How can domain-specific modeling languages (DSML) help us formalize customer requirements?

EJCP 2022

Florian Galinier

SPILEn

IRIT

SM@RT

Who am I?

Florian Galinier PhD
SPILEn CEO

IRIT SM@RT

SPILEn

- 2020 - now: SPILEn, CEO

  Toulouse Tech Transfer technology transfer and financial support

- 2016 - 2021: PhD Thesis

  Seamless development of complex systems: a multirequirements approach

How can domain-specific modeling languages (DSML) help us **formalize customer requirements**?

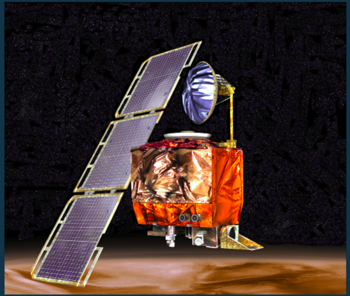
Context: consequences of bad specification

Patriot missile battery

Error introduce by a truncation of radar timestamps.

Therac 25

Several problems, including lack of specification and traceability with previous models.



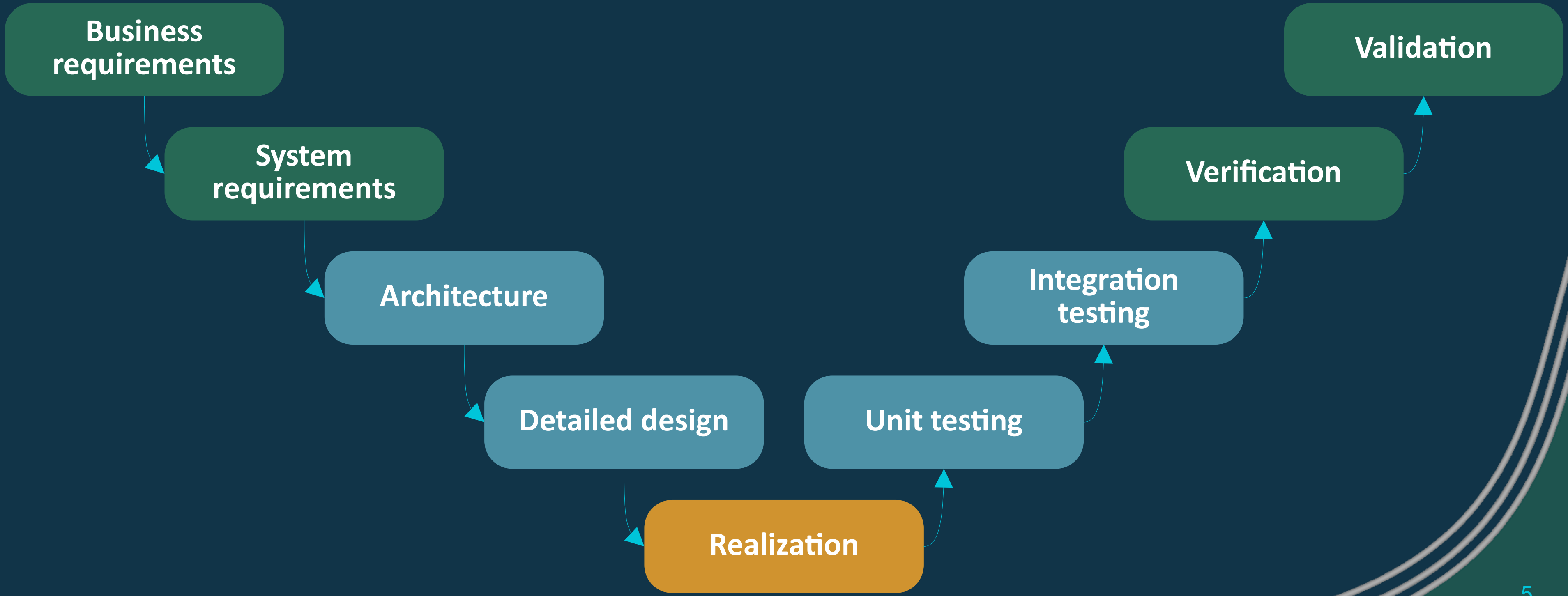
Mars Climate Orbiter

Different units of measurement.

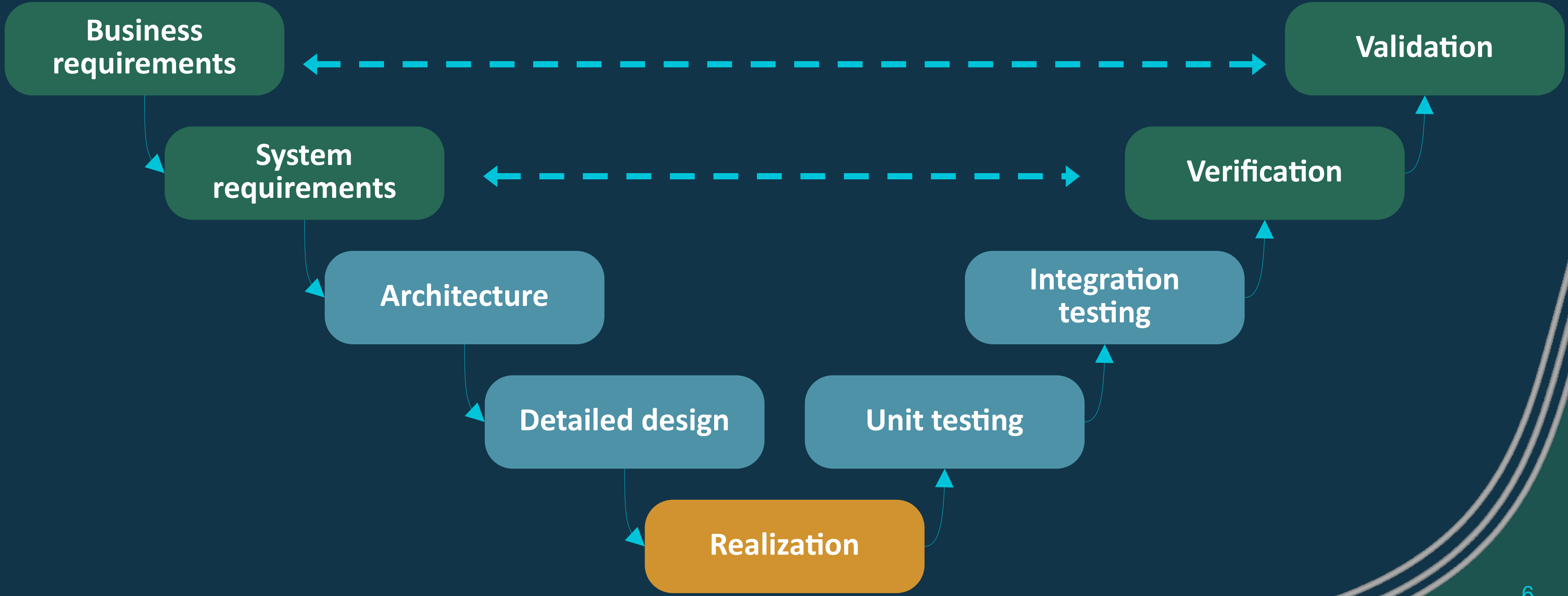
Ariane 5 flight 501

Arithmetic overflow.

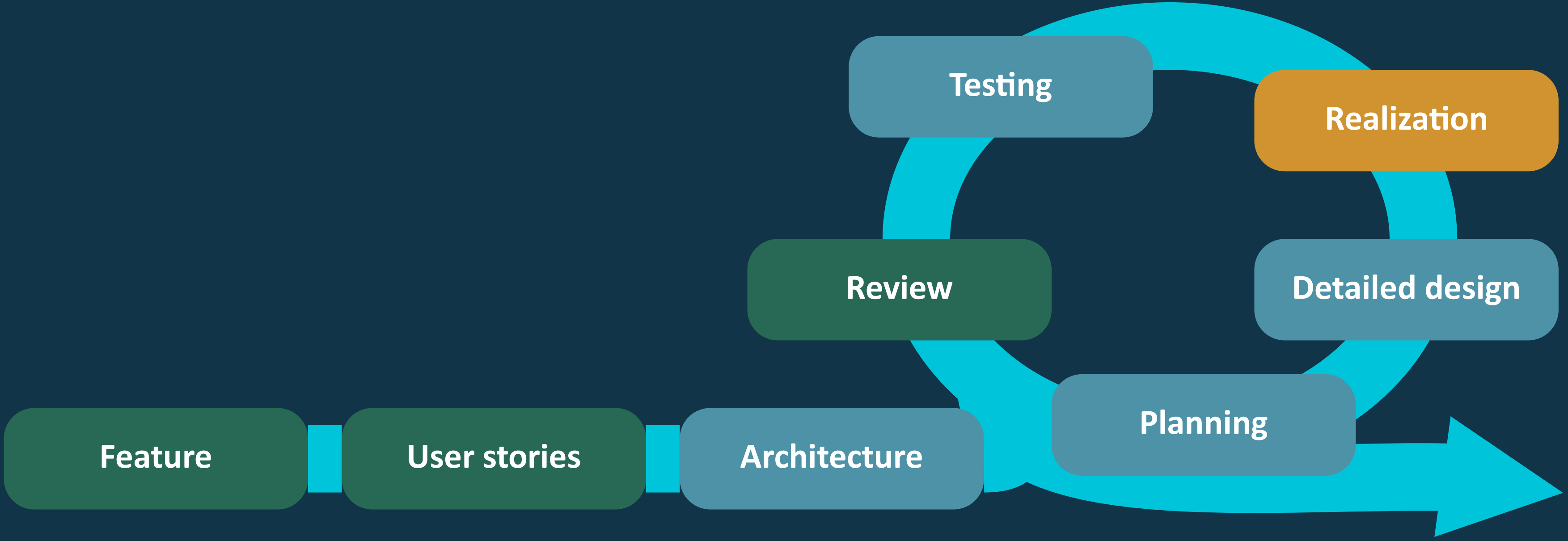
Requirements: basic building bricks of the system



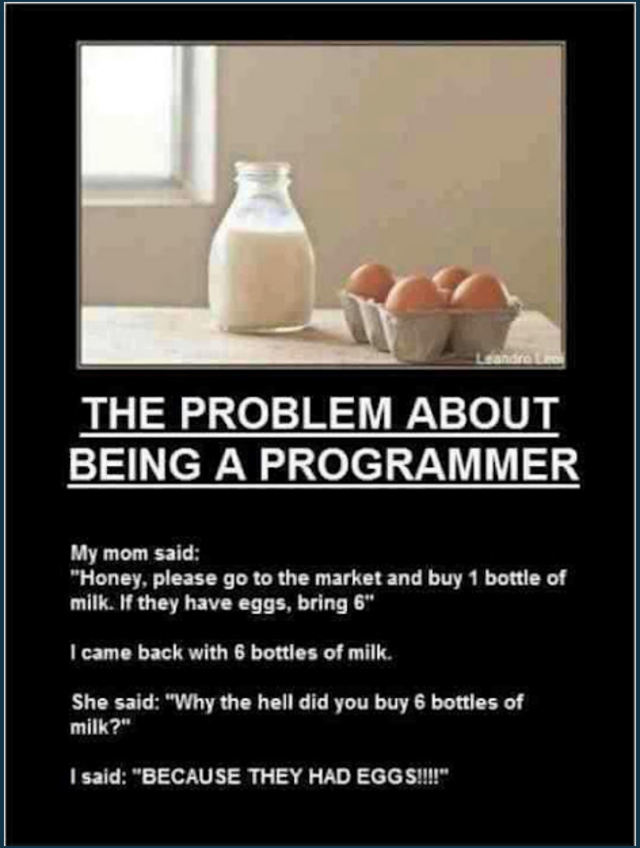
Requirements: basic building bricks of the system



Requirements: basic building bricks of the system



What is a good requirement?



THE PROBLEM ABOUT BEING A PROGRAMMER

My mom said:
"Honey, please go to the market and buy 1 bottle of milk. If they have eggs, bring 6"

I came back with 6 bottles of milk.

She said: "Why the hell did you buy 6 bottles of milk?"

I said: "BECAUSE THEY HAD EGGS!!!!"

What is a good requirement?

A good requirement is:

- necessary
- complete
- unambiguous
- singular
- feasible
- correct
- verifiable

Formal representation of requirements

A requirement *R* is:

- a set of properties $\mathrm{P}_R = \{P_1, \ldots, P_m\}$ that

the system shall meet

- in a context $\mathrm{C}_R = \{C_1, \ldots, C_n\}$.

$$sat(R) \equiv hold(\mathrm{C}_R) \rightarrow hold(\mathrm{P}_R)$$

Formal representation of requirements: example

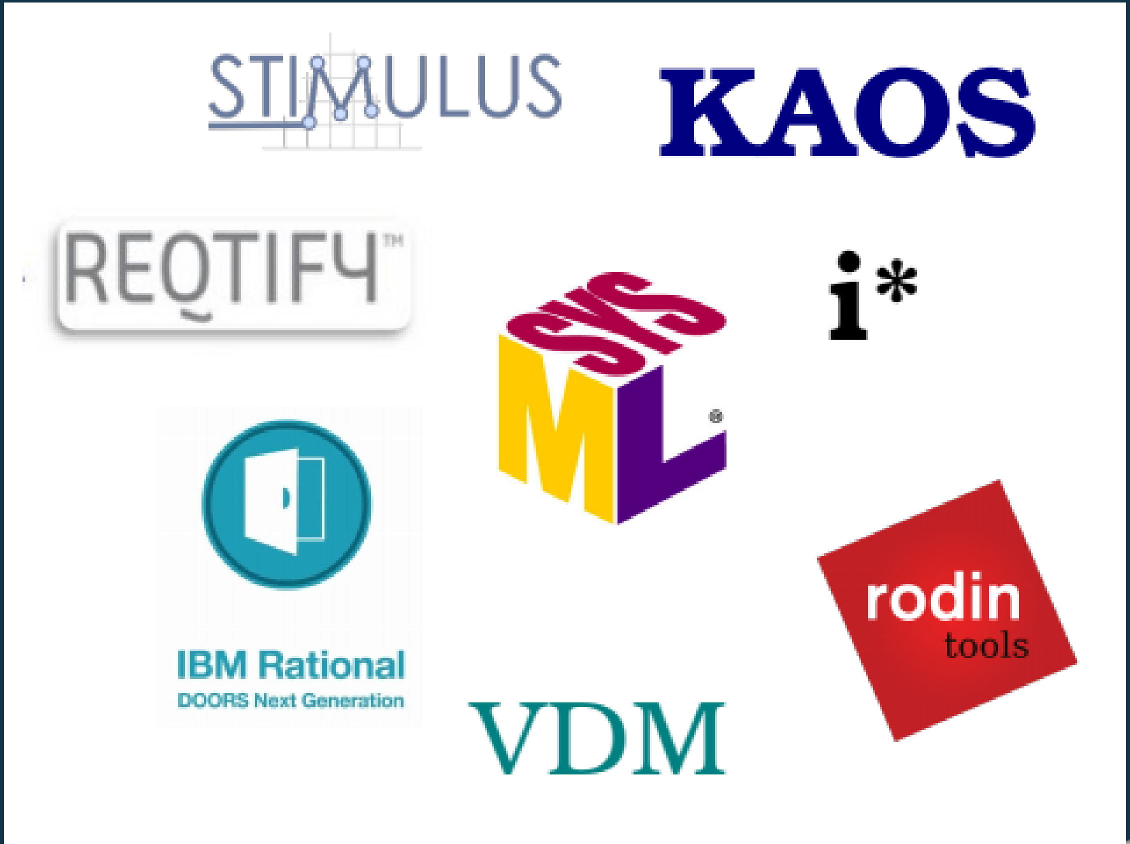
R1 - When package status is assigned and destination is not null then status shall be mobilized.

C_{R1} = { package status = assigned ; destination ≠ null }

P_{R1} = { status = mobilized }

sat(R1) ≡ (package status = assigned ∧ destination ≠ null)
→ status = mobilized

How to express formal requirements?



How to express formal requirements?

- Express system and requirements in a same formalism (Single Model Principle[1,2])
- Use verification and validation tool (e.g., formal method[3], autoproof[4])

[1] **Richard F. Paige and Jonathan S. Ostroff. “The Single Model Principle”.** In: Proceedings of the Fifth IEEE International Symposiumon Requirements Engineering. RE ’01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 292–311.

[2] **Bertrand Meyer. “Multirequirements”.** In: Modelling and Quality in Requirements Engineering (Martin Glinz Festscrhift) (2013).Ed. by Norbert Seyff and Anne Koziolek.

[3] **Amel Mammar and Régine Laleau. “On the Use of Domain and System Knowledge Modeling in Goal-Based Event-B Specifications”**. en. In: Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques. Ed. by Tiziana Margaria and Bernhard Steffen. LNCS 9952. Springer Int. Publishing, Oct. 2016, pp. 325–339.

[4] **Alexandr Naumchev, Bertrand Meyer, and Víctor Rivera. “Unifying Requirements and Code: an Example”.** In: CoRRabs/1602.05395(2016).

Expressing formal requirements

```
Variable package_status: Set.
Variable assigned: package_status.
Variable unassigned: package_status.
Axiom package_status_value: forall x : package_status, x = assigned \/ x = unassigned.

Variable drone_status: Set.
Variable mobilized: drone_status.
Variable standby: drone_status.
Axiom drone_status_value: forall x : drone_status, x = mobilized \/ x = standby.

Variable package : package_status.
Variable destination : nat * nat.
Variable drone : drone_status.

Theorem R1 : package = assigned /\ ~ destination = (0 , 0) -> drone = mobilized.
```

Expressing the requirement as an artifact of code:

- contracts as proof obligations
- documentation as an understandable expression of the requirement

The SIRCOD approach: principle

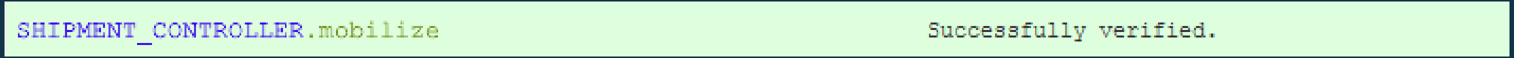
```
invariant
  requirement_1:
    ((package_status = assigned) and (destination /= Void))
       implies
   (drone_status = mobilized)
```

The SIRCOD approach: principle

```
feature
  requirement_1
  note
    src: "{SHIPMENT_REQUIREMENTS}.requirement_1_doc"
  require
    package_assigned: (package_status = assigned)
    has_destination:  (destination /= Void)
  deferred
  ensure
    check_drone_status: (drone_status = mobilized)
  end
```

Refinement: inheriting requirements features

```
class SHIPMENT_CONTROLLER
  inherit
    SHIPMENT_FORMAL_REQUIREMENTS
  rename
    requirement_1 as mobilize
  end

  feature
    mobilize
    do
      drone_status := mobilized
    end
end
```



```
SHIPMENT_CONTROLLER.mobilize                    Successfully verified.
```

One project; several stakeholders

« The drone shall avoid obstacles »

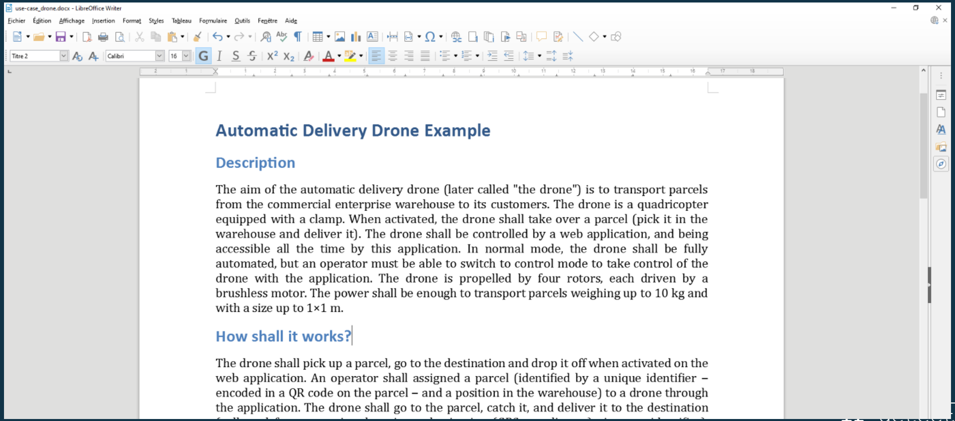
One project; several stakeholders

« The drone shall avoid obstacles »

```
avoid_obstacle
do
        detected_object := detect_object
        if detected_object /= Void then
                obstacle_detected :=
                  detected_object.distance_to (Current) <
                  2 * stopping_distance
                if obstacle_detected then
                        path := compute_new_path
                end
        end
ensure
        change_path_if_obstacle:
                obstacle_detected implies path /= old path
end
```

One project; several stakeholders

« The drone shall avoid obstacles »



use-case_drone.docx - LibreOffice Writer

Automatic Delivery Drone Example

Description

The aim of the automatic delivery drone (later called "the drone") is to transport parcels from the commercial enterprise warehouse to its customers. The drone is a quadricopter equipped with a clamp. When activated, the drone shall take over a parcel (pick it in the warehouse and deliver it). The drone shall be controlled by a web application, and being accessible all the time by this application. In normal mode, the drone shall be fully automated, but an operator must be able to switch to control mode to take control of the drone with the application. The drone is propelled by four rotors, each driven by a brushless motor. The power shall be enough to transport parcels weighing up to 10 kg and with a size up to 1×1 m.

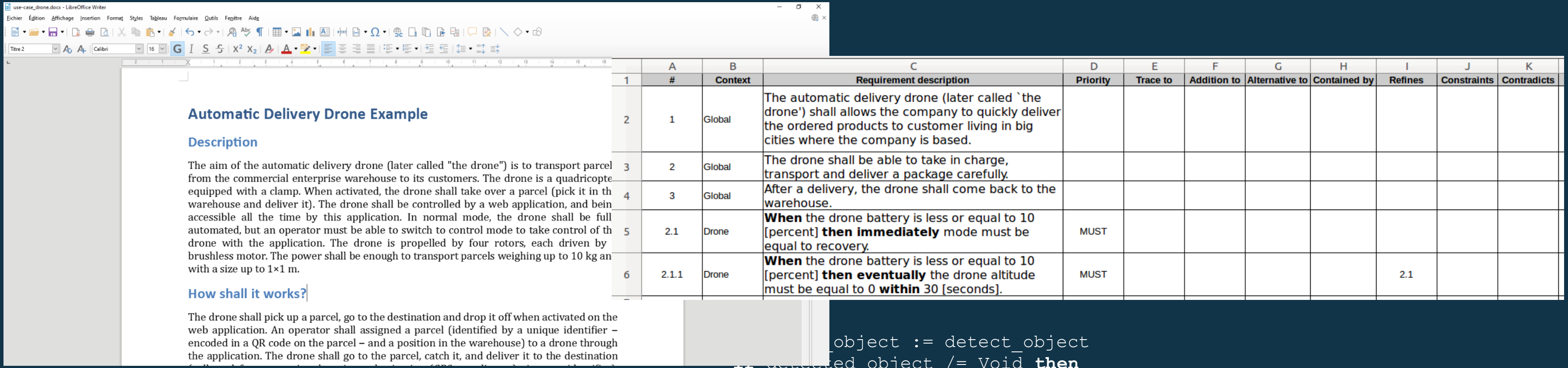
How shall it works?

The drone shall pick up a parcel, go to the destination and drop it off when activated on the web application. An operator shall assigned a parcel (identified by a unique identifier – encoded in a QR code on the parcel – and a position in the warehouse) to a drone through the application. The drone shall go to the parcel, catch it, and deliver it to the destination

```
                object := detect_object
        if detected_object /= Void then
               obstacle_detected :=
                 detected_object.distance_to (Current) <
                 2 * stopping_distance
               if obstacle_detected then
                      path := compute_new_path
               end
        end
ensure
        change_path_if_obstacle:
               obstacle_detected implies path /= old path
end
```

One project; several stakeholders

« The drone shall avoid obstacles »


use-case_drone.docx - LibreOffice Writer
Fichier Édition Affichage Insertion Format Styles Tableau Formulaire Outils Fenêtre Aide
Titre 2
Calibri
16

Automatic Delivery Drone Example

Description

The aim of the automatic delivery drone (later called "the drone") is to transport parcel
from the commercial enterprise warehouse to its customers. The drone is a quadricopte
equipped with a clamp. When activated, the drone shall take over a parcel (pick it in th
warehouse and deliver it). The drone shall be controlled by a web application, and bein
accessible all the time by this application. In normal mode, the drone shall be full
automated, but an operator must be able to switch to control mode to take control of th
drone with the application. The drone is propelled by four rotors, each driven by
brushless motor. The power shall be enough to transport parcels weighing up to 10 kg an
with a size up to 1×1 m.

How shall it works?

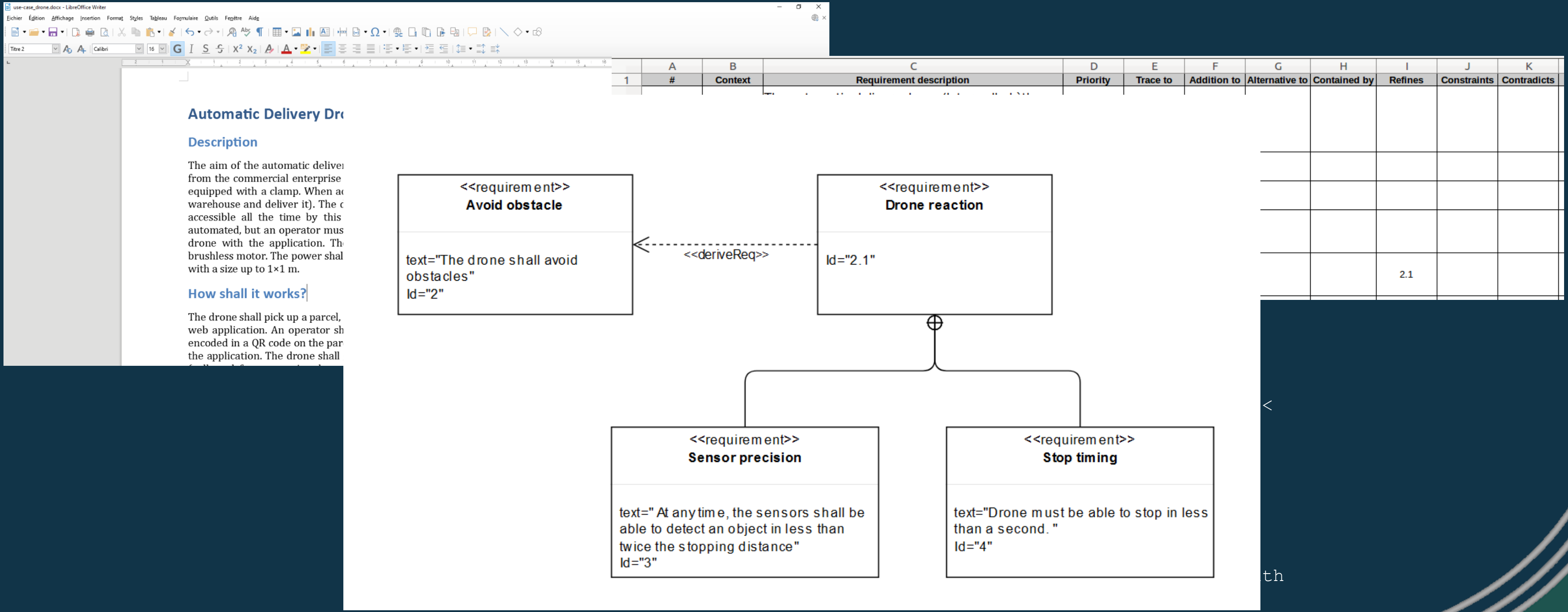
The drone shall pick up a parcel, go to the destination and drop it off when activated on the web application. An operator shall assigned a parcel (identified by a unique identifier – encoded in a QR code on the parcel – and a position in the warehouse) to a drone through the application. The drone shall go to the parcel, catch it, and deliver it to the destination

	A	B	C	D	E	F	G	H	I	J	K
1	#	Context	Requirement description	Priority	Trace to	Addition to	Alternative to	Contained by	Refines	Constraints	Contradicts
2	1	Global	The automatic delivery drone (later called `the drone') shall allows the company to quickly deliver the ordered products to customer living in big cities where the company is based.								
3	2	Global	The drone shall be able to take in charge, transport and deliver a package carefully.								
4	3	Global	After a delivery, the drone shall come back to the warehouse.								
5	2.1	Drone	**When** the drone battery is less or equal to 10 [percent] **then immediately** mode must be equal to recovery.	MUST							
6	2.1.1	Drone	**When** the drone battery is less or equal to 10 [percent] **then eventually** the drone altitude must be equal to 0 **within** 30 [seconds].	MUST					2.1		

```
           object := detect_object
     if detected_object /= Void then
          obstacle_detected :=
           detected_object.distance_to (Current) <
           2 * stopping_distance
          if obstacle_detected then
               path := compute_new_path
          end
     end
ensure
     change_path_if_obstacle:
          obstacle_detected implies path /= old path
end
```

One project; several stakeholders

« The drone shall avoid obstacles »



use-case_drone.docx - LibreOffice Writer
Fichier Édition Affichage Insertion Format Styles Tableau Formulaire Outils Fenêtre Aide
Titre 2
Calibri
16

Automatic Delivery Dr

Description

The aim of the automatic delive
from the commercial enterprise
equipped with a clamp. When a
warehouse and deliver it). The
accessible all the time by this
automated, but an operator mus
drone with the application. Th
brushless motor. The power sha
with a size up to 1×1 m.

How shall it works?

The drone shall pick up a parcel,
web application. An operator sh
encoded in a QR code on the par
the application. The drone shall

	A	B	C	D	E	F	G	H	I	J	K
1	#	Context	Requirement description	Priority	Trace to	Addition to	Alternative to	Contained by	Refines	Constraints	Contradicts
									2.1		

<<requirement>>
Avoid obstacle

text="The drone shall avoid obstacles"
Id="2"

<<deriveReq>>

<<requirement>>
Drone reaction

Id="2.1"

<<requirement>>
Sensor precision

text=" At any time, the sensors shall be able to detect an object in less than twice the stopping distance"
Id="3"

<<requirement>>
Stop timing

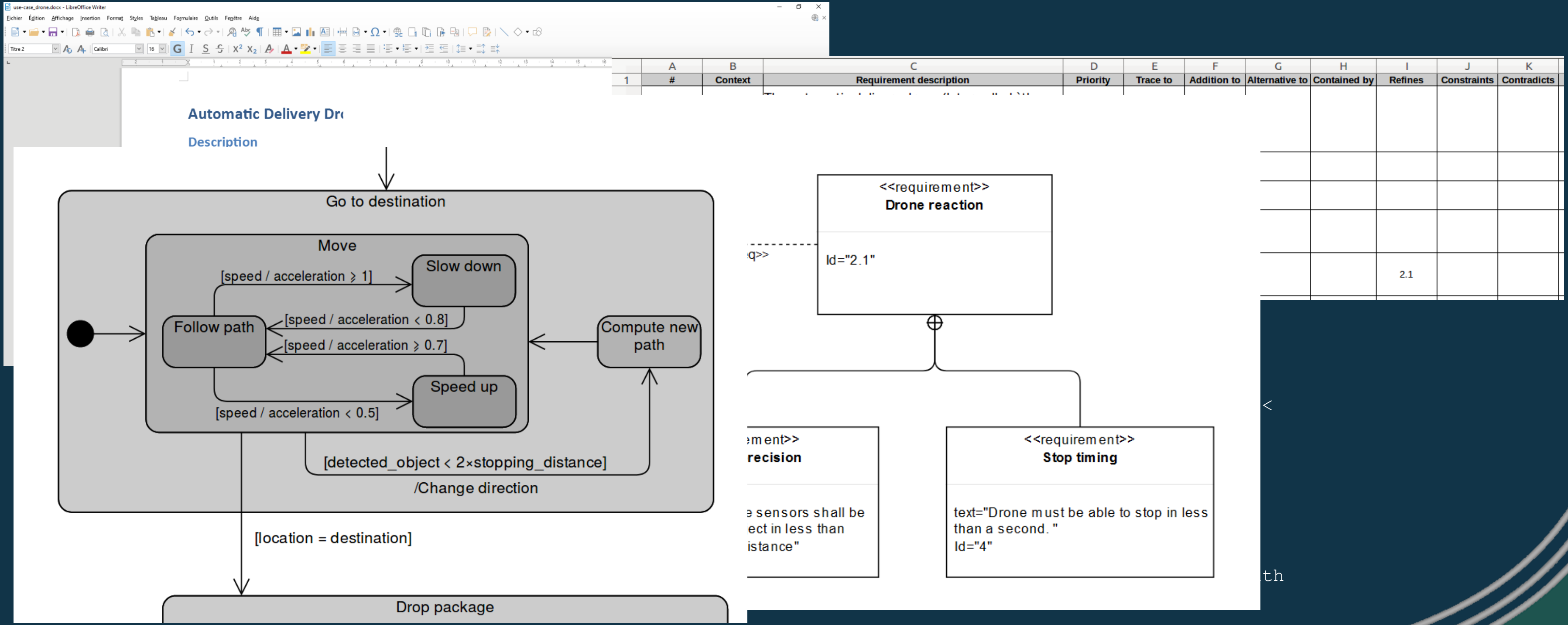
text="Drone must be able to stop in less than a second. "
Id="4"

<

th

One project; several stakeholders

« The drone shall avoid obstacles »



use-case_drone.docx - LibreOffice Writer

Fichier Édition Affichage Insertion Format Styles Tableau Formulaire Outils Fenêtre Aide
Titre 2
Calibri
16

Automatic Delivery Dr

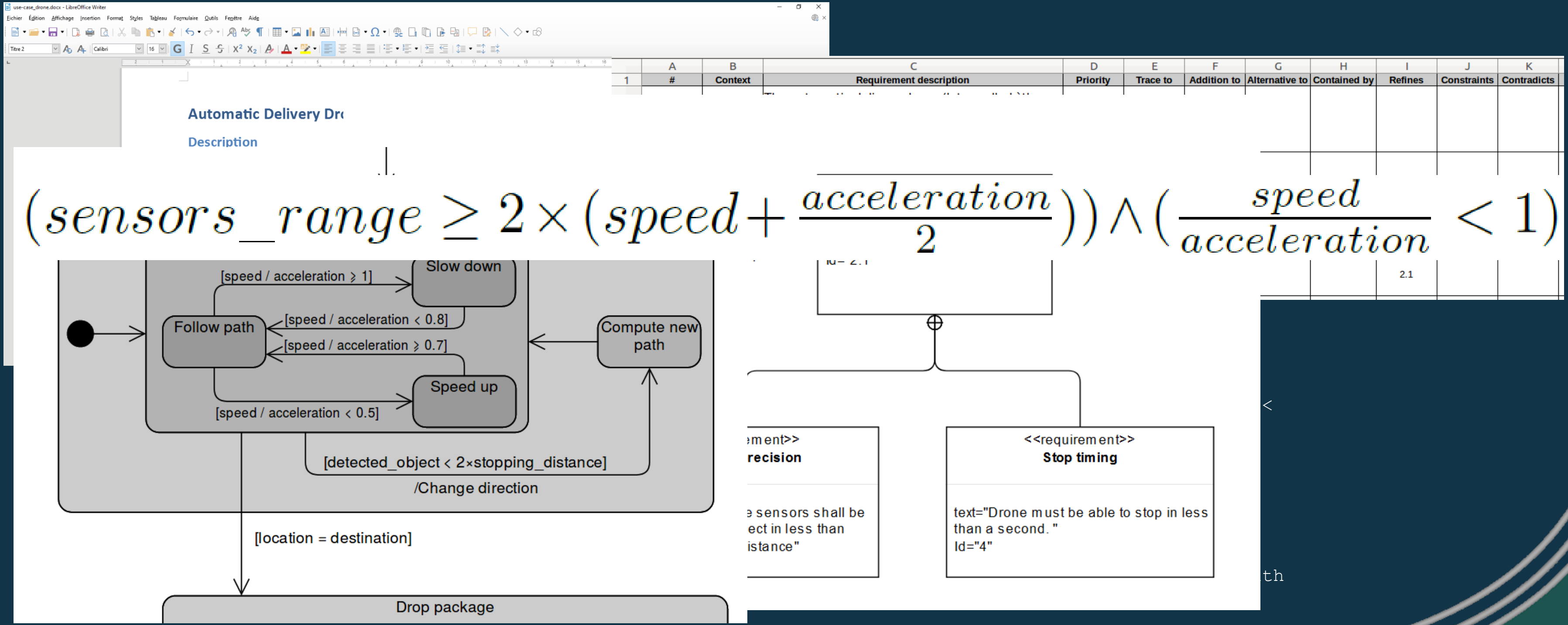
Description

	A	B	C	D	E	F	G	H	I	J	K
1	#	Context	Requirement description	Priority	Trace to	Addition to	Alternative to	Contained by	Refines	Constraints	Contradicts
									2.1		

Go to destination
Move
Slow down
[speed / acceleration > 1]
Follow path
[speed / acceleration < 0.8]
[speed / acceleration > 0.7]
Speed up
[speed / acceleration < 0.5]
Compute new path
[detected_object < 2×stopping_distance]
/Change direction
[location = destination]
Drop package
<<requirement>>
Drone reaction
Id="2.1"
q>>
ement>>
recision
e sensors shall be
ect in less than
istance"
<<requirement>>
Stop timing
text="Drone must be able to stop in less than a second. "
Id="4"
<
th

One project; several stakeholders

« The drone shall avoid obstacles »



use-case_drone.docx - LibreOffice Writer

Automatic Delivery Dr

Description

	A	B	C	D	E	F	G	H	I	J	K
1	#	Context	Requirement description	Priority	Trace to	Addition to	Alternative to	Contained by	Refines	Constraints	Contradicts
									2.1		

$$(sensors\_range \geq 2\times(speed+\frac{acceleration}{2}))\wedge(\frac{speed}{acceleration}<1)$$

Slow down
[speed / acceleration > 1]
Follow path
[speed / acceleration < 0.8]
[speed / acceleration > 0.7]
Compute new path
Speed up
[speed / acceleration < 0.5]
[detected_object < 2×stopping_distance]
/Change direction
[location = destination]
Drop package
Id= 2.1
<

ment>>
recision

e sensors shall be ect in less than istance"

<<requirement>>
Stop timing

text="Drone must be able to stop in less than a second."
Id="4"

How can **domain-specific modeling languages (DSML)** help us formalize customer requirements?

What is a DSL?

Domain Specific Language

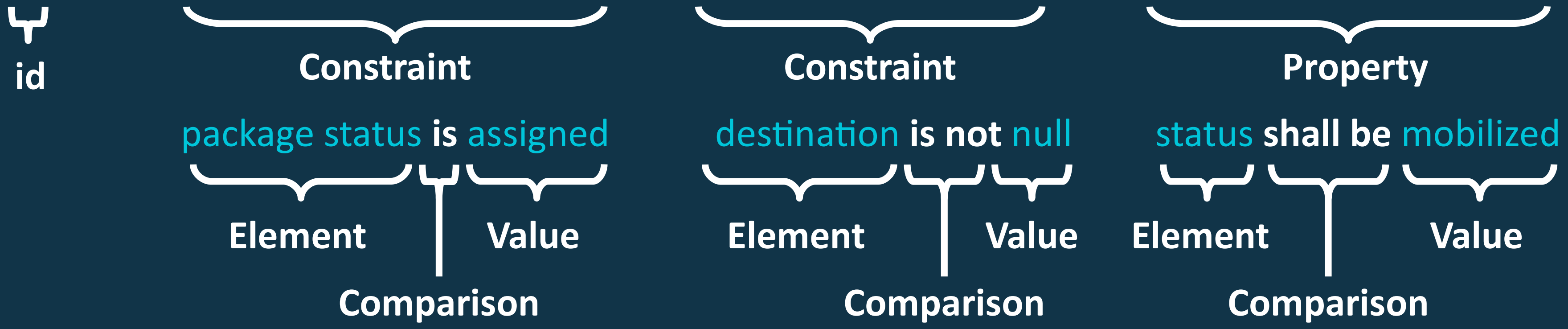
```
ALTER TABLE `app\model\project` ADD `jira` VARCHAR(200) NULL DEFAULT NULL AFTER `gitlab`;
```

```
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault"
aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-awesome fas fa-bars"></span>
        <span class="navbar-toggler-awesome fas fa-times"></span>
 </button>
```

R1 - **When** package status **is** assigned **and** destination **is not** null **then** status **shall be** mobilized**.**

A requirement DSL: example

R1 - **When** package status **is** assigned **and** destination **is not** null **then** status **shall be** mobilized.



id

Constraint

package status **is** assigned

Element

Comparison

Value

Constraint

destination **is not** null

Element

Comparison

Value

Property

status **shall be** mobilized

Element

Comparison

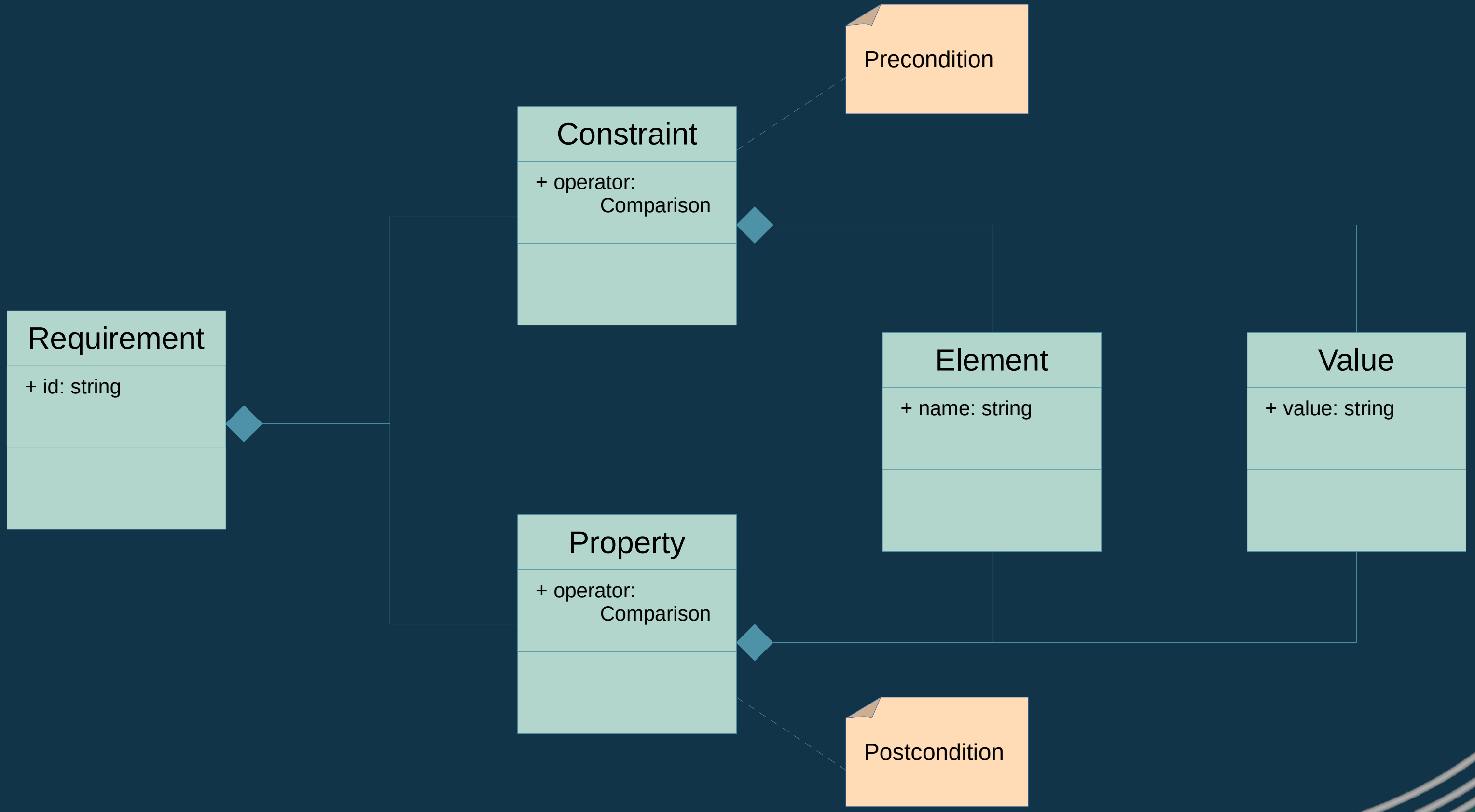
Value

Why model a requirement?

- Because a formal representation is already a model
- Because we can analyze and work with models
- Because we can transform models
- Because there are many tools for models and DSML*

*I'll talk about the M in DSML later

Modeling a requirement



Tools for creating DSML

Gemoc Studio

MDElib

emf

ECLIPSE MODELING FRAMEWORK

MPS

Bison

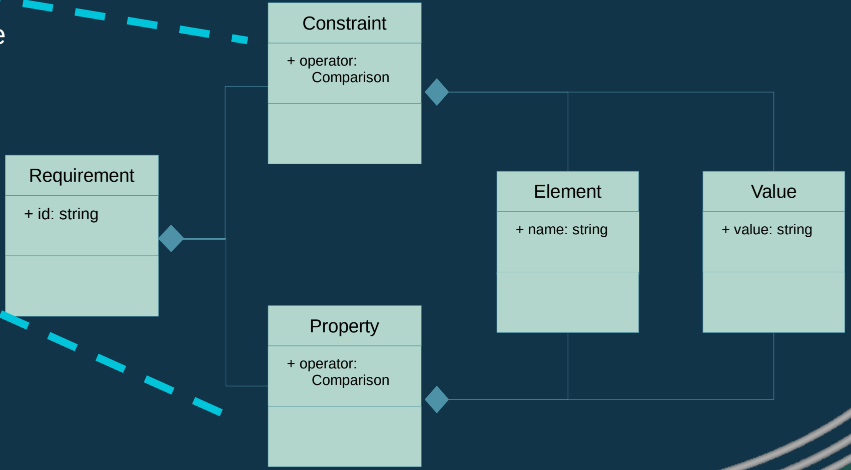
Xtext

A DSL for our requirements: define a grammar

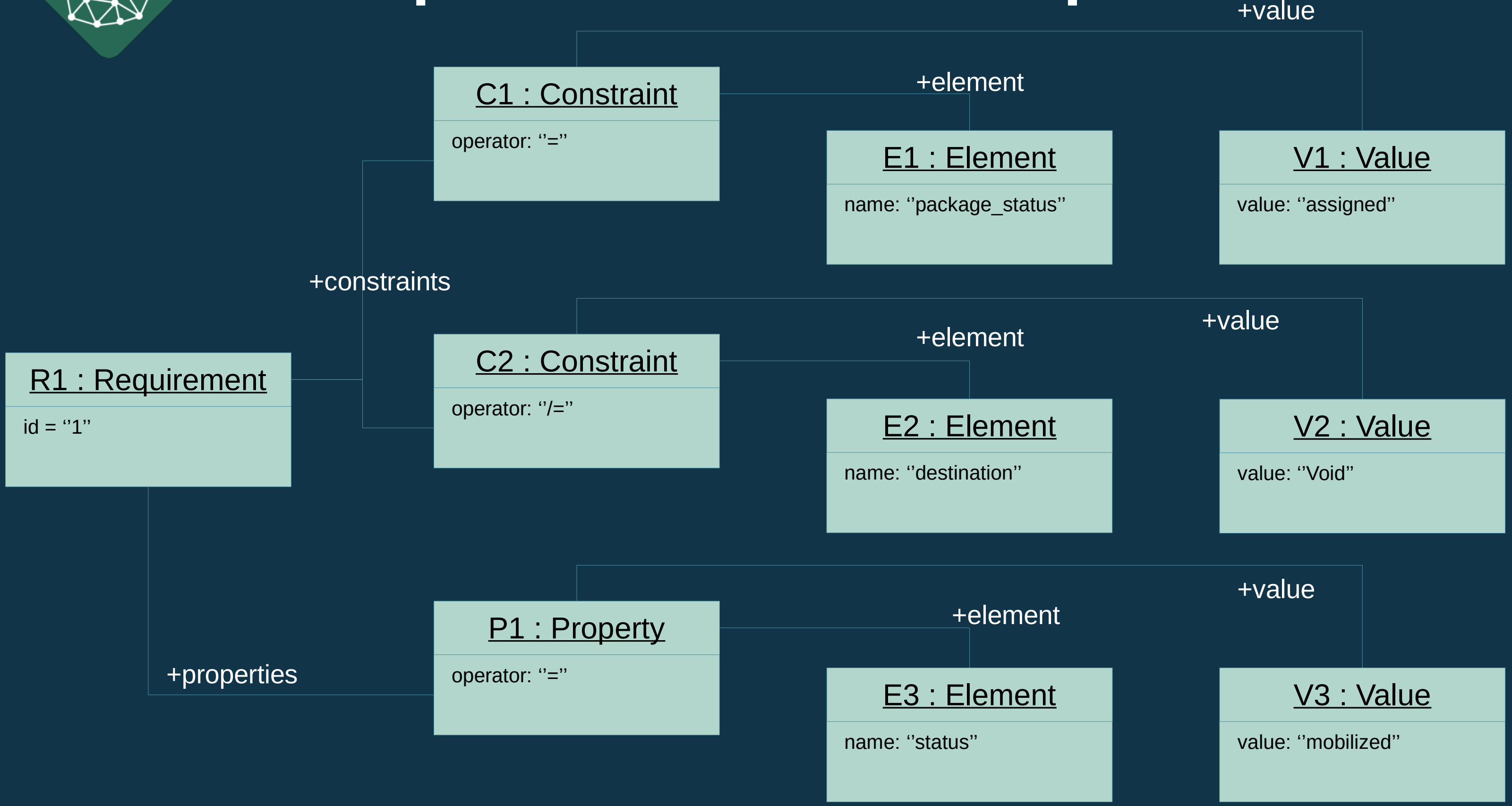
```
Requirement:
	id=REQID '-' "When" constraints+=Constraint ("and" constraints+=Constraint)*
			"then" properties+=Property ("and" properties+=Property)* '.'

Constraint:
	element=Element operator=Comparison value=Value

Property:
	element=Element operator=ModalComparison value=Value

terminal REQID:
	'R' INT ('.' REQID)* ;
```

A DSL for our requirements: define a grammar

```
Requirement:
        id=REQID '-' "When" constraints+=Constraint ("and" constraints+=Constraint)*
                                "then" properties+=Property ("and" properties+=Property)* '.'

Constraint:
        element=Element operator=Comparison value=Value


Property:
        element=Element operator=ModalComparison value=Value

terminal REQID:
        'R' INT ('.' REQID)* ;
```



A requirement DSL: example



R1 : Requirement

id = ‘’1’’

+constraints

C1 : Constraint

operator: ‘’=’’

+element

E1 : Element

name: ‘’package_status’’

+value

V1 : Value

value: ‘’assigned’’

C2 : Constraint

operator: ‘’/=’’

+element

E2 : Element

name: ‘’destination’’

+value

V2 : Value

value: ‘’Void’’

+properties

P1 : Property

operator: ‘’=’’

+element

E3 : Element

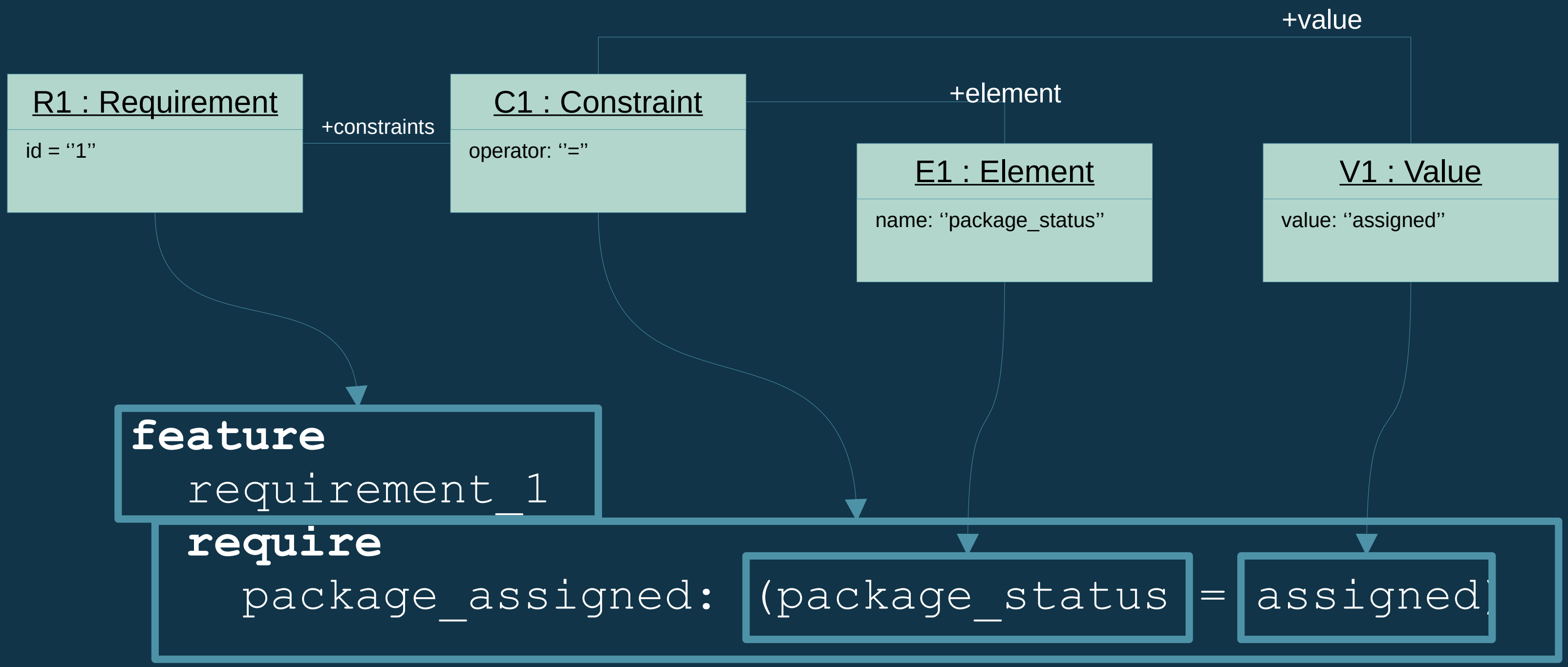
name: ‘’status’’

+value

V3 : Value

value: ‘’mobilized’’

A requirement DSL: example



R1 : Requirement

id = ‘’1’’

+constraints

C1 : Constraint

operator: ‘’=’’

+element

E1 : Element

name: ‘’package_status’’

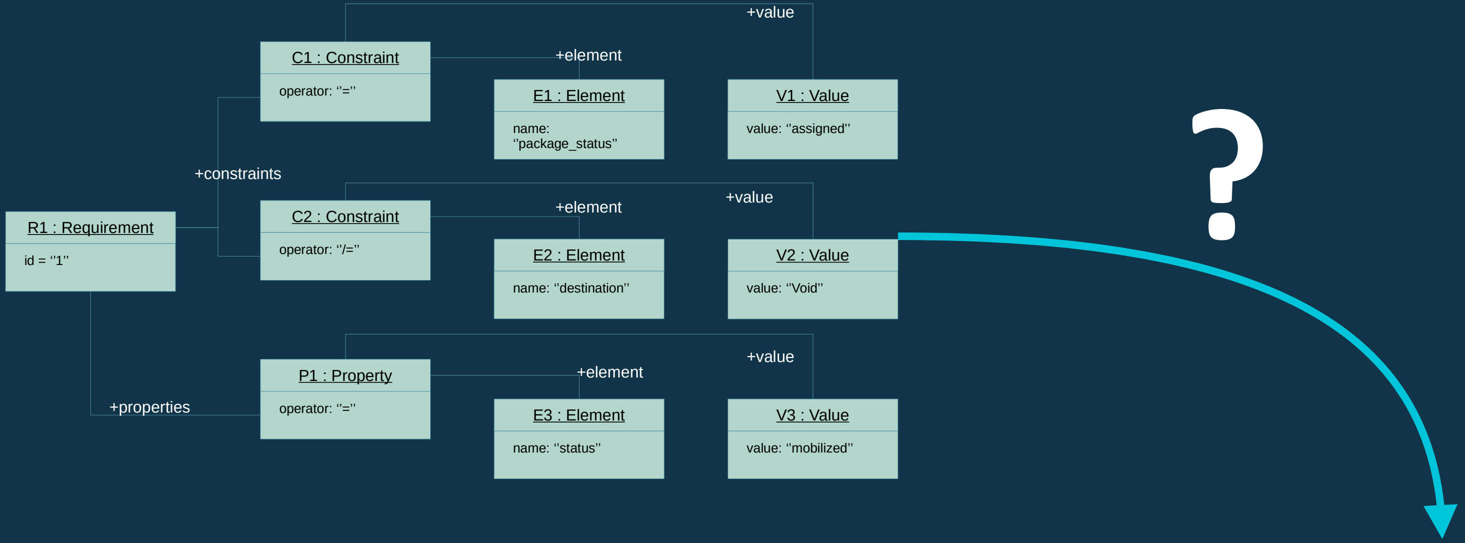
+value

V1 : Value

value: ‘’assigned’’

```
feature
  requirement_1
 require
    package_assigned: (package_status = assigned)
```

A requirement DSL: code generation



R1 : Requirement
id = ‘‘1’’

+constraints

C1 : Constraint
operator: ‘‘=’’

+element

E1 : Element
name: ‘‘package_status’’

+value

V1 : Value
value: ‘‘assigned’’

C2 : Constraint
operator: ‘‘/=’’

+element

E2 : Element
name: ‘‘destination’’

+value

V2 : Value
value: ‘‘Void’’

+properties

P1 : Property
operator: ‘‘=’’

+element

E3 : Element
name: ‘‘status’’

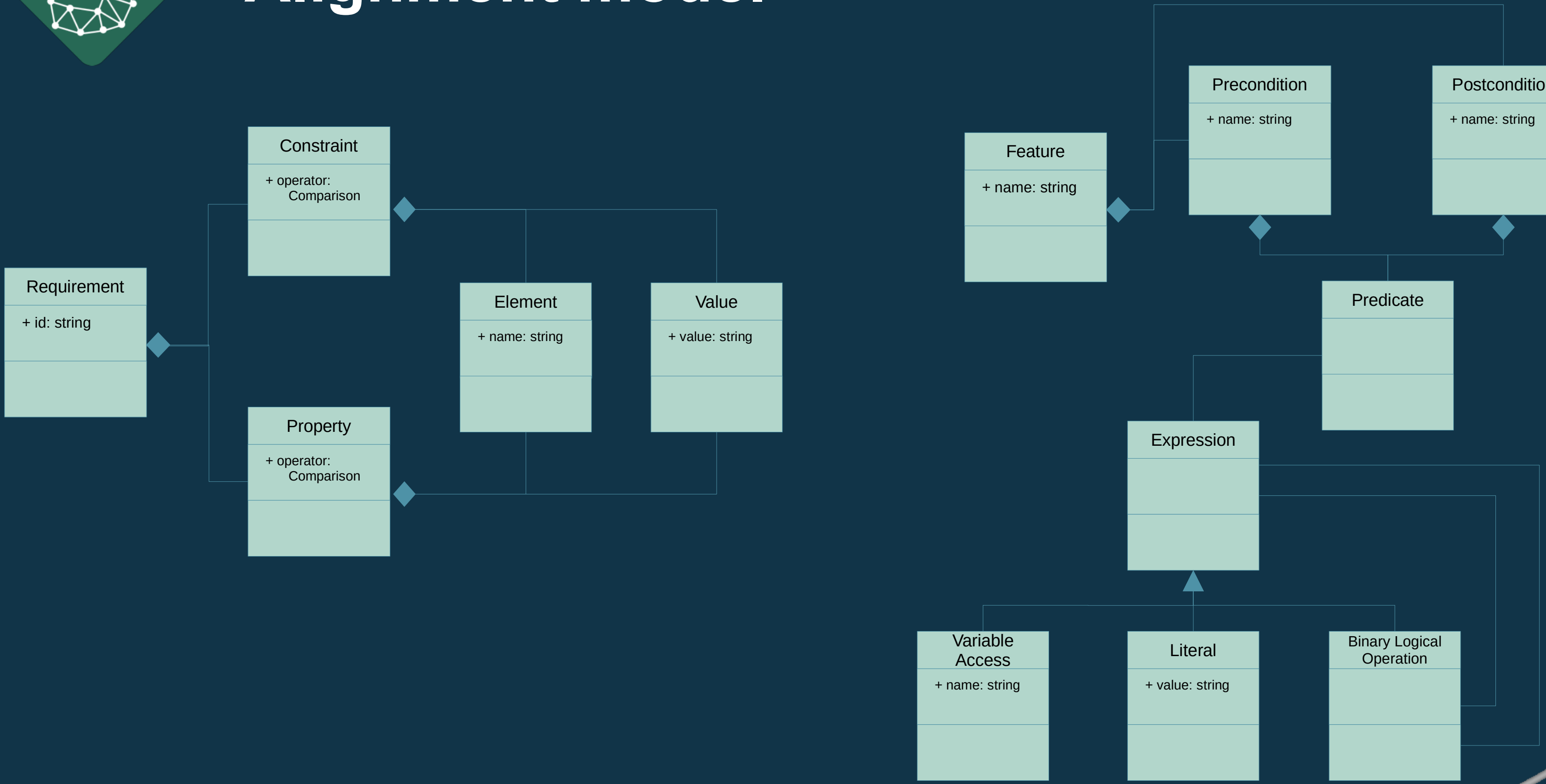
+value

V3 : Value
value: ‘‘mobilized’’

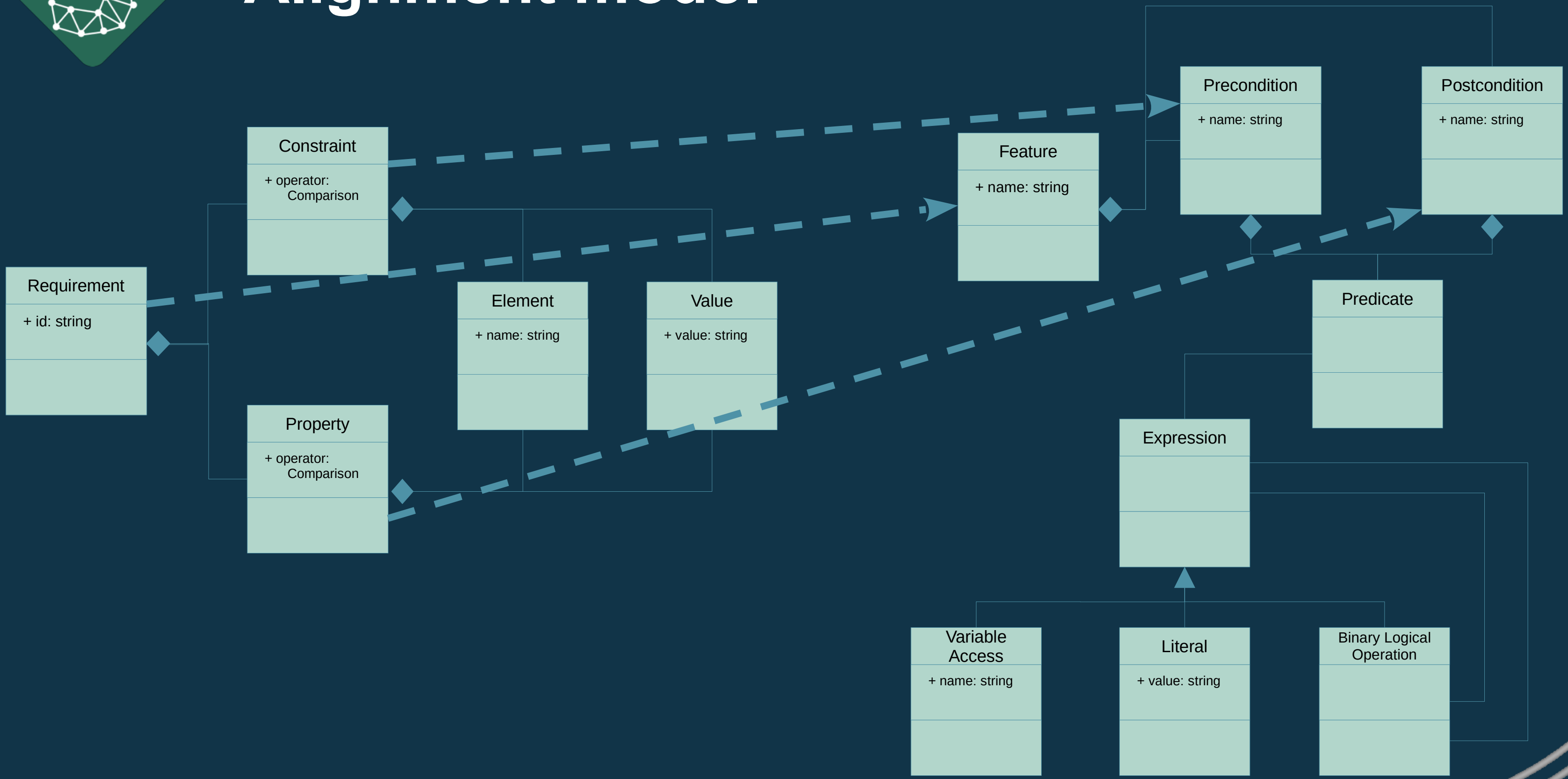
?

```
feature
  requirement_1
  note
    src: "{SHIPMENT_REQUIREMENTS}.requirement_1_doc"
  require
    package_assigned: (package_status = assigned)
    has_destination:  (destination /= Void)
  deferred
  ensure
    check_drone_status: (drone_status = mobilized)
  end
```

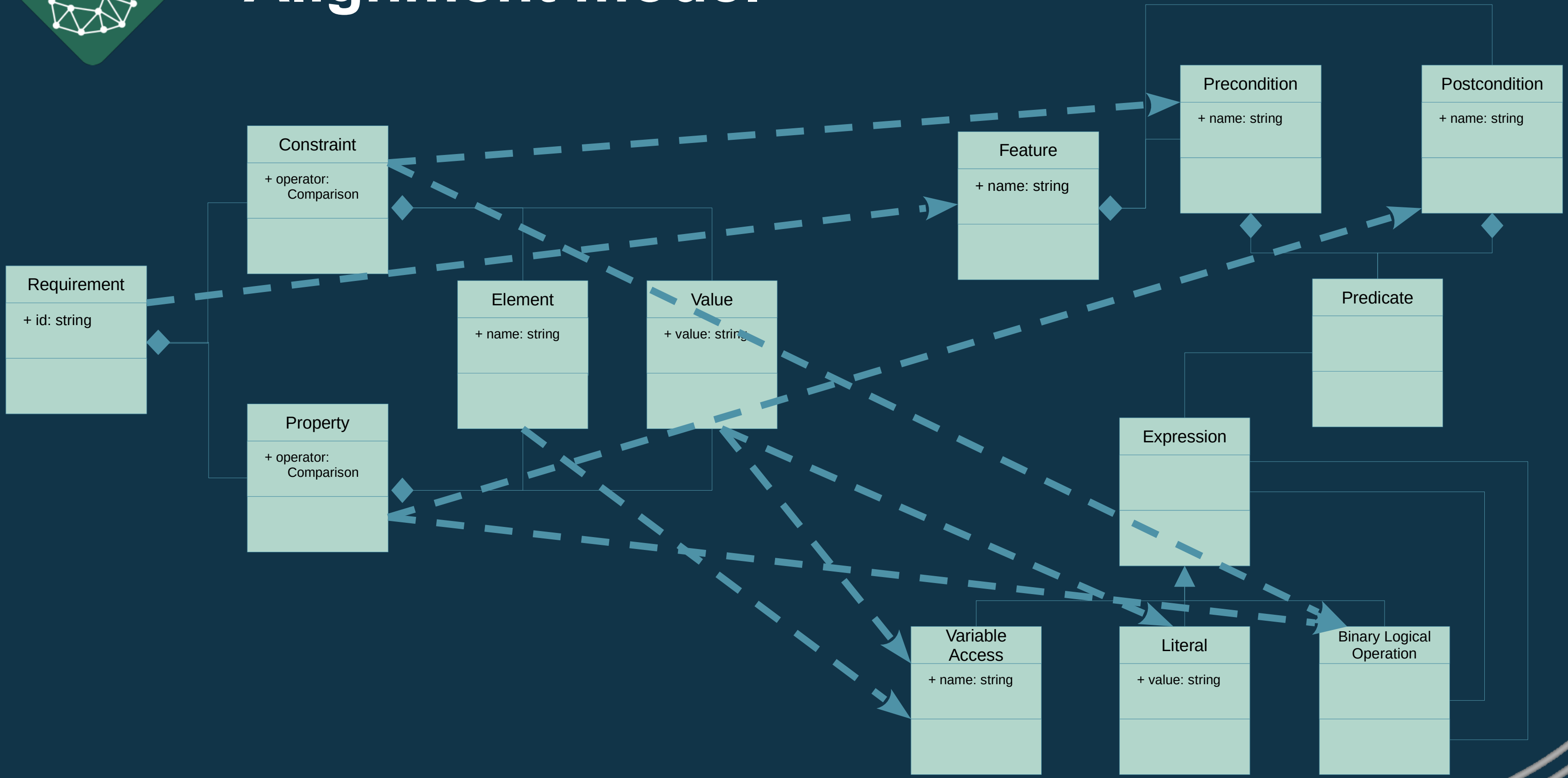
Alignment model



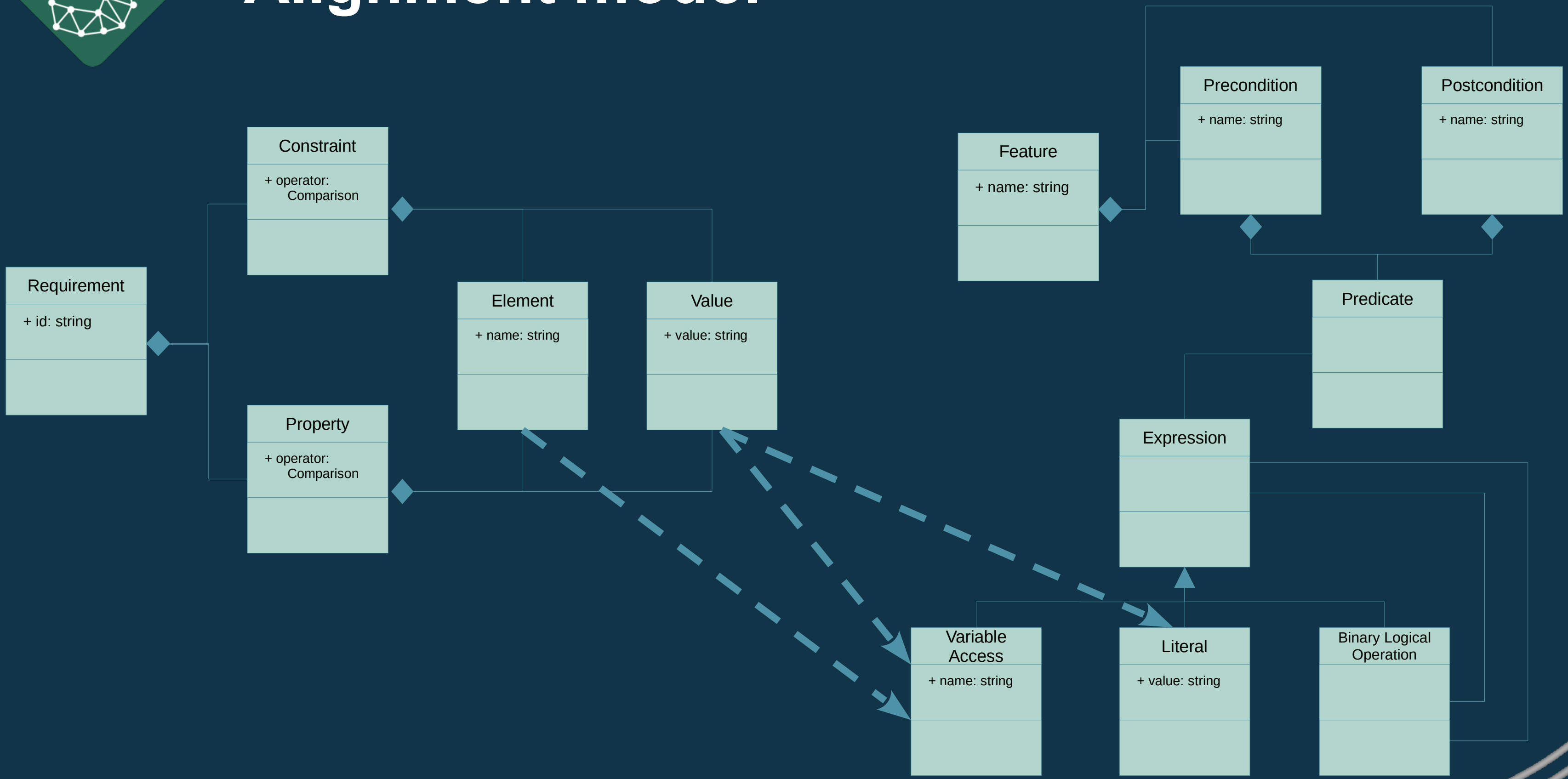
Alignment model



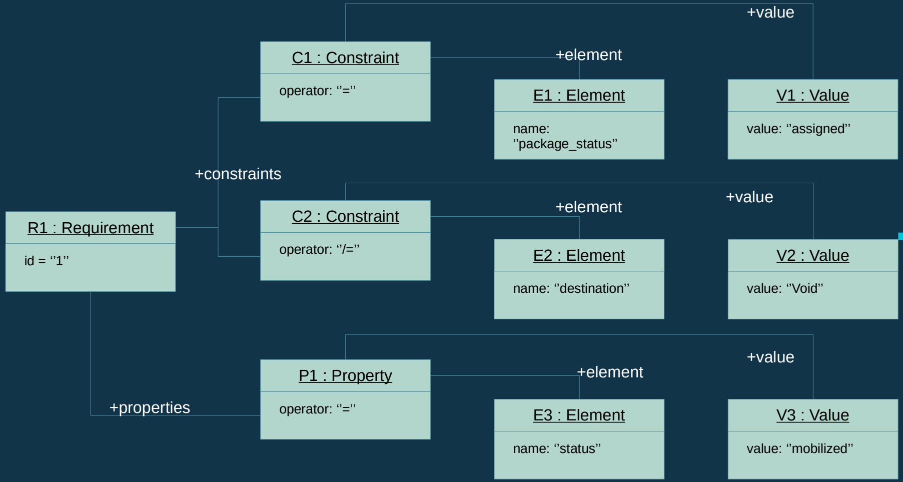
Alignment model



Alignment model



A requirement DSL: code generation



```
def compile(Requirement r) '''
feature
  «r.formattedName»
  note
    src: "«r.docClass.formattedName».«r.formattedName»_doc"
  «IF r.constraints.size > 0»
  require
    «FOR pre : r.constraints.toList»
      «pre.compileWithName»
    «ENDFOR»
  «ENDIF»
  deferred
  «IF r.assertions.size > 0»
  ensure
    «FOR post : r.assertions.toList»
      «post.compileWithName»
    «ENDFOR»
  «ENDIF»
end
'''
```

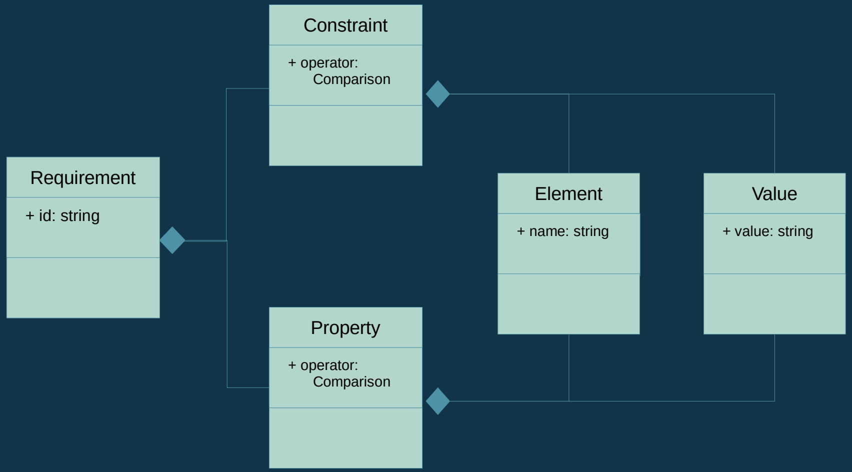
```
feature
  requirement_1
  note
    src: "{SHIPMENT_REQUIREMENTS}.requirement_1_doc"
  require
    package_assigned: (package_status = assigned)
    has_destination:  (destination /= Void)
  deferred
  ensure
    check_drone_status: (drone_status = mobilized)
  end
```

Demo

MPS

Why DSML?

Domain Specific **Modeling** Language



Requirement

+ id: string

Constraint

+ operator: Comparison

Property

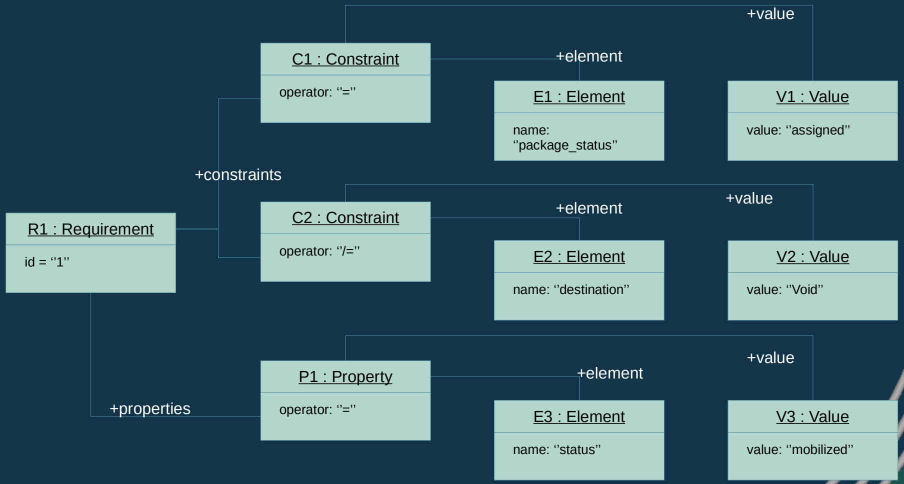
+ operator: Comparison

Element

+ name: string

Value

+ value: string



R1 : Requirement

id = “1”

+constraints

+properties

C1 : Constraint

operator: “=”

C2 : Constraint

operator: “/=”

P1 : Property

operator: “=”

E1 : Element

name: “package_status”

E2 : Element

name: “destination”

E3 : Element

name: “status”

V1 : Value

value: “assigned”

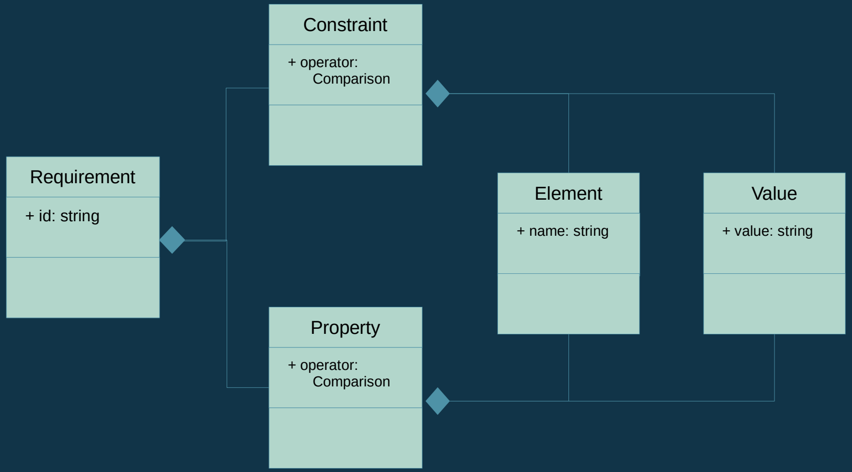
V2 : Value

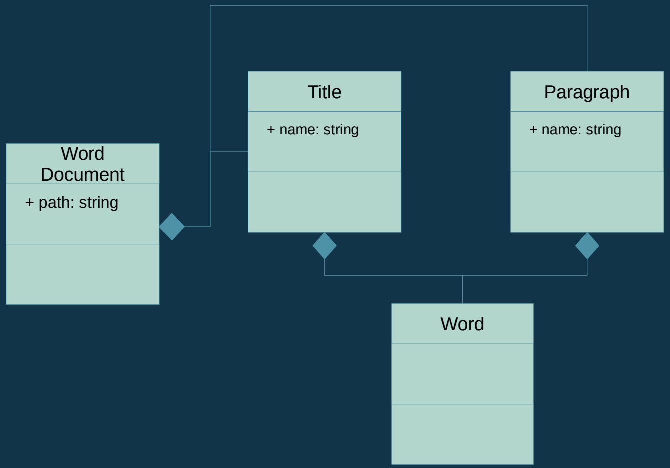
value: “Void”

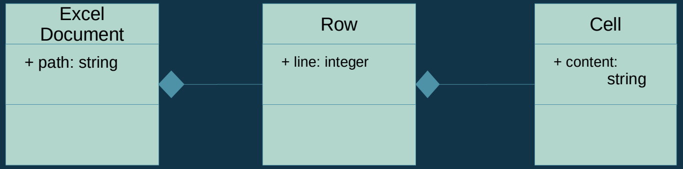
V3 : Value

value: “mobilized”

Alignment model

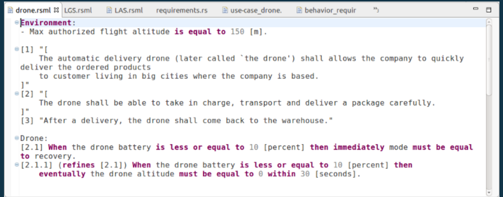




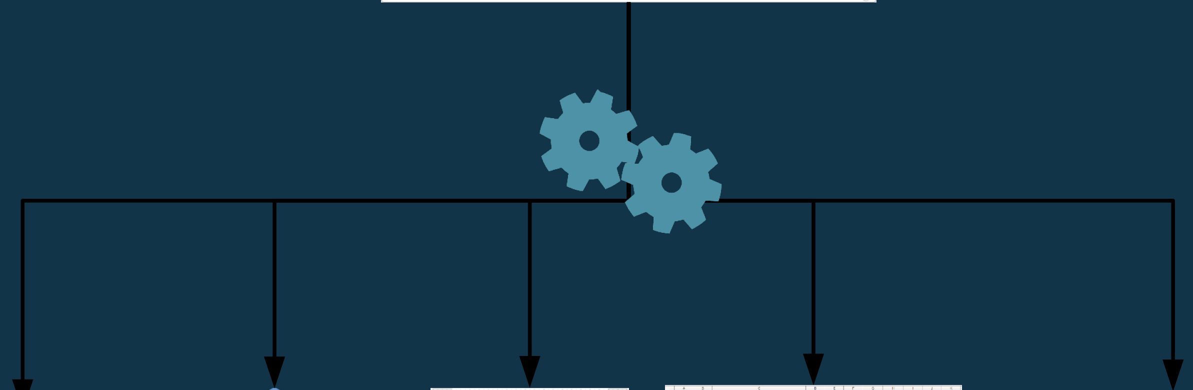


Example: RSML

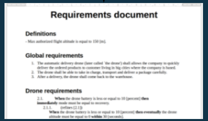
RSML



```
Environment:
- Max authorized flight altitude is equal to 150 [m].

[1] "[
    The automatic delivery drone (later called `the drone') shall allows the company to quickly
deliver the ordered products
    to customer living in big cities where the company is based.
]"
[2] "[
    The drone shall be able to take in charge, transport and deliver a package carefully.
]"
[3] "After a delivery, the drone shall come back to the warehouse."

Drone:
[2.1] When the drone battery is less or equal to 10 [percent] then immediately mode must be equal
to recovery.
[2.1.1] (refines [2.1]) When the drone battery is less or equal to 10 [percent] then
    eventually the drone altitude must be equal to 0 within 30 [seconds].
```



Eiffel



Requirements document

Definitions

- Max authorized flight altitude is equal to 150 [m].

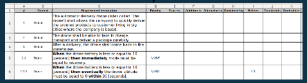
Global requirements

1. The automatic delivery drone (later called `the drone') shall allows the company to quickly deliver the ordered products to customer living in big cities where the company is based.
2. The drone shall be able to take in charge, transport and deliver a package carefully.
3. After a delivery, the drone shall come back to the warehouse.

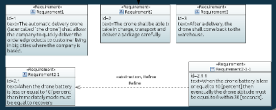
Drone requirements

2.1. **When** the drone battery is less or equal to 10 [percent] **then immediately** mode must be equal to recovery.

2.1.1. (refines [2.1])
When the drone battery is less or equal to 10 [percent] **then eventually** the drone altitude must be equal to 0 **within** 30 [seconds].



	A	B	C	D	E	F	G	H	I	J	K
1	id	Context	Requirement description	Priority	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	1	Global	The automatic delivery drone (later called `the drone') shall allows the company to quickly deliver the ordered products to customer living in big cities where the company is based.								
3	2	Global	The drone shall be able to take in charge, transport and deliver a package carefully.								
4	3	Global	After a delivery, the drone shall come back to the warehouse.								
5	2.1	Drone	**When** the drone battery is less or equal to 10 [percent] **then immediately** mode must be equal to recovery.	MUST							
6	2.1.1	Drone	**When** the drone battery is less or equal to 10 [percent] **then eventually** the drone altitude must be equal to 0 **within** 30 [seconds].	MUST					2.1		



«Requirement»
Requirement1
id=1
text=The automatic delivery drone (later called `the drone') shall allows the company to quickly deliver the ordered products to customer living in big cities where the company is based.

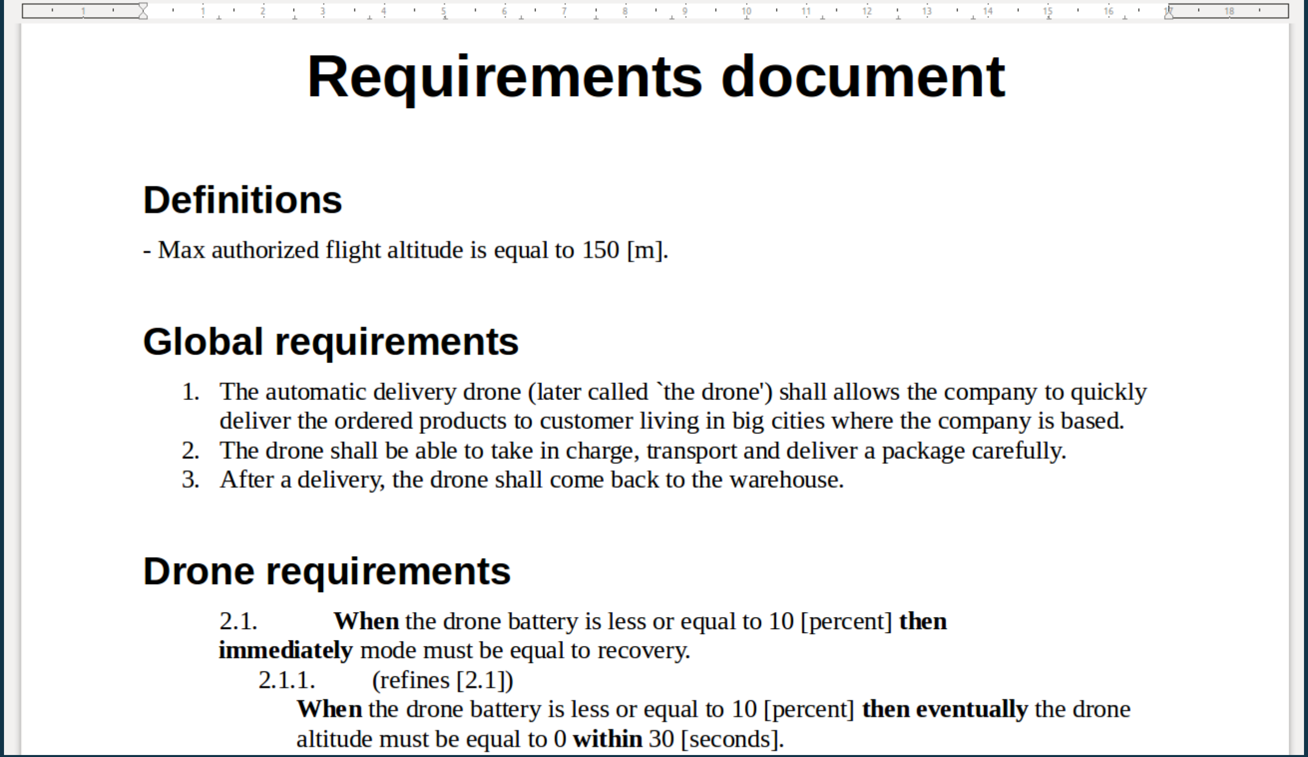
«Requirement»
Requirement2
id=2
text=The drone shall be able to take in charge, transport and deliver a package carefully.

«Requirement»
Requirement3
id=3
text=After a delivery, the drone shall come back to the warehouse.

«Requirement»
Requirement2.1
id=2.1
text=When the drone battery is less or equal to 10 [percent] then immediately mode must be equal to recovery.

«Requirement»
Requirement2.1.1
id=2.1.1
text=When the drone battery is less or equal to 10 [percent] then eventually the drone altitude must be equal to 0 within 30 [seconds].

Example: From RSML to docx



Requirements document

Definitions

- Max authorized flight altitude is equal to 150 [m].

Global requirements

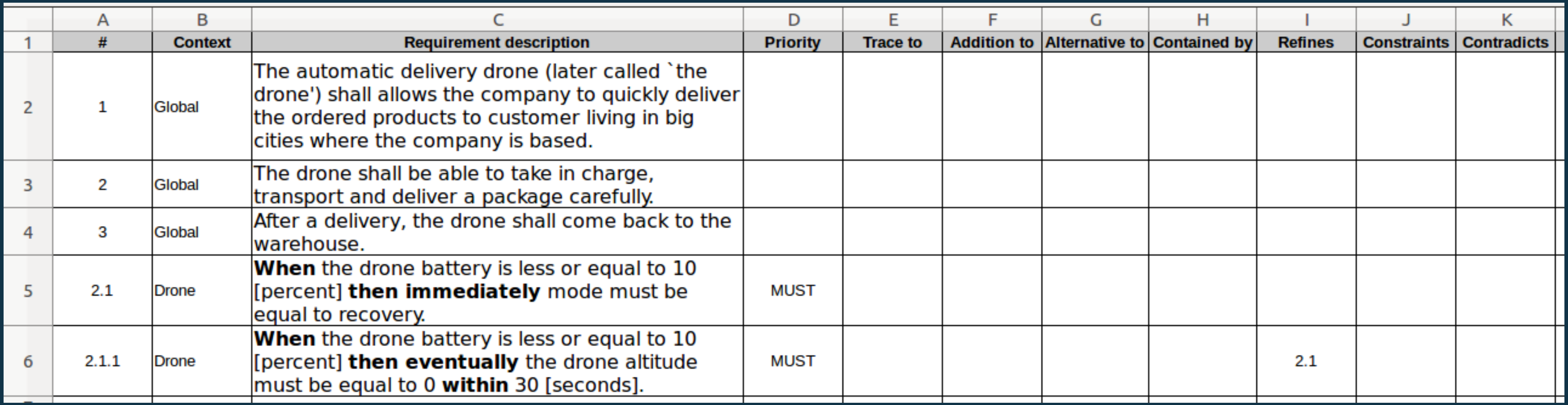
1. The automatic delivery drone (later called `the drone') shall allows the company to quickly deliver the ordered products to customer living in big cities where the company is based.
2. The drone shall be able to take in charge, transport and deliver a package carefully.
3. After a delivery, the drone shall come back to the warehouse.

Drone requirements

2.1. **When** the drone battery is less or equal to 10 [percent] **then immediately** mode must be equal to recovery.

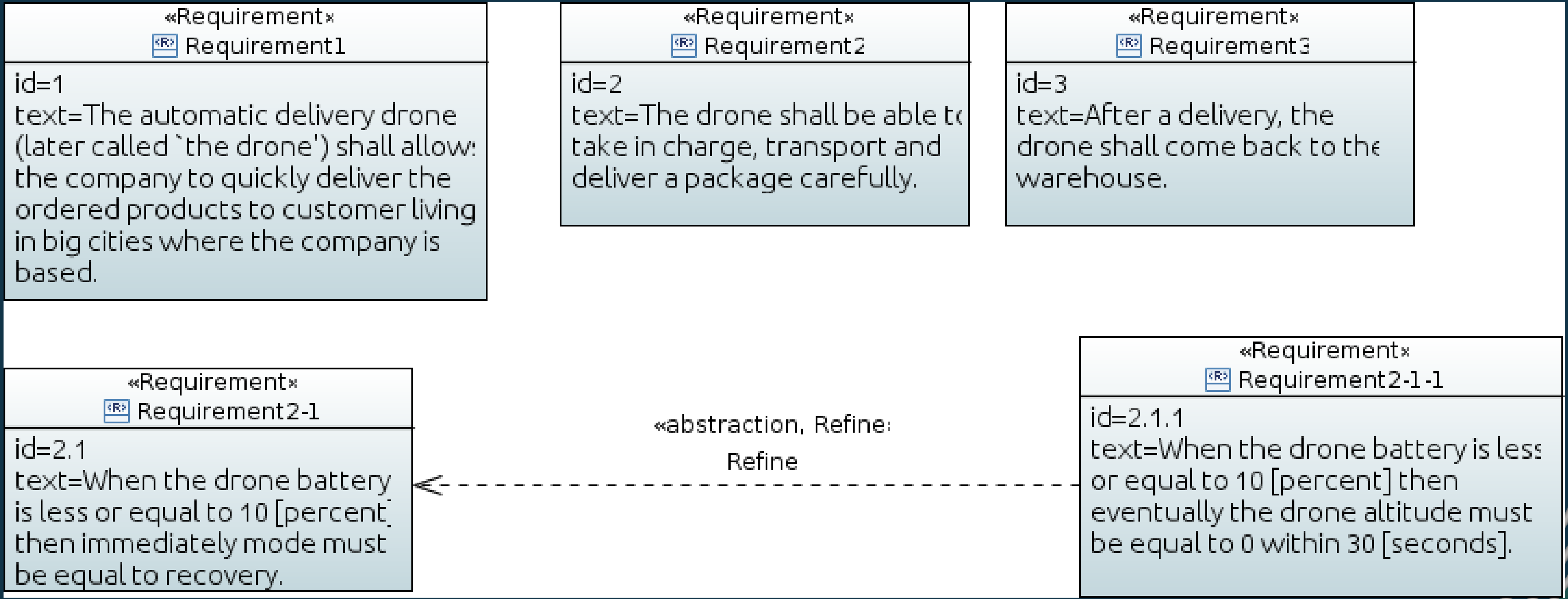
2.1.1. (refines [2.1])
When the drone battery is less or equal to 10 [percent] **then eventually** the drone altitude must be equal to 0 **within** 30 [seconds].

Example: From RSML to spreadsheet



	A	B	C	D	E	F	G	H	I	J	K
1	**#**	**Context**	**Requirement description**	**Priority**	**Trace to**	**Addition to**	**Alternative to**	**Contained by**	**Refines**	**Constraints**	**Contradicts**
2	1	Global	The automatic delivery drone (later called `the drone') shall allows the company to quickly deliver the ordered products to customer living in big cities where the company is based.								
3	2	Global	The drone shall be able to take in charge, transport and deliver a package carefully.								
4	3	Global	After a delivery, the drone shall come back to the warehouse.								
5	2.1	Drone	**When** the drone battery is less or equal to 10 [percent] **then immediately** mode must be equal to recovery.	MUST							
6	2.1.1	Drone	**When** the drone battery is less or equal to 10 [percent] **then eventually** the drone altitude must be equal to 0 **within** 30 [seconds].	MUST					2.1		

Example: From SysML to RSML



«Requirement»
Requirement1

id=1
text=The automatic delivery drone (later called `the drone') shall allow: the company to quickly deliver the ordered products to customer living in big cities where the company is based.

«Requirement»
Requirement2

id=2
text=The drone shall be able t take in charge, transport and deliver a package carefully.

«Requirement»
Requirement3

id=3
text=After a delivery, the drone shall come back to the warehouse.

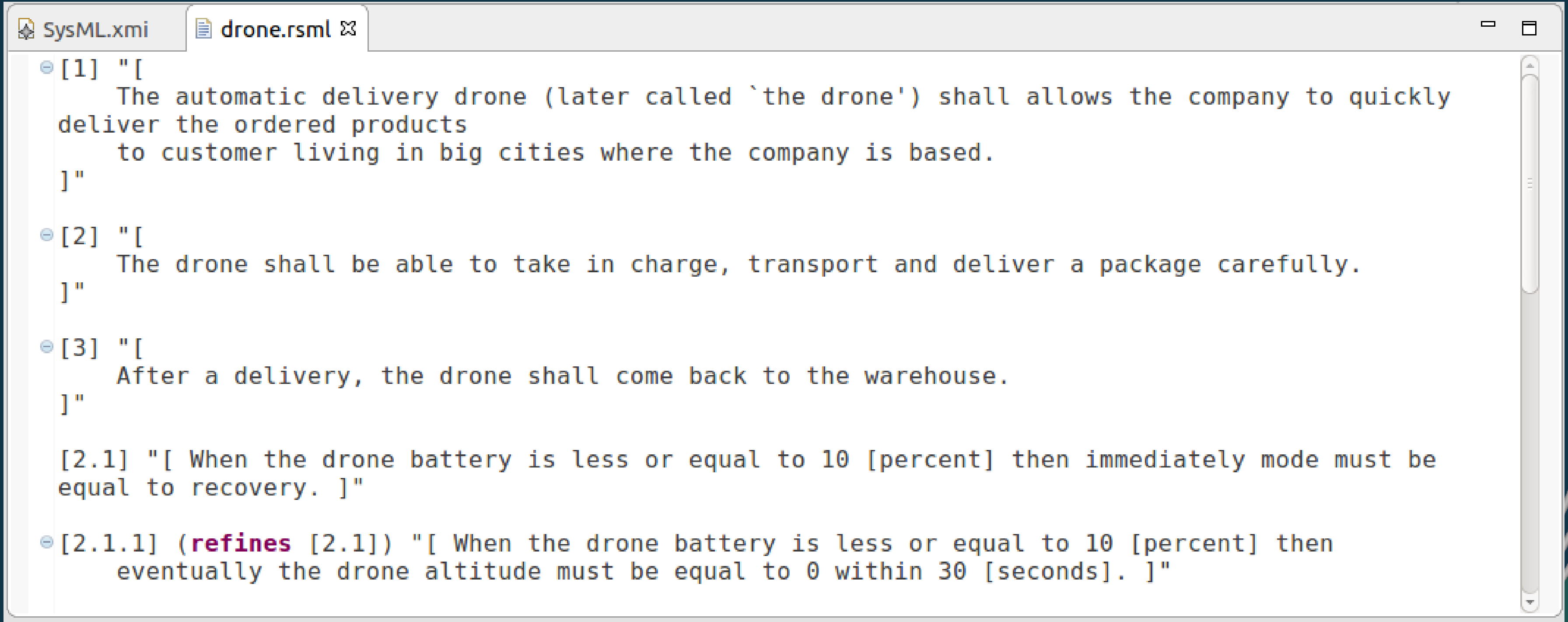
«Requirement»
Requirement2-1

id=2.1
text=When the drone battery is less or equal to 10 [percent then immediately mode must be equal to recovery.

«Requirement»
Requirement2-1-1

id=2.1.1
text=When the drone battery is less or equal to 10 [percent] then eventually the drone altitude must be equal to 0 within 30 [seconds].

Example: From SysML to RSML



```
[1] "[
    The automatic delivery drone (later called `the drone') shall allows the company to quickly
deliver the ordered products
    to customer living in big cities where the company is based.
]"

[2] "[
    The drone shall be able to take in charge, transport and deliver a package carefully.
]"

[3] "[
    After a delivery, the drone shall come back to the warehouse.
]"

[2.1] "[ When the drone battery is less or equal to 10 [percent] then immediately mode must be
equal to recovery. ]"

[2.1.1] (refines [2.1]) "[ When the drone battery is less or equal to 10 [percent] then
    eventually the drone altitude must be equal to 0 within 30 [seconds]. ]"
```

Is it really useful?

Is it really useful?

SPILEn

TOULOUSE
TECH
TRANSFER

DOC
DOC

DOC
D'occitanie
Le programme qui enrichit ton doctorat

- Startup created in December 2020
- *Toulouse Tech Transfer* technology transfer and financial support
- Laureate Doc d'Occitanie
- Test projects with two companies in the industry

Who are we?

Clément Simon - COO

Project manager

Jimmy Lopez - CRO

Multi-skilled developer

Manuel Chataigner - CTO

Technical expert

Florian Galinier PhD - CEO

Leader, project leader

SPILEn

Improving the technological world of tomorrow

Develop and provide:

innovative software engineering **tools**

for:

software companies

Our tools

MDElib

Inspecto need

for project managers and product owners

Instant verification of the compliance of the specifications with the customers' requests.

Project managers and product owners secure the start of projects by eliminating human interpretation errors when writing specifications.

Inspecto code

for developers

Continuous verification of program compliance with specifications.

Developers can detect and correct programming errors very early in the development cycle.

Demo

Inspecto

need

PILEn

Solutions Providers and Innovative Lab of Engineering