reqT.org

Towards a Semi-Formal, Open and Scalable Requirements Modeling Tool

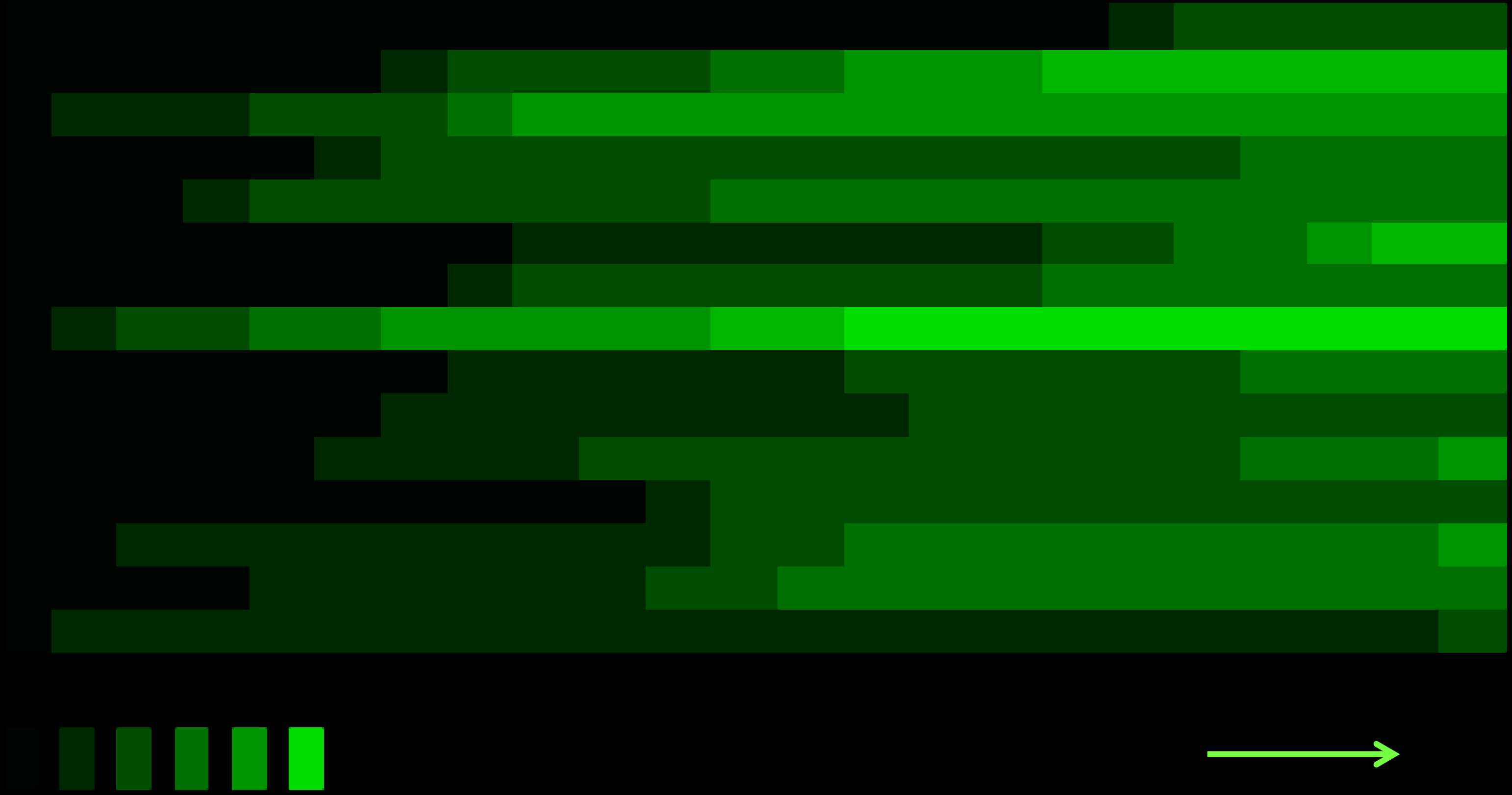
Björn Regnell

cs.lth.se/bjornregnell

LundUniversity.lu.se

```
var myRequirements = Model( ??? )
```

Good enough Requirements?



3 software engineering trends

- **D**ecentralize, **D**istribute, **D**ocument less
  - Agile teams
  - No centrally controlled, detailed "master plan"
  - Continuous integration & deployment
  - Increased parallelization
  - Distributed Version Control, e.g. Git
  - "Code is king"

```
> ls challenge:
```

How to help
code-focused, agile
software engineers
to do good
requirements engineering?

Provide an interesting tool that is...

Goal	Design	Rationale
Semi-formal	• Use graph structures • Mix Natural Language (NL) with RE semantics	• Graphs are well-known by software engineers and powerful for expressing structure and flexible for search. • NL is well-known and powerful...
Open	• Free, permissive license • Cross-platform: JVM	• Allow integration of existing code bases • Enable academic usage and contribution
Scalable	• Internal DSL in Scala http://www.scala-lang.org/	• Open-ended language • Scala is scalable, powerful, concise, typesafe, scriptable, ...

Requirements == Code

- Requirements as computational entities
- Serialize as self-generating code
- Flexible meta-model and semantics:
  > warn, don't force

Requirements modelling in reqT

A reqT model includes sequences of graph parts

`<Entity> <Edge> <NodeSet>`

separated by comma and wrapped inside a **`Model( )`**

```
Model(
  Feature("f1") has (Spec("A good spec."), Status(SPECIFIED)),
  Feature("f1") requires (Feature("f2"), Feature("f3")),
  Stakeholder("s1") assigns(Prio(1)) to Feature("f2")
)
```

Requirements modelling in reqT

A reqT model includes sequences of graph parts

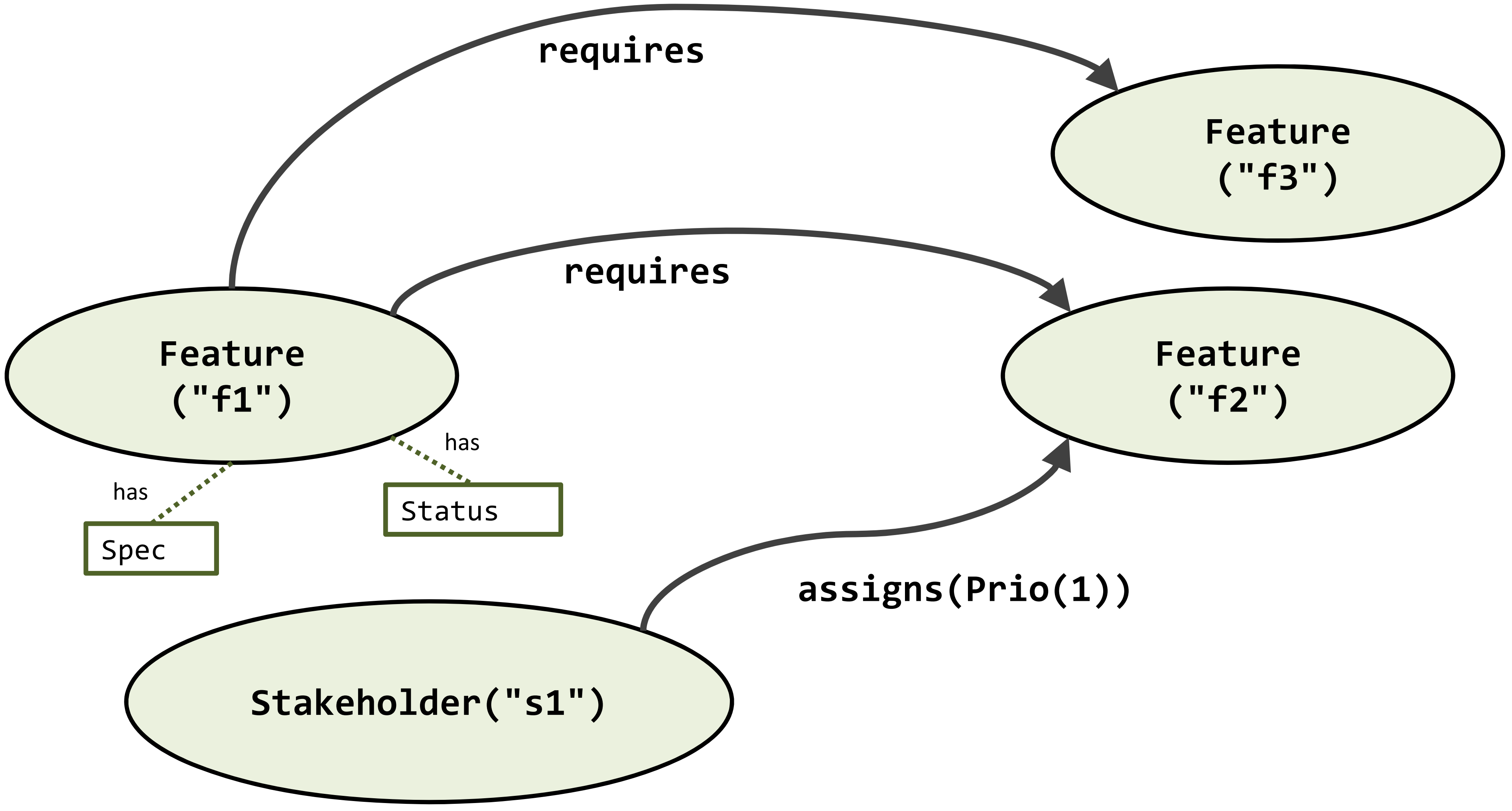
<Entity> <Edge> <NodeSet>

separated by comma and wrapped inside a **`Model( )`**

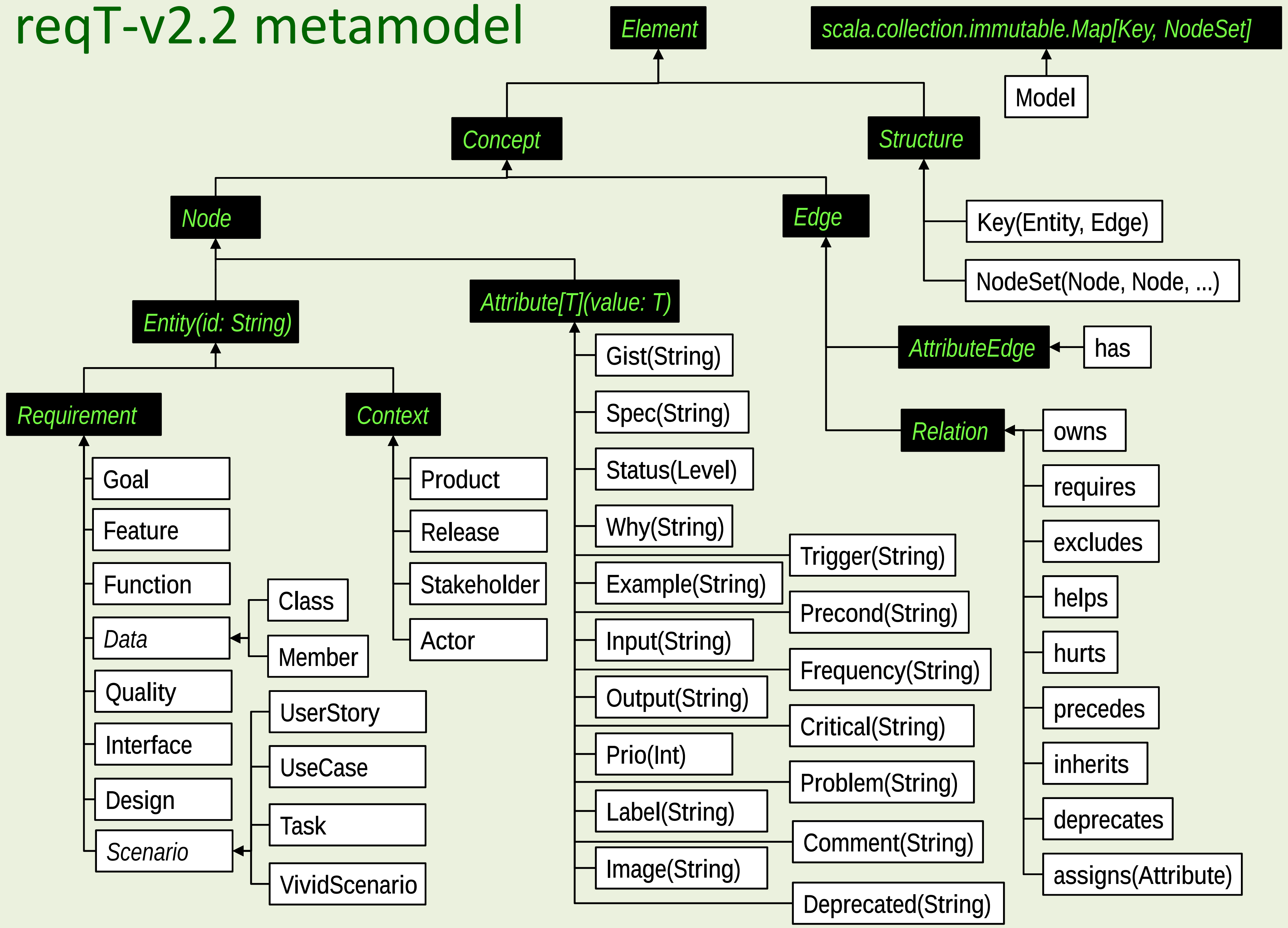
```
Model(
  Feature("f1") has (Spec("A good spec."), Status(SPECIFIED)),
  Feature("f1") requires (Feature("f2"), Feature("f3")),
  Stakeholder("s1") assigns(Prio(1)) to Feature("f2")
)
```

Implied reqT graph structure

```
Model(
  Feature("f1") has (Spec("A good spec."), Status(SPECIFIED)),
  Feature("f1") requires (Feature("f2"), Feature("f3")),
  Stakeholder("s1") assigns(Prio(1)) to Feature("f2")
)
```



reqT-v2.2 metamodel

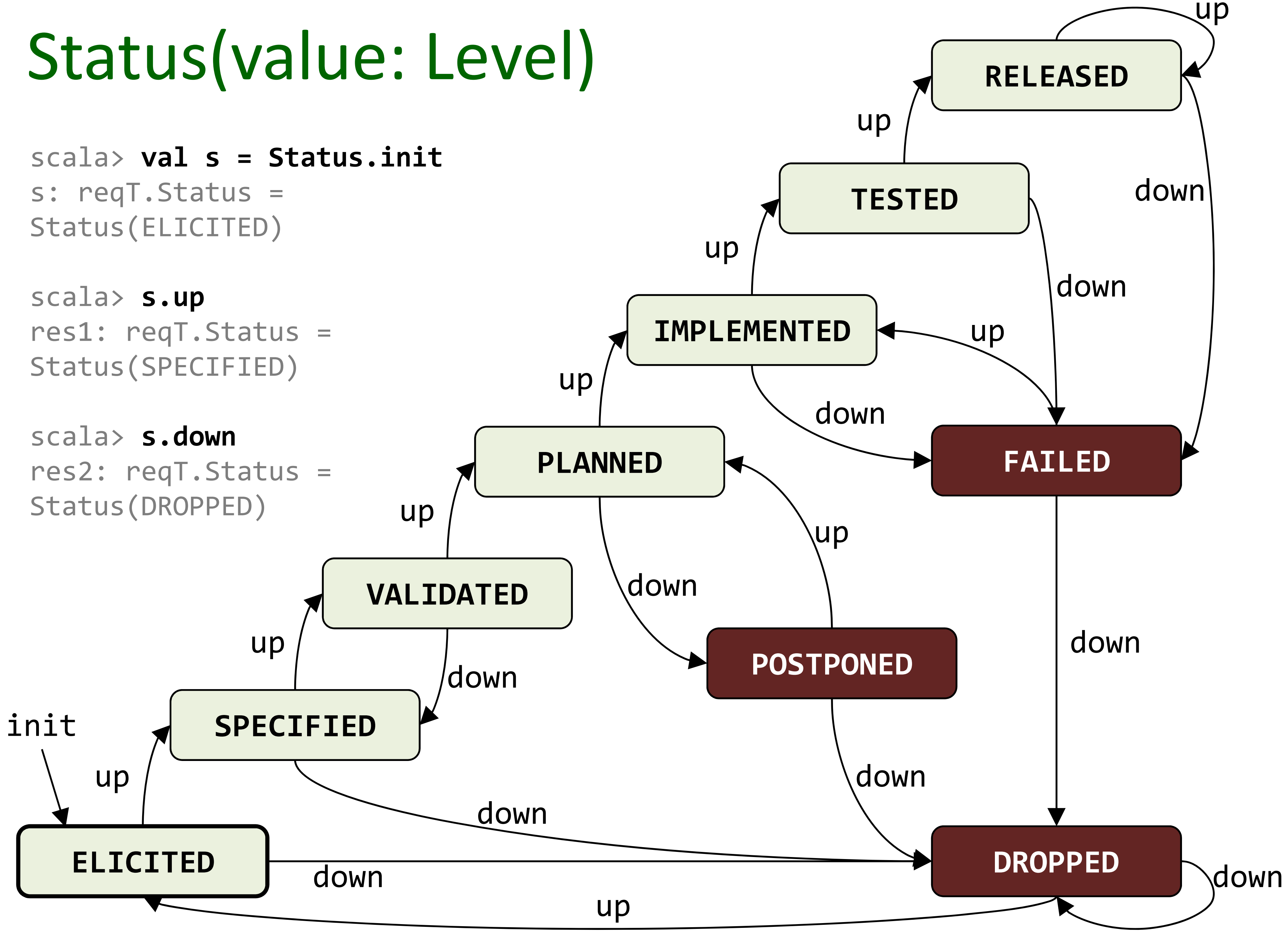


Status(value: Level)

```
scala> val s = Status.init
s: reqT.Status =
Status(ELICITED)

scala> s.up
res1: reqT.Status =
Status(SPECIFIED)

scala> s.down
res2: reqT.Status =
Status(DROPPED)
```



reqT Task description example

```
Model(
  Task("reception work") owns (Task("check in"), Task("booking")),
  Task("check in") has (
    Why("Give guest a room. Mark it as occupied. Start account."),
    Trigger("A guest arrives"),
    Frequency("Average 0.5 checkins/room/day"),
    Critical("Group tour with 50 guests.")
  ),
  Task("check in") owns (
    Task("find room"), Task("record guest"), Task("deliver key")),
  Task("record guest") has Spec(
    "variants: a) Guest has booked in advance, b) No suitable room"
  )
)
```

[Example modified from Lauesen: "Software Requirements – Styles and Techniques"]

Features of reqT-v2.2 publ. @ REFSQ13

- 2nd generation of DSL based on student feedback
- Deep integration with Scala collections
- A rich set of operators and methods for:
  - extracting models parts (restrict, exclude, DFS, ...)
  - finding elements of models
  - updating and analyzing models
- Import/Export
  - tabsep for integration with spreadsheet programs
  - template-based HTML requirements doc generator

Features of reqT-v2.3

- New experimental features in v2.3-RC1
  - Constraints on models, inside models
  - Integration with constraint solver JaCoP
    - prioritization
    - release planning
  - "Deep models" using recursive structures
    - Submodel as attribute of any entity
    - Modularization of models in subdomains
    - ModelVector, ModelFiles
  - Executable test cases as requirements in models

Themes planned for future releases

- GUI Editor & Visualizer
- GUI for Prioritization & Release Planning
- NLP Support
- Git Integration & History Analyzer
- Semantic checks as plugins
- …

Interested in trying out or contributing to reqT.org?

- Download at http://reqT.org
- Contact bjorn.regnell@cs.lth.se
- Clone https://github.com/reqT/reqT
- Pull-requests are welcome!
- Seeking strategic partnerships with research groups that have competence in e.g.
  NLP, PLE, SPM, GORE, ...