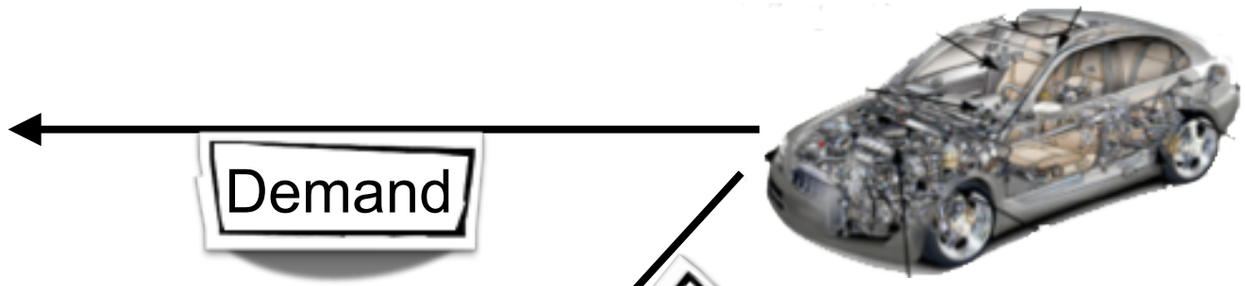
Feature clouds infrastructure

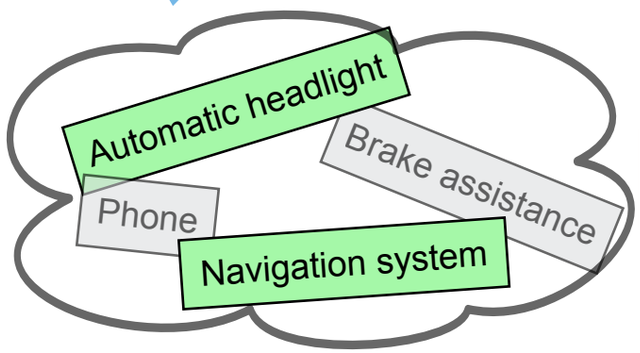


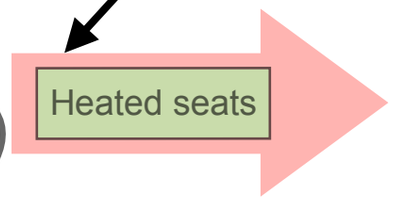



Features on Demand





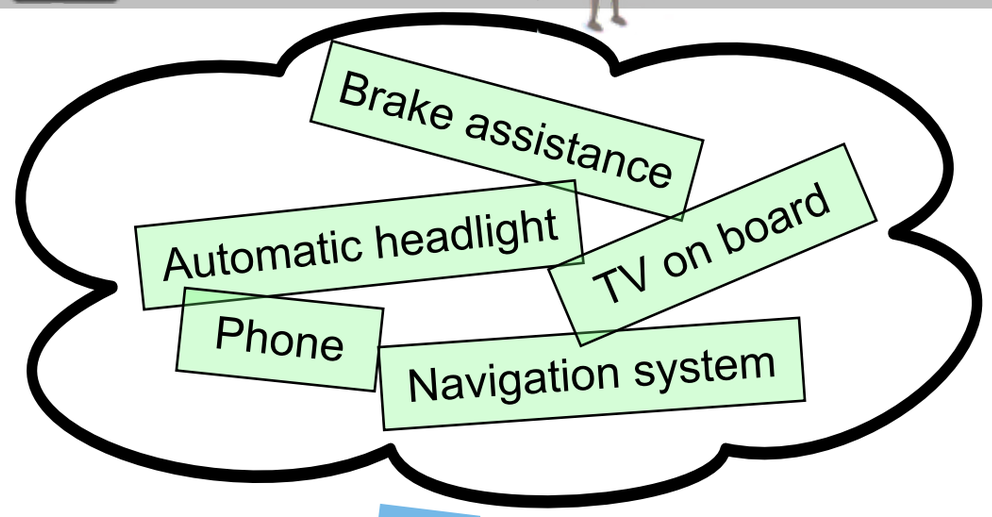




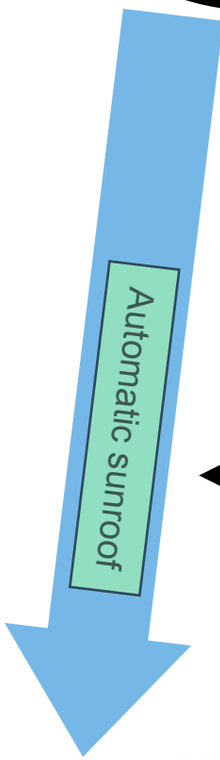
Feature clouds infrastructure

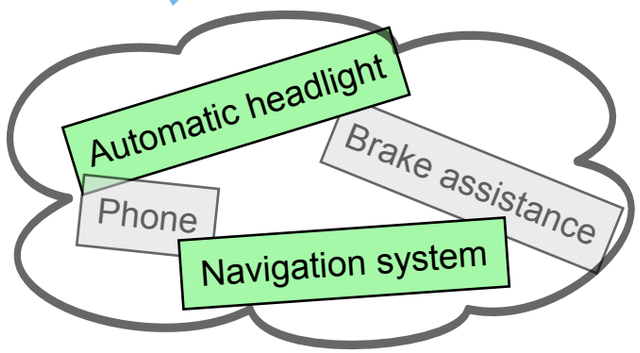


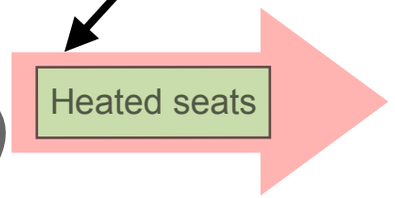

Feature Store



Features on Demand



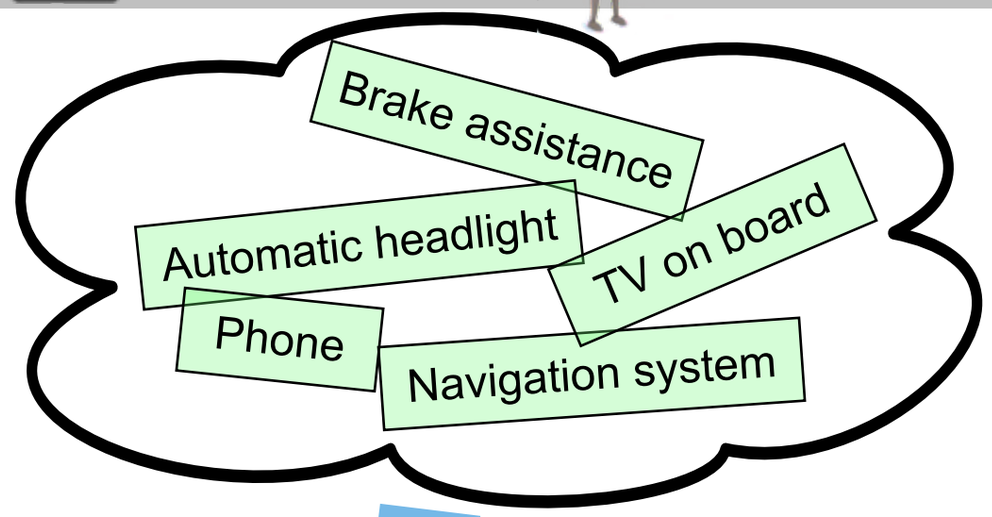




Feature clouds infrastructure




Feature Store



Automatic sunroof

Features on Demand

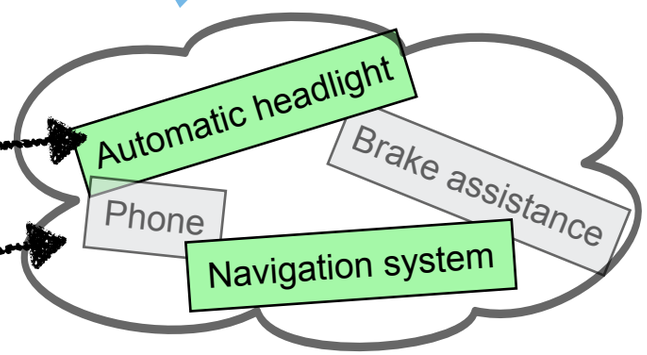
Demand

Drop

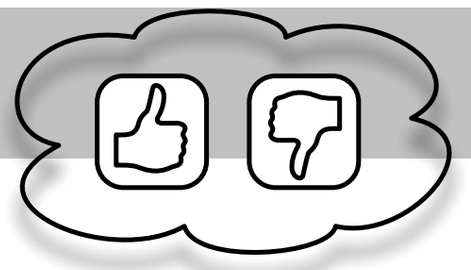
Feature Manager

Active features

Inactive features



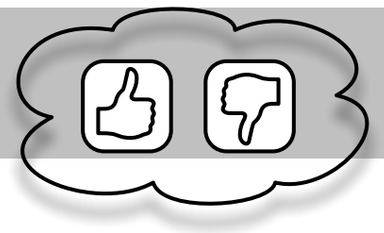
Heated seats



Conclusion

Summary



Feature Clouds

- ✓ Infrastructure for automated service composition

Feature Store

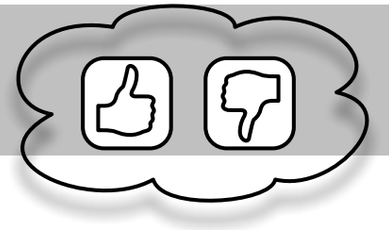
- ✓ Fine-grained feature definition

Feature Manager

- ✓ Dynamic selection and composition
- ✓ Feature interaction

Open Questions

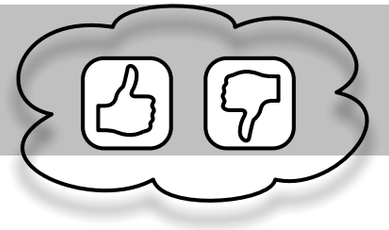


- Granularity of features
  - does the approach remain manageable for a user or developer at this fine level of granularity?
- Third party features
- Run-time composition and verification
- Feature clouds infrastructure
  - need for a more holistic approach: in addition to behavioural aspect, also need to consider the data and UI aspects

Open Questions



- Granularity of features
  - does the approach remain manageable for a user or developer at this fine level of granularity?
- Third party features
- Run-time composition and verification
- Feature clouds infrastructure
  - need for a more holistic approach: in addition to behavioural aspect, also need to consider the data and UI aspects

Questions?