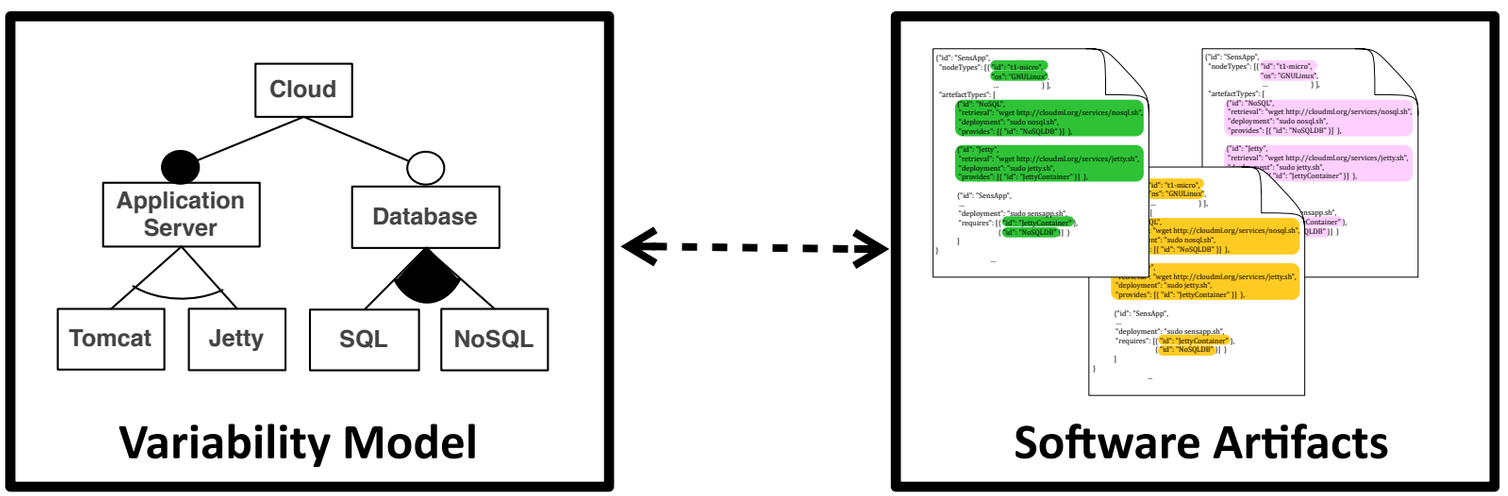
Evolving Dynamic Software Product Lines

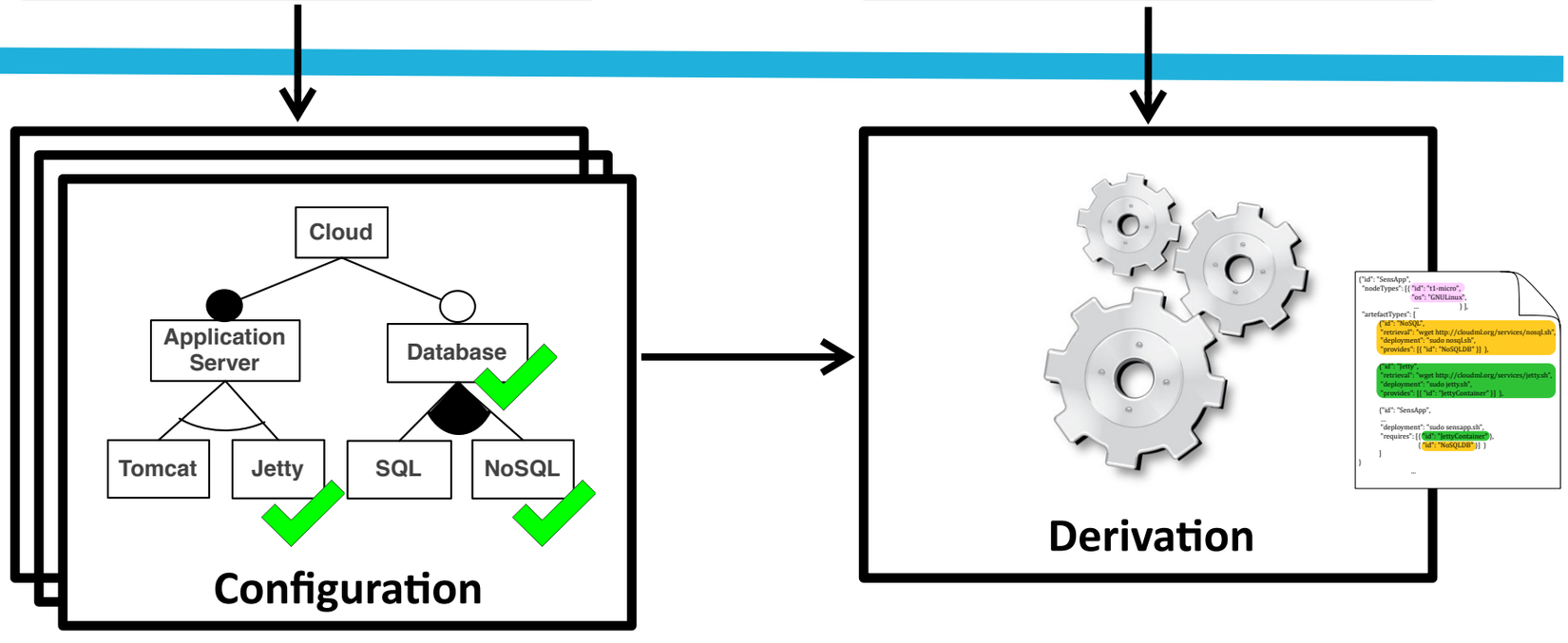
Clément Quinton, Luciano Baresi

EASSy, September 7-10, 2015 - Japan

Software Product Lines

Domain Engineering

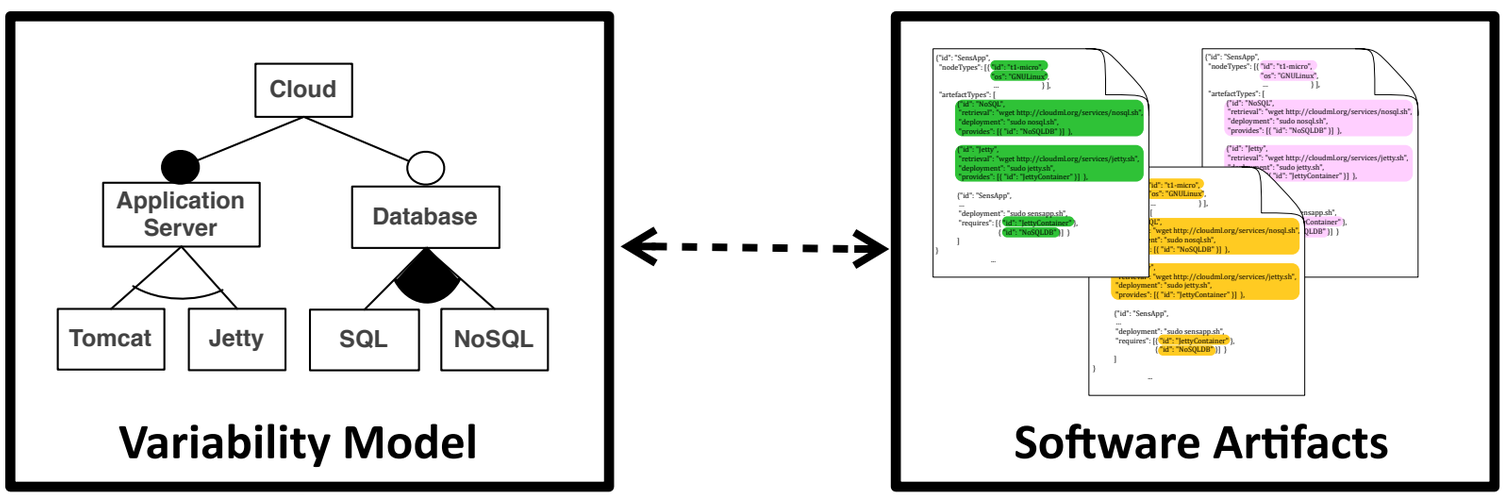


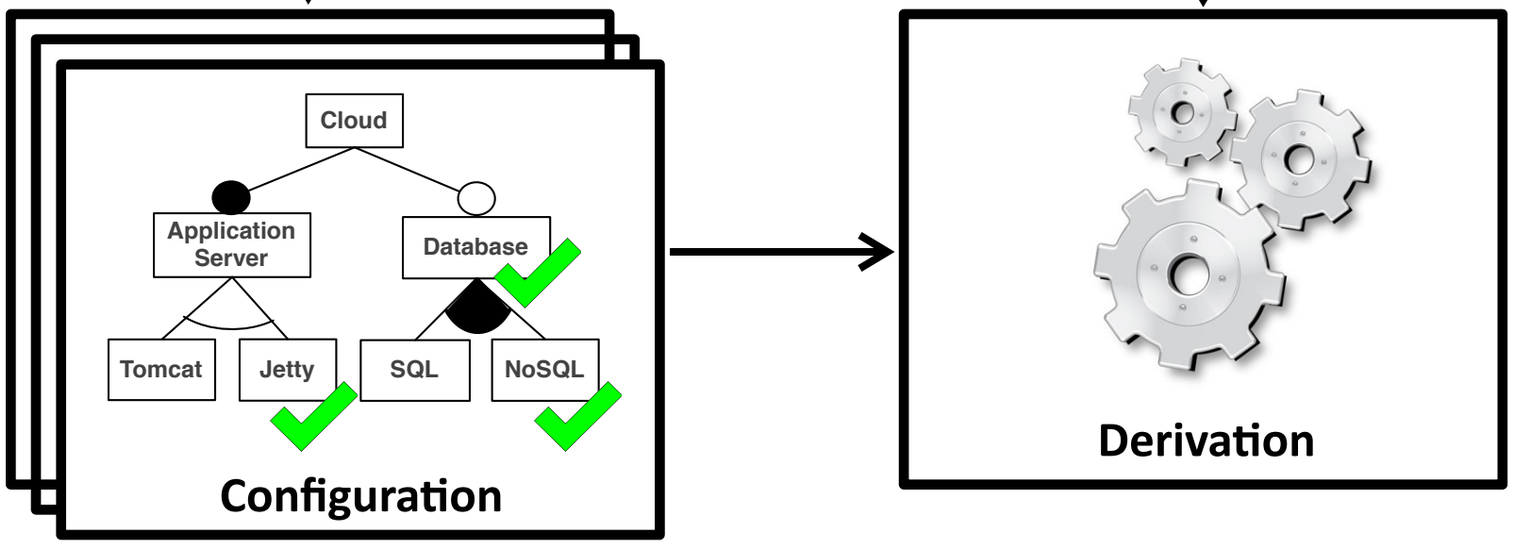


Application Engineering

Software Product Lines

Domain Engineering



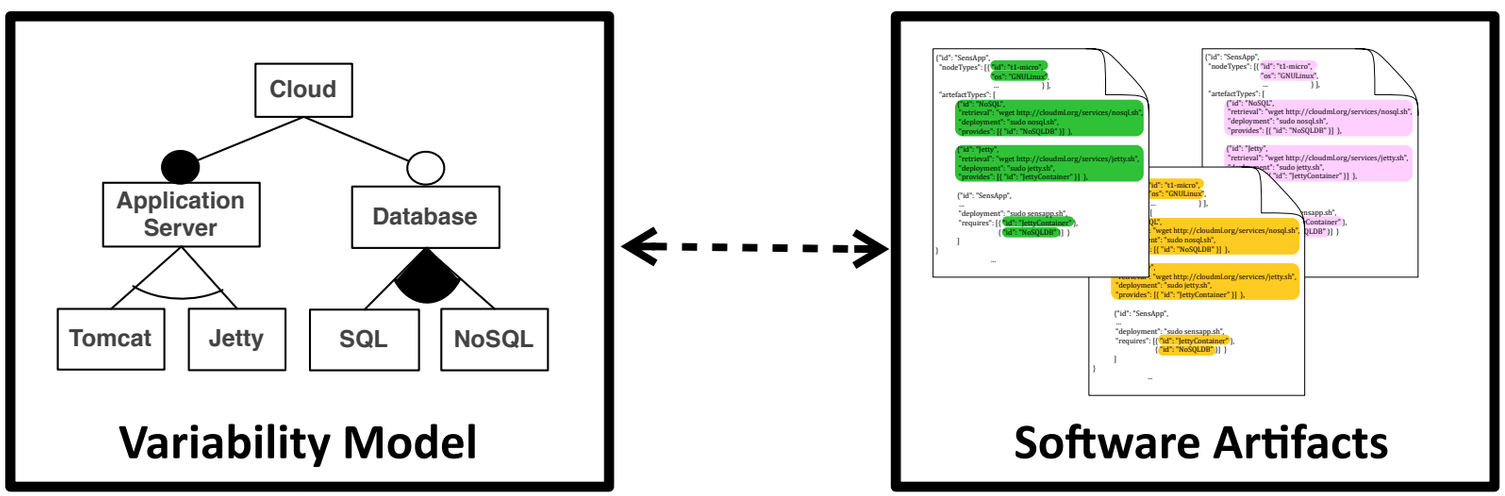


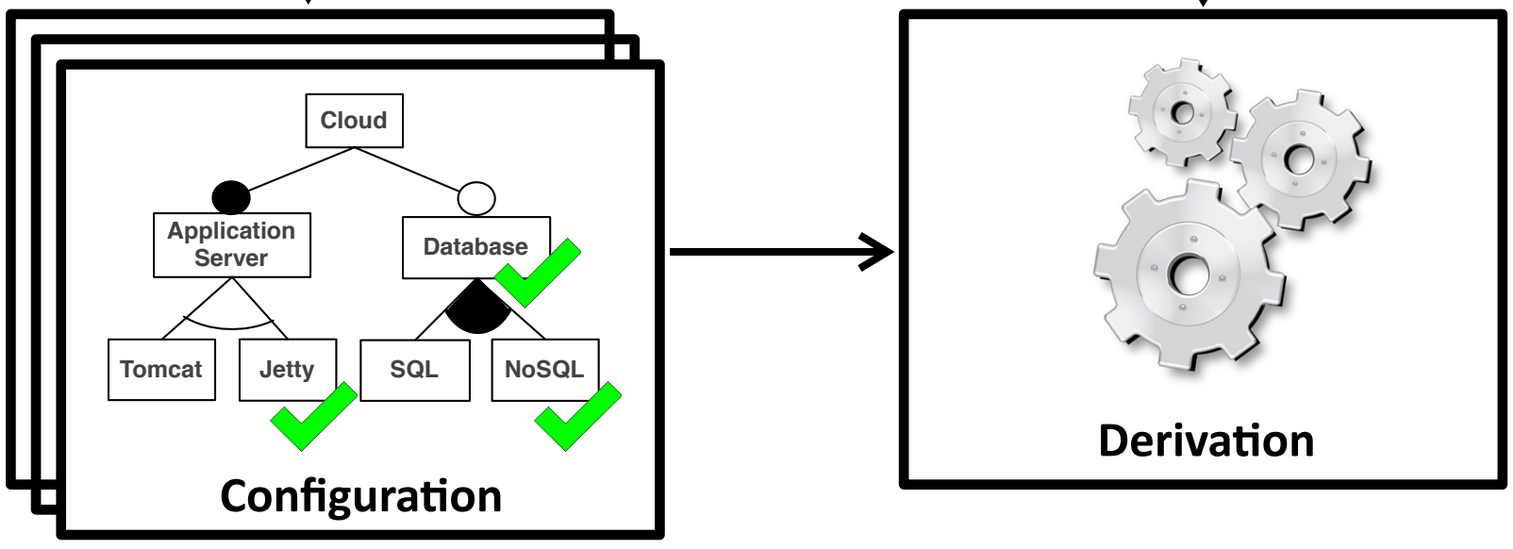
Application Engineering

Design Time | Runtime

Dynamic Software Product Lines

Domain Engineering



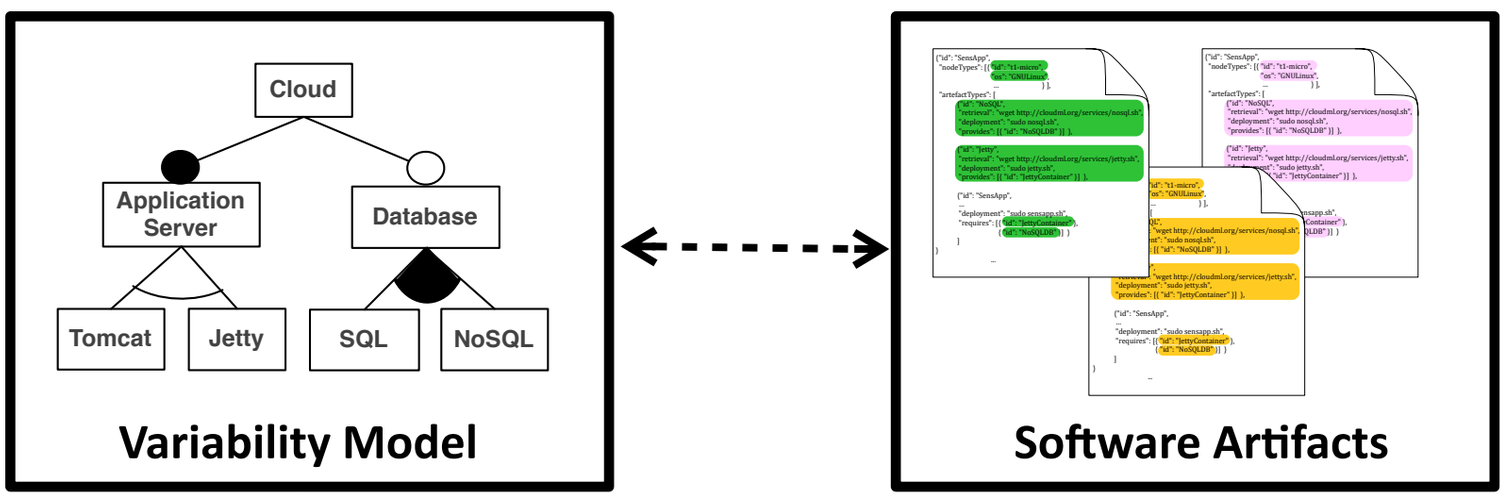


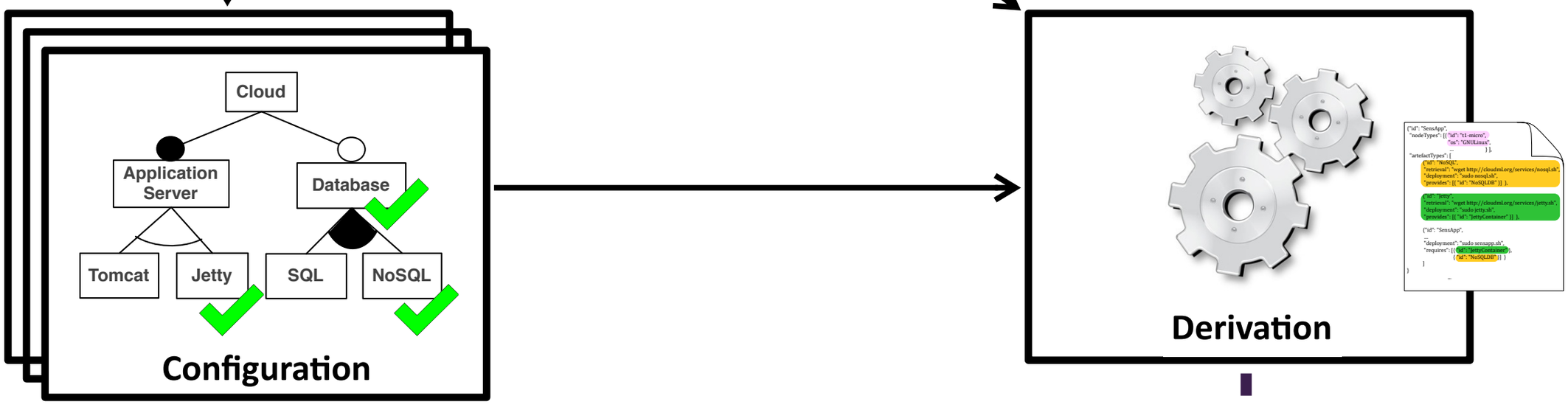
Application Engineering

Design Time | Runtime

Dynamic Software Product Lines

Domain Engineering



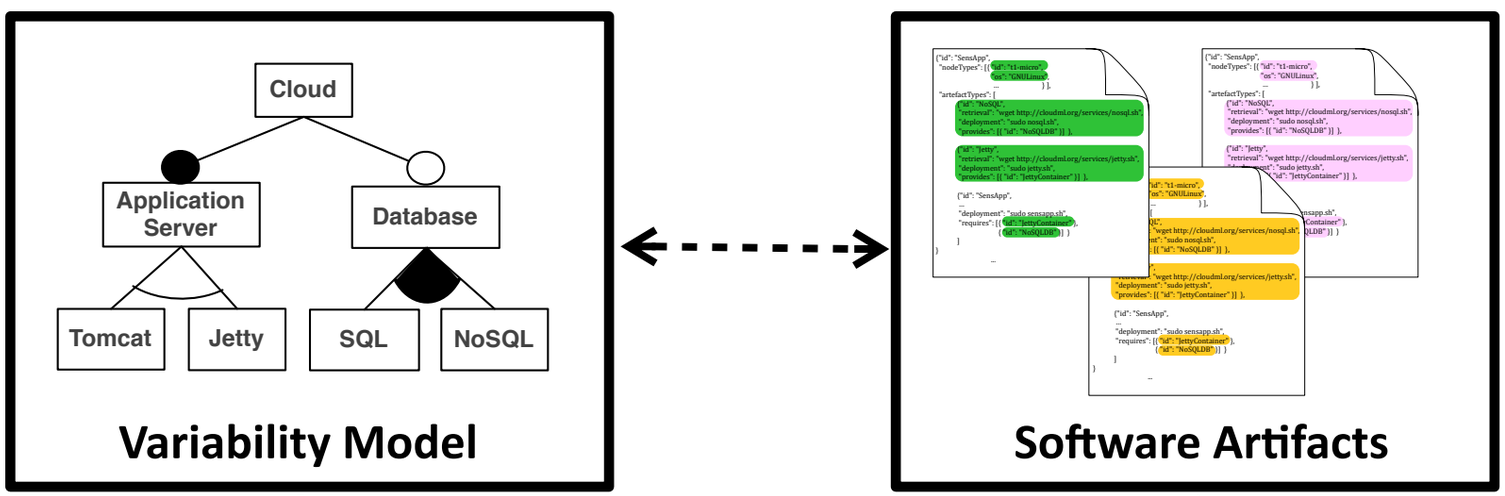


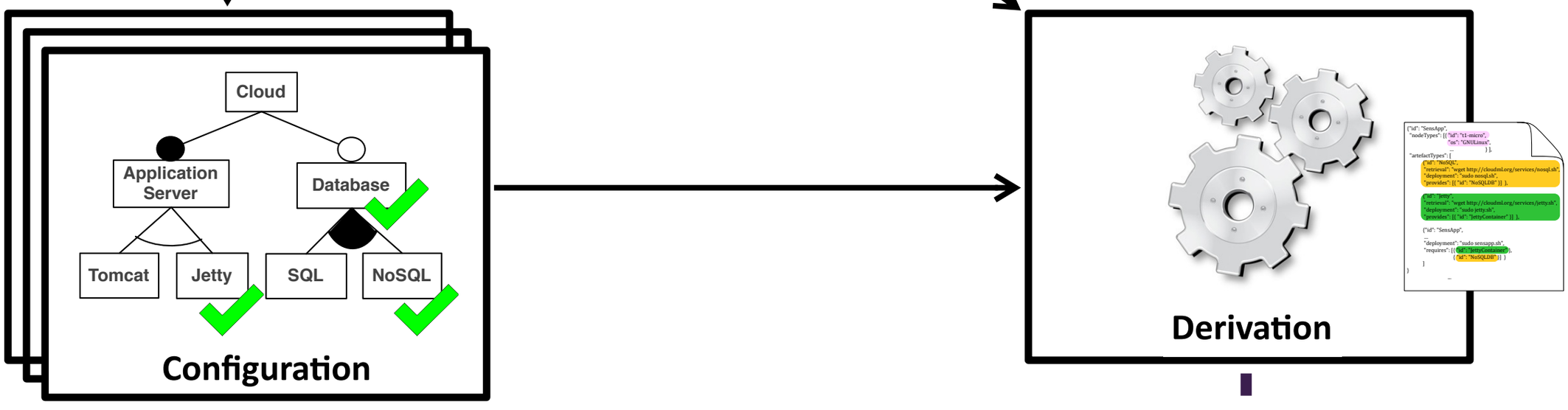
Application Engineering

Design Time | **Runtime**

Dynamic Software Product Lines

Domain Engineering





Application Engineering

Adaptation

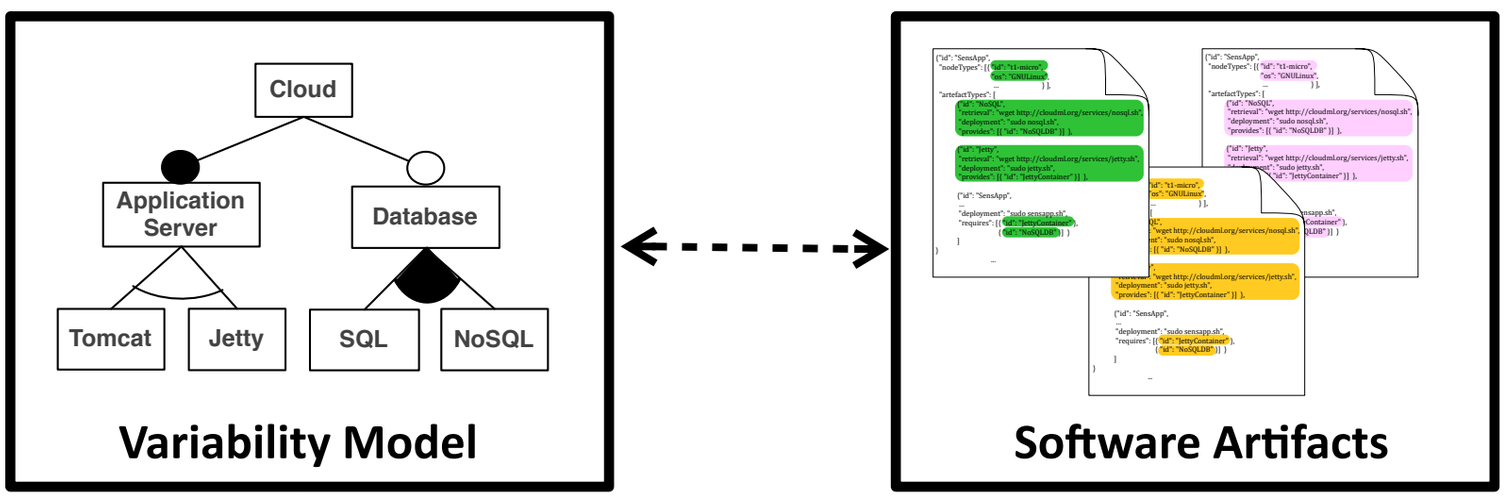
Reconfiguration

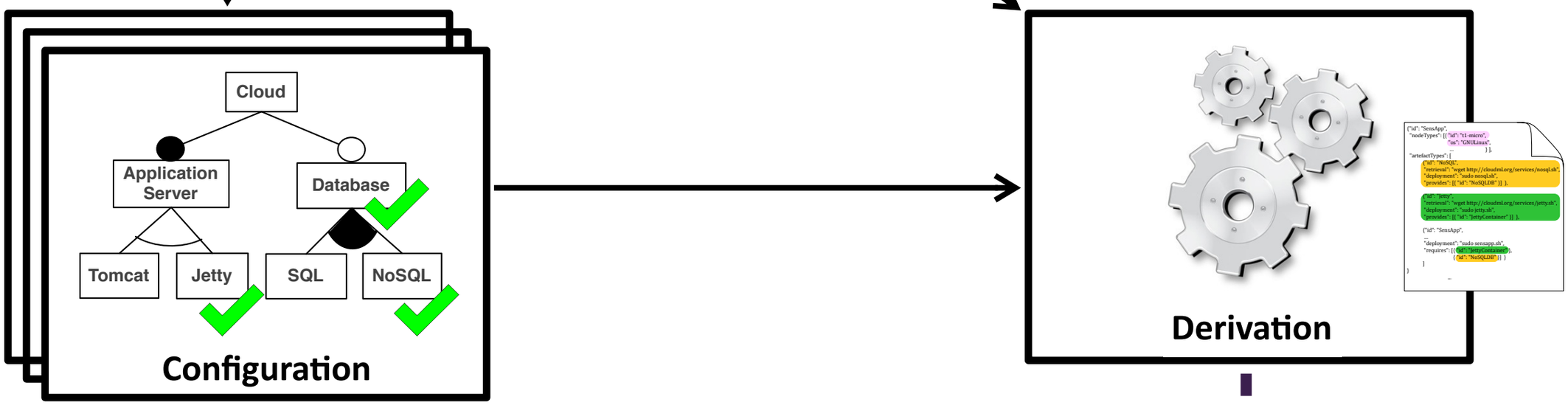
Design Time

Runtime

Dynamic Software Product Lines

Domain Engineering





Application Engineering

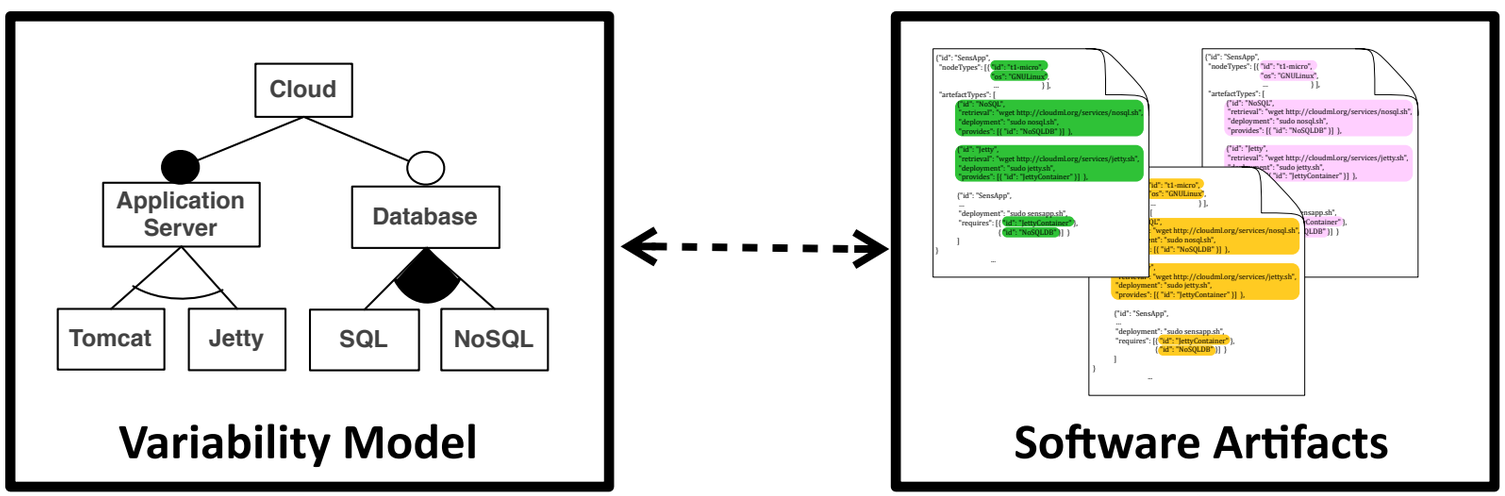
Adaptation

Reconfiguration

Design Time | Runtime

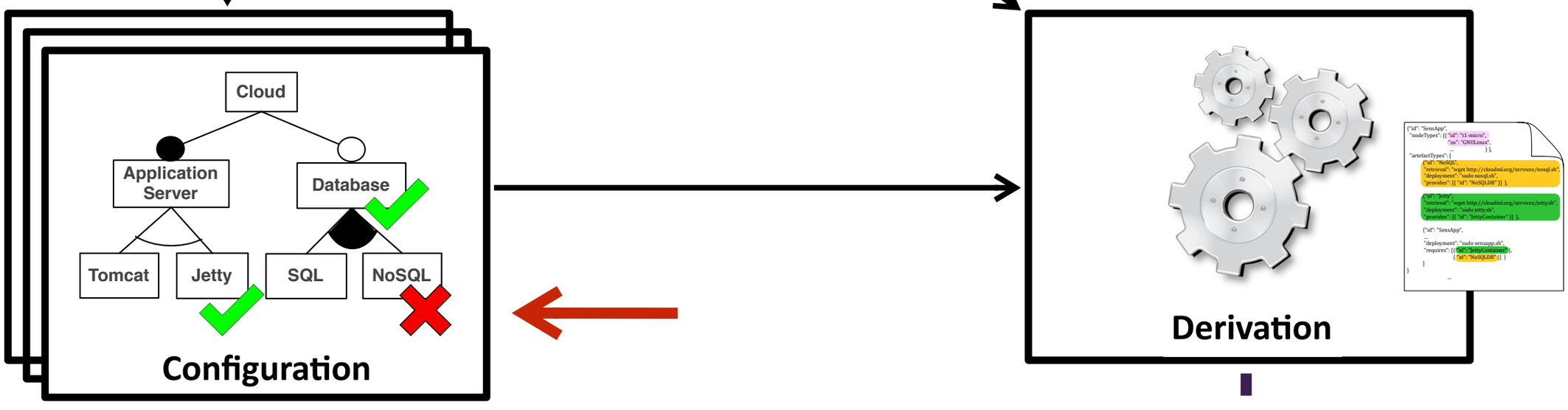
Dynamic Software Product Lines

Domain Engineering



Adaptation

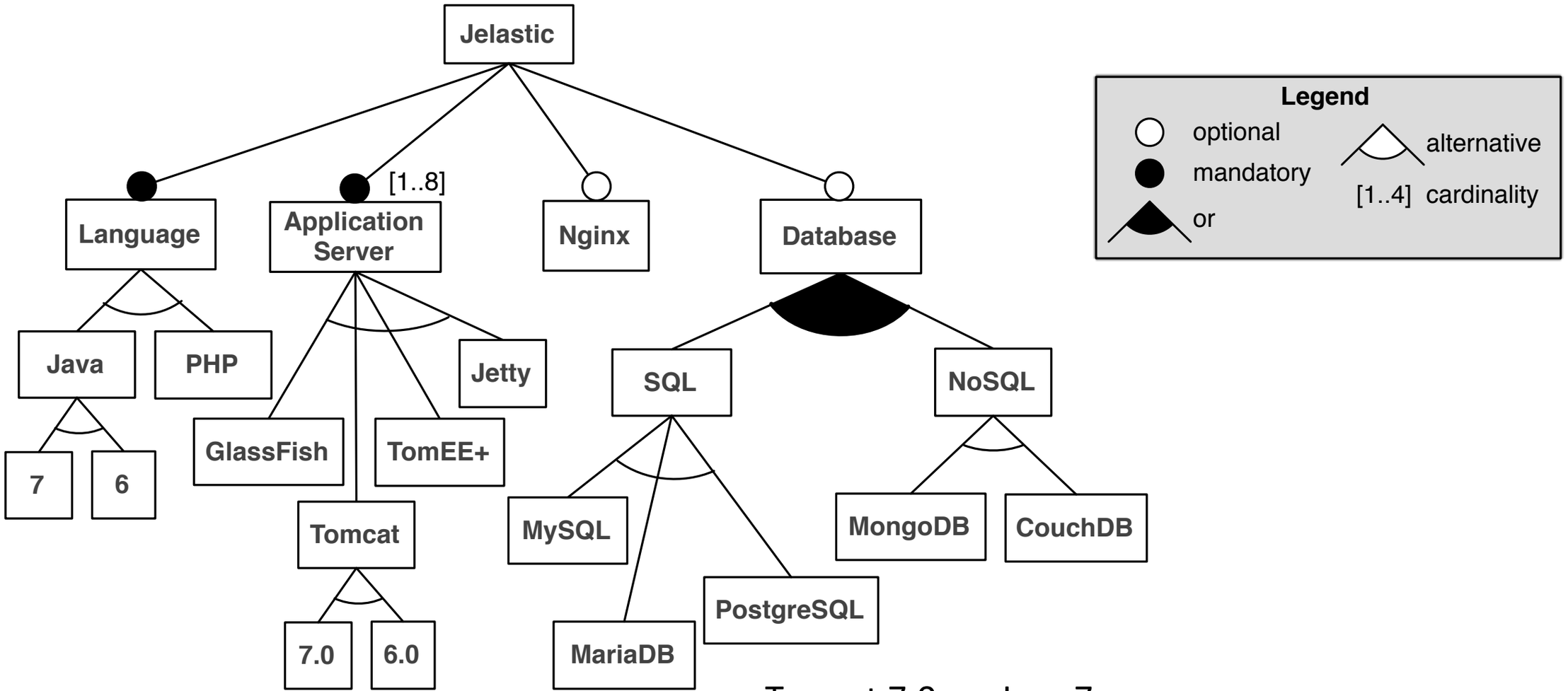
Reconfiguration



Application Engineering

Design Time | Runtime

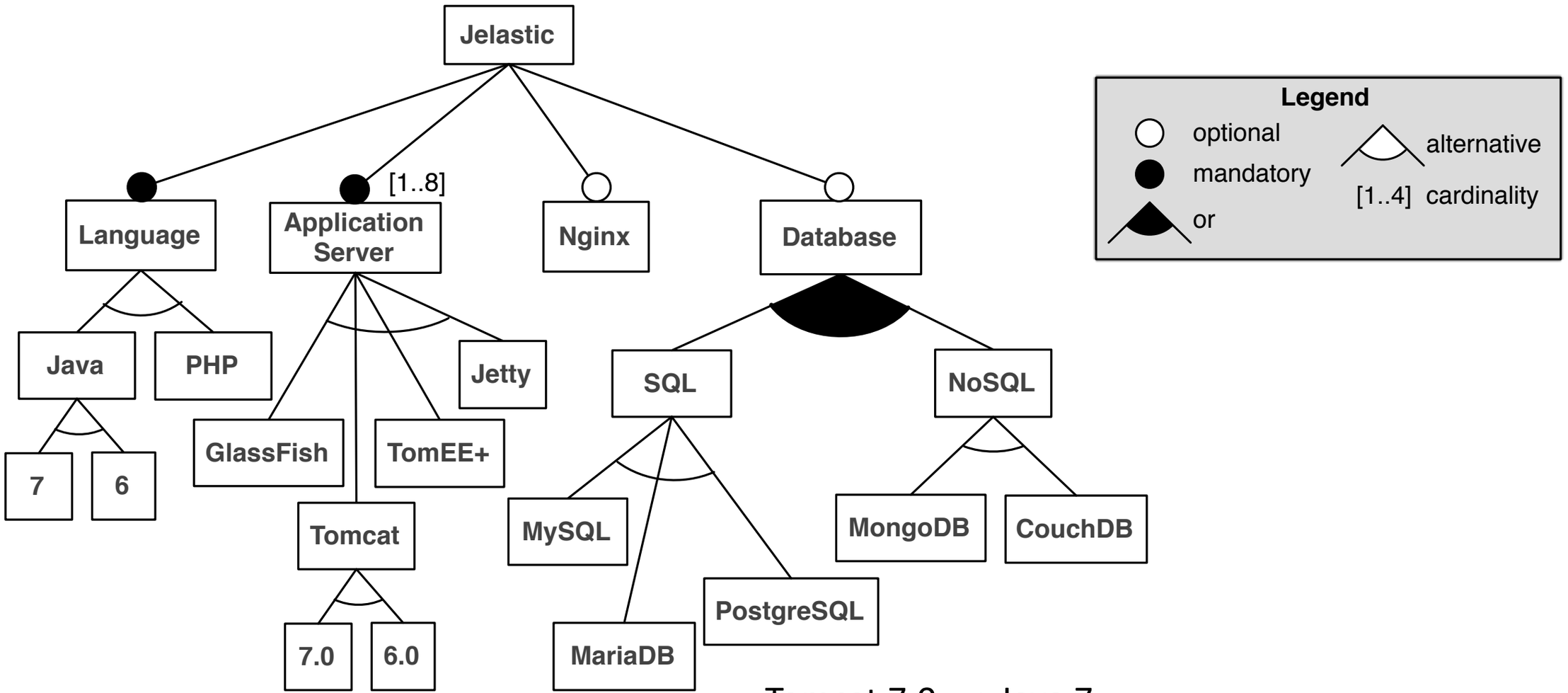
Why DSPL for Adaptive Systems?



Tomcat 7.0 ⟶ Java 7

[2,*] Application Server ⟶ Nginx

Why DSPL for Adaptive Systems?

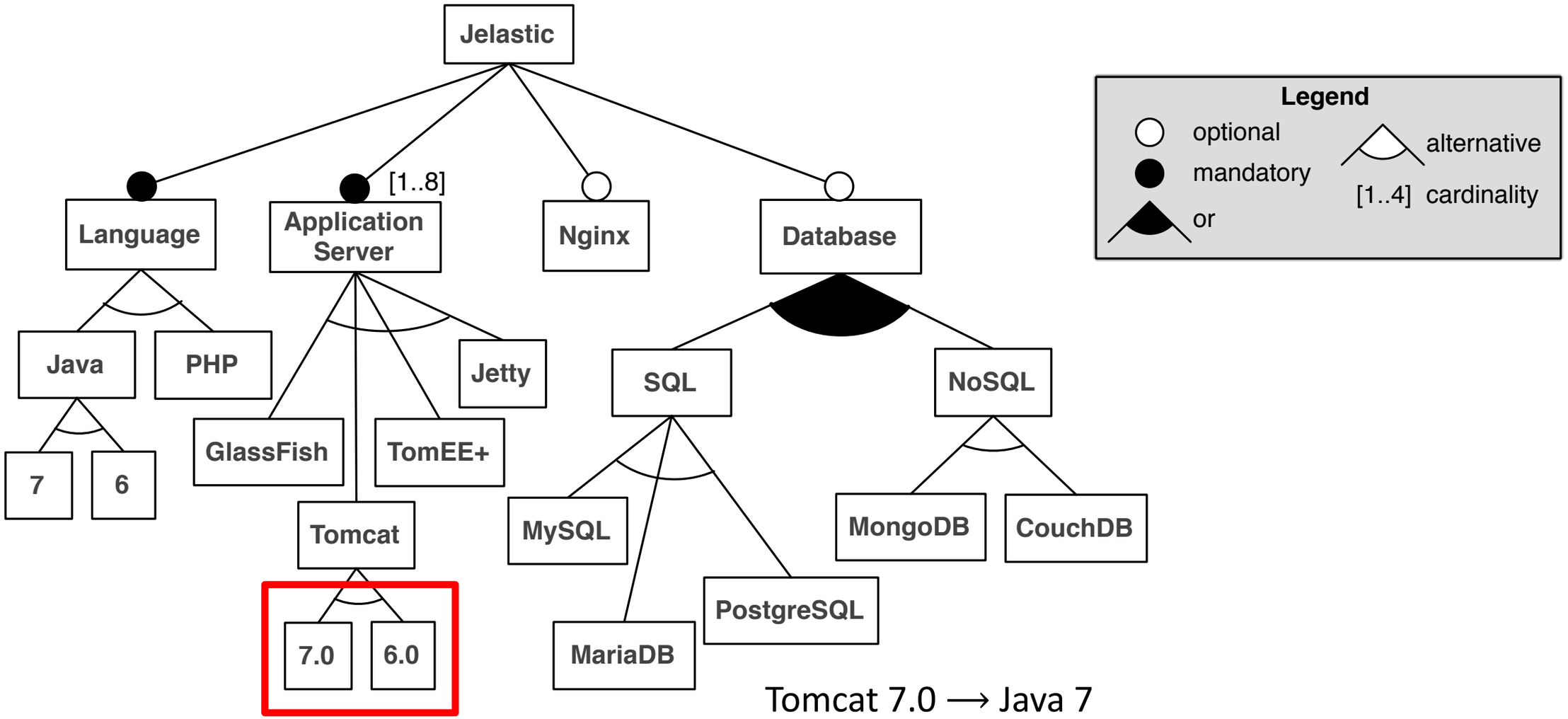


Tomcat 7.0 ⟶ Java 7

[2,*] Application Server ⟶ Nginx

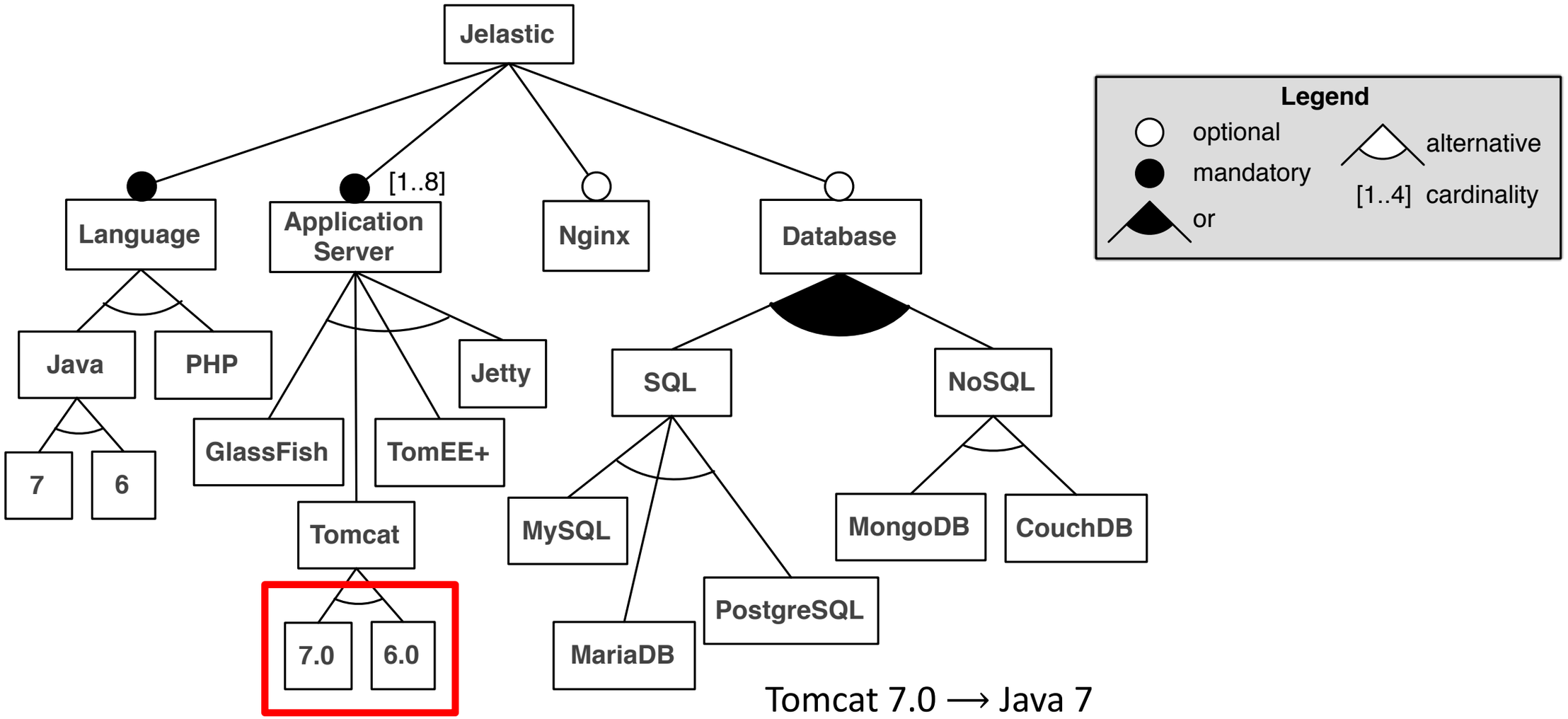
- Adaptation capabilities are modeled (boundaries!)

Why DSPL for Adaptive Systems?



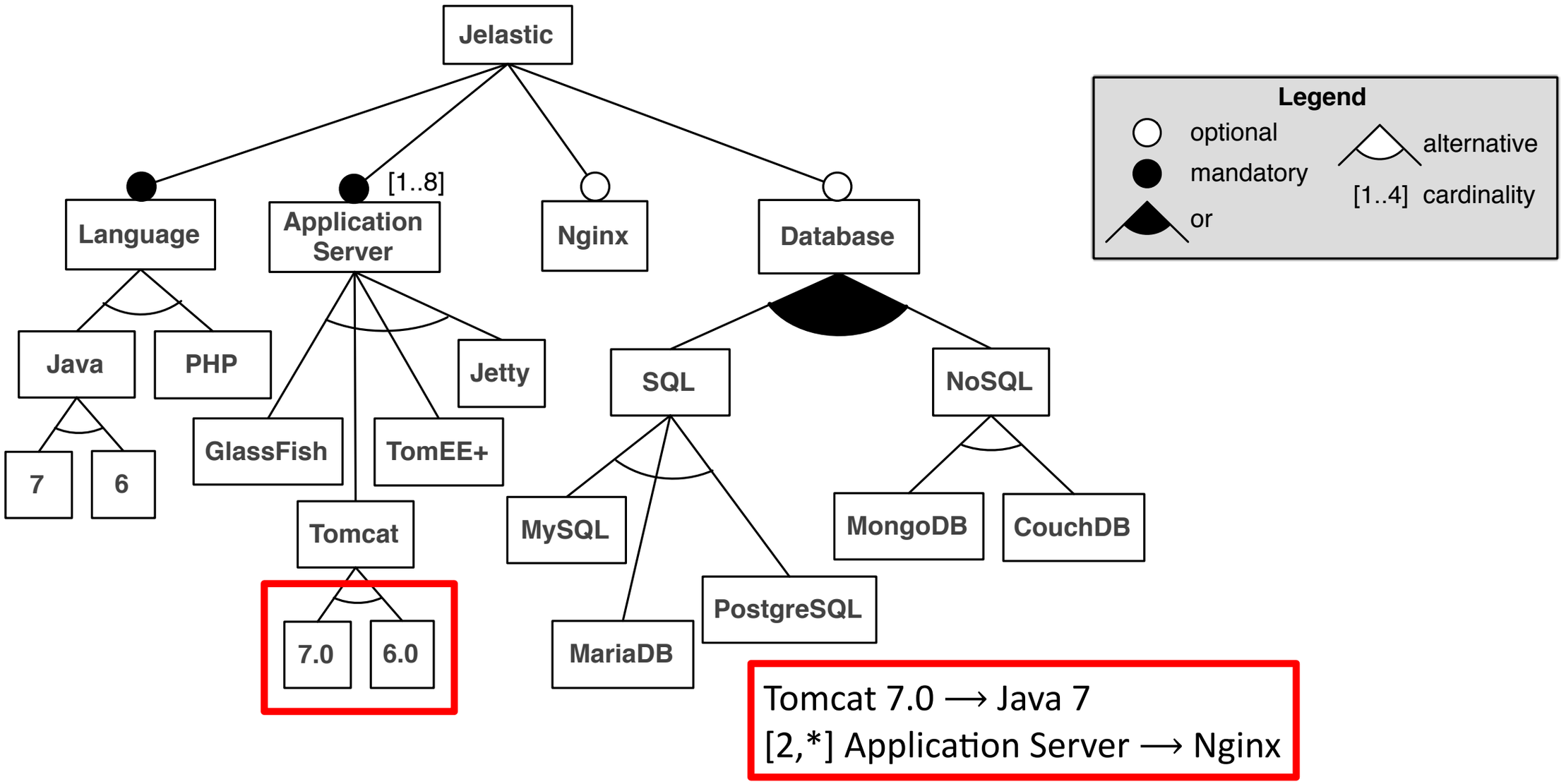
- Adaptation capabilities are modeled (boundaries!)

Why DSPL for Adaptive Systems?



- Adaptation capabilities are modeled (boundaries!)
- Automated reasoning on (re)configurations

Why DSPL for Adaptive Systems?



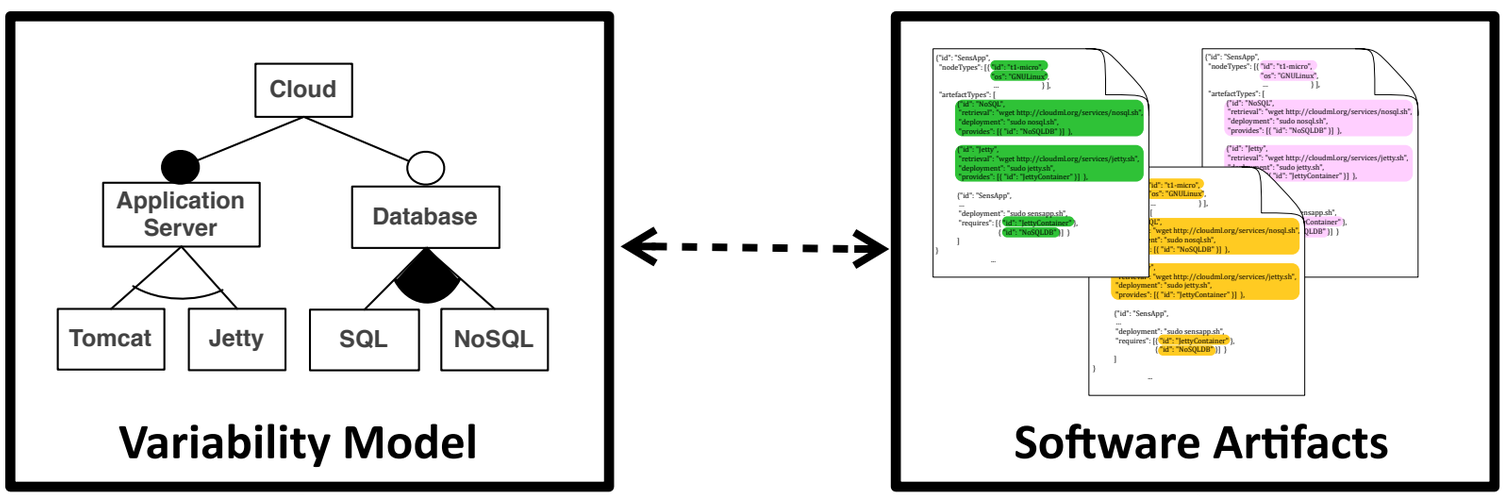
- Adaptation capabilities are modeled (boundaries!)
- Automated reasoning on (re)configurations

”Design evolution is absolutely inevitable“

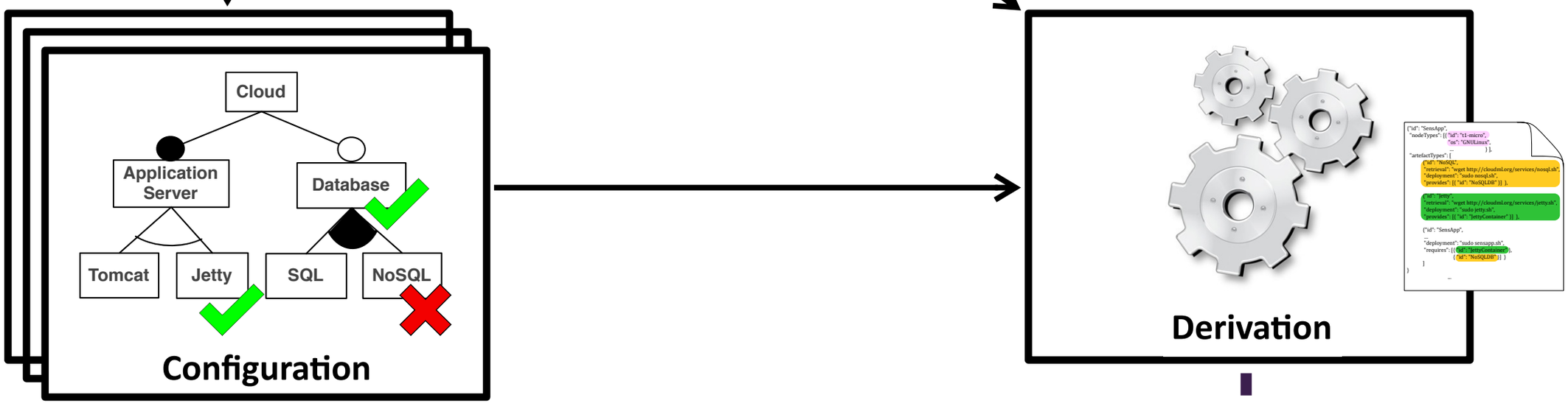
Kent Beck, *The Inevitability of Evolution*, IEEE Software, 2010

Evolution of Dynamic Software Product Lines

Domain Engineering



Application Engineering

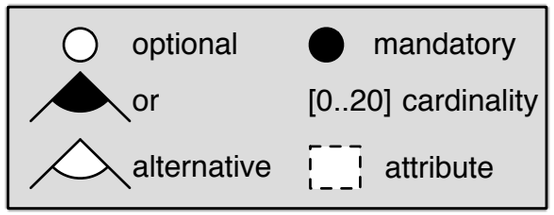


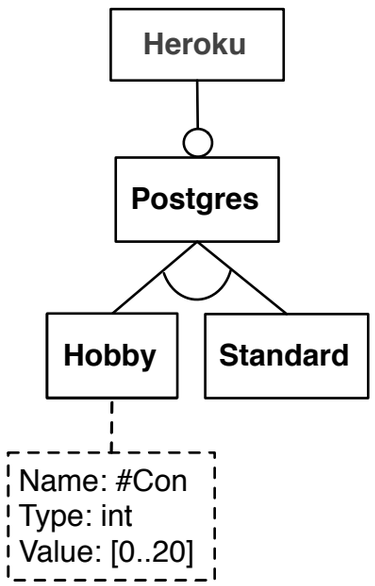
Adaptation

Reconfiguration

Design Time | Runtime

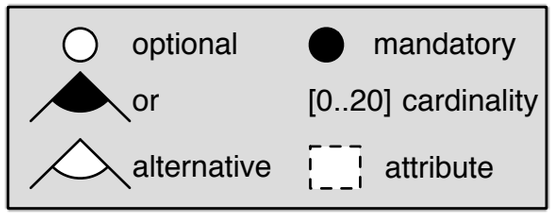
Example

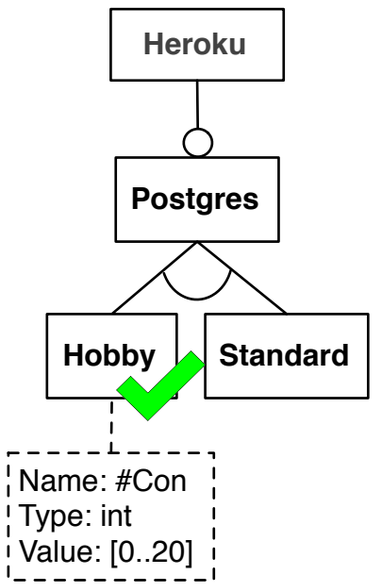


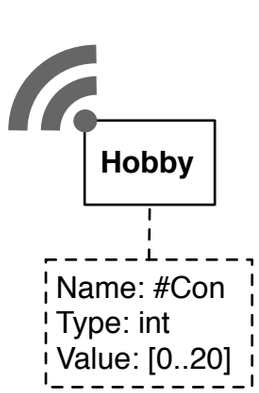


Design Time | **Runtime**

Example

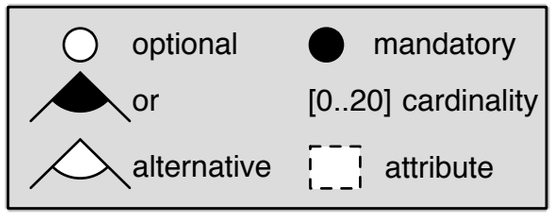


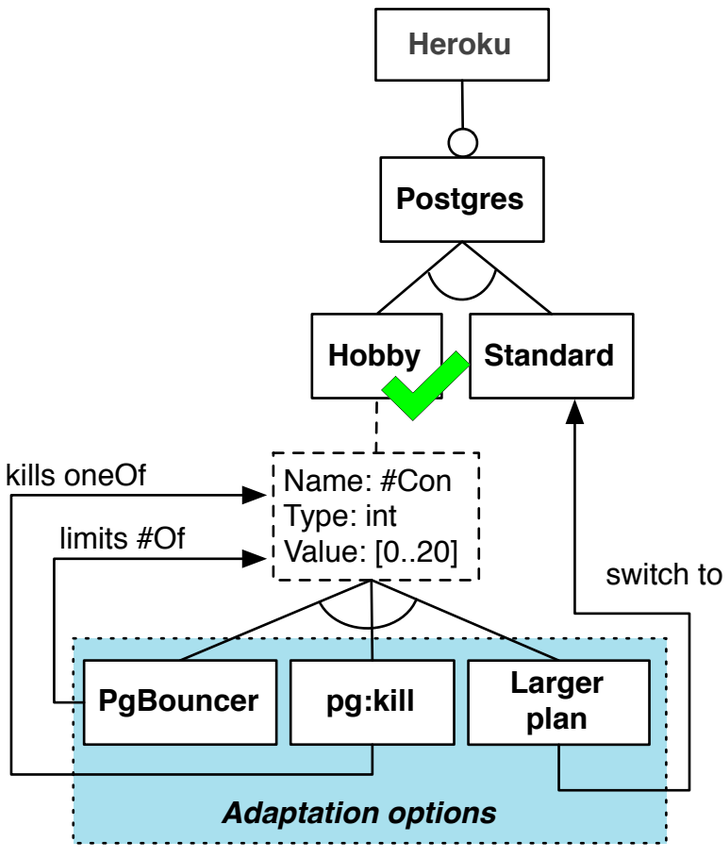


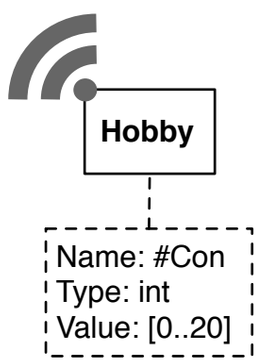


Design Time | **Runtime**

Example

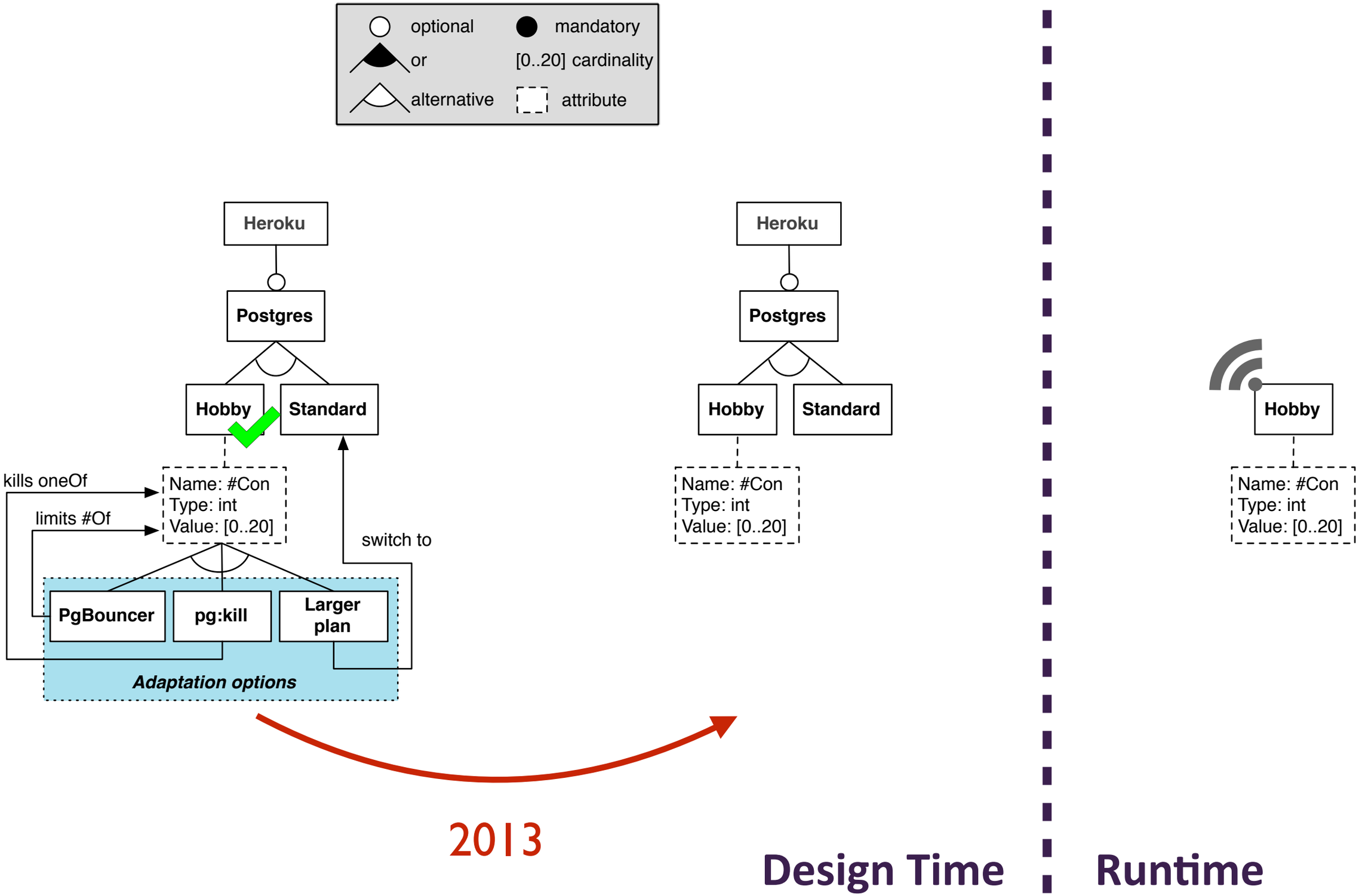




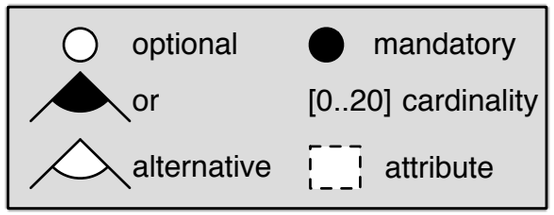


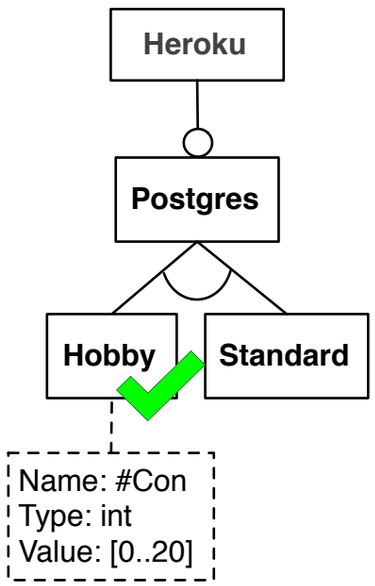
Design Time | **Runtime**

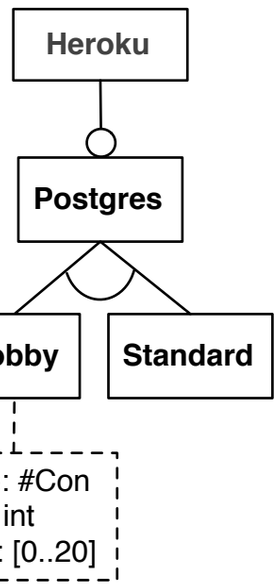
Example


optional
mandatory
or
[0..20] cardinality
alternative
attribute
Heroku
Postgres
Hobby
Standard
kills oneOf
limits #Of
Name: #Con
Type: int
Value: [0..20]
switch to
PgBouncer
pg:kill
Larger plan
Adaptation options
Heroku
Postgres
Hobby
Standard
Name: #Con
Type: int
Value: [0..20]
Hobby
Name: #Con
Type: int
Value: [0..20]
2013
Design Time
Runtime

Example





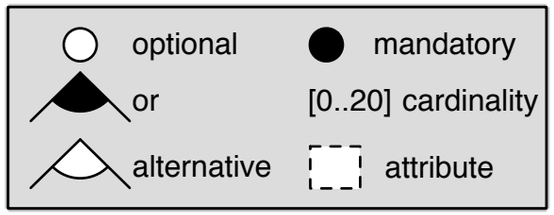


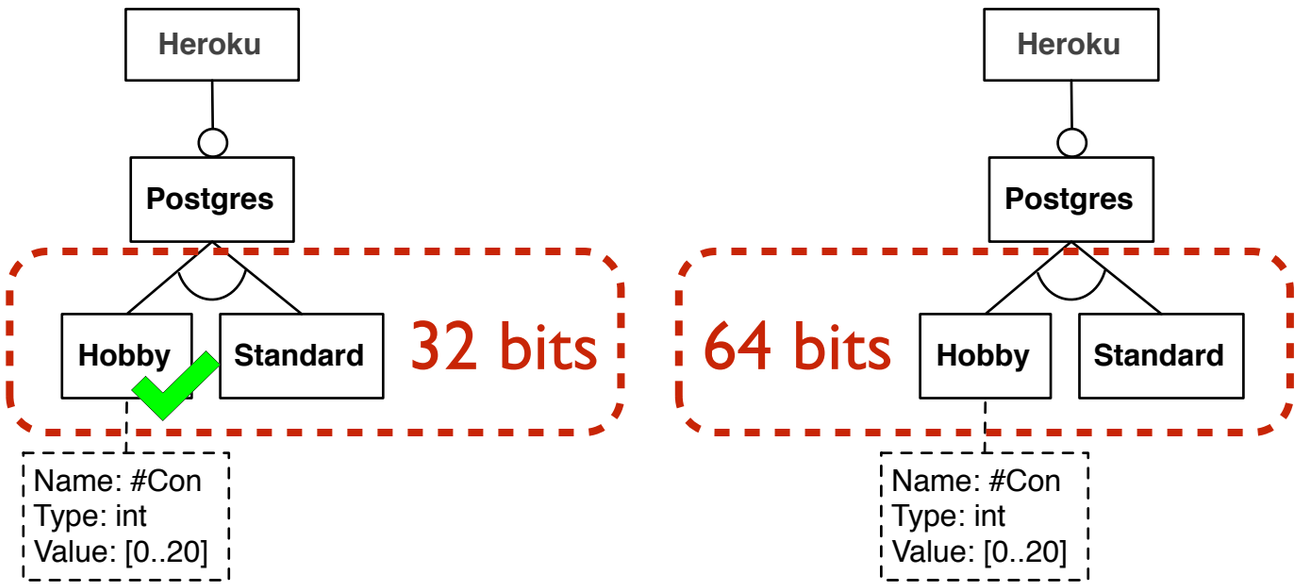
2013

Design Time

Runtime

Example



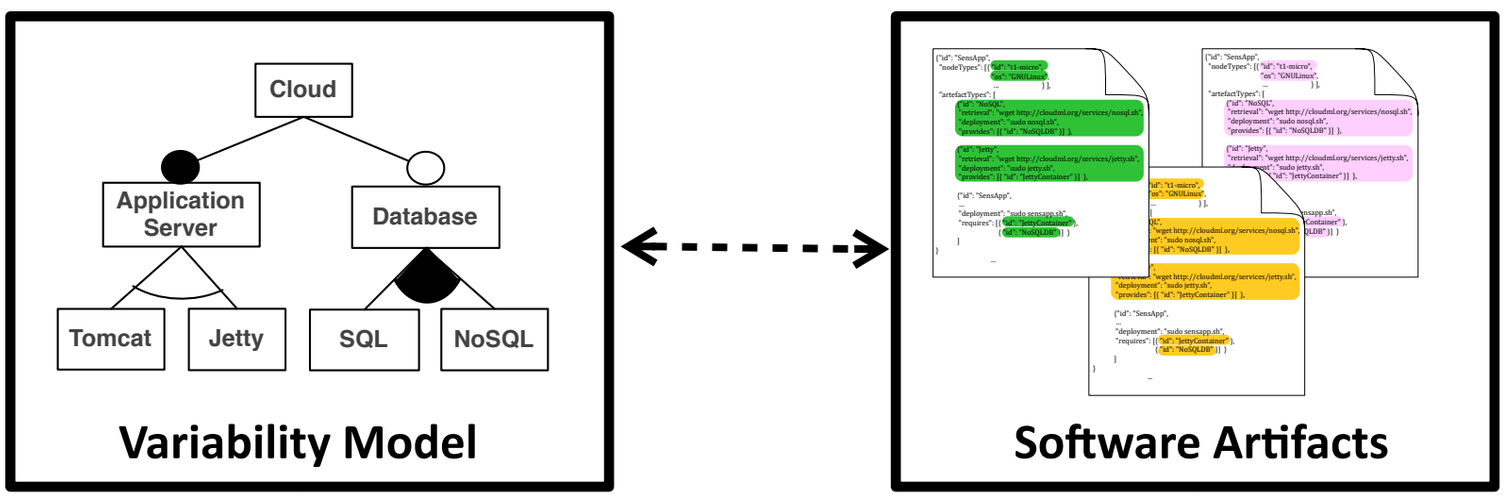


2013

Design Time **Runtime**

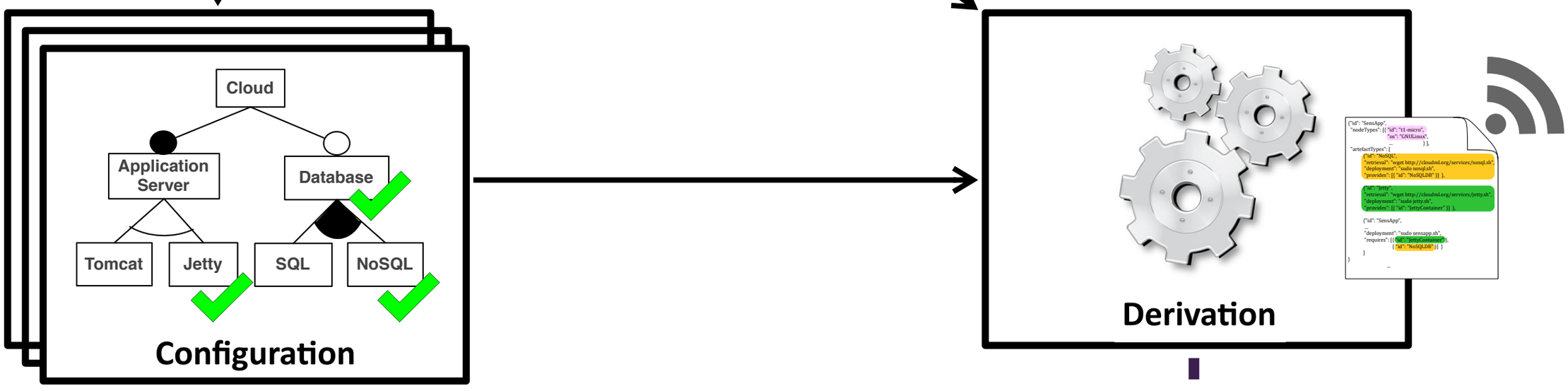
Dynamic Software Product Lines

Domain Engineering



Adaptation

Reconfiguration

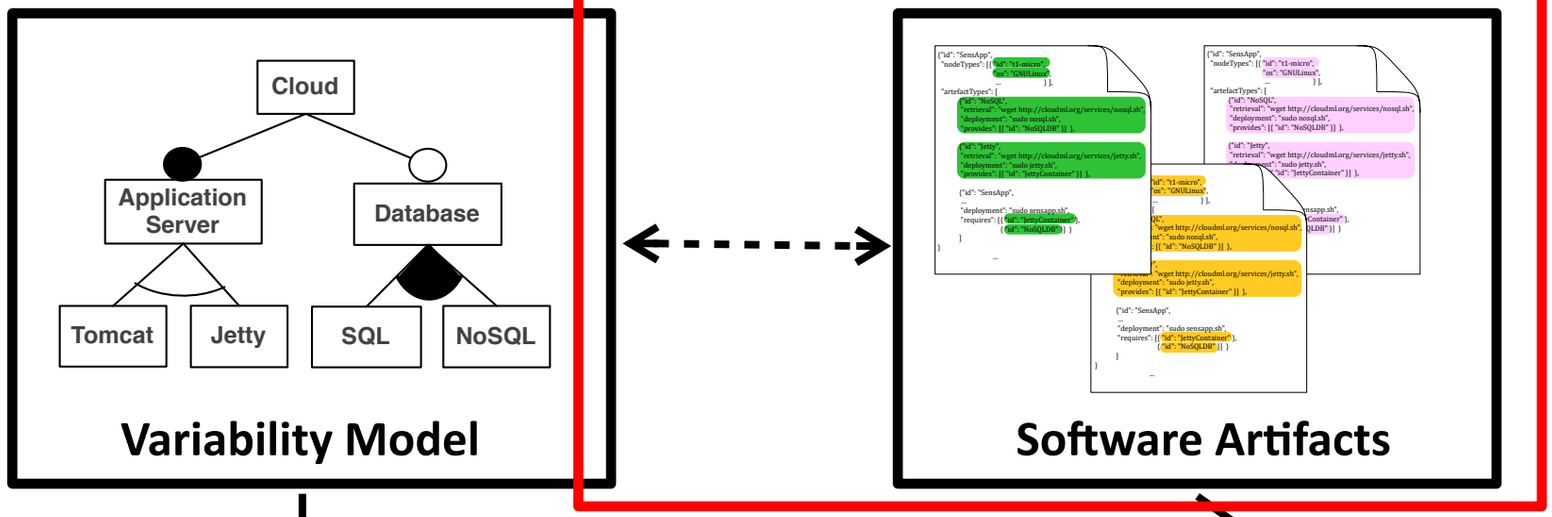


Application Engineering

Design Time | Runtime

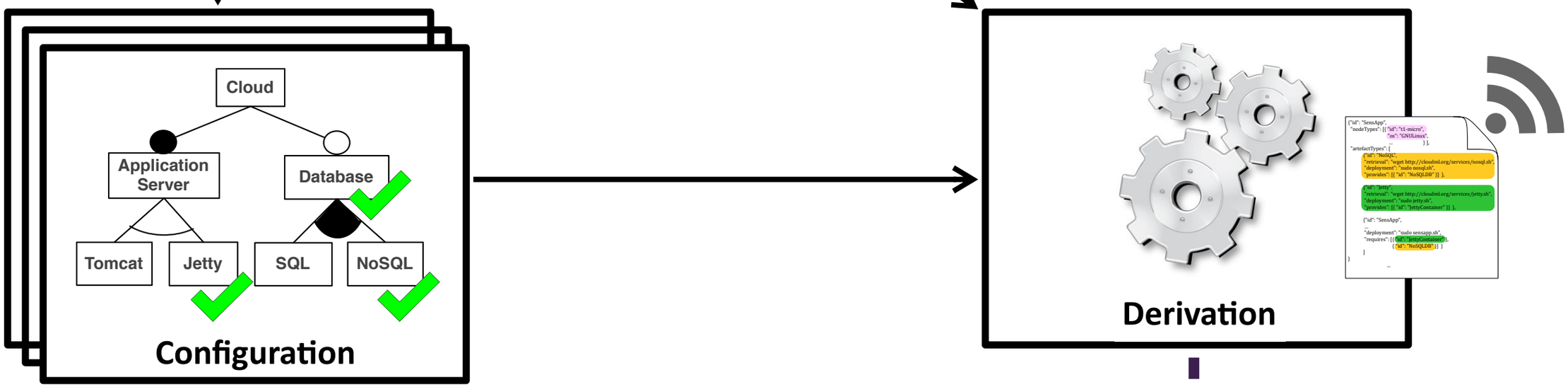
Dynamic Software Product Lines

Domain Engineering



Adaptation

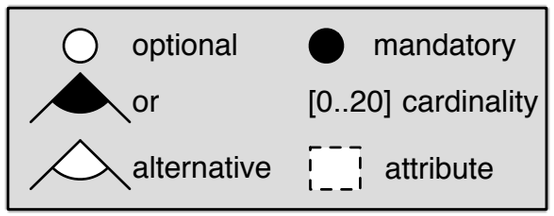
Reconfiguration

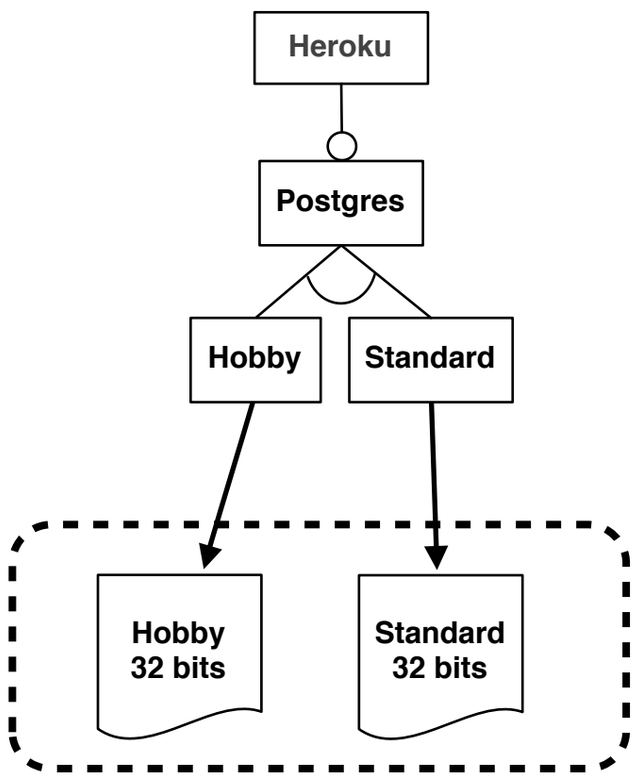


Application Engineering

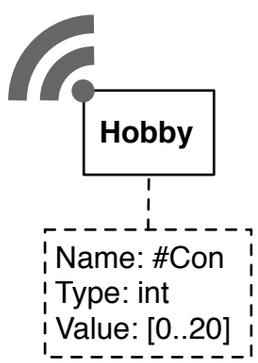
Design Time | Runtime

Example



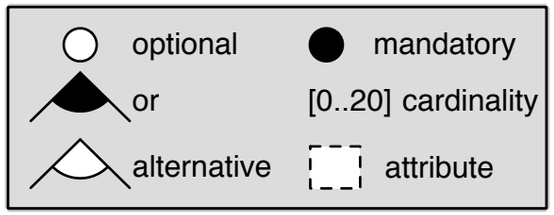


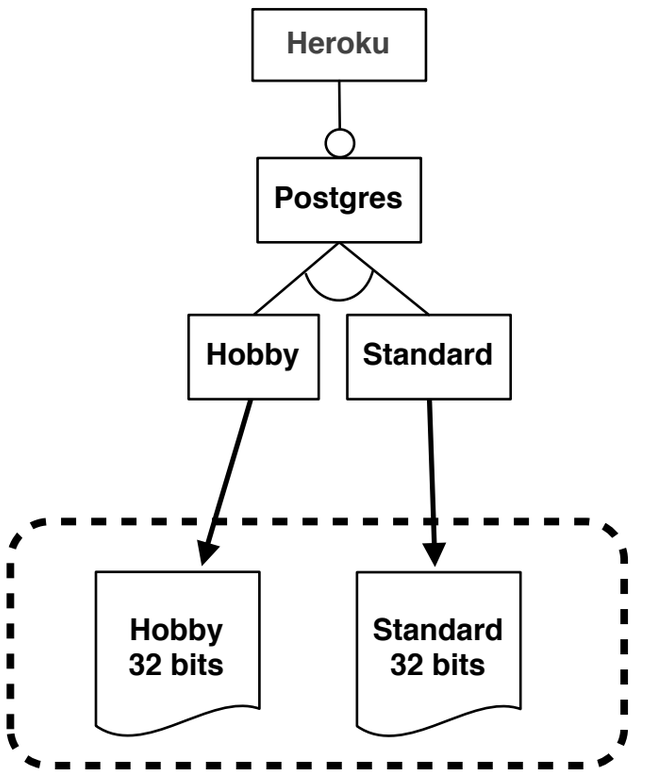
Artifacts repository



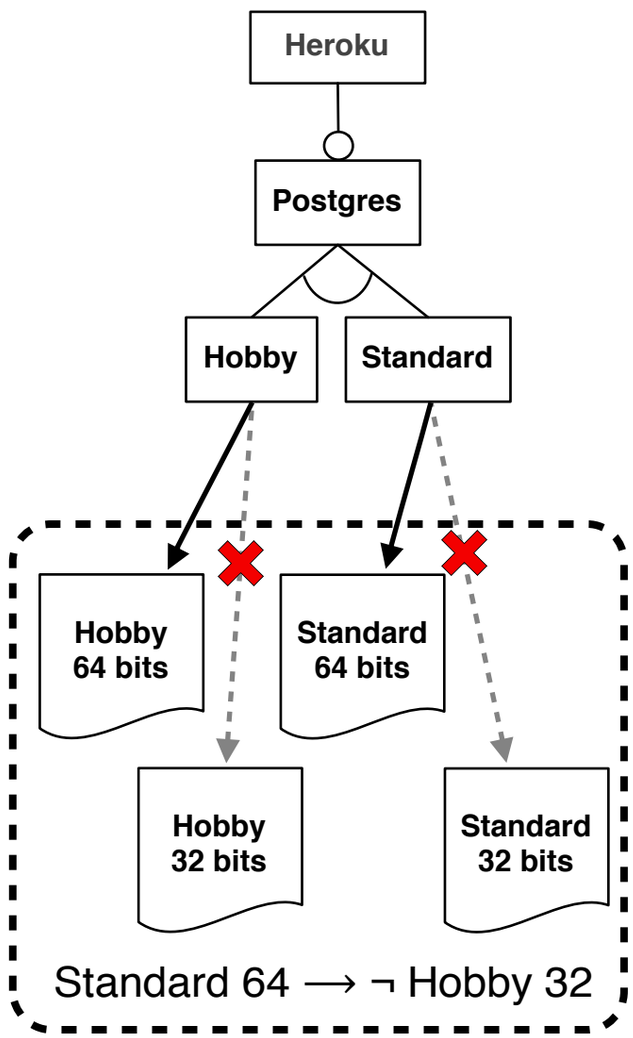
Design Time | **Runtime**

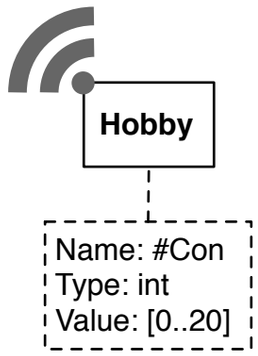
Example





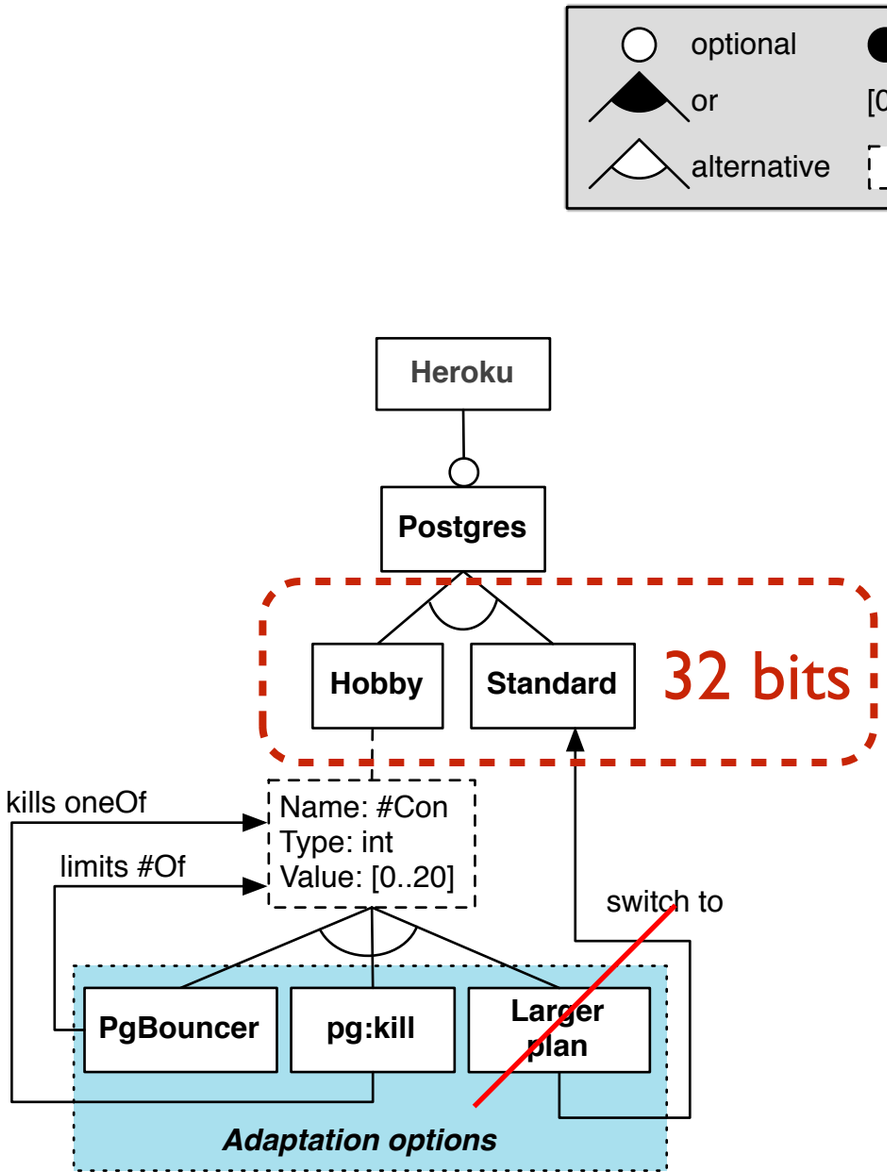
Artifacts repository

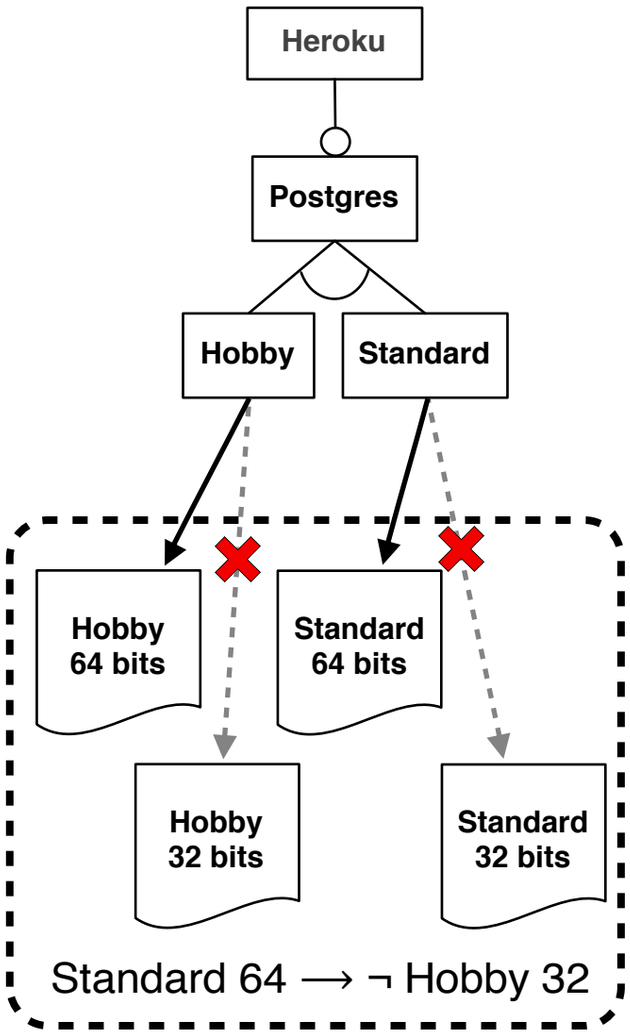


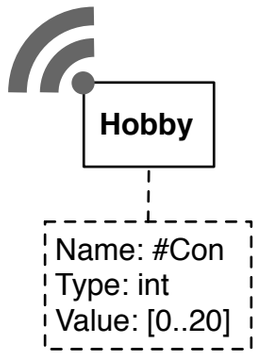


Design Time | **Runtime**

Example







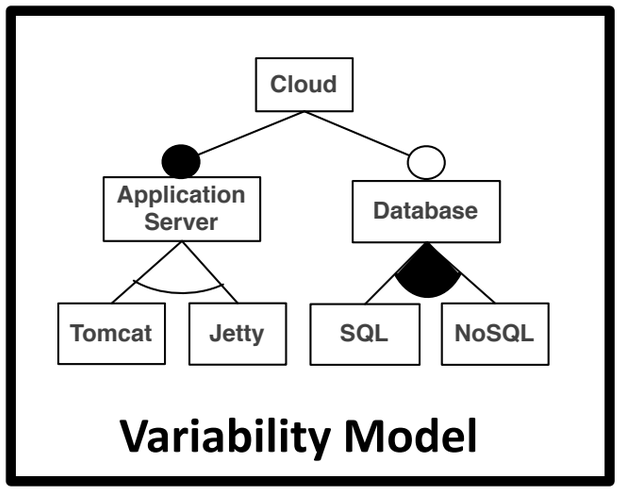
Design Time **Runtime**

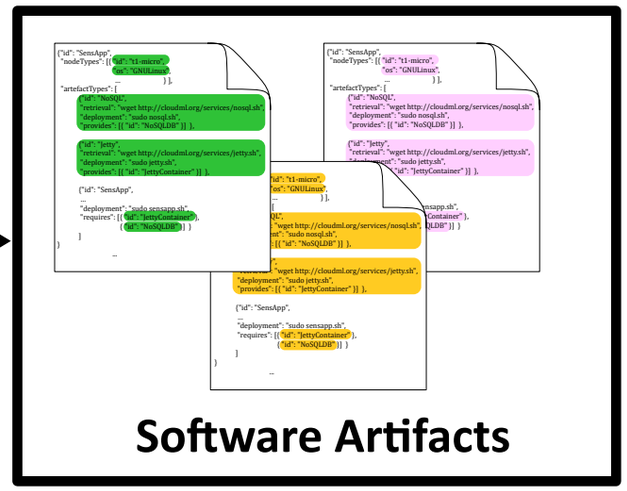
The problem: consistency

Two dimensions: Space and Time

Space

Problem Space

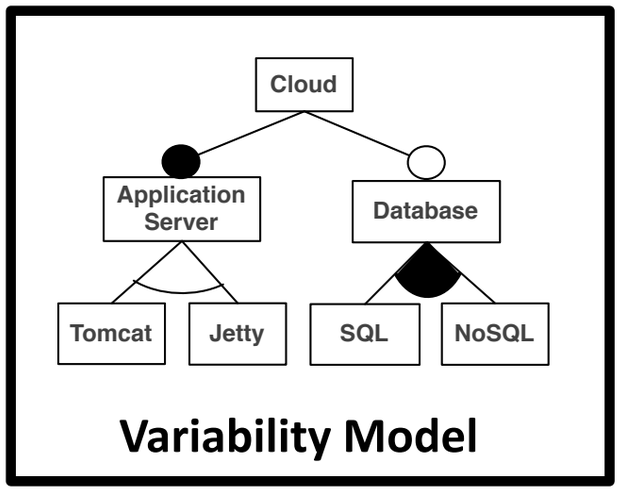


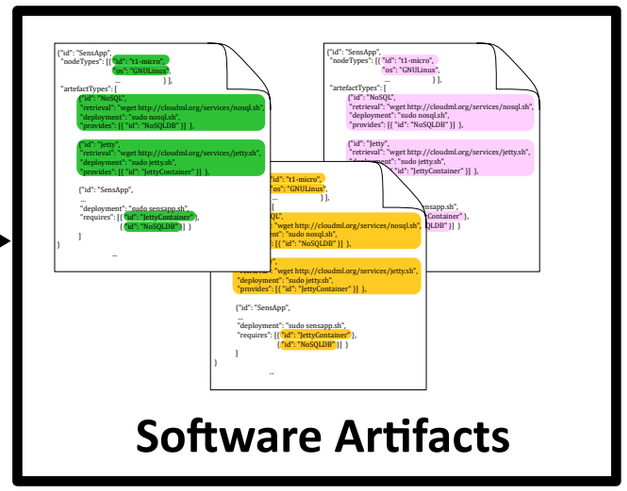


Solution Space

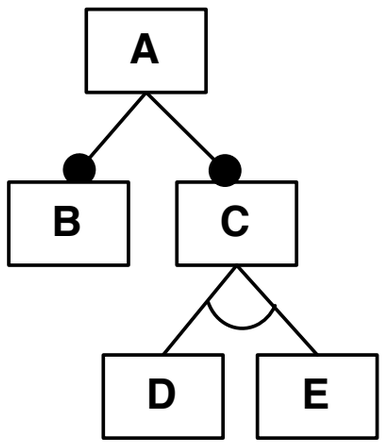
Space

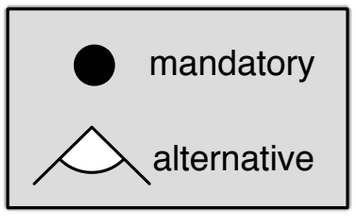
Problem Space





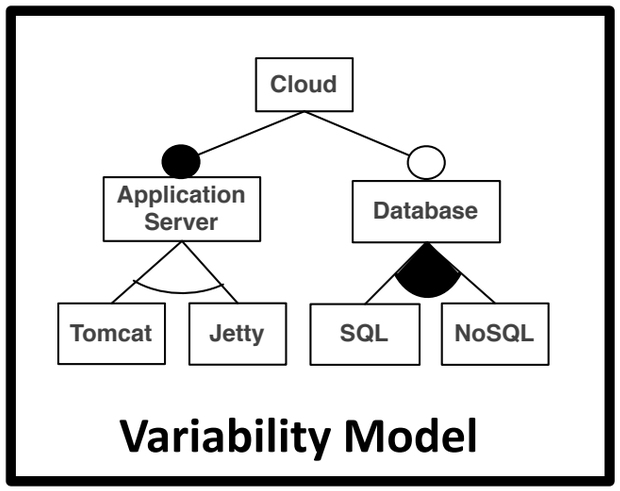
Solution Space

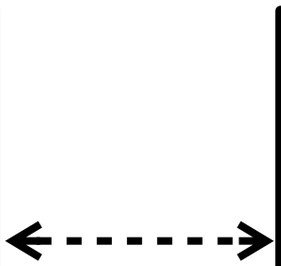


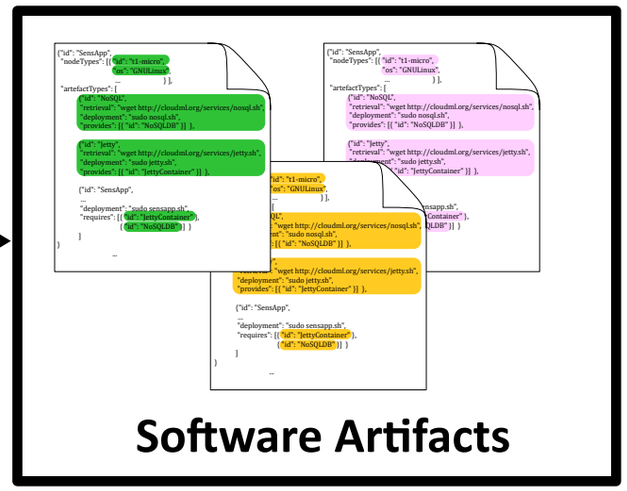


Space

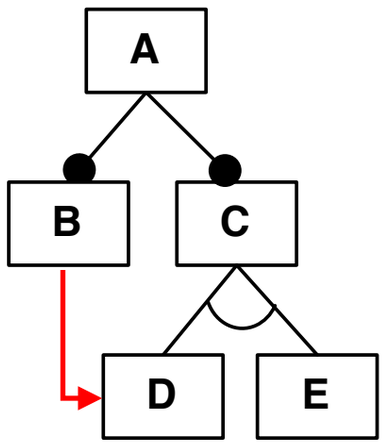
Problem Space

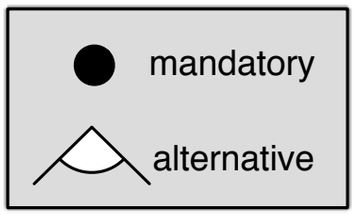






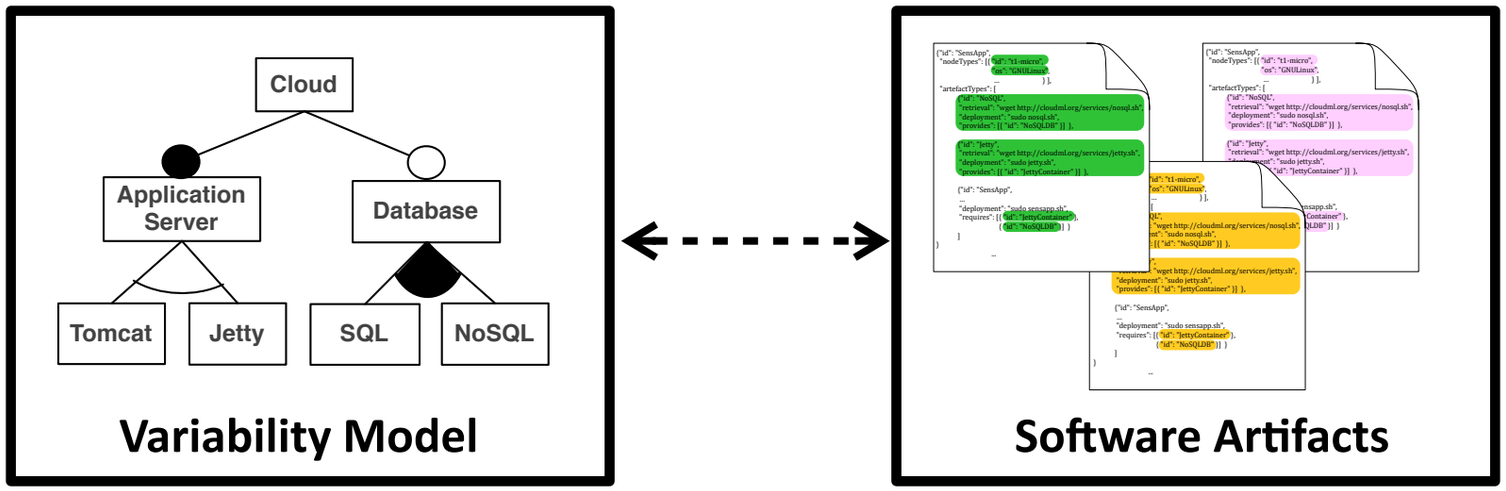
Solution Space





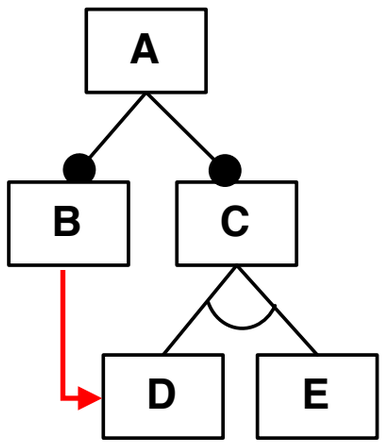
Space

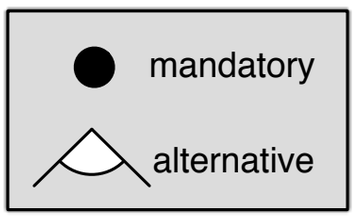
Problem Space



Solution Space

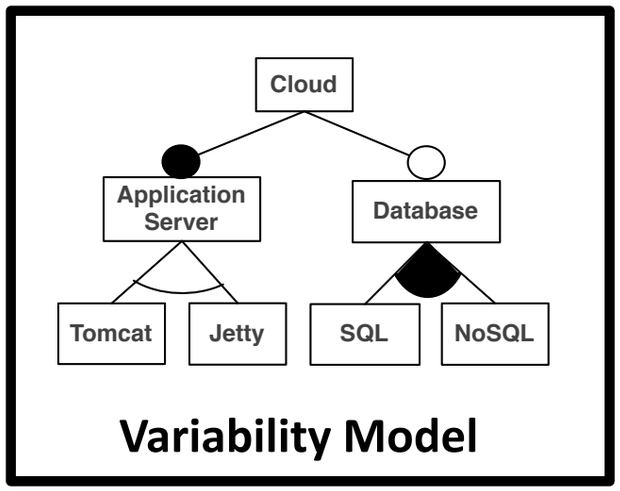
E is a dead feature

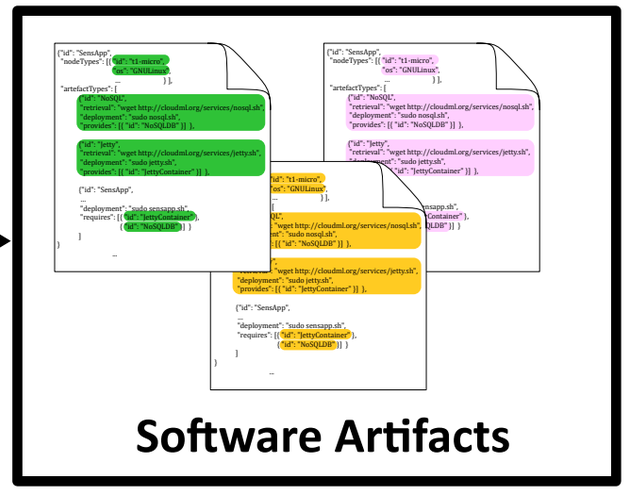




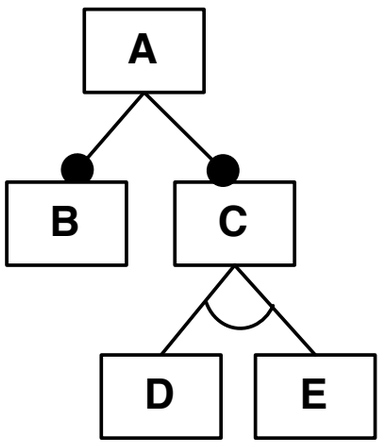
Space

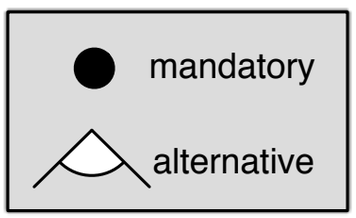
Problem Space

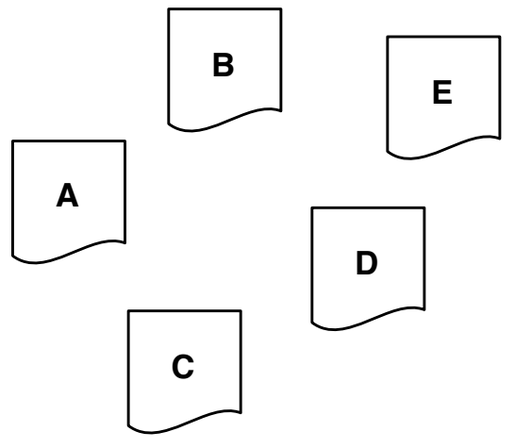




Solution Space



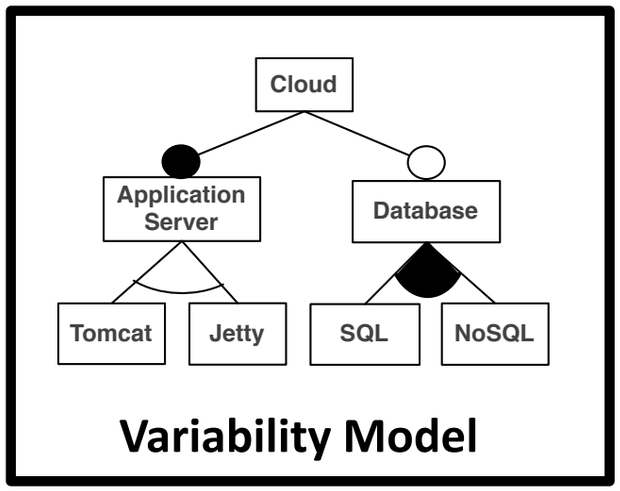


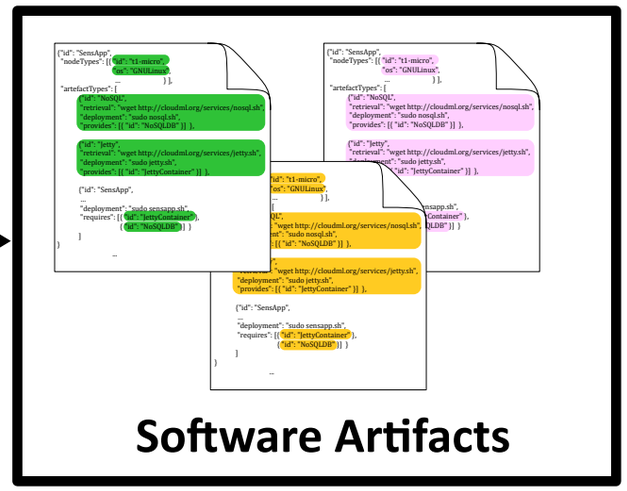


A ⟶ B
A ⟶ C
C ⟶ D
C ⟶ E

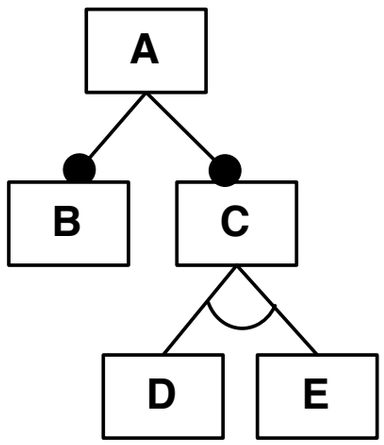
Space

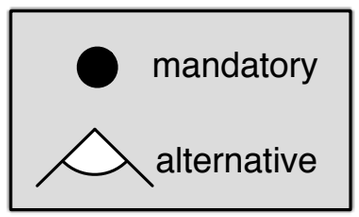
Problem Space

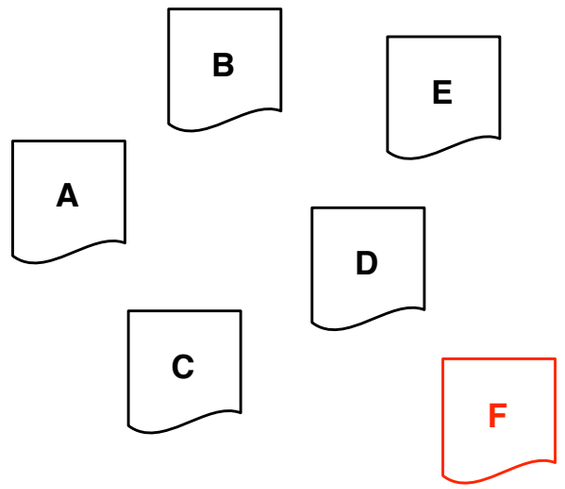




Solution Space



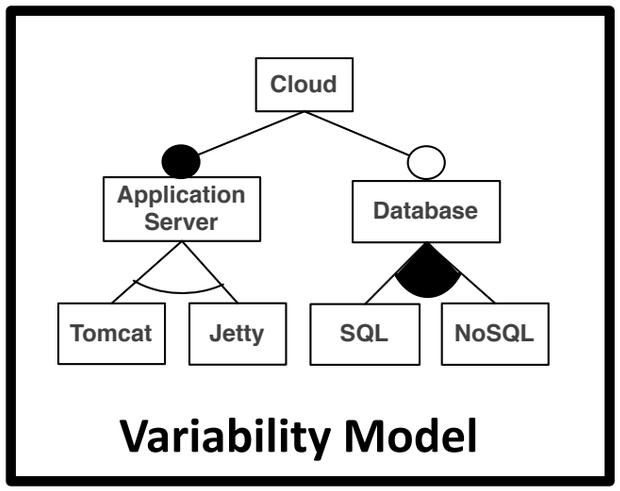


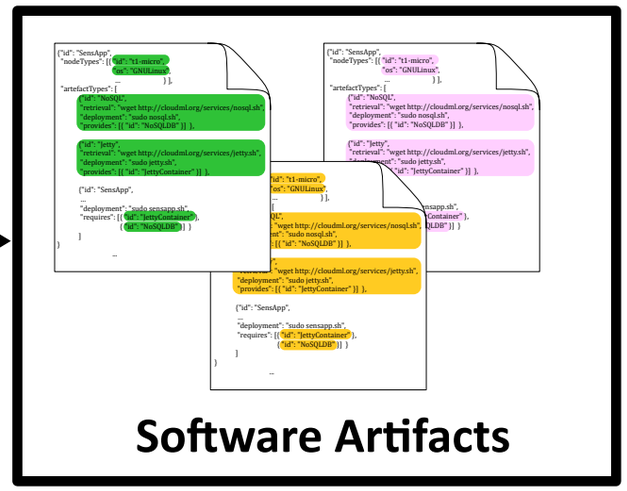


$A \longrightarrow B$
$A \longrightarrow C$
$C \longrightarrow D$
$C \longrightarrow E$
$F \longrightarrow C$
$F \longrightarrow \neg E$

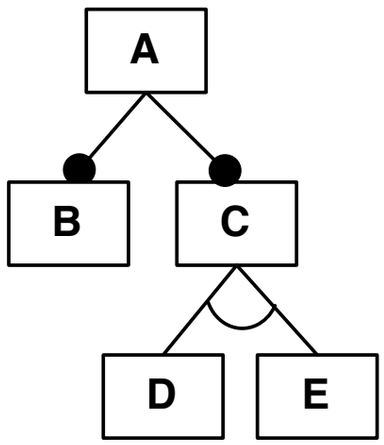
Space

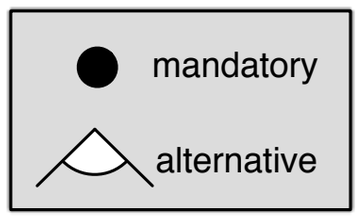
Problem Space

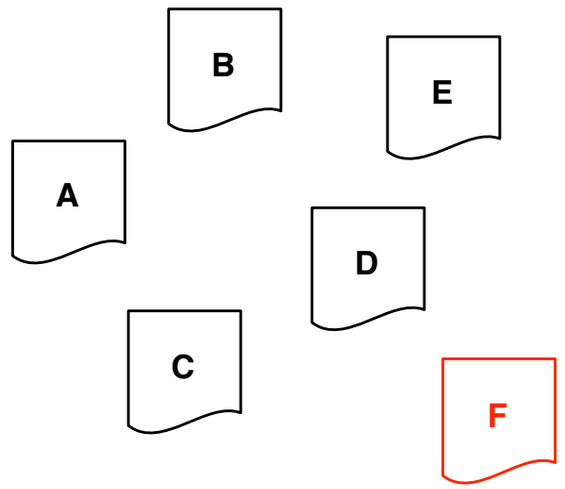




Solution Space



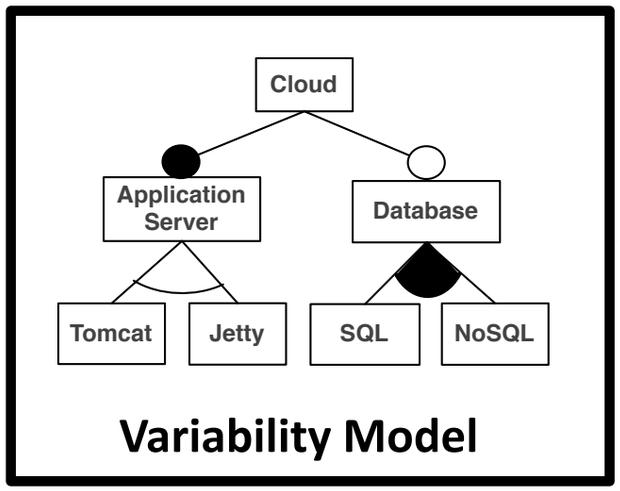


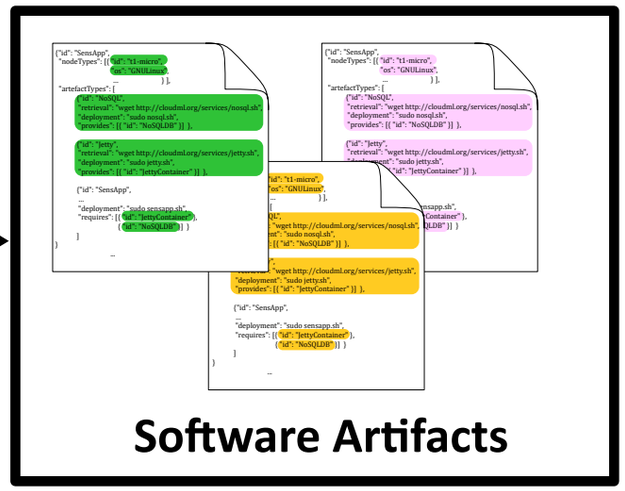


$A \longrightarrow B$
$A \longrightarrow C$
$C \longrightarrow D$
$C \longrightarrow E$
$F \longrightarrow C$
$F \longrightarrow \neg E$

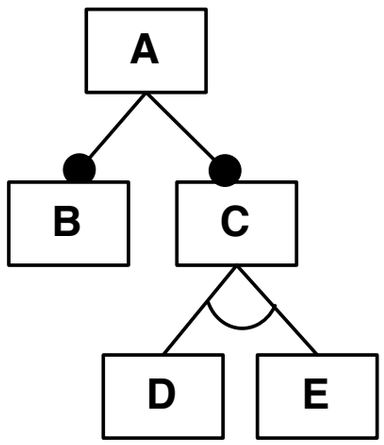
Space

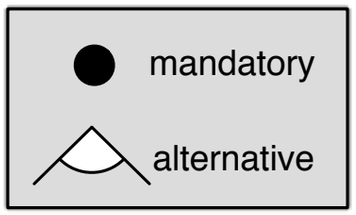
Problem Space

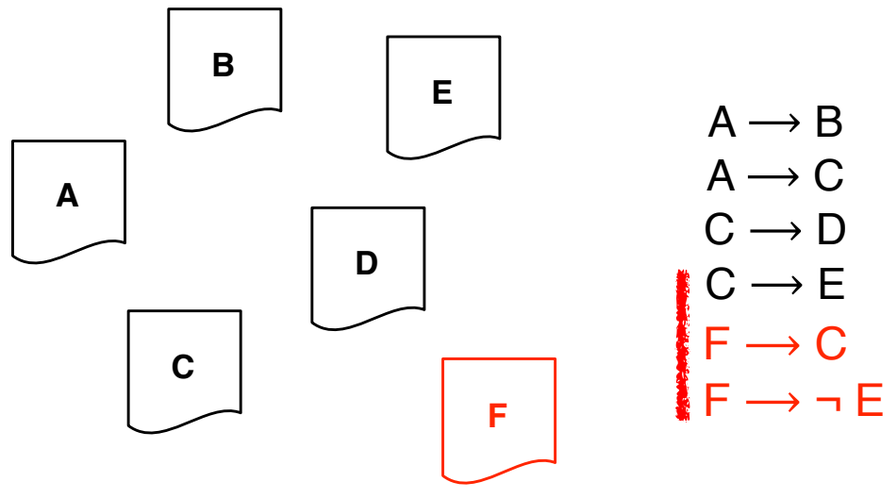




Solution Space



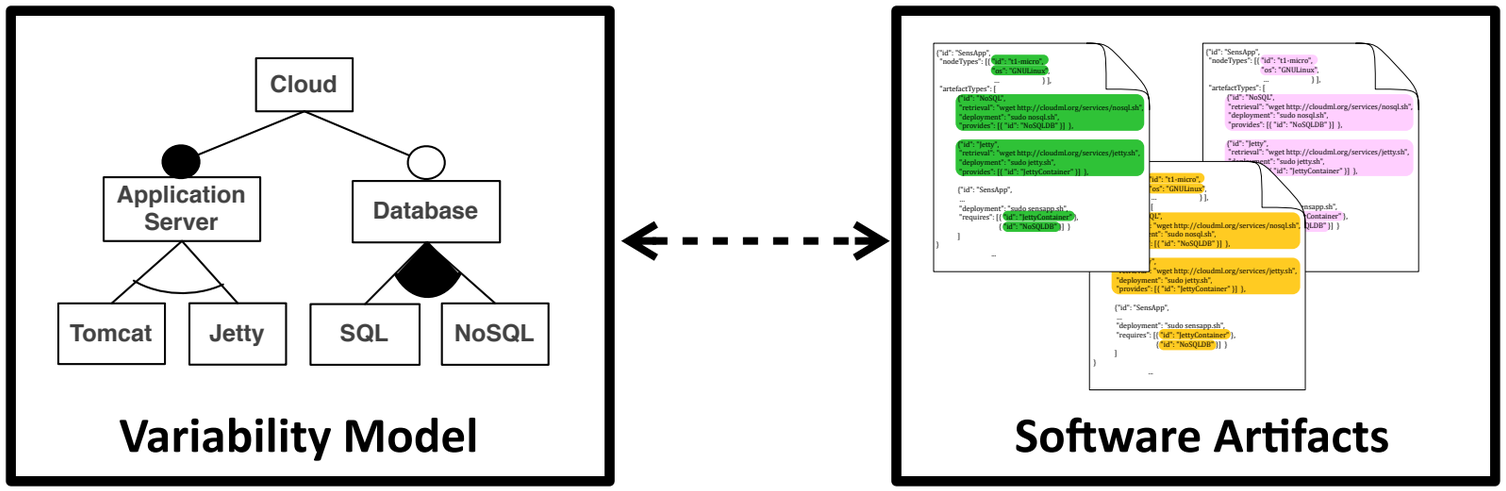




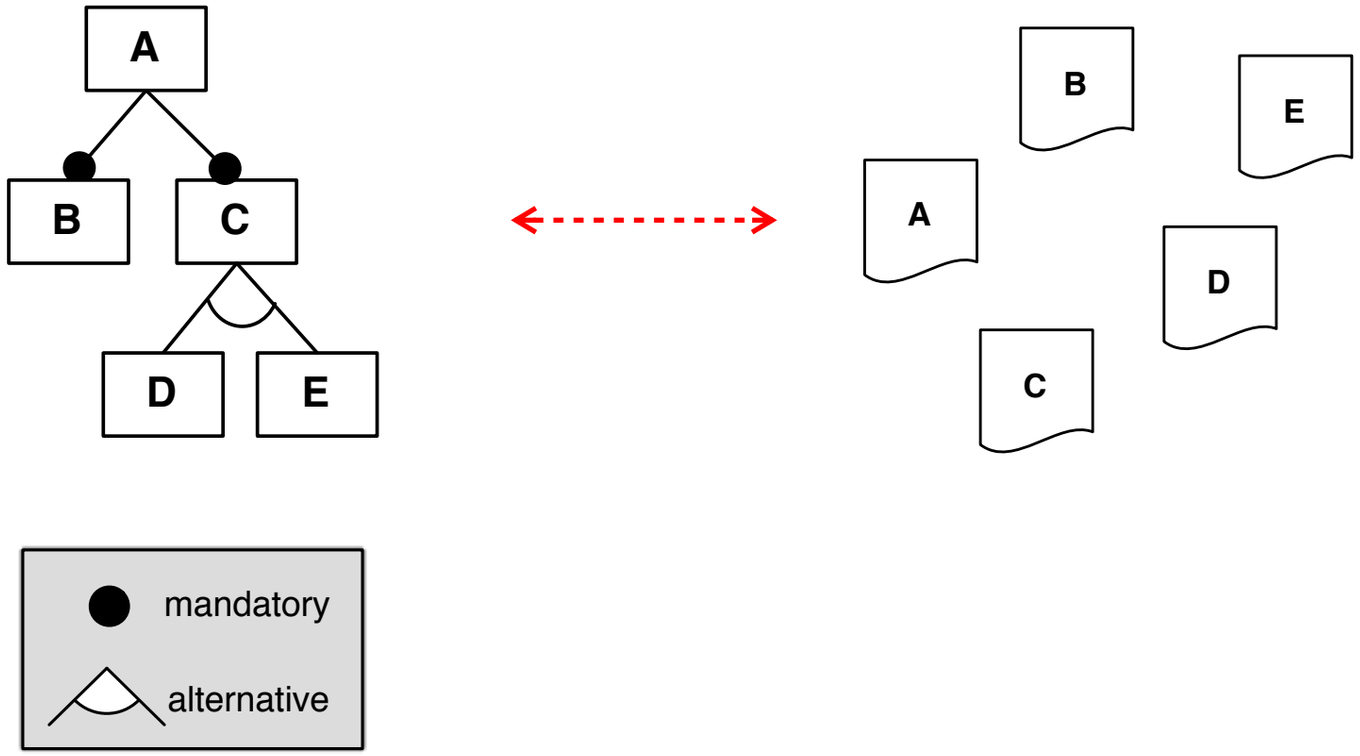
Incompatibilities between
- packages (e.g. Linux)
- plugins (e.g. Eclipse)

Space

Problem Space

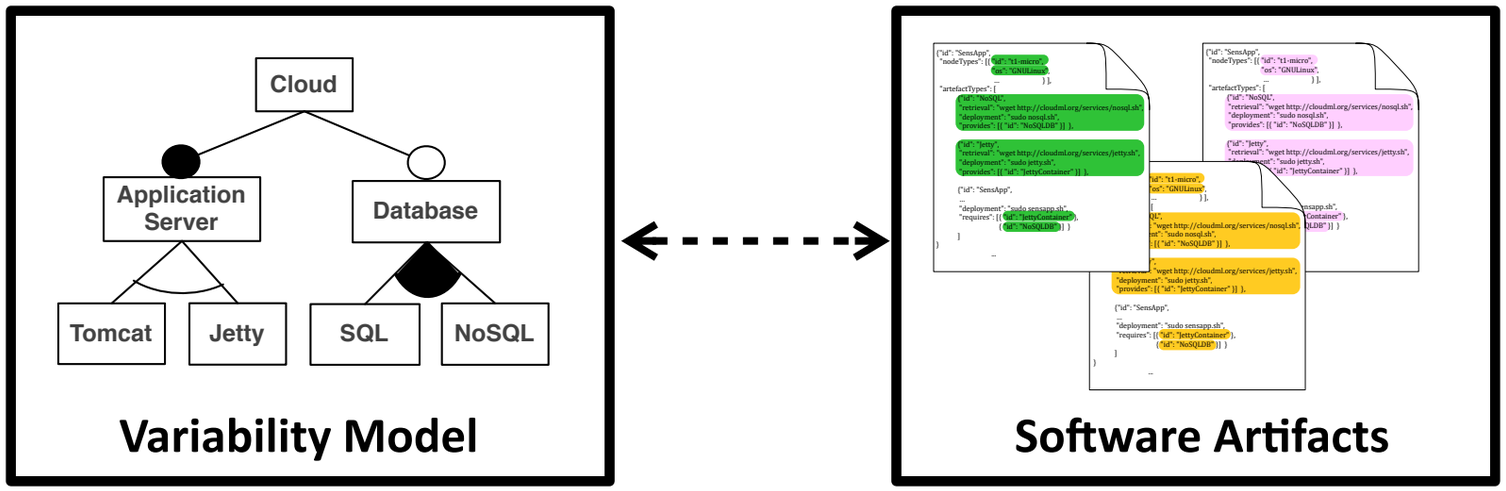


Solution Space

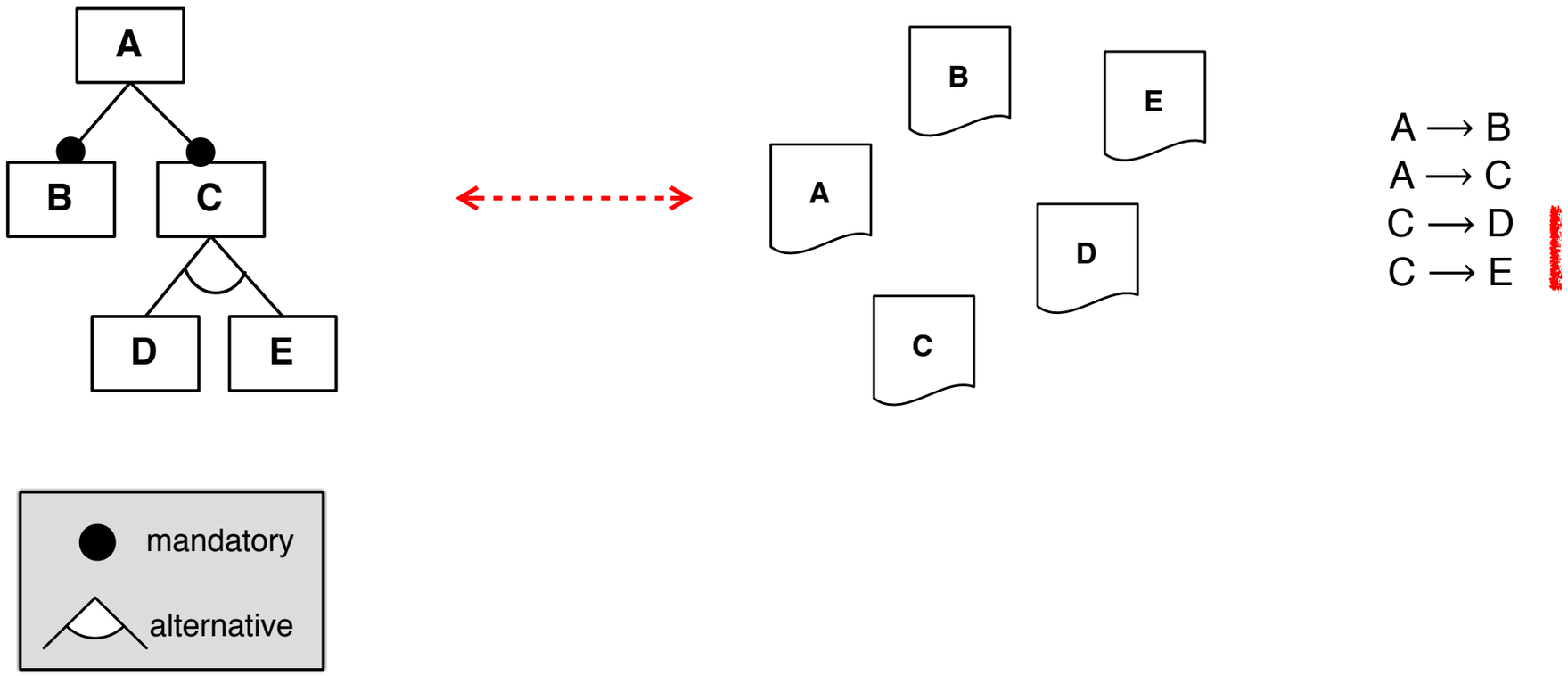


Space

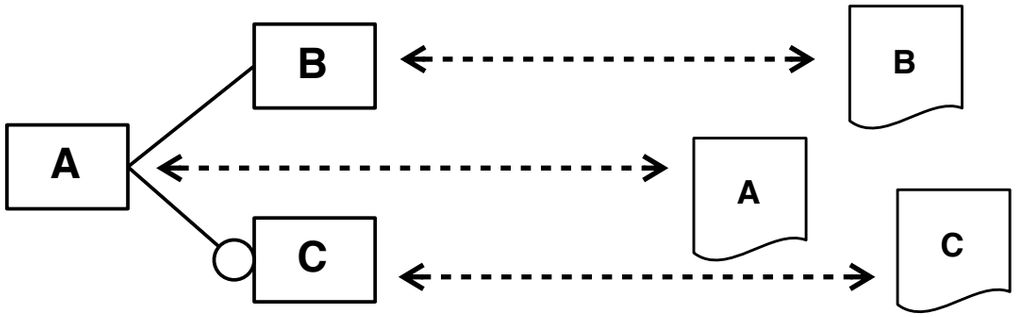
Problem Space



Solution Space



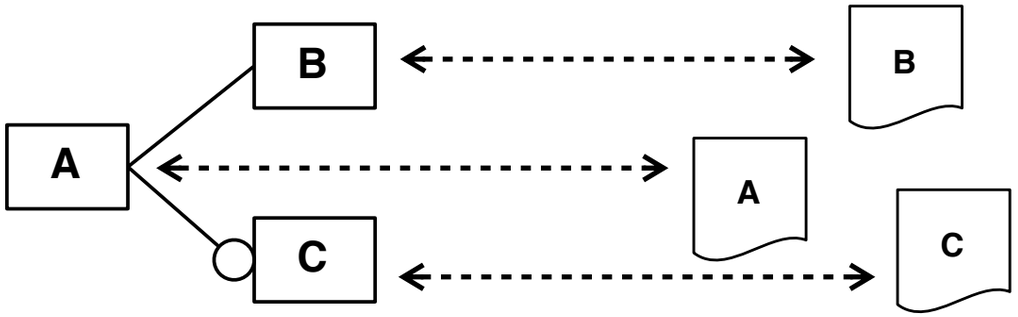
Time



○ optional

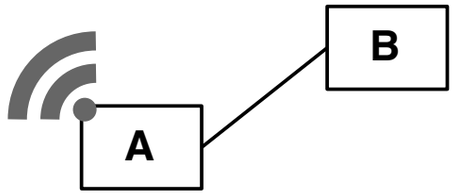
Design Time | **Runtime**

Time



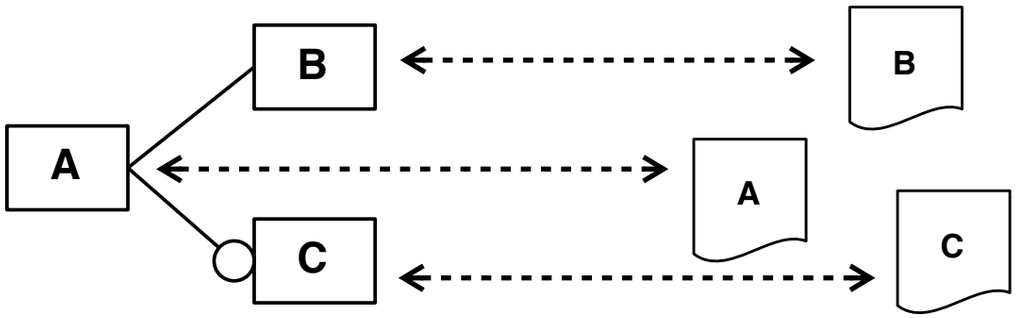
optional

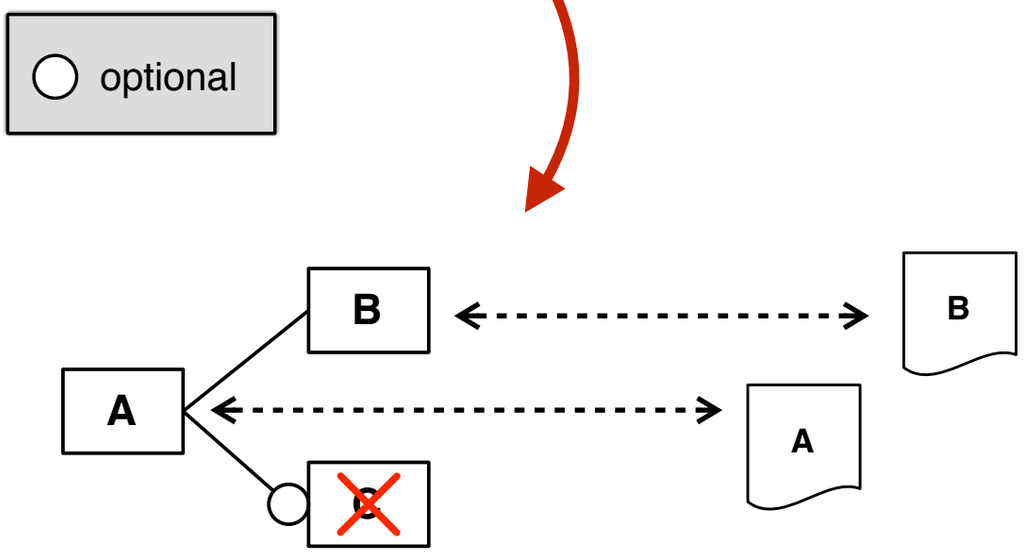
Adaptation rule
if *condition* then C



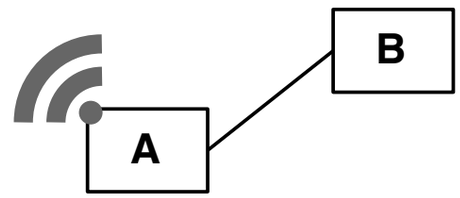
Design Time | **Runtime**

Time



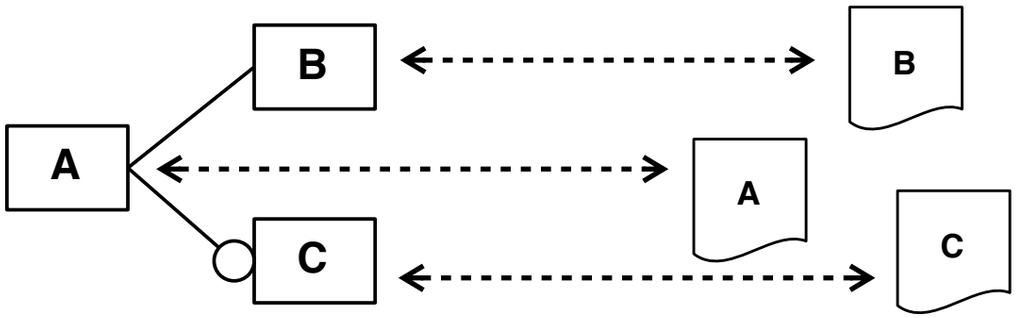


Adaptation rule
if *condition* then C

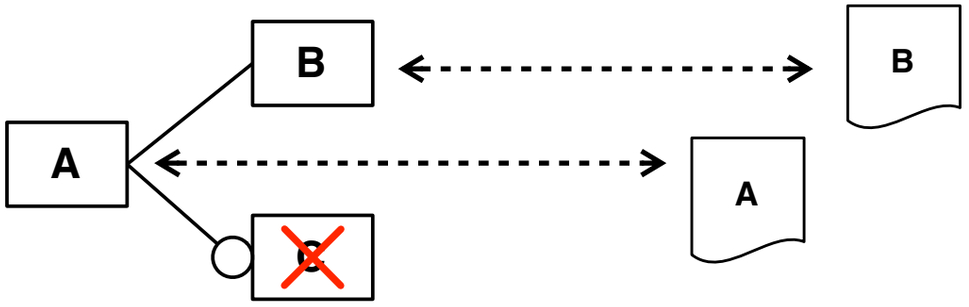


Design Time | **Runtime**

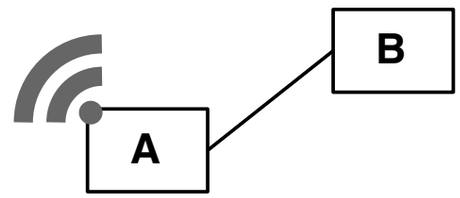
Time



optional



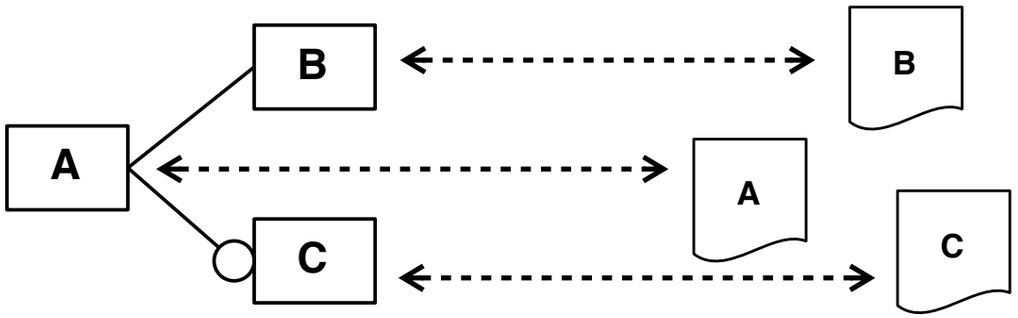
Adaptation rule
if *condition* then C

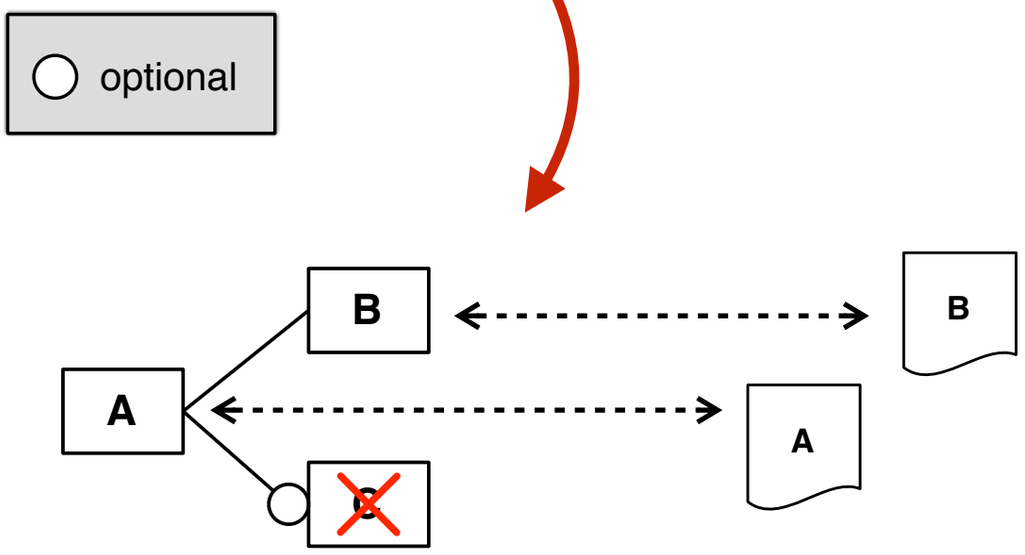


condition is met

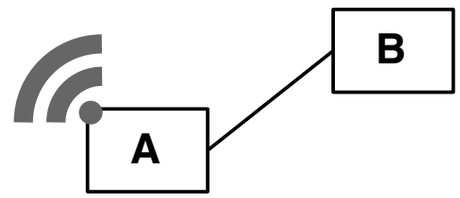
Design Time **Runtime**

Time





Adaptation rule
if *condition* then C

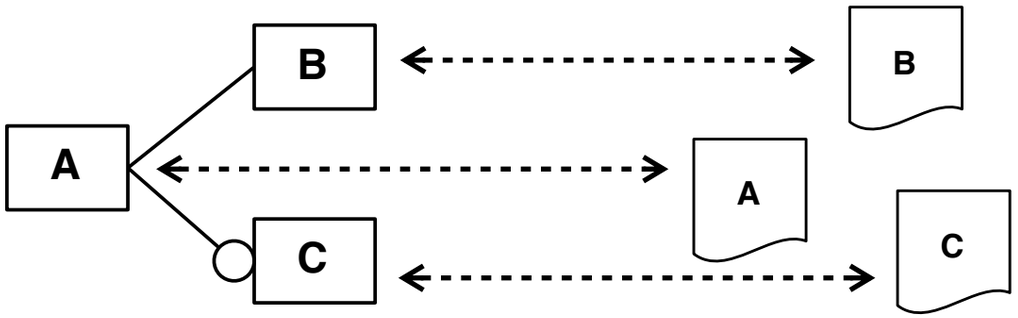


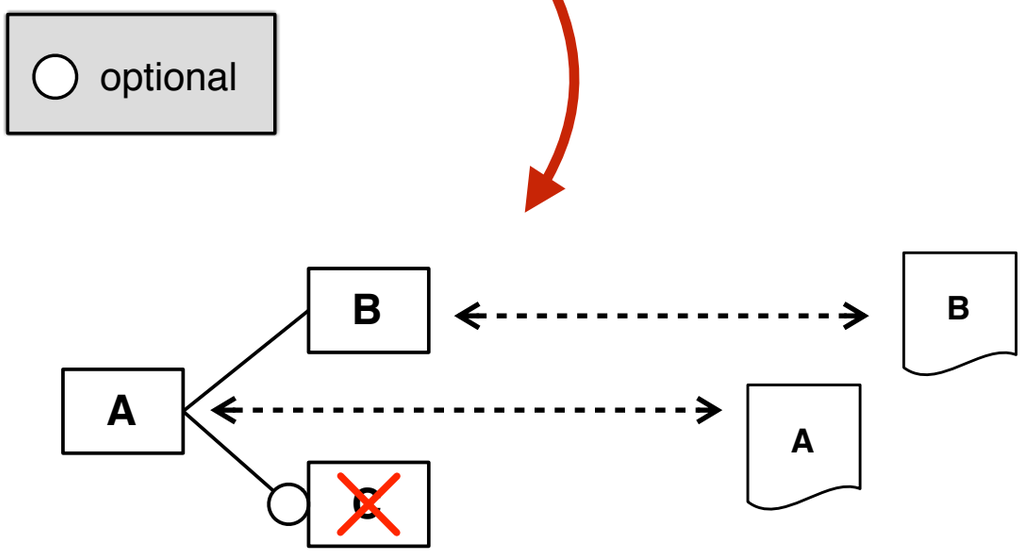
condition is met
then

?

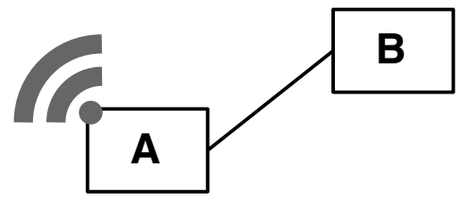
Design Time **Runtime**

Time





Adaptation rule
if *condition* then C



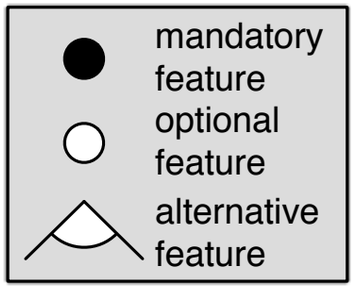
condition is met
then
- nothing
- something else

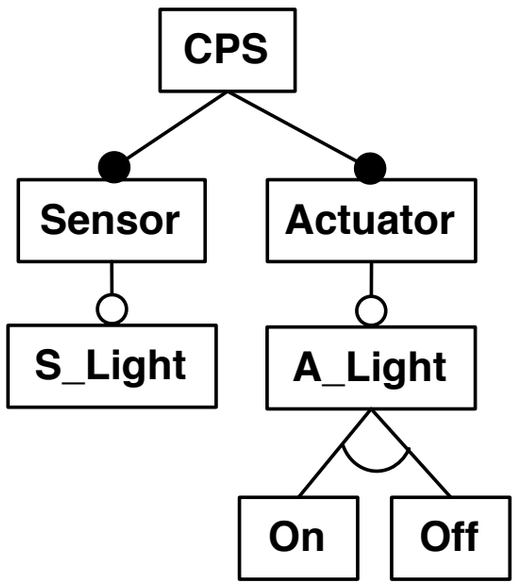
?

Design Time **Runtime**

It depends

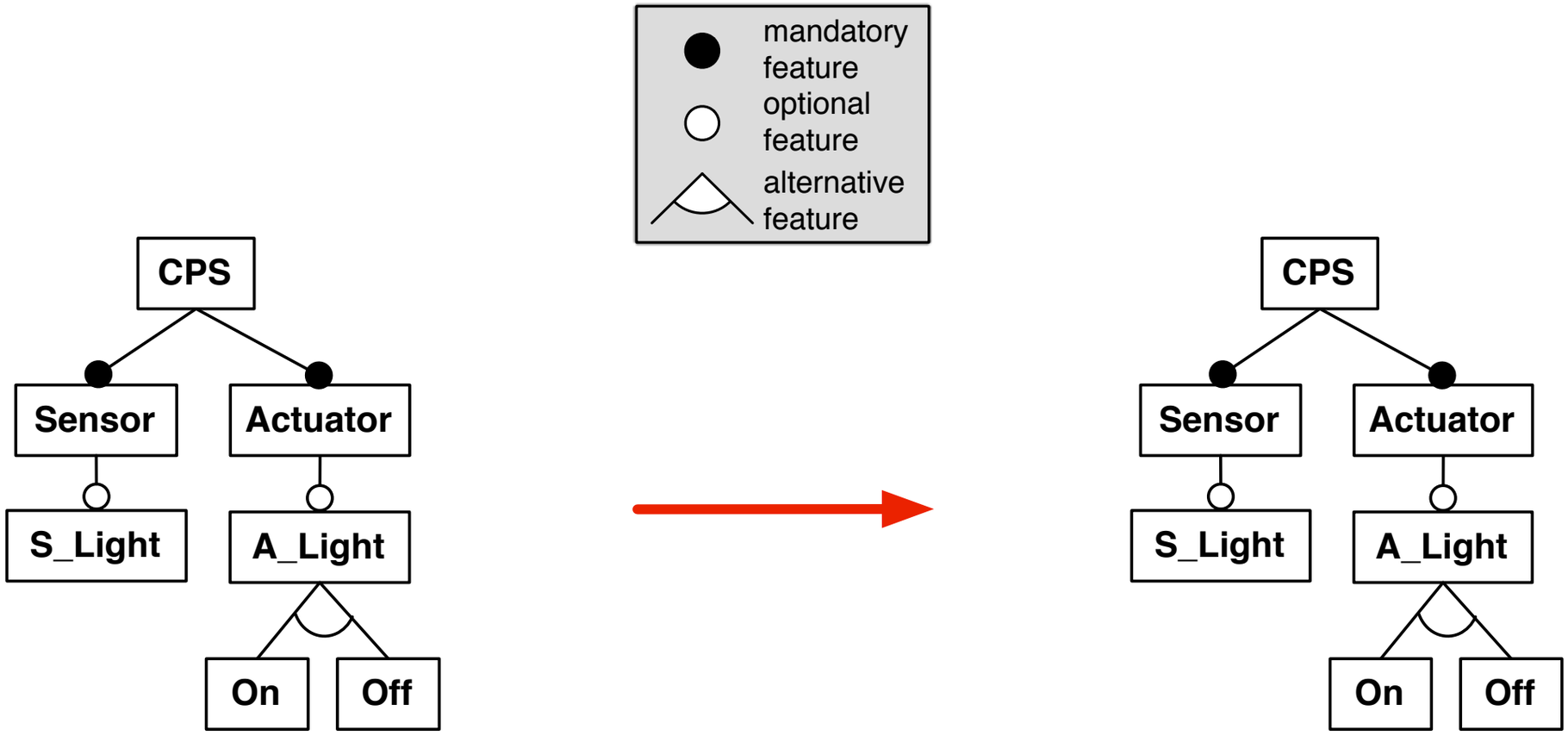
It depends





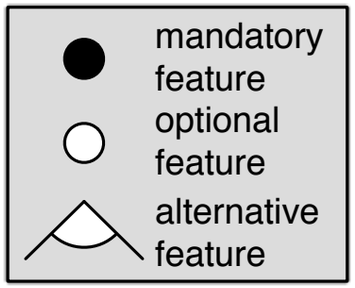
luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

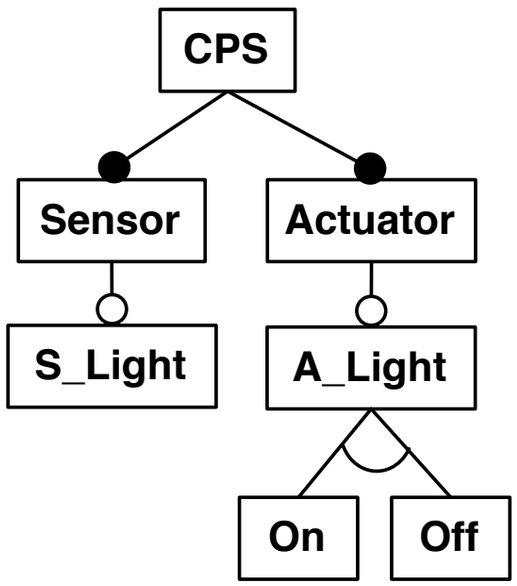
It depends

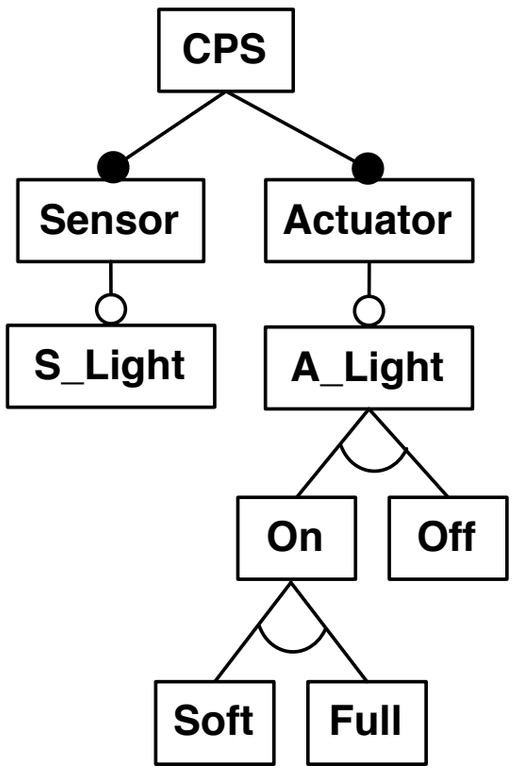


luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

It depends

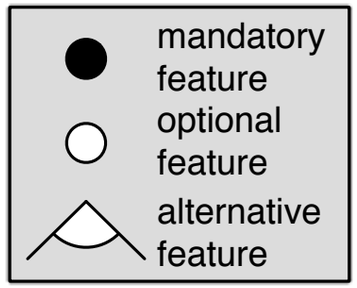


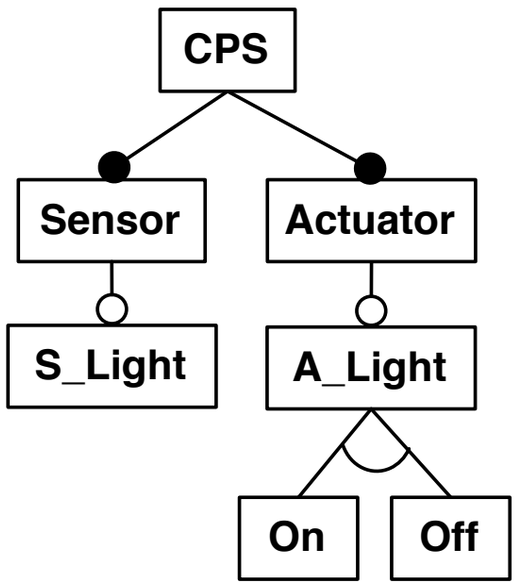


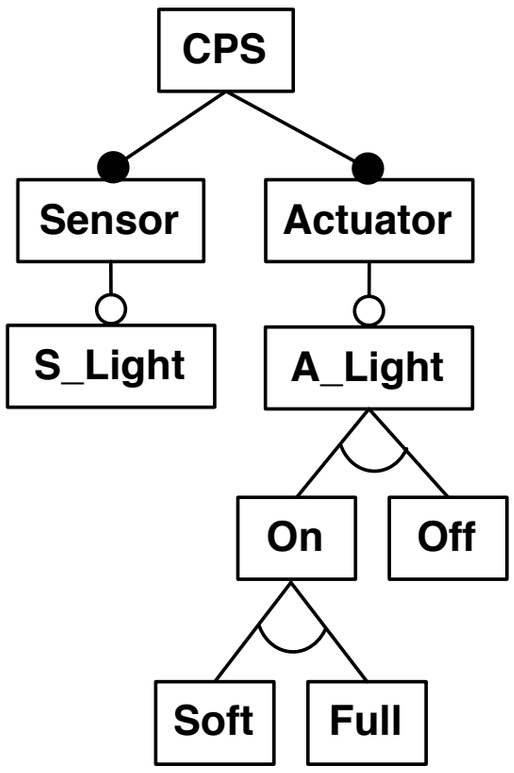


luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

It depends



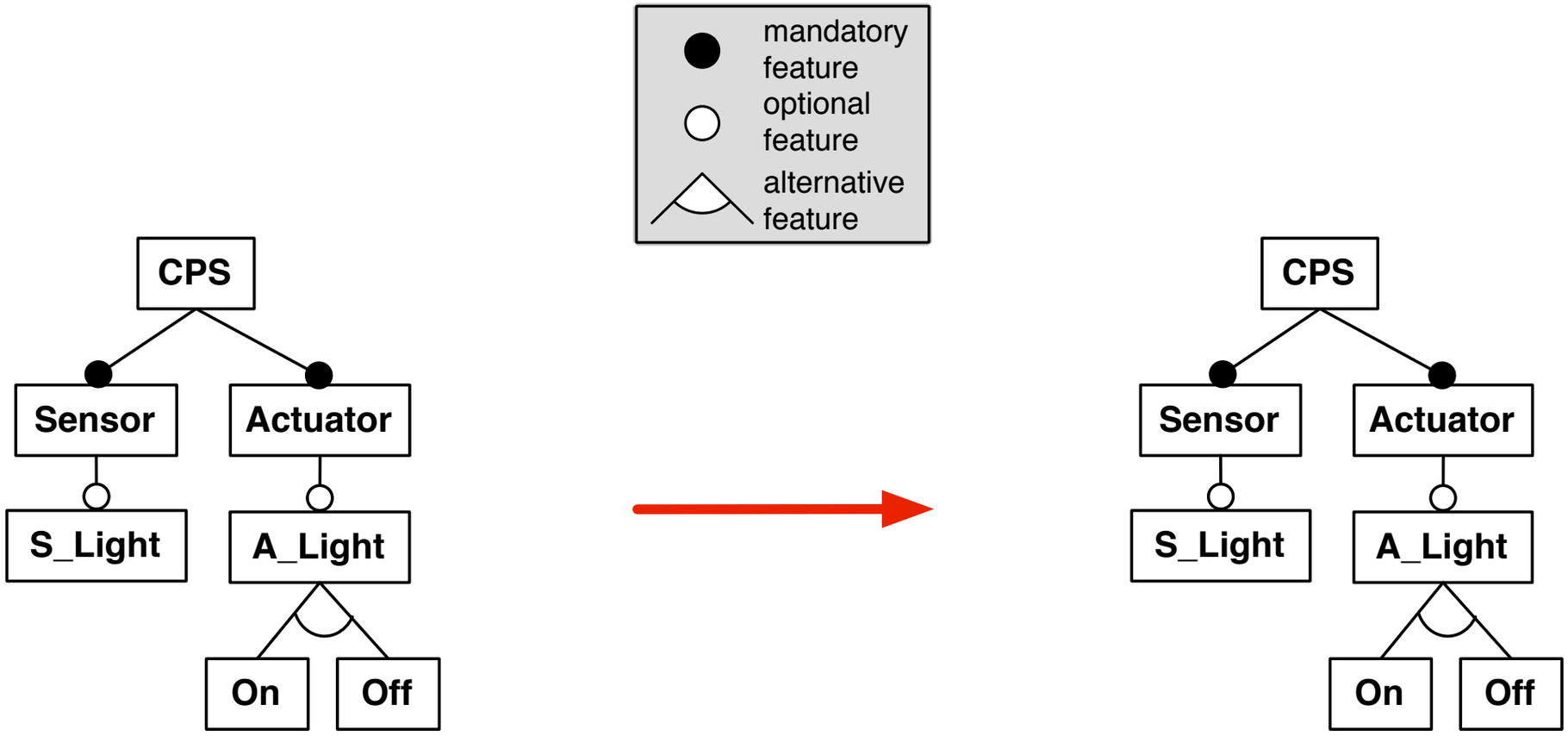




luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

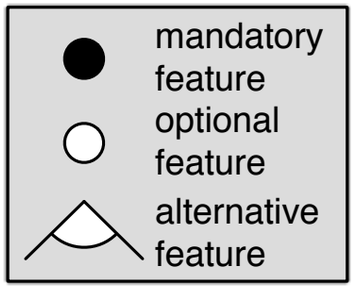
Automated evolution (based on rules, default solver choice)
Manual evolution (user choice)

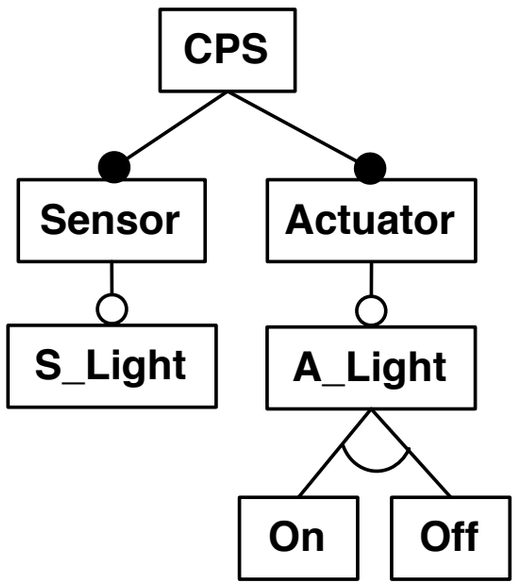
It depends

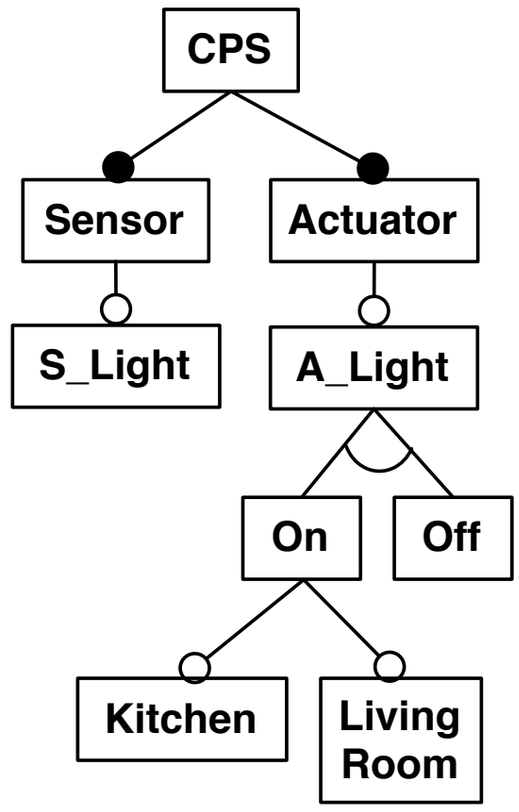


luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

It depends

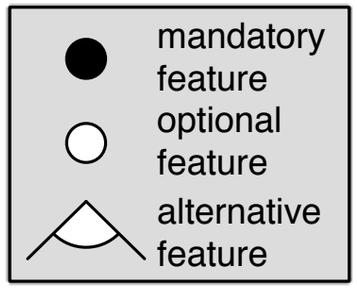


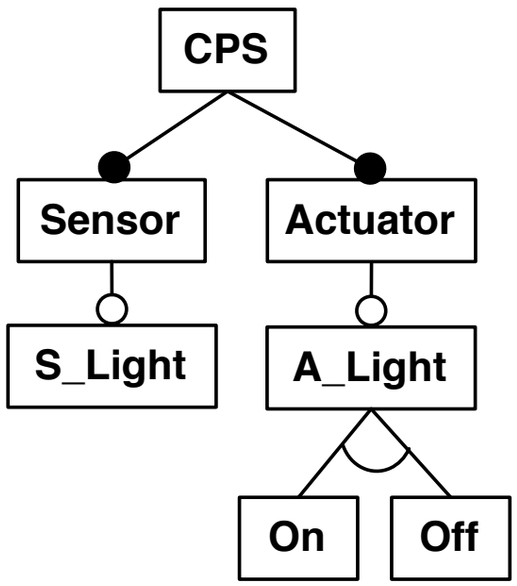




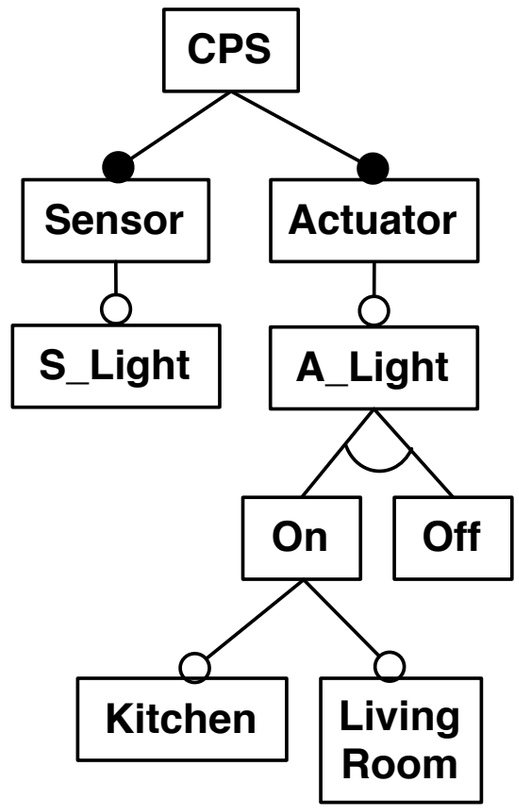
luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

It depends



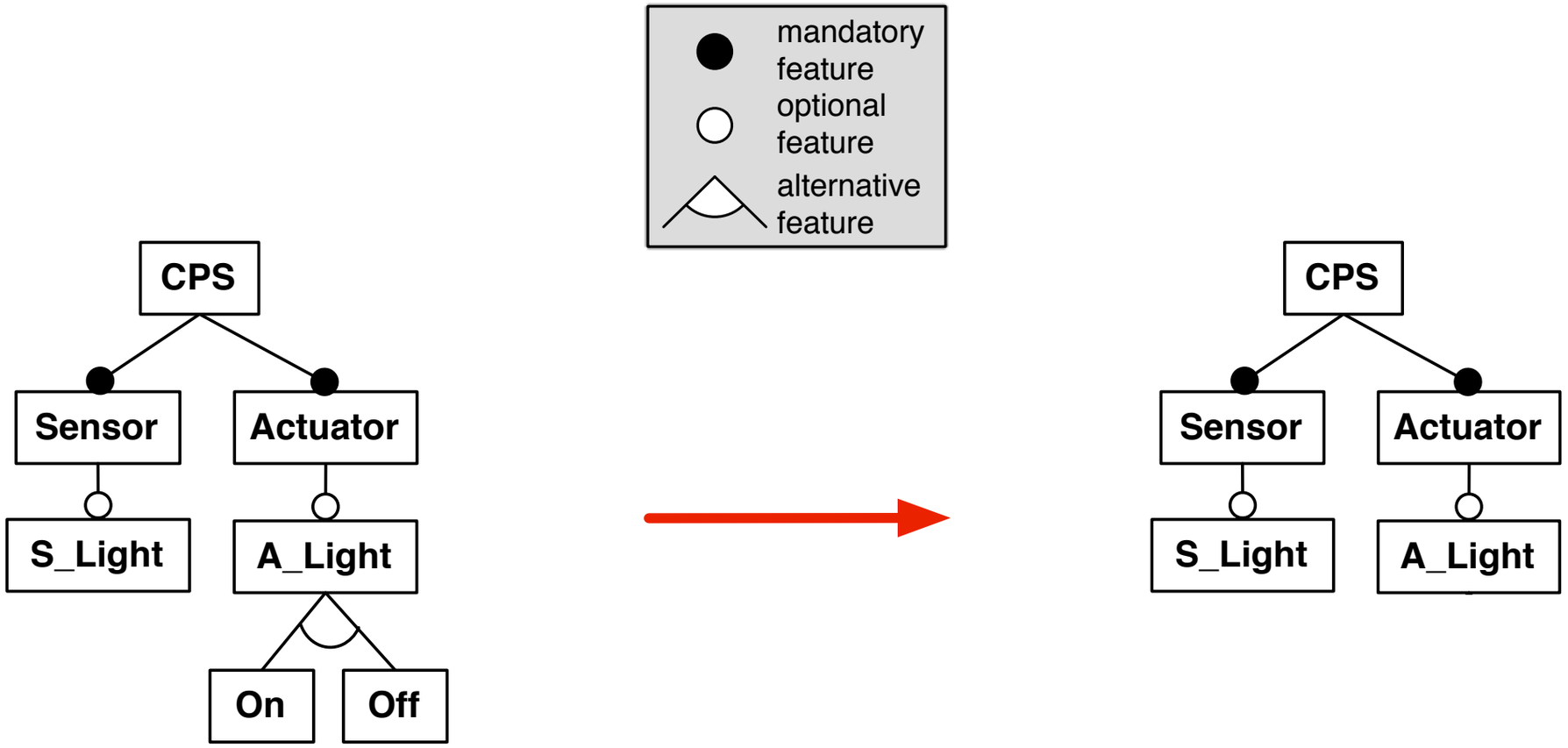


luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)



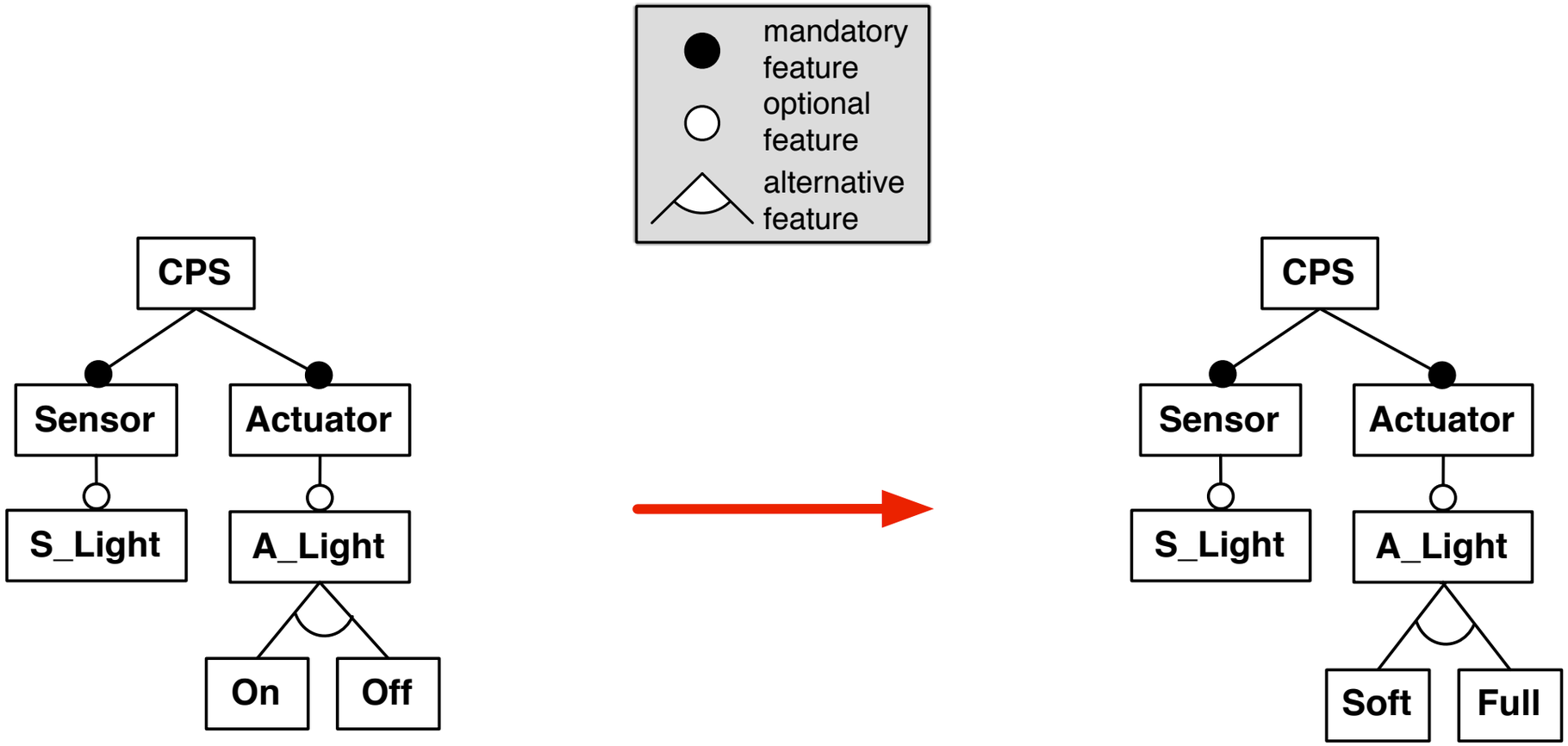
Automated evolution (based on rules, default solver choice)
Manual evolution (user choice)

It depends



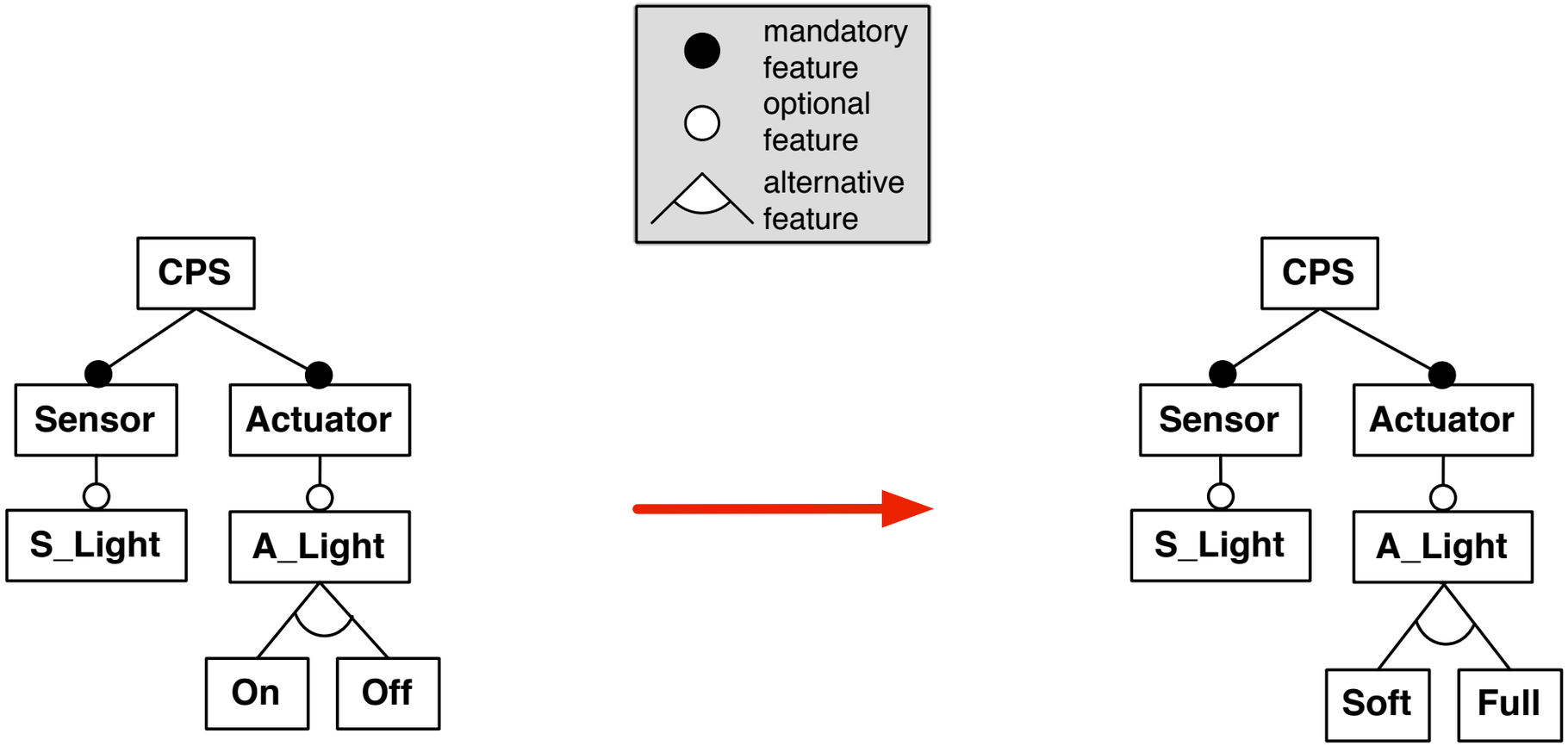
luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

It depends



luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

It depends



luminosity < 40 lumens —> turnLightOn()
(activates the *On* feature)

Manual evolution only

To sum up

The adaptation rules must evolve to maintain the DSPL consistency

To sum up

The adaptation rules must evolve to maintain the DSPL consistency

But it’s not an easy task!

To sum up

The adaptation rules must evolve to maintain the DSPL consistency

But it's not an easy task!

```
/* Adaptation rules for functionalities */

rule BecomeDA : // Becomes a DA
  condition ElectedDA and not LowBatt and not DA
  effect DA

rule StopDA : // Stop being a DA
  condition (LowBatt or not ElectedDA) and DA
  effect not DA

rule BecomeUA : // Become a User Agent
  condition  SrvReq=UA and not UA
  effect UA and not SA

rule BecomeSA : // Become a Service Agent
  condition SrvReq=SA and not SA
  effect not UA and SA
```

Modeling and Validating Dynamic Adaptation, MODELS 2009

To sum up

The adaptation rules must evolve to maintain the DSPL consistency

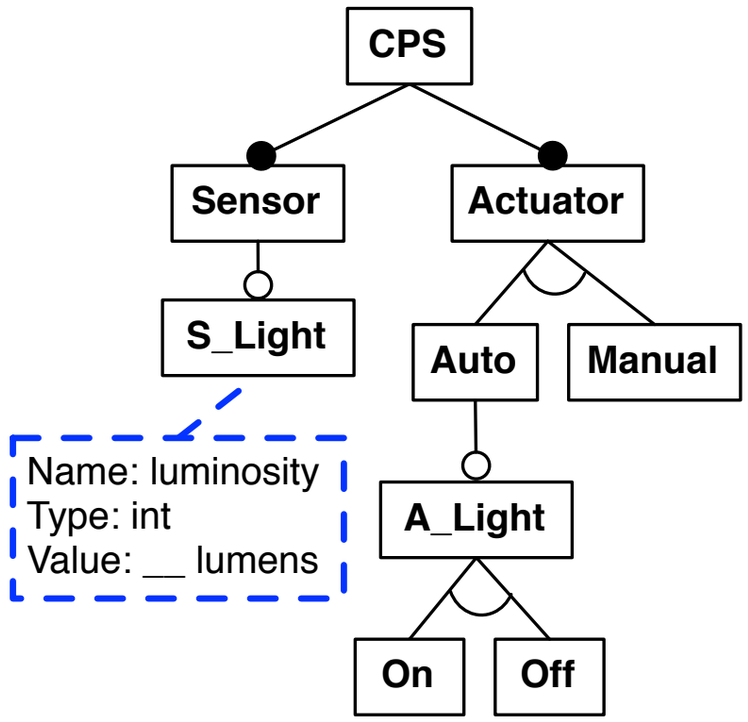
But it's not an easy task!

```
/* Adaptation rules for functionalities */

rule BecomeDA : // Becomes a DA
  condition ElectedDA and not LowBatt and not DA
  effect DA

rule StopDA : // Stop being a DA
  condition (LowBatt or not ElectedDA) and DA
  effect not DA

rule BecomeUA : // Become a User Agent
  condition  SrvReq=UA and not UA
  effect UA and not SA

rule BecomeSA : // Become a Service Agent
  condition SrvReq=SA and not SA
  effect not UA and SA
```

How to detect?

Modeling and Validating Dynamic Adaptation, MODELS 2009

Our idea



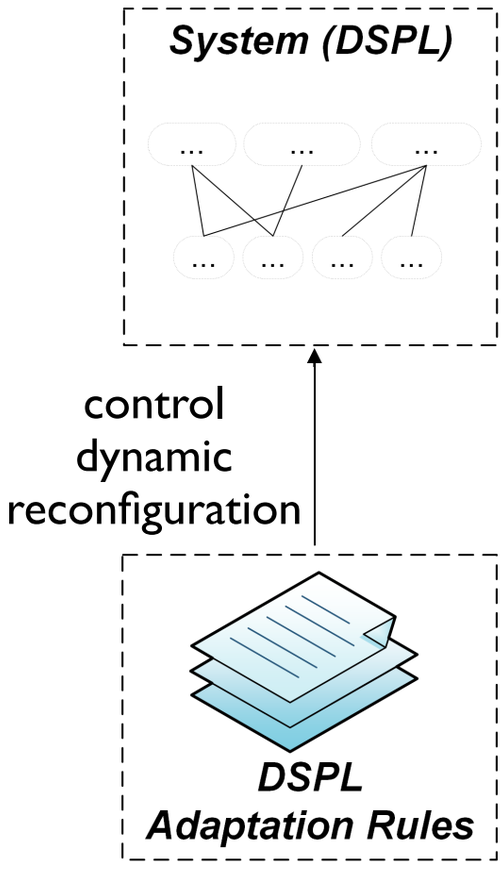
Configuration: A_Light ⟶ S_Light

Adaptation: S_Light.luminosity < 40 ⟶ On

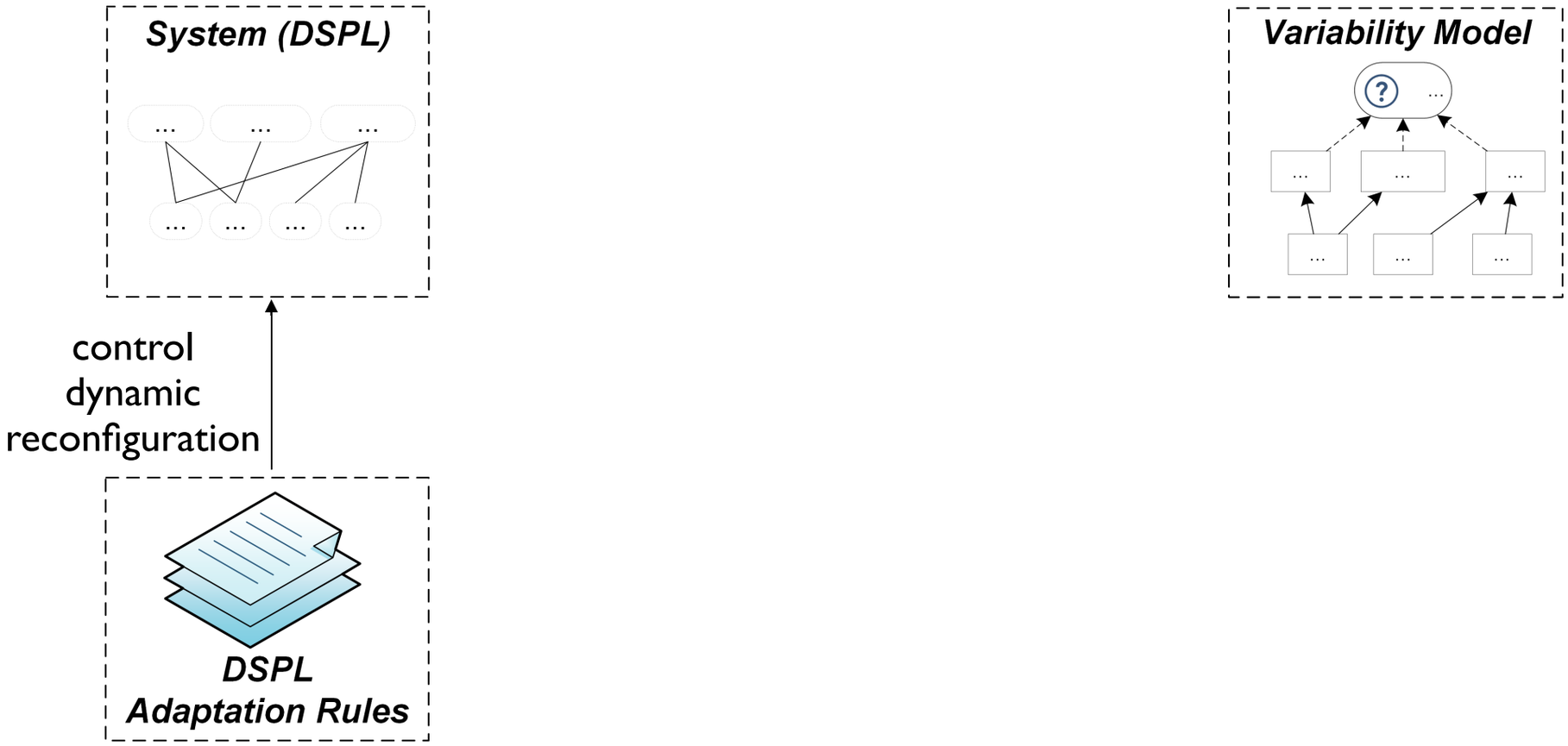
Runtime elements

Current and Future Work

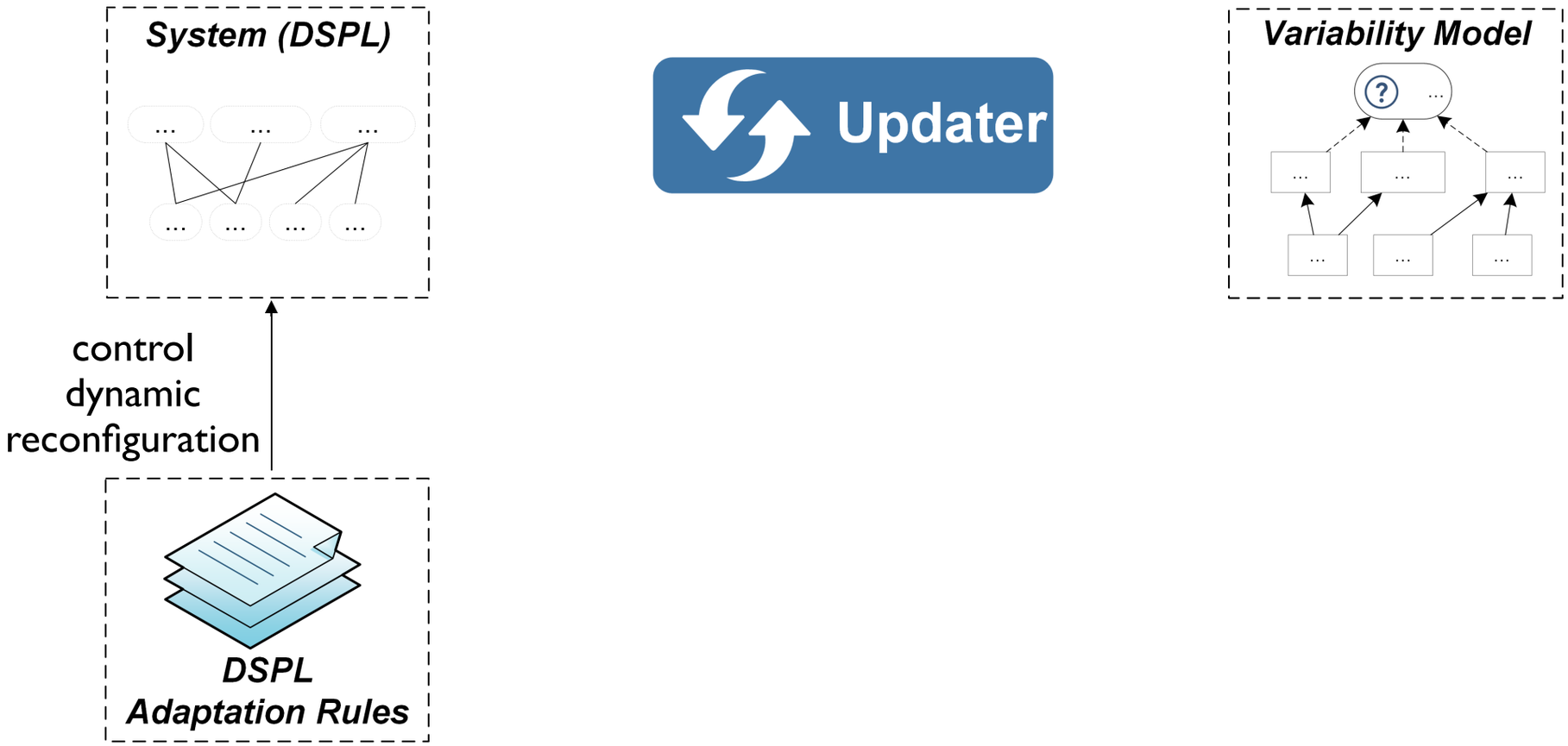
Current and Future Work



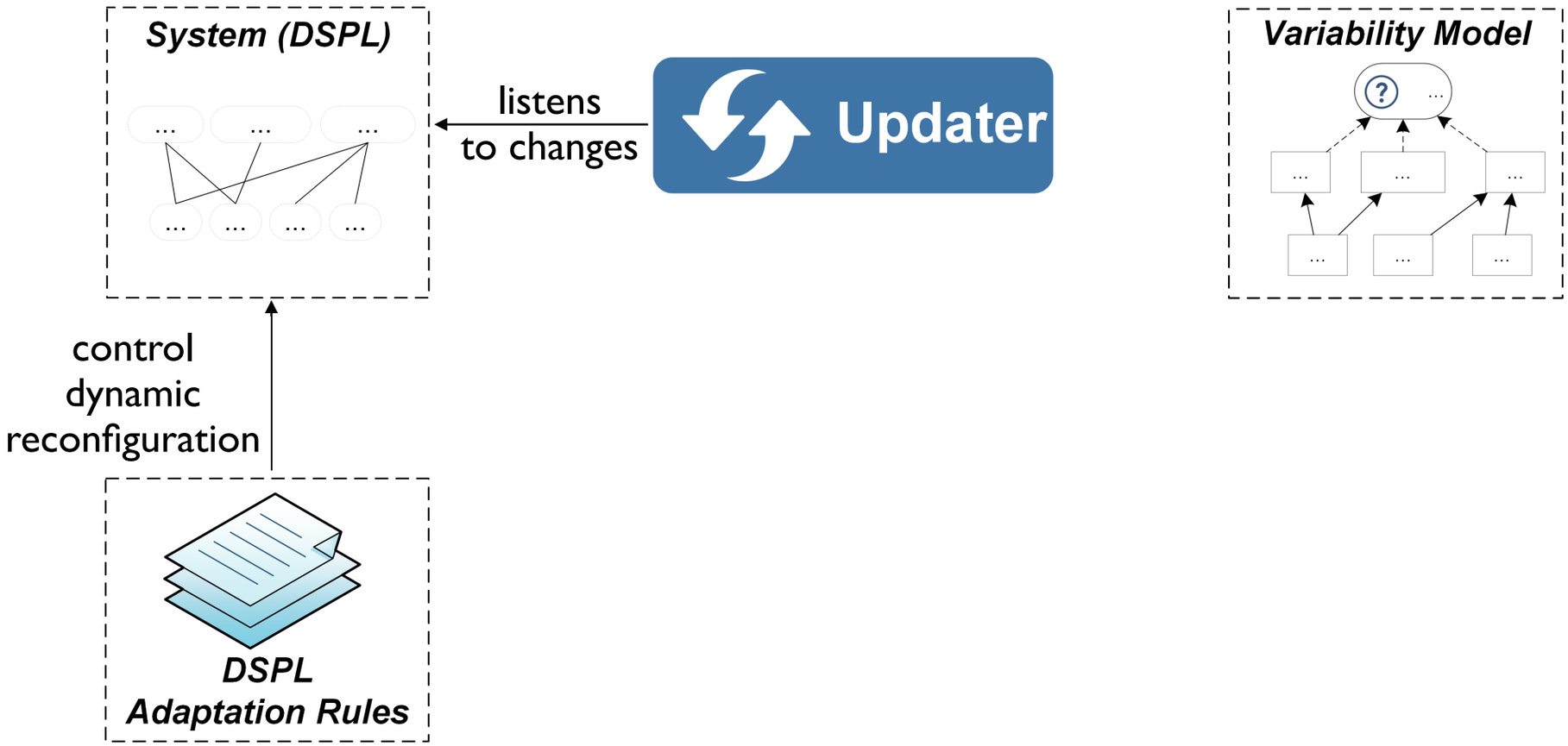
Current and Future Work



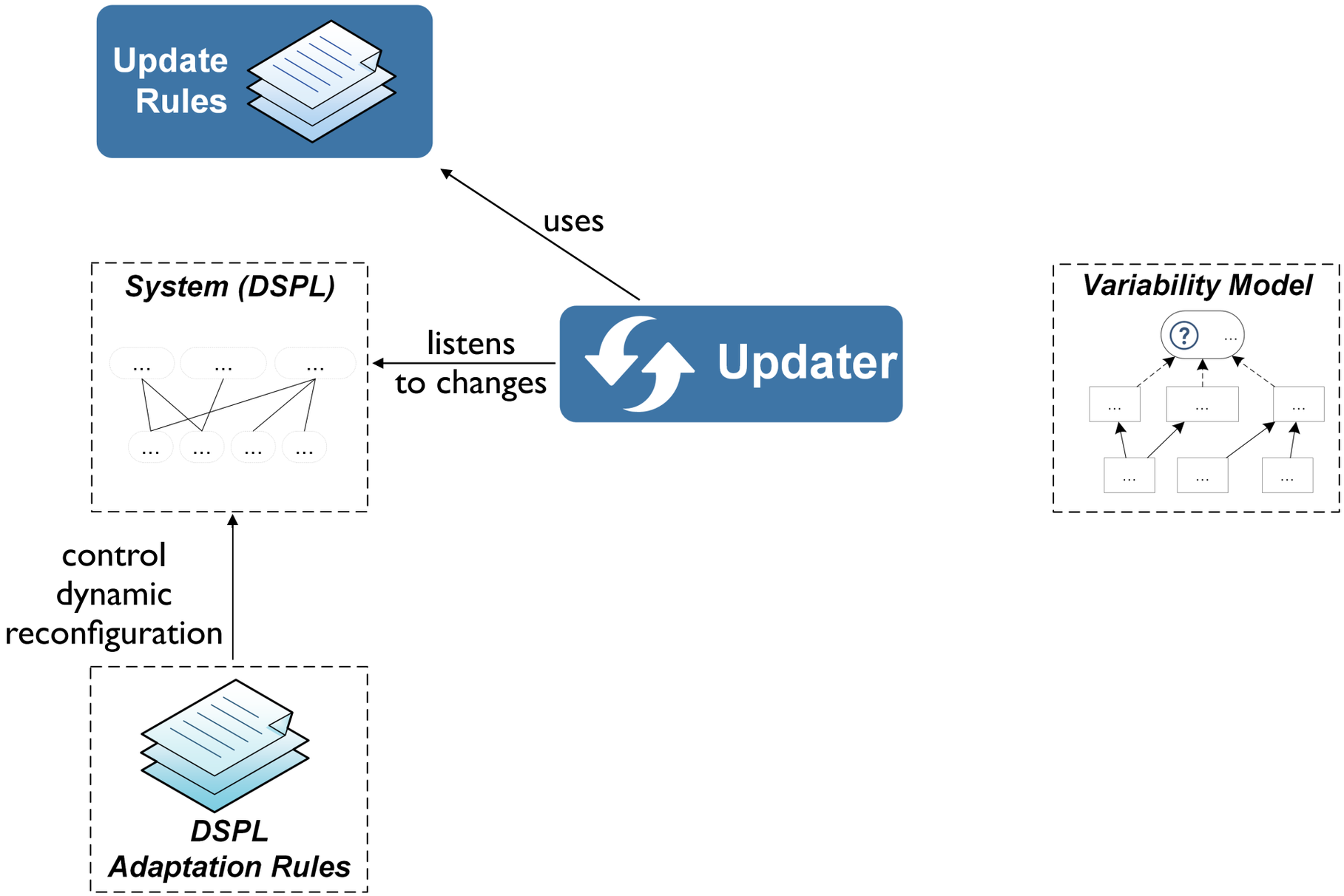
Current and Future Work



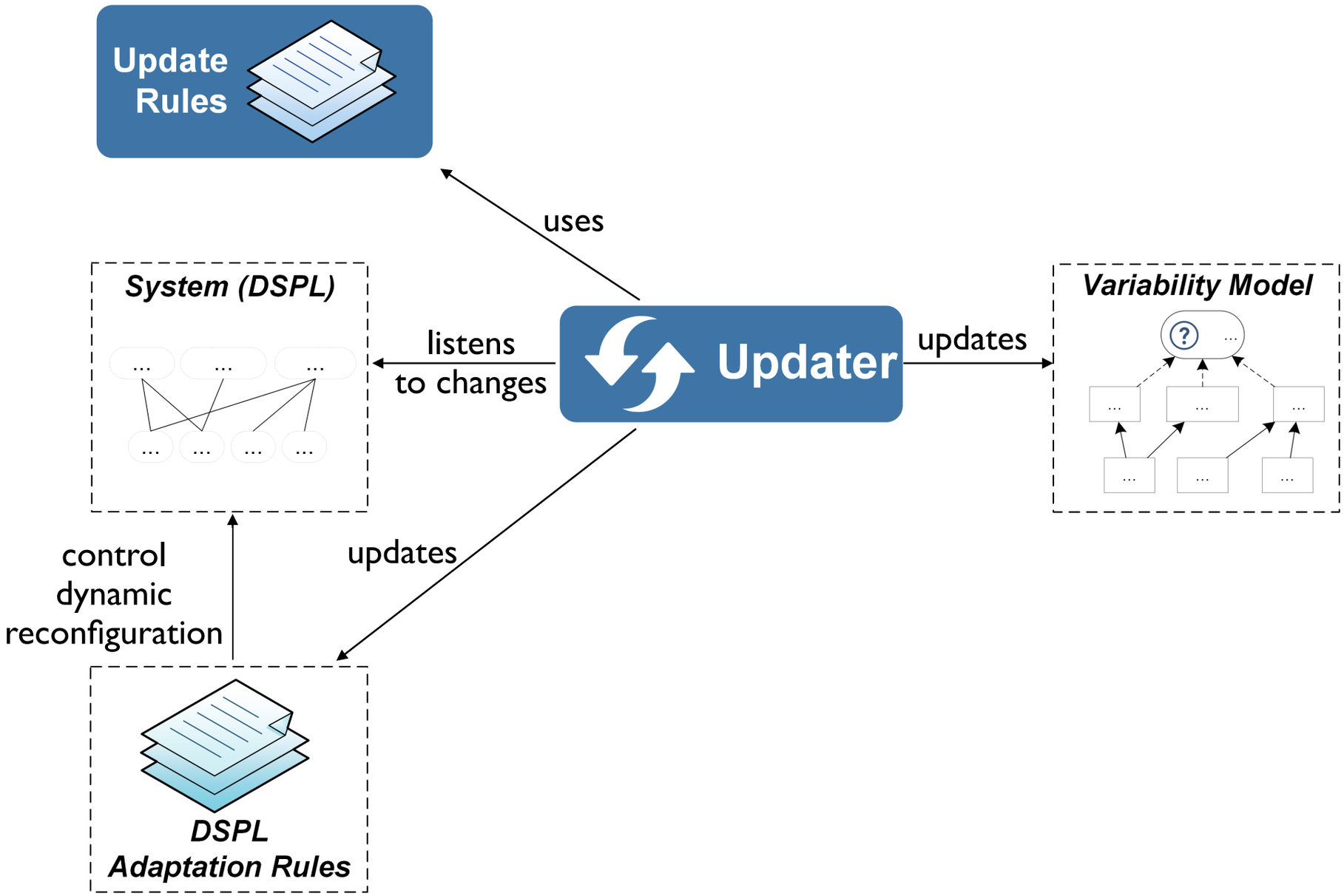
Current and Future Work



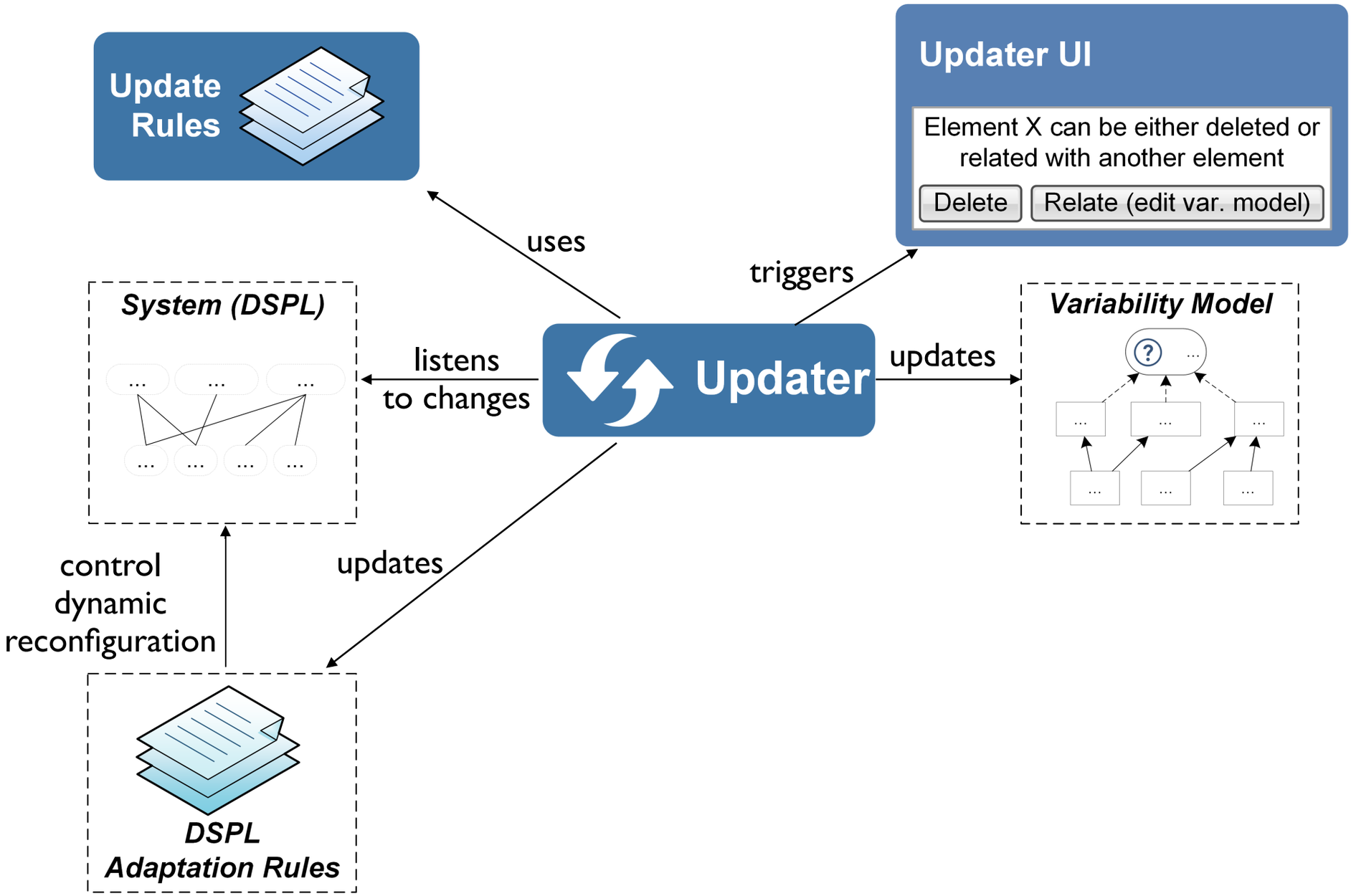
Current and Future Work



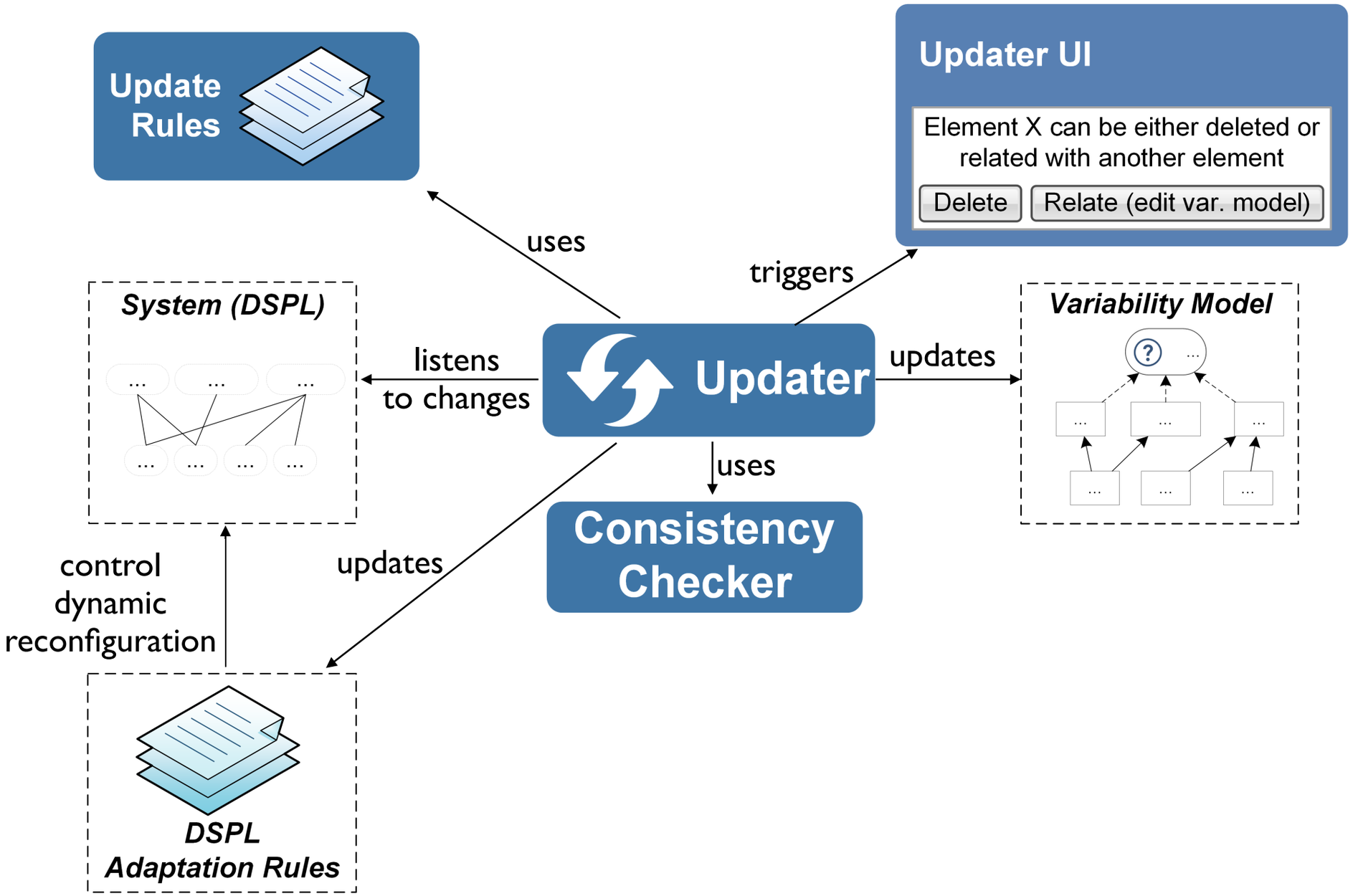
Current and Future Work



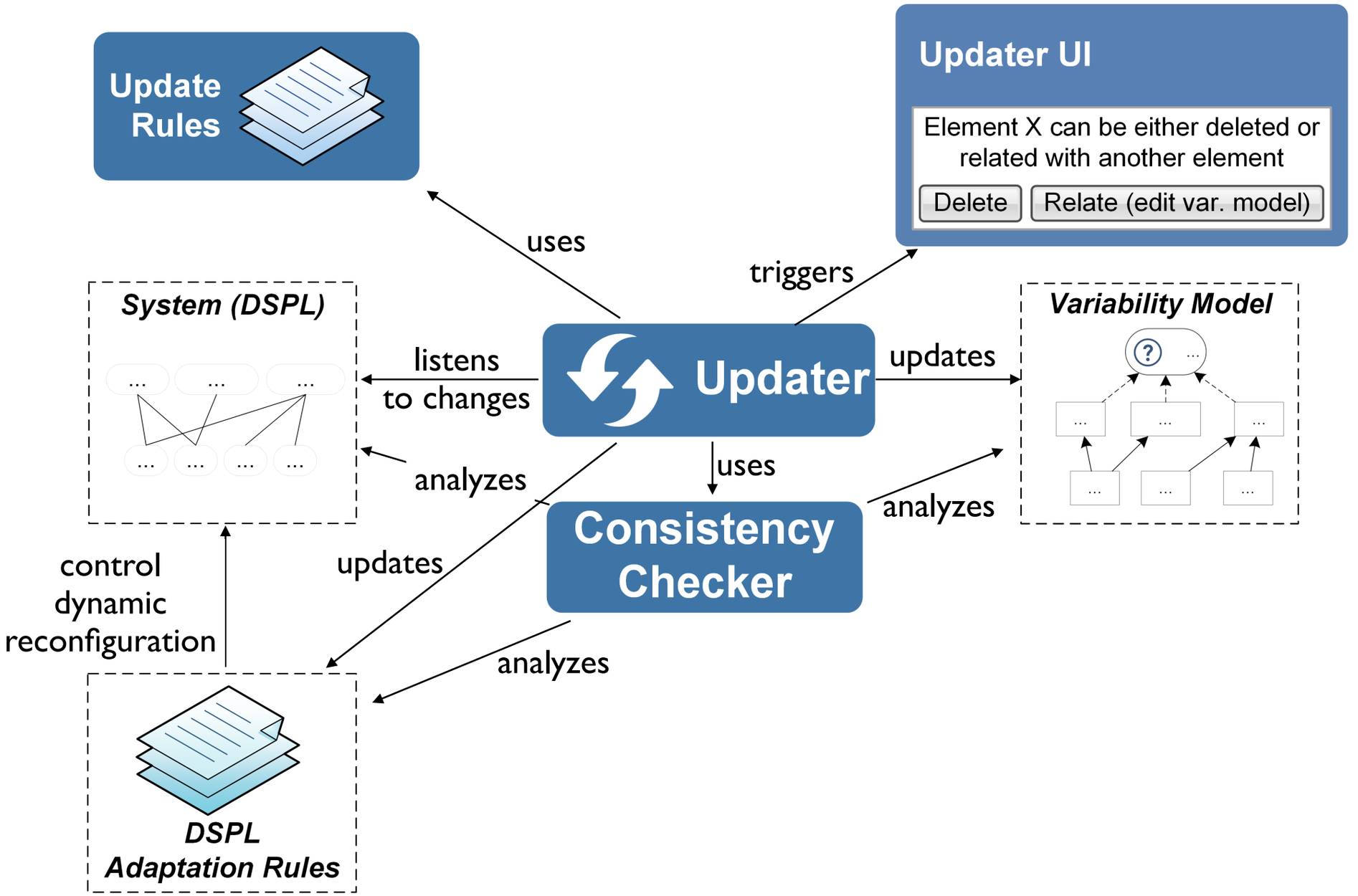
Current and Future Work



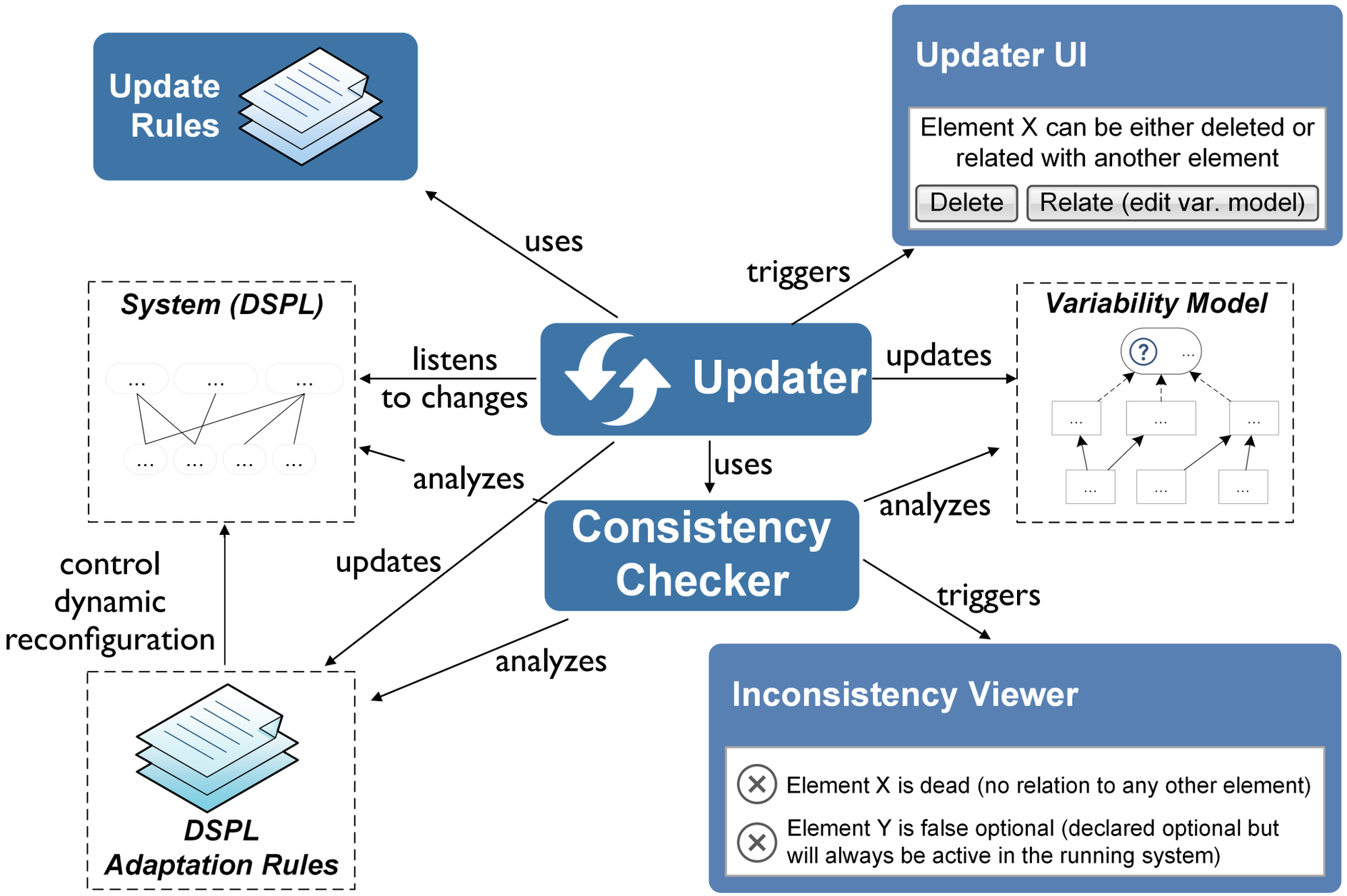
Current and Future Work



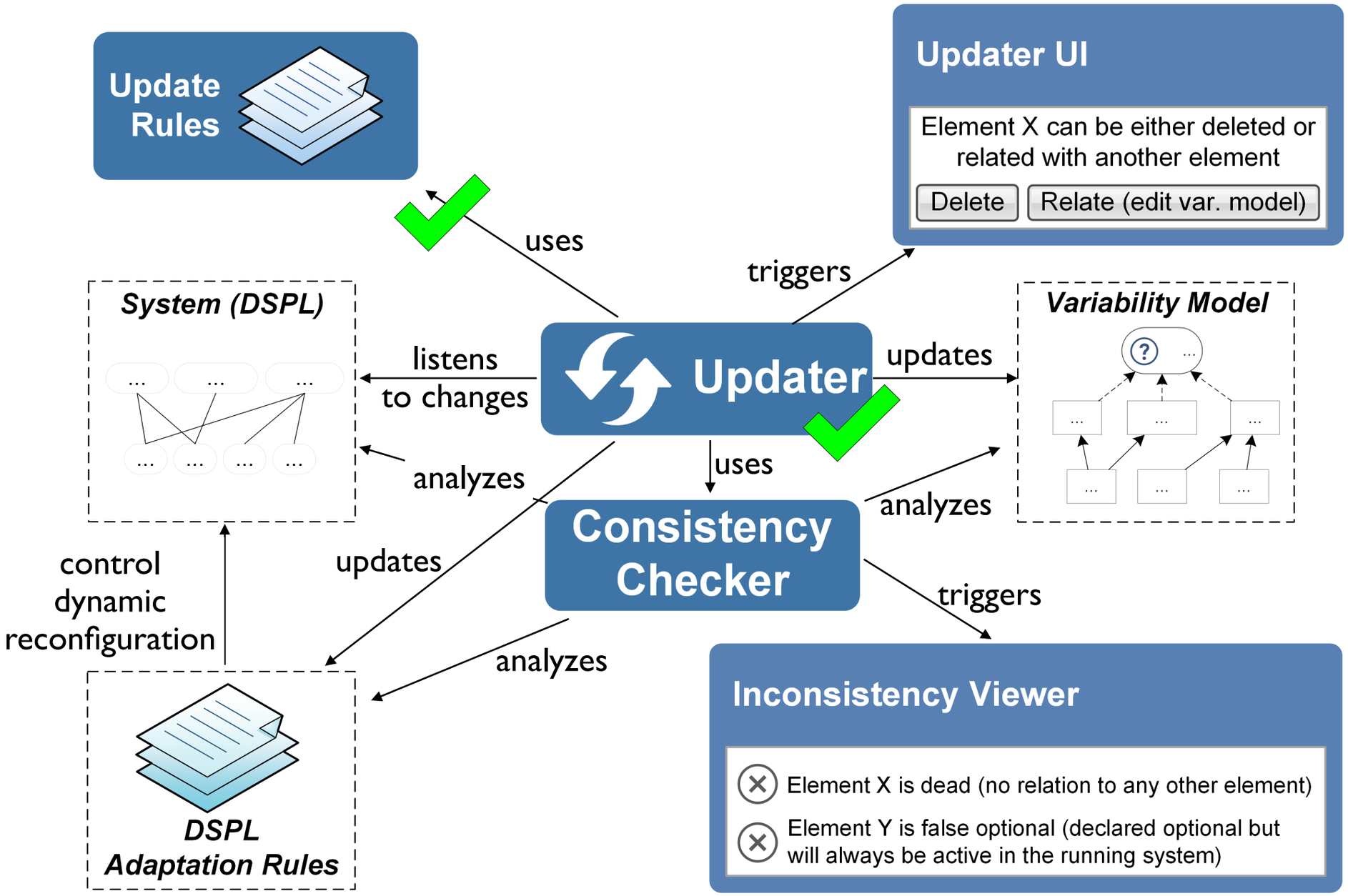
Current and Future Work



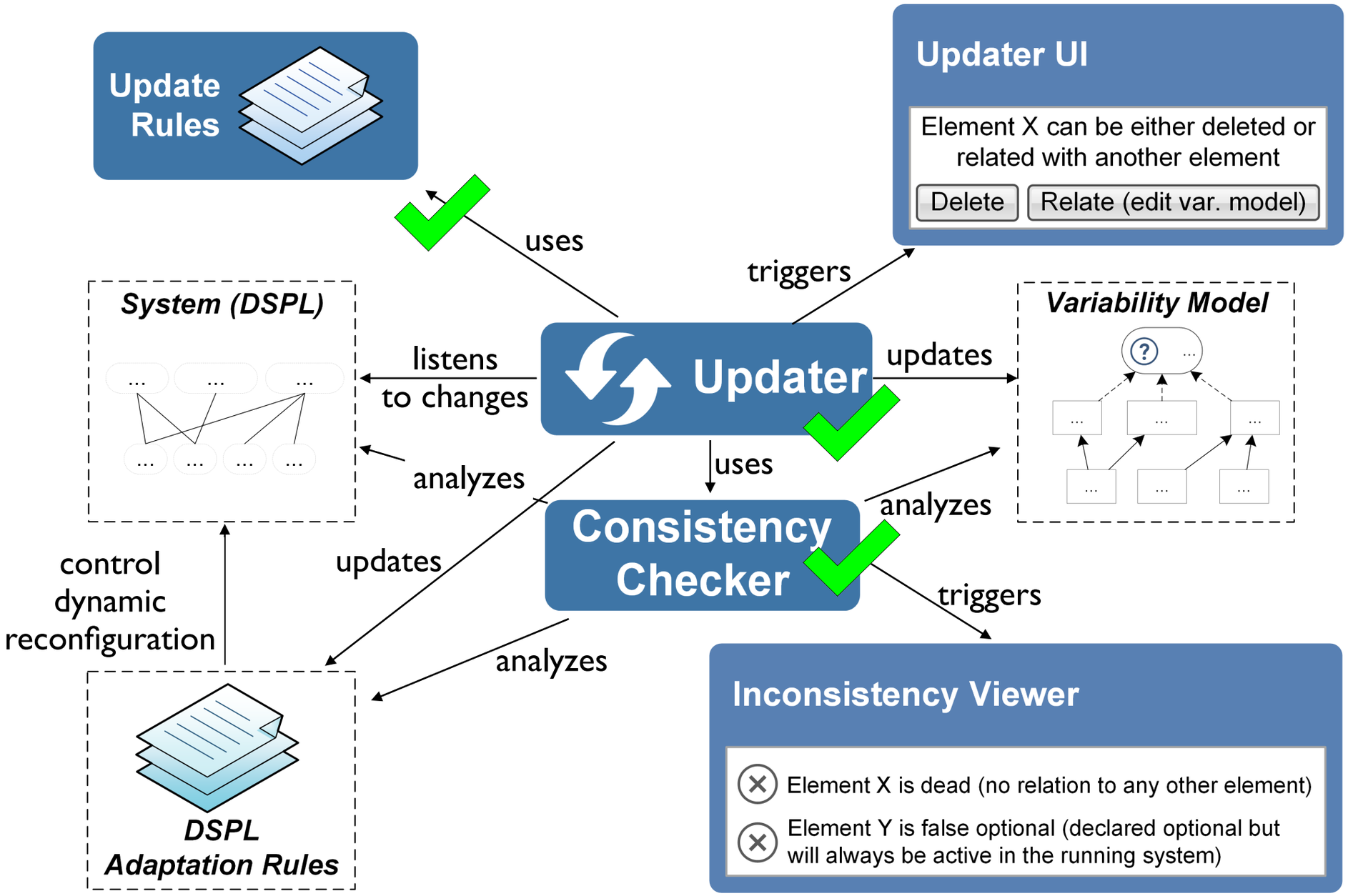
Current and Future Work



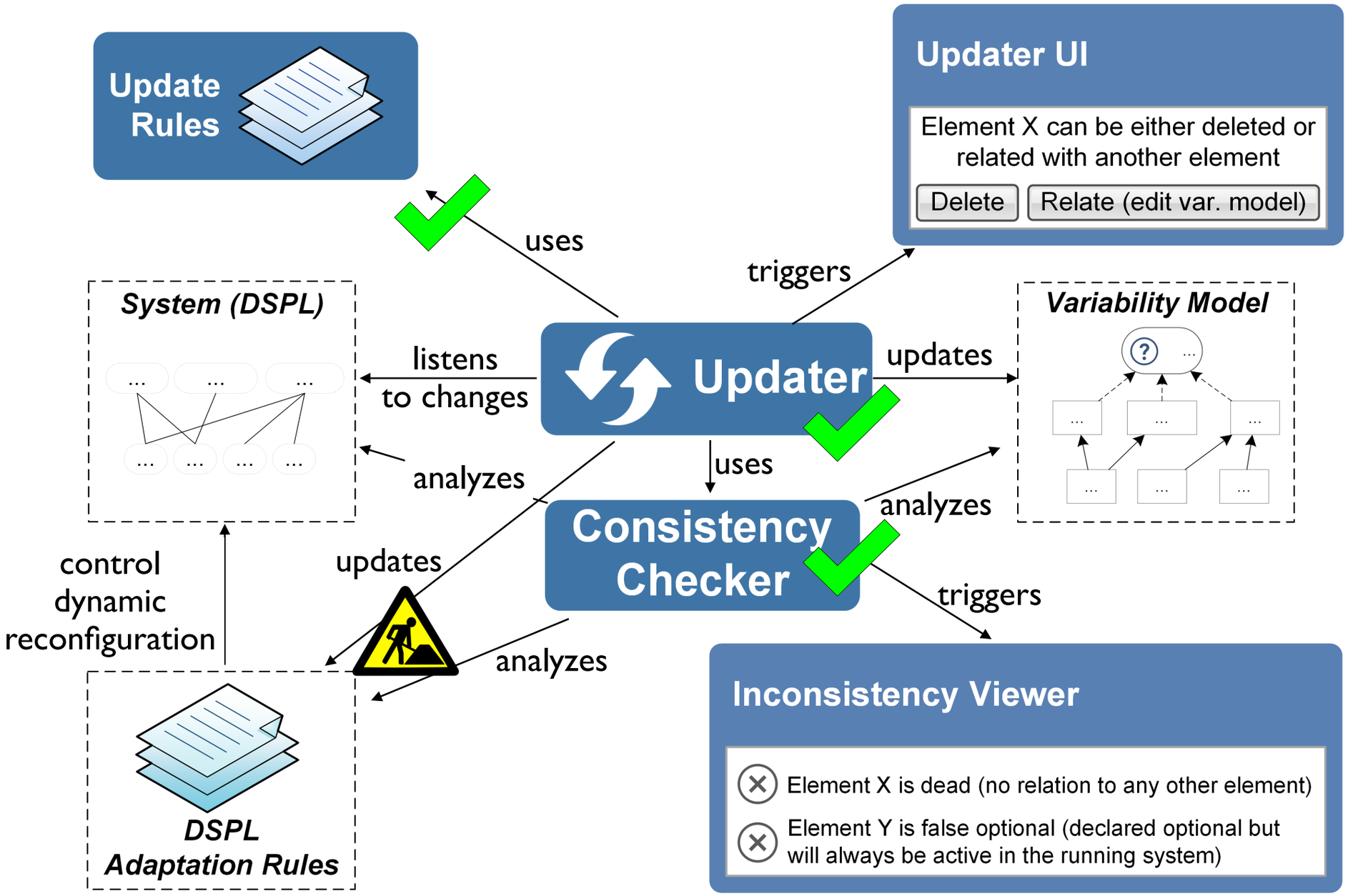
Current and Future Work



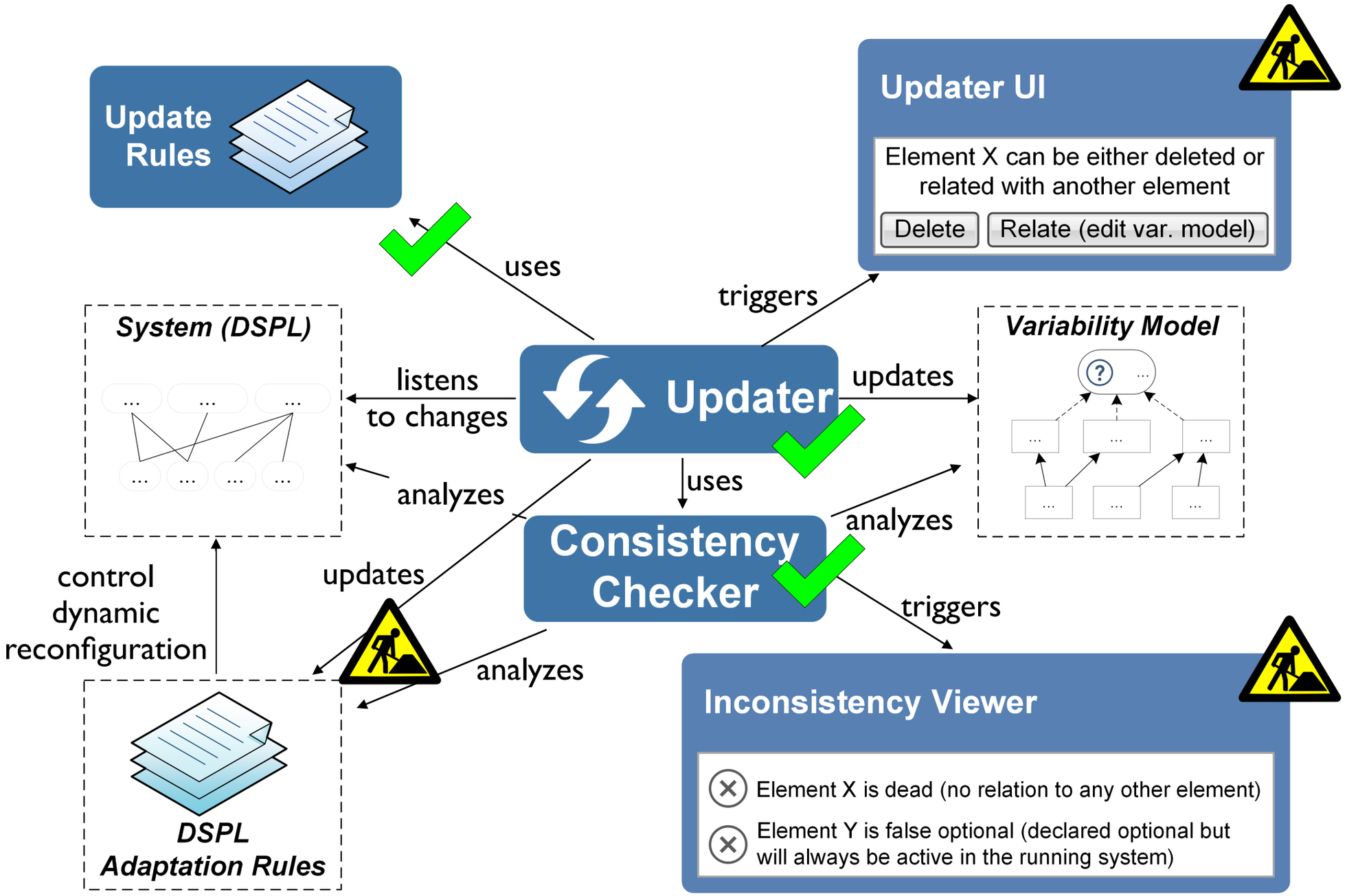
Current and Future Work



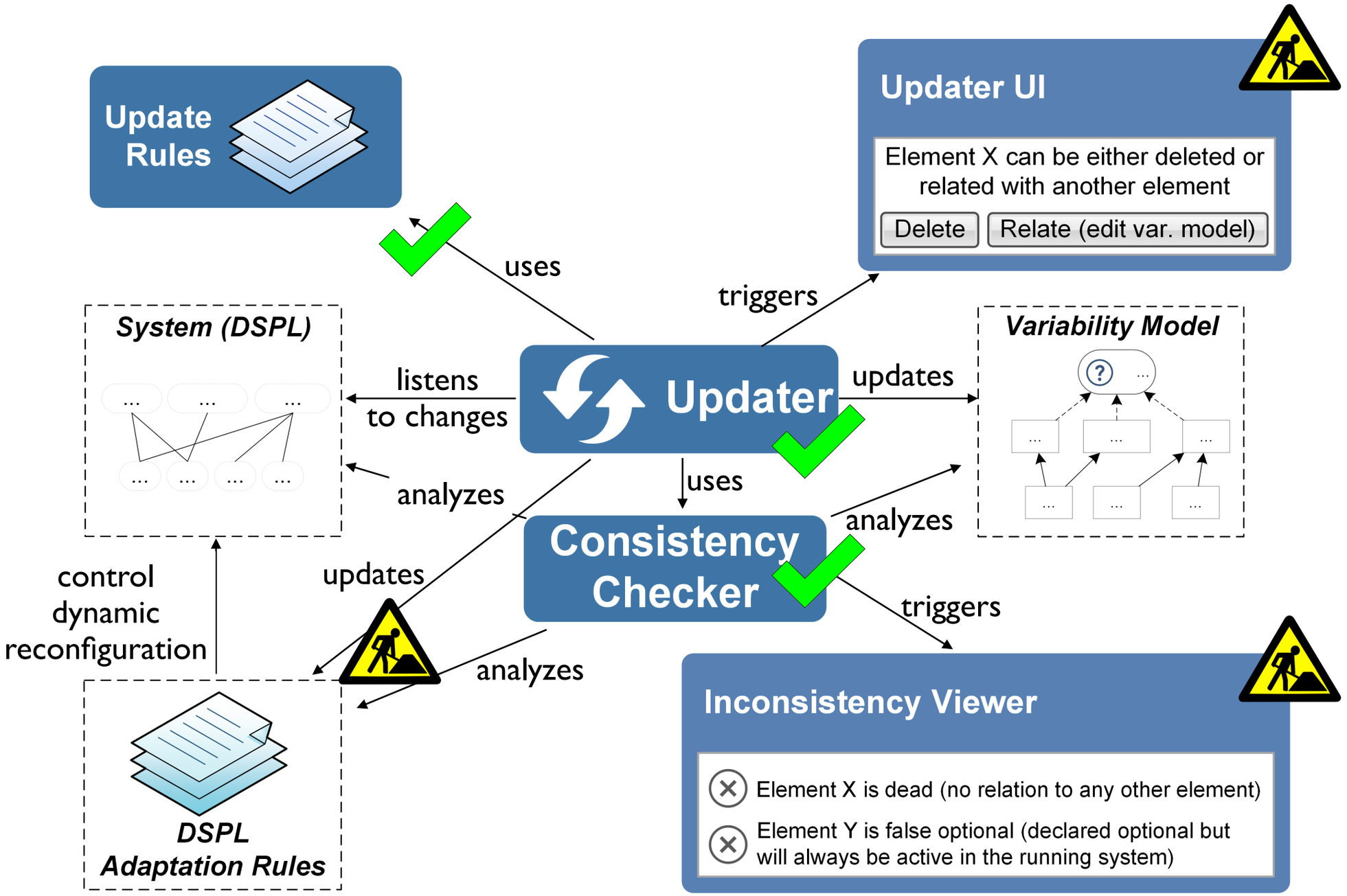
Current and Future Work



Current and Future Work



Current and Future Work



Human in the loop!

Conclusion

- DSPL must evolve while systems are running
  - Systems cannot be stopped
- Need for tools/approaches dealing with
  - Consistency maintenance
  - Uncertainty
  - History
  - Proactive adaptations

Questions / Comments ?

Thank you!