Model-based testing

November2017

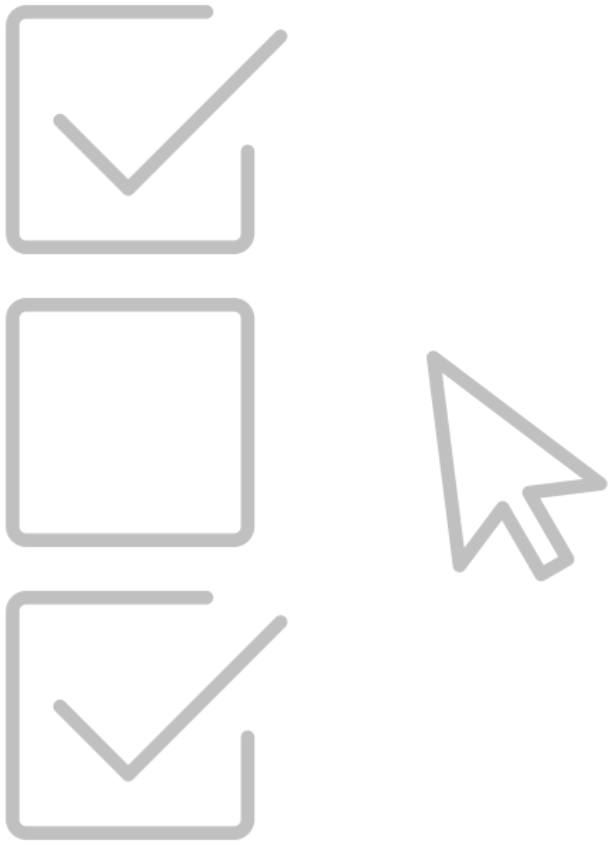
www.axini.com


axini

Overview

axini

- Who am I?
- Why MBT?
- What is MBT?
- MBT theory
- Conclusion



What do you want to hear?

axini

Comparing levels of test automation

axini

Development steps	Manual	Script	MBT
Create specification	✋	✋	✋
Interpret specification	✋	✋	✋
Create model	–	–	✋
Create test	✋	✋	✓
Predict outcome	✋	✋	✓
Script test	–	✋	✓
Execute test	✋	✓	✓
Check outcome	✋	✓	✓

✋ manual step

✓ automated step

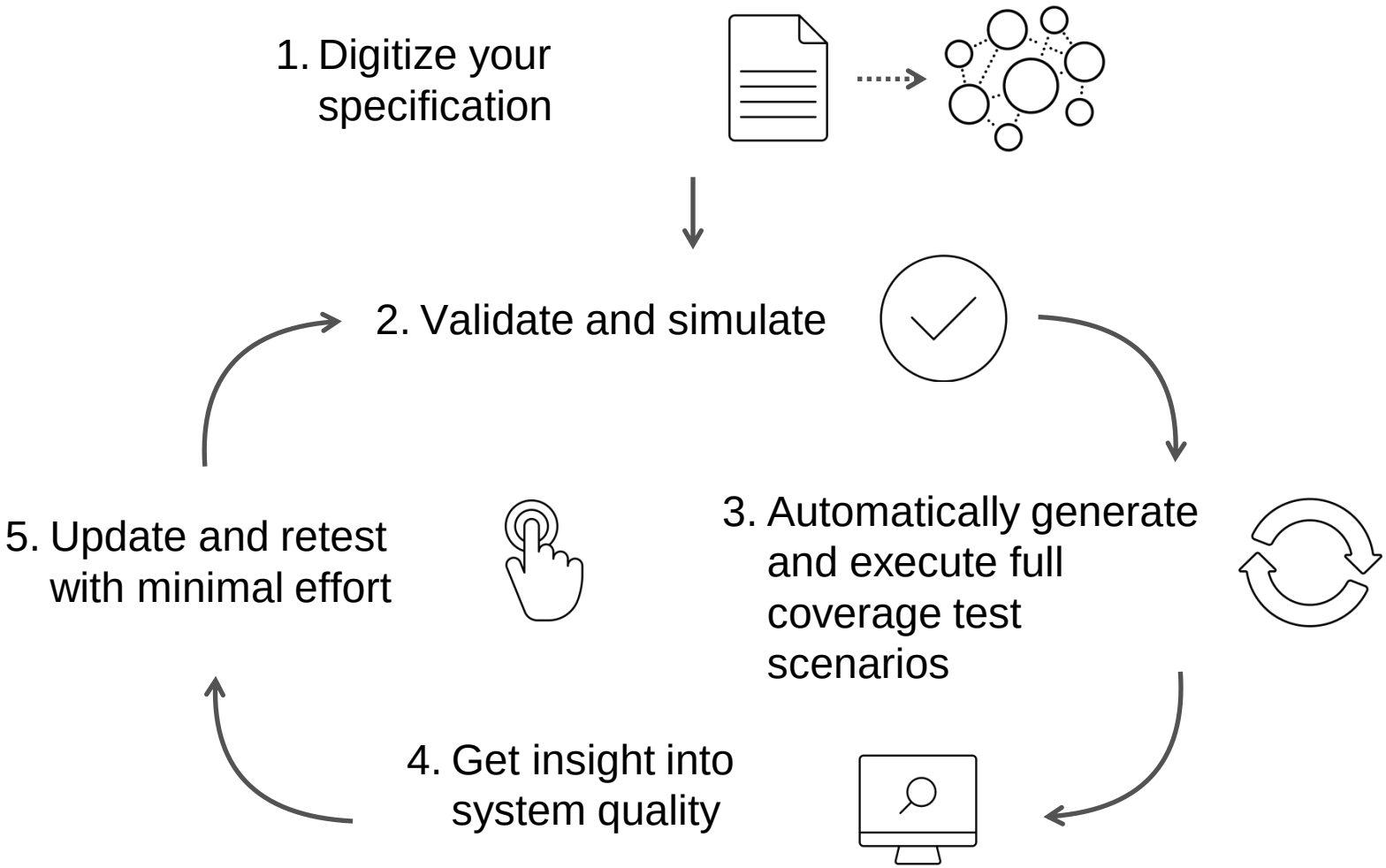
✋ use the model as specification

100% automation

More coverage = More certainty

Axini MBT in a nutshell

axini



Introducing Axini

axini

- Goal: industrialize model-based testing (MBT) as a highly-rewarding step towards model-based engineering
- Foundation: 25+ years of R&D
- Result-driven and fully funded by commercial MBT
- Proven technology since 2007

Customer base and motivation

axini

✓ Shorten release cycles, increase predictability

✓ Prevent production issues

✓ Reduce TCO

Several high tech companies

- Technical interfaces and protocols
- Ease system integration: single truth for all parties
- Cover timing, parallelism, robustness, bad weather

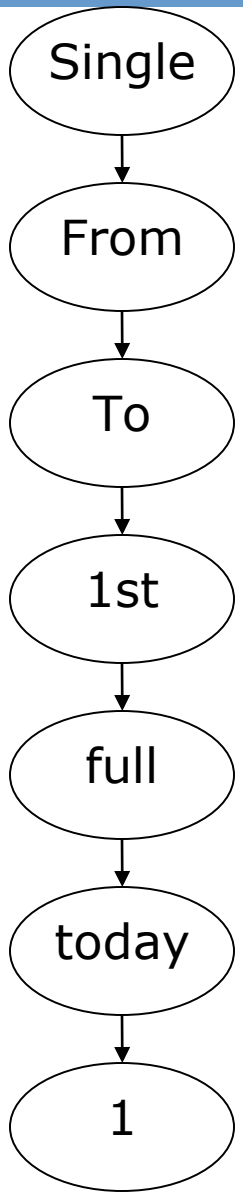
Top-3 bank, top-3 insurer

- Complex business logic with large data sets
- Cover unique situations, find hard to detect errors
- Simulate changes before implementing them

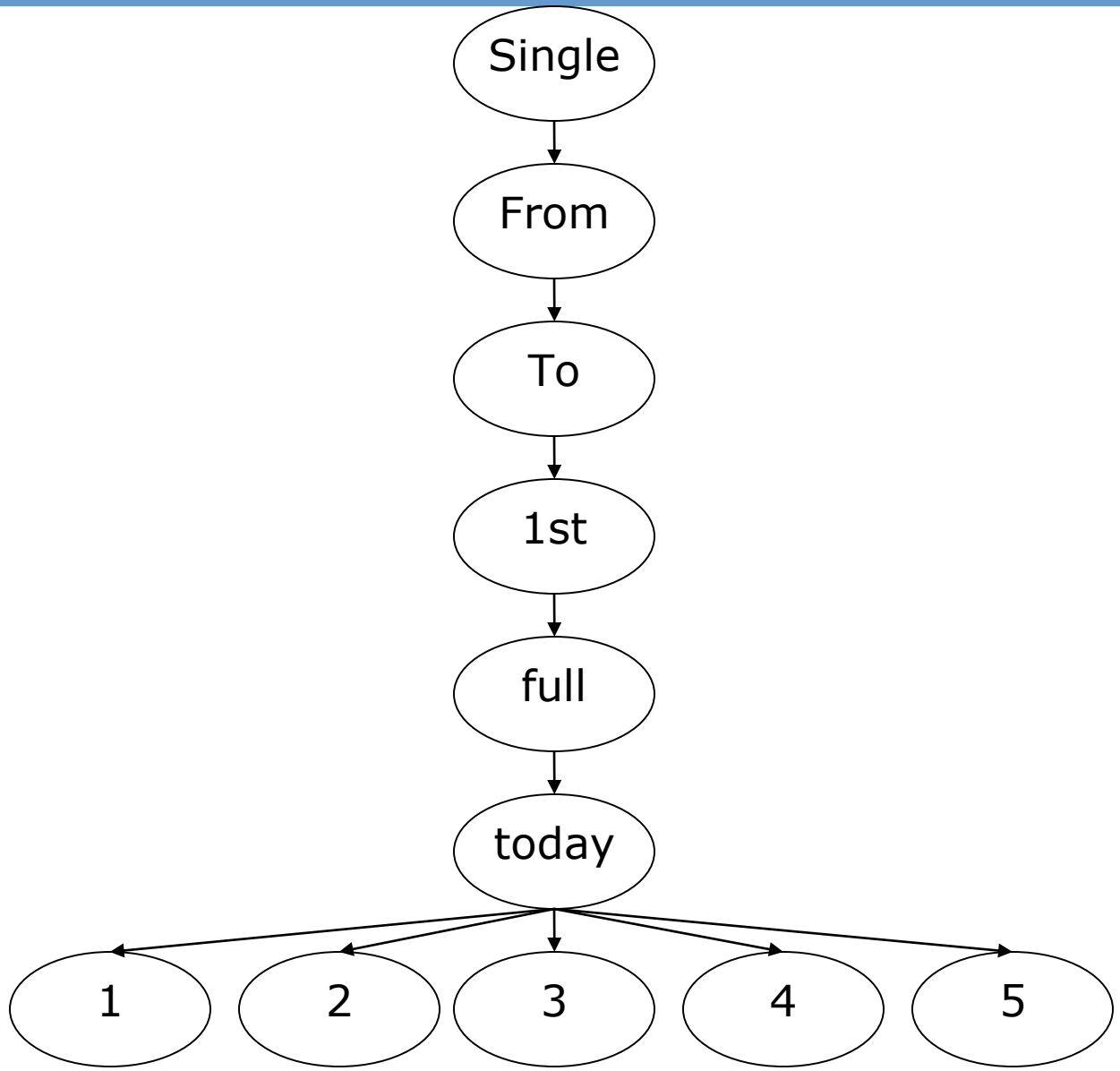

Single
From:
Rotterdam Centraal
1st class
Full fare
Valid today
1 ticket
Day Return
To:
Amsterdam Centraal
2nd class
Discount
Open date
2 tickets
5 Return ticket
To change route: press a white box above.
3 tickets
Weekend Return
4 tickets
Railrunner
4-11 (incl.) years
Select number of tickets
Other tickets
'Via' station
Nederlands
English
Stop
Clear all

Test case 1

axini

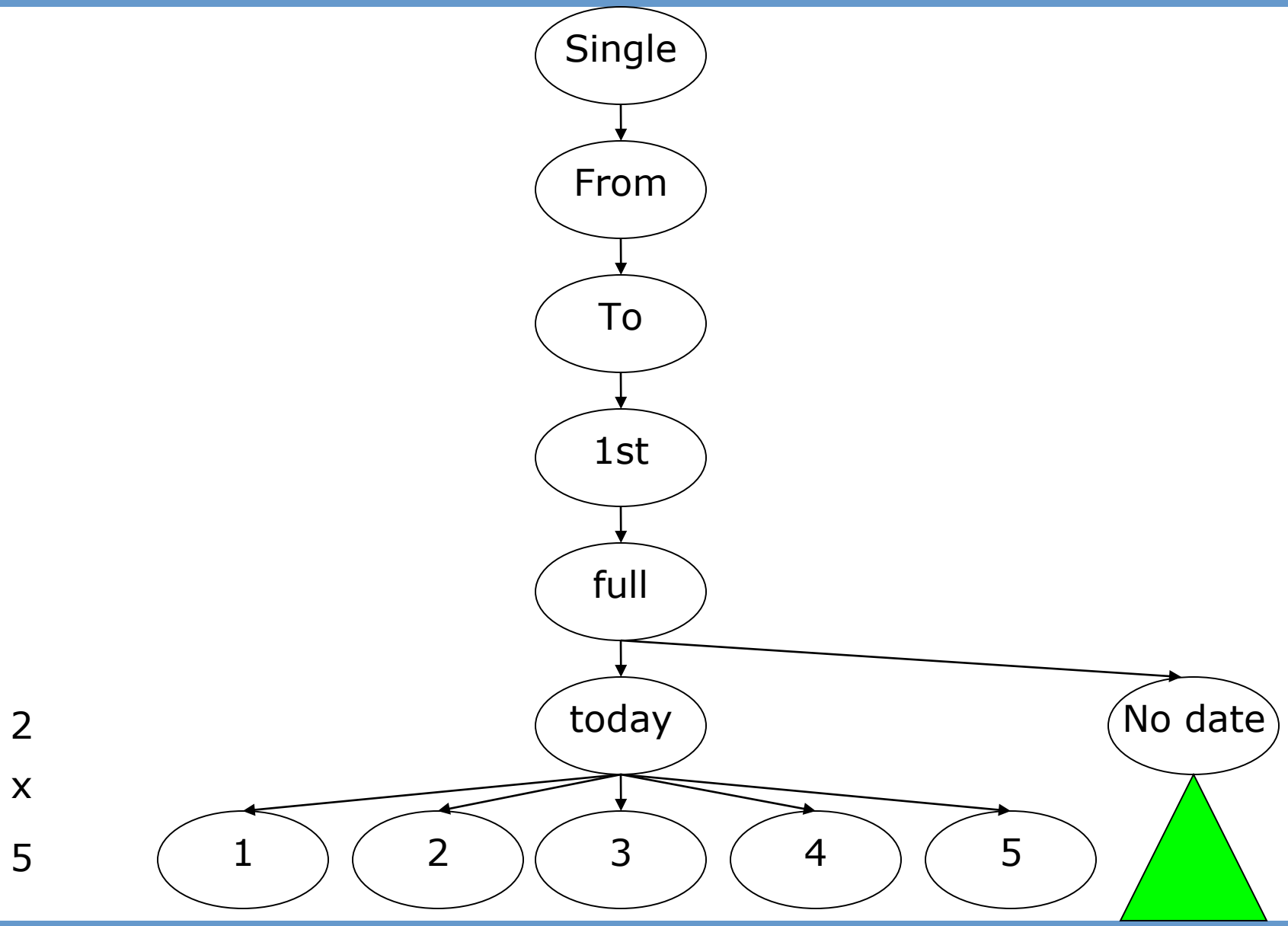


Test case 1-5



Test case 1-10

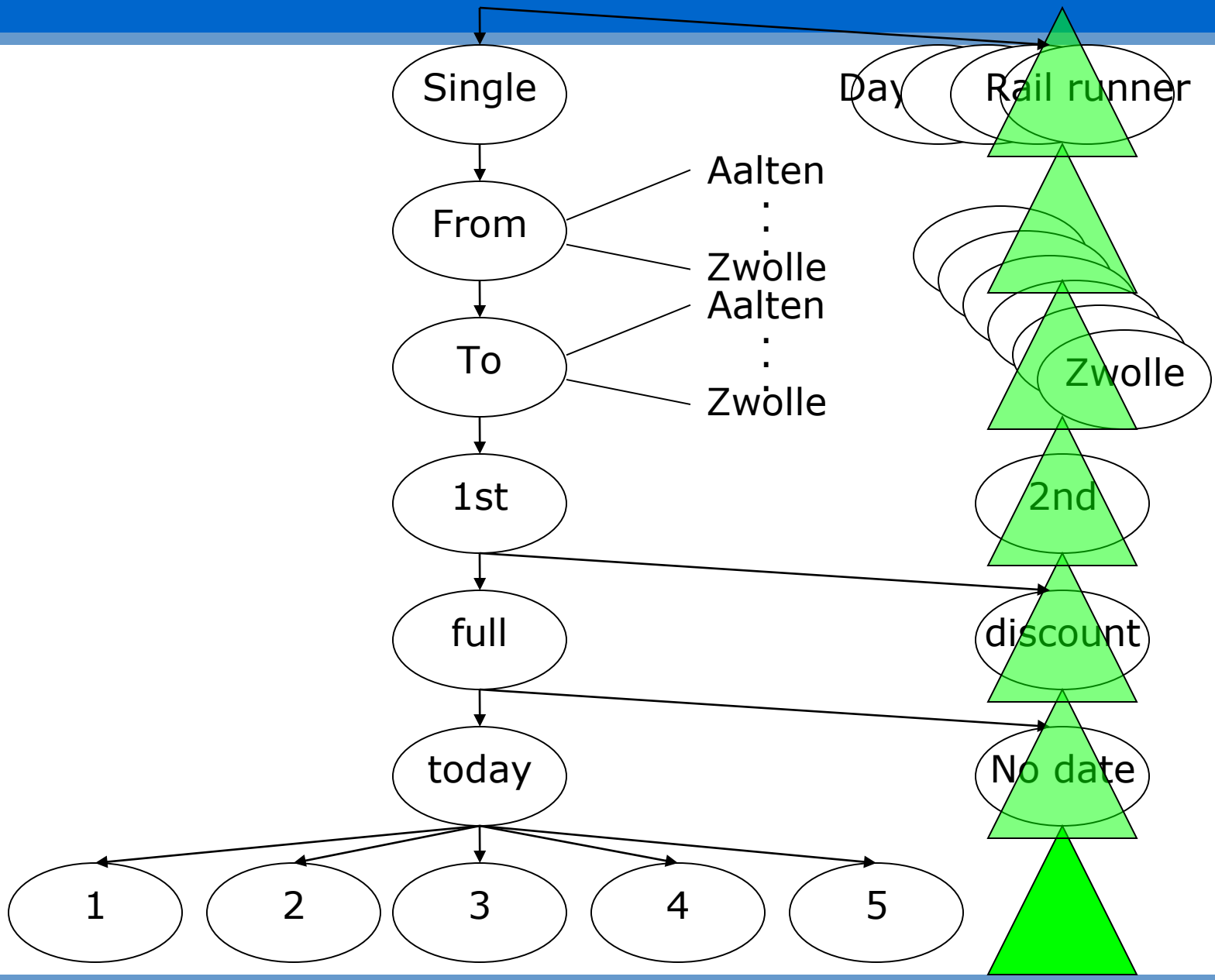
axini



Test case 1-30732800

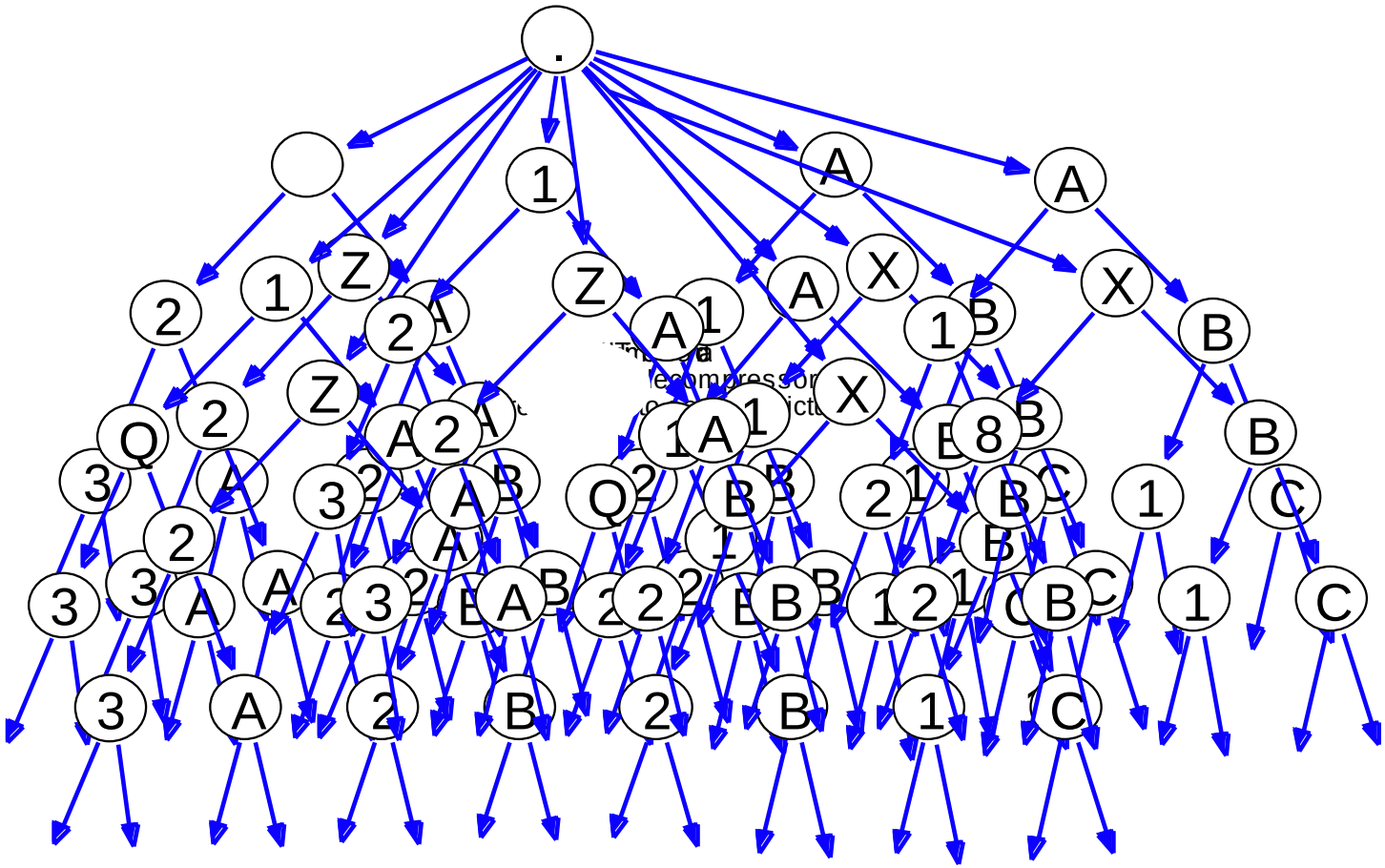
axini

5
x
392
x
392
x
2
x
2
x
2
x
5
= 30732800



Combinatorial explosion

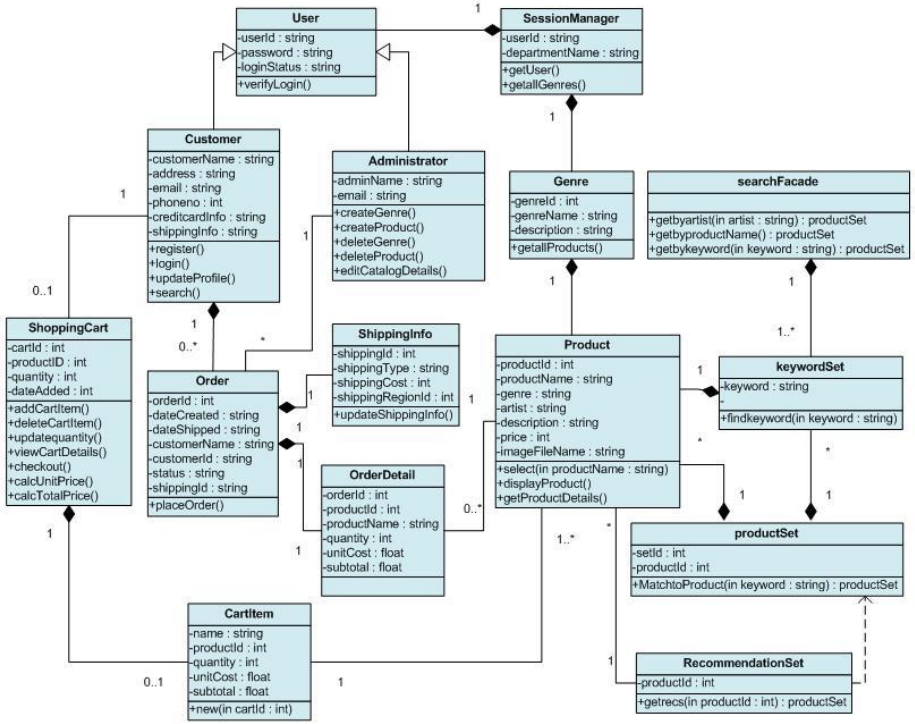
axini



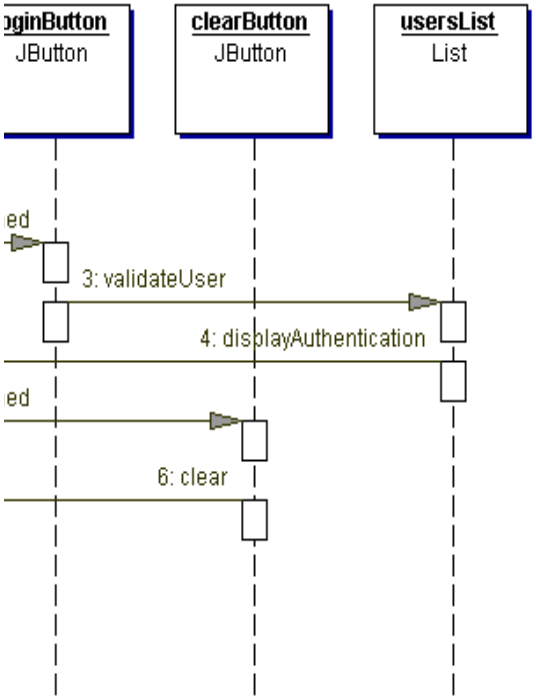
The trouble with testing

axini

Data



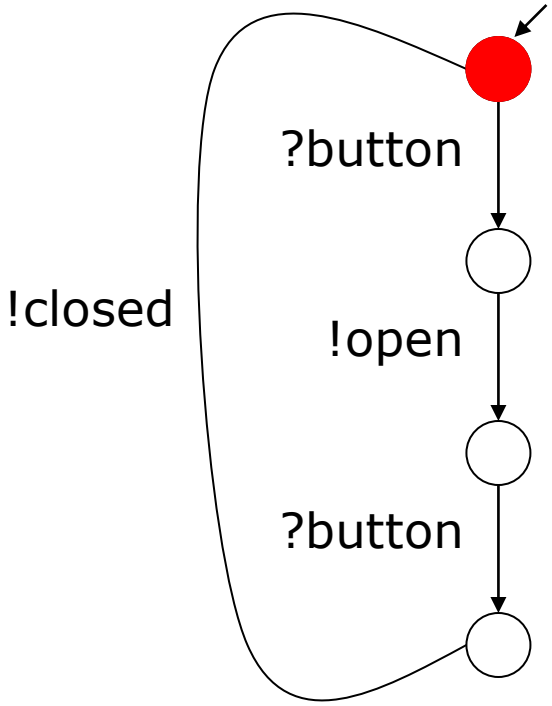
Interaction



INFINITY INFINITY

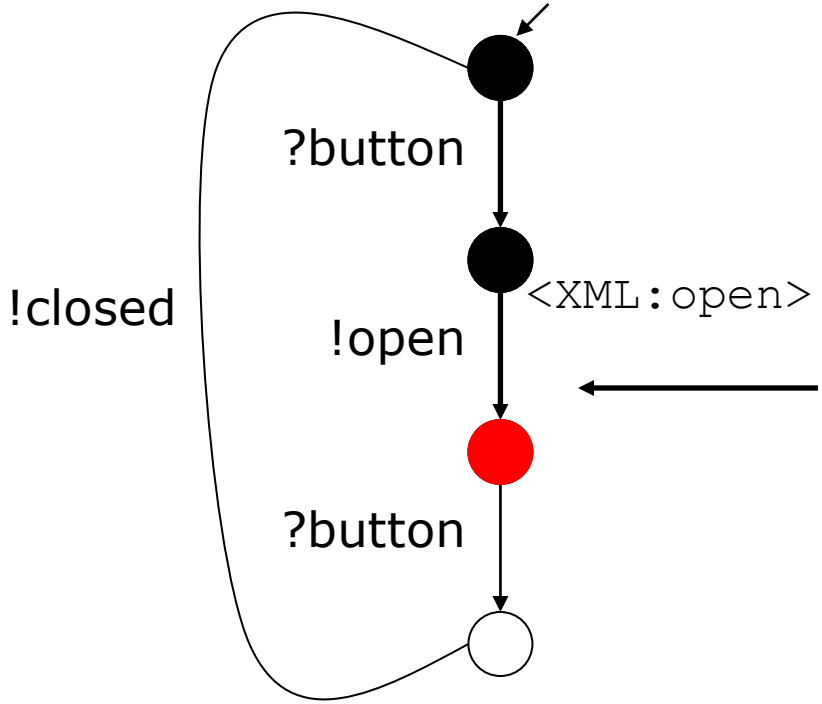
Smart-lock

axini

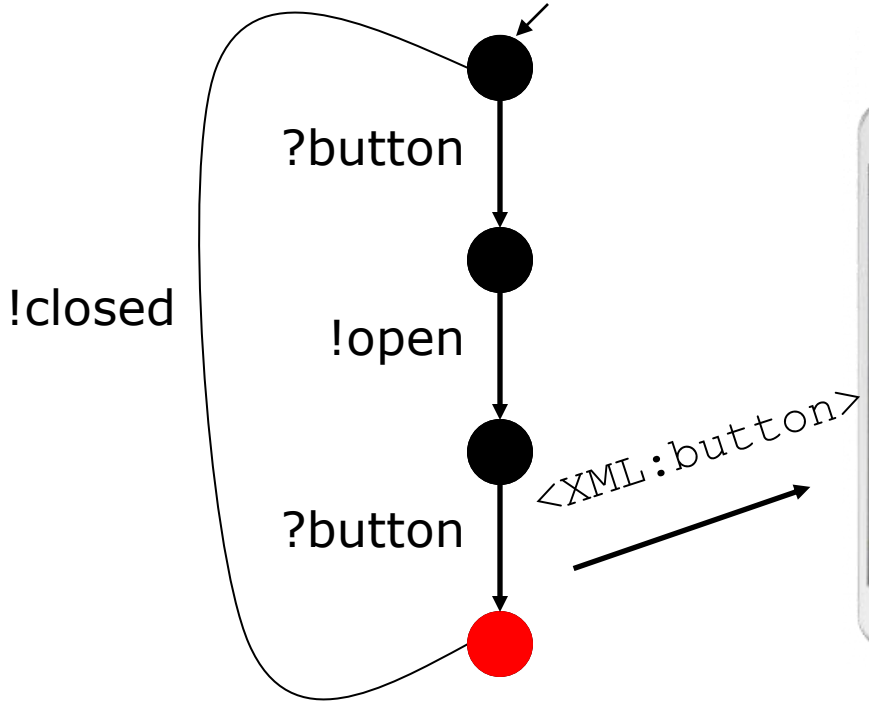

?button
!closed
!open
?button


OK DOKEYS
Bluetooth
SMART

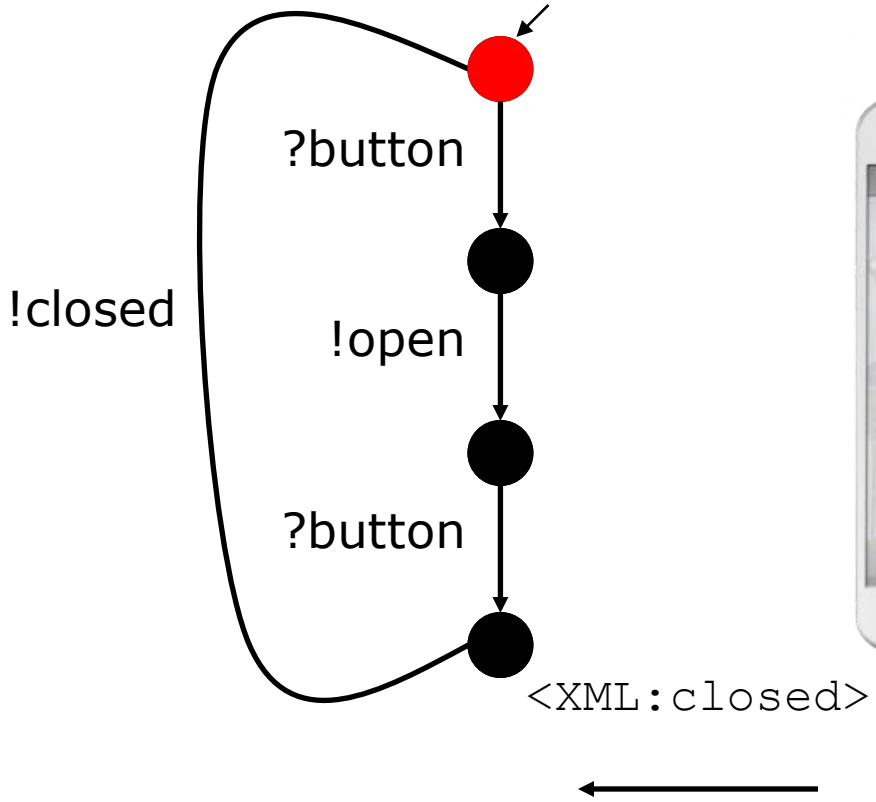

?button
!closed
!open
?button
<XML:button>
OK DOKEYS
Bluetooth
SMART


?button
!closed
<XML:open>
!open
?button


OK DOKEYS
Bluetooth
SMART


!closed
?button
!open
?button
<XML:button>


OK DOKEYS
Bluetooth
SMART

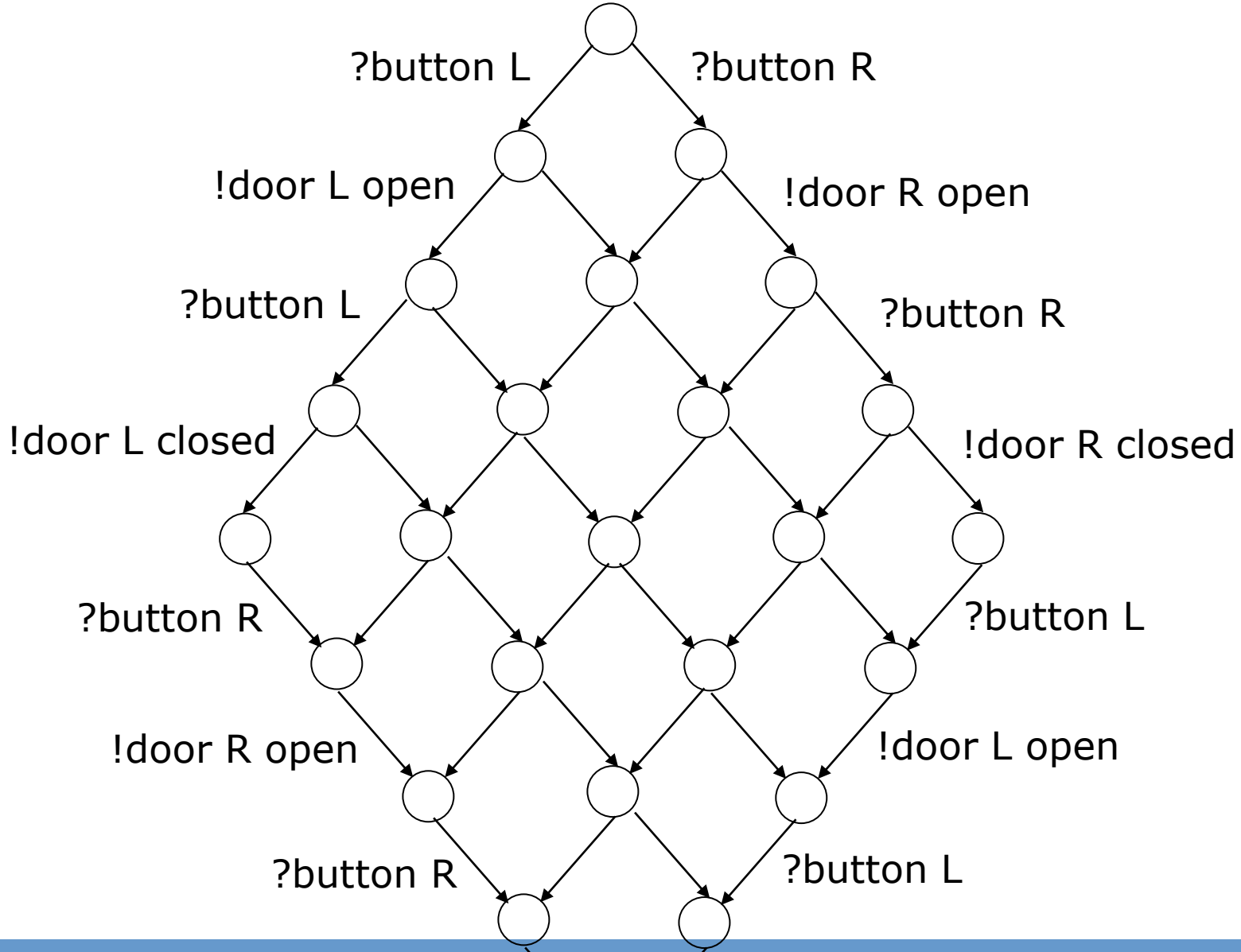

?button
!closed
!open
?button
<XML:closed>


OK DOKEYS
Bluetooth
SMART


OK DOKEYS
Bluetooth
SMART
OK DOKEYS
Bluetooth
SMART

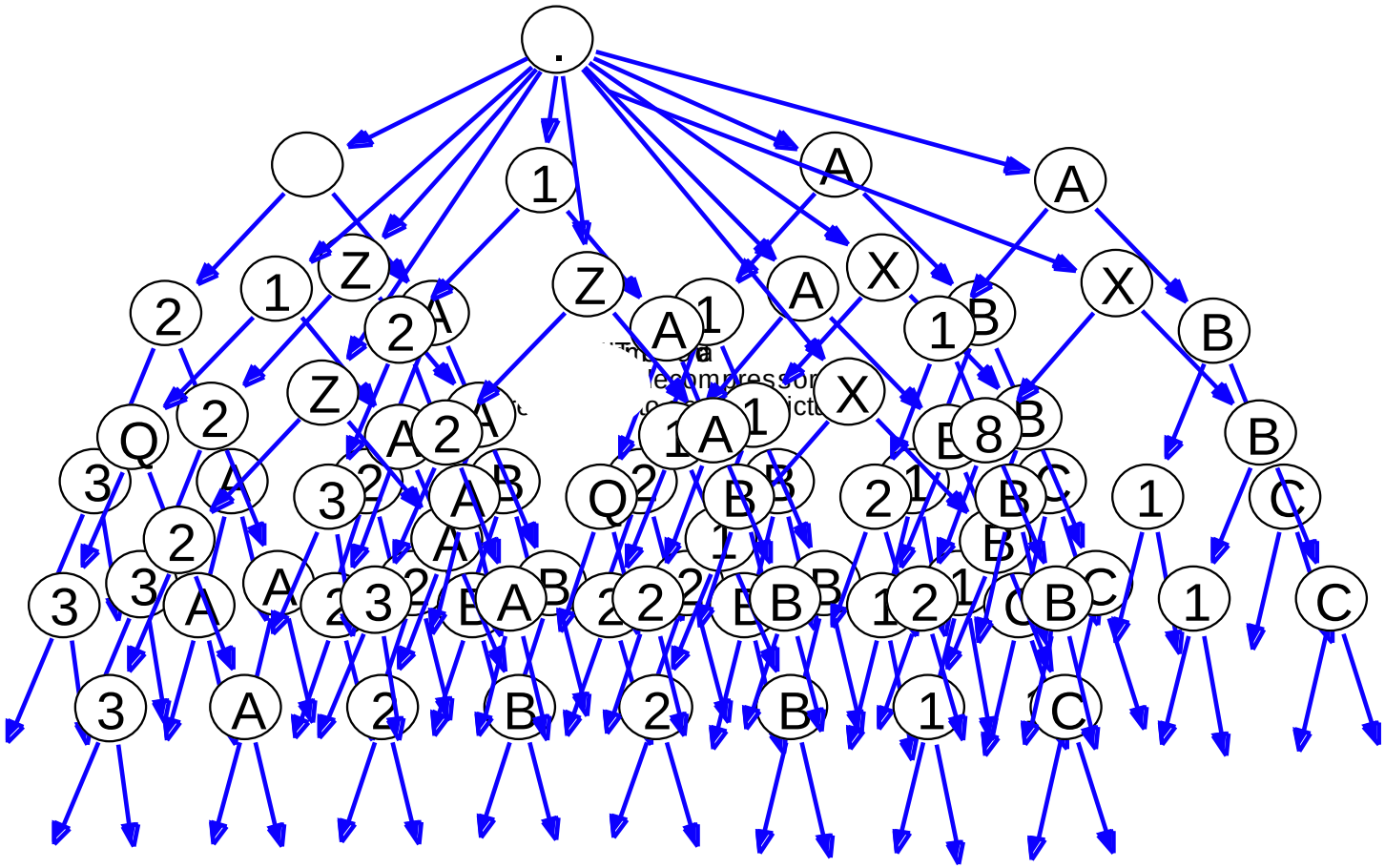
Testing parallel components

axini



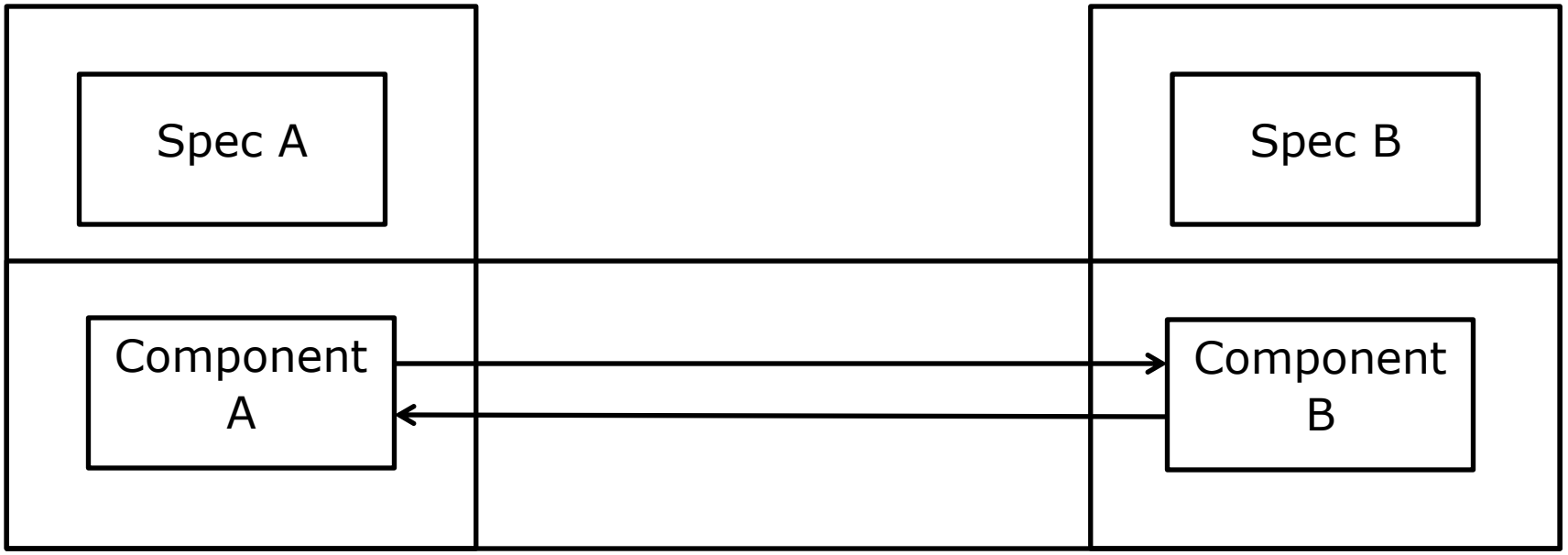
Combinatorial explosion

axini



Theory: compositionality

axini



Compositionality - theory

axini

- When you specify your components precisely
- And you test the components thoroughly in isolation
- Then you do not have to test the integration of the components

Compositionality - theory

axini

- When you specify your components **precisely**
- And you test the components **thoroughly** in isolation
- Then you do not have to test the integration of the components

Modeling in practice

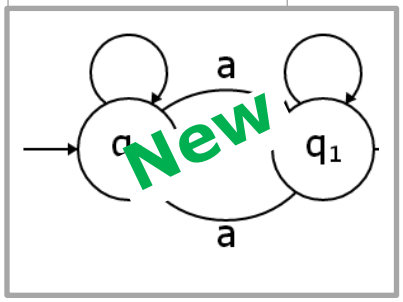
axini

Input:
Paper specifications

Missing knowledge

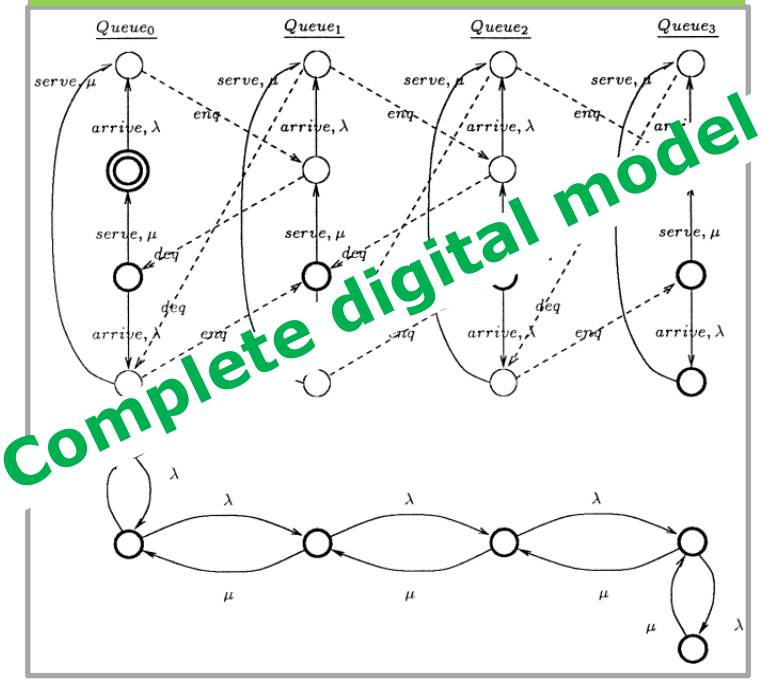
Step 1:
Combine and structure

interface
workflow
behavior
entity



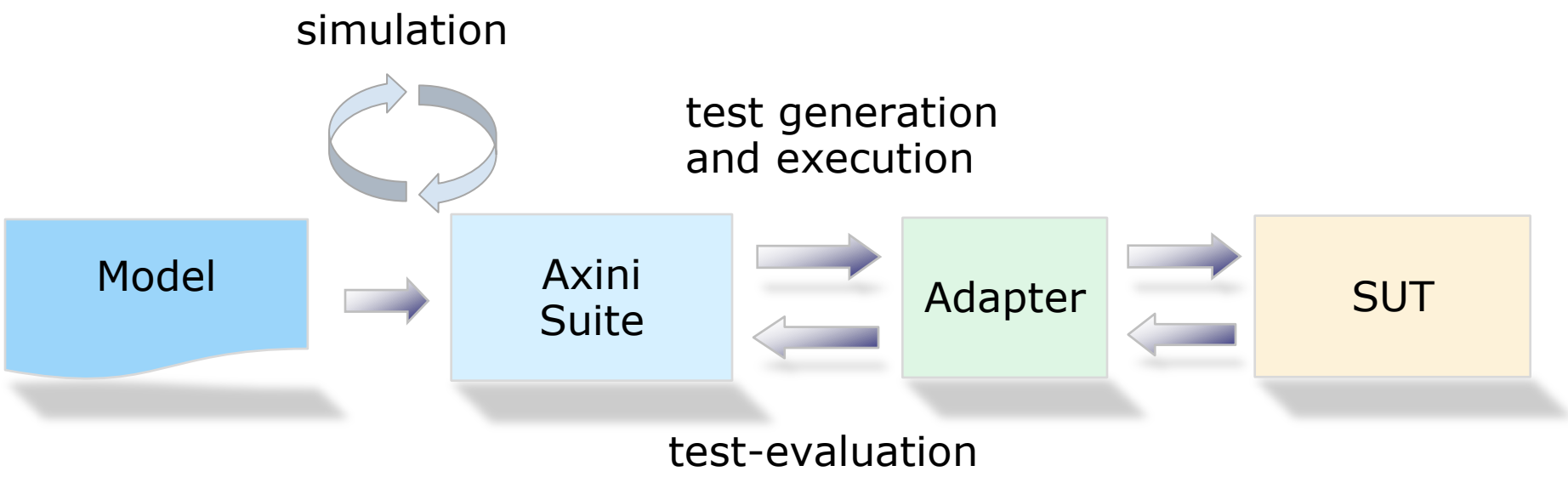
Step 2:
Add missing information
Add bad/sad weather

interface
workflow
behavior
entity



MBT ingredients

axini

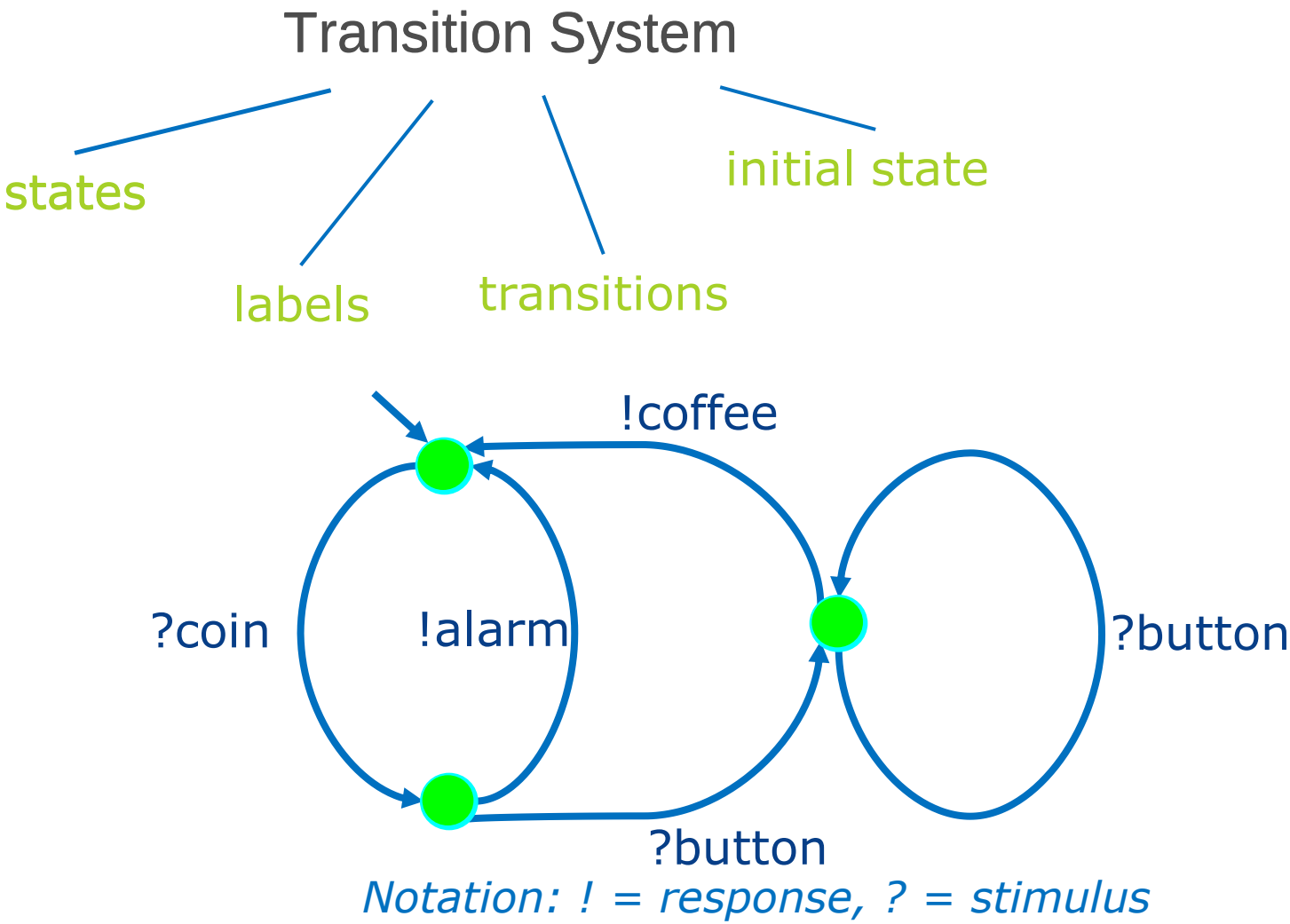


Questions?

axini

Transition Systems

axini



Notation: ! = response, ? = stimulus

loco, notion of correctness

axini

$out(s) = \{ \lambda \varepsilon \quad U_\delta \mid s \text{ –}\lambda\text{->} \}$

$(s \text{ after } \sigma) = \{ s' \mid s =\sigma=>s' \}$

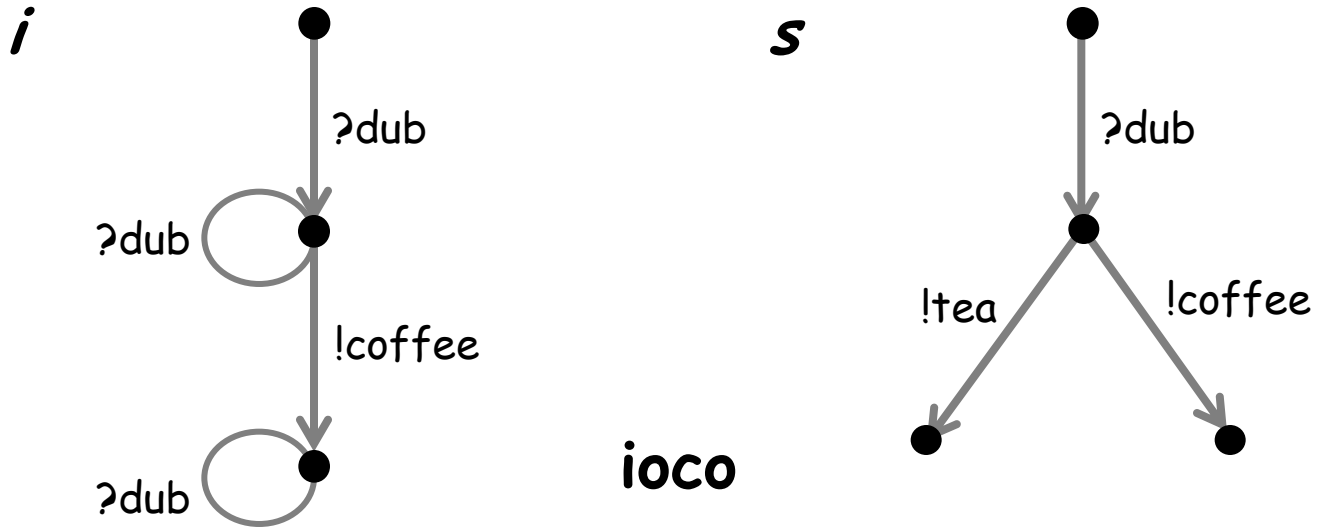
$Straces(s) = \{ \sigma \varepsilon L^*_\delta \mid s =\sigma=>s' \}$

$\forall \sigma \in Straces(s)$: $out(i \text{ after } \sigma) \subseteq out(s \text{ after } \sigma)$

Implementation Relation ioco (sheets from Jan Tretmans)

axini

$$i \textbf{ ioco } s \;=_{def}\; \forall \sigma \in \mathit{Straces}(s) : \mathit{out}(i \textbf{ after } \sigma) \subseteq \mathit{out}(s \textbf{ after } \sigma)$$



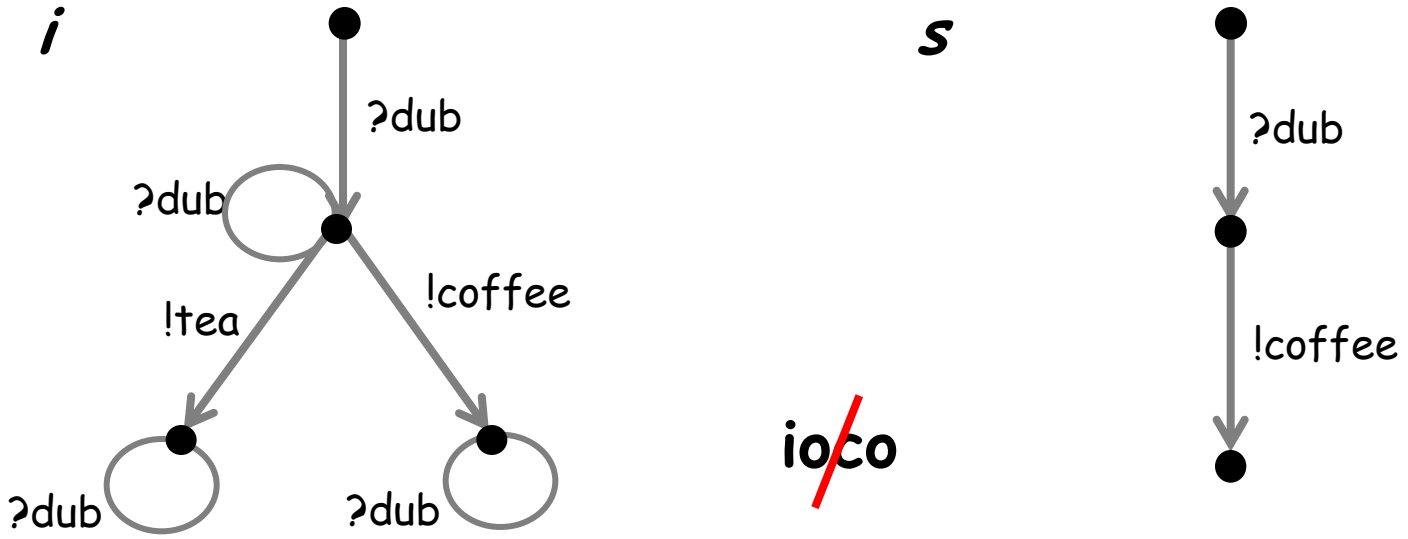
$\mathit{out}(i \textbf{ after } ?dub) = \{ !coffee \}$

$\mathit{out}(s \textbf{ after } ?dub) = \{ !coffee, !tea \}$

Implementation Relation ioco

axini

$$i \textbf{ ioco } s \;=_{def}\; \forall \sigma \in \mathit{Straces}(s) : \mathit{out}(i \textbf{ after } \sigma) \subseteq \mathit{out}(s \textbf{ after } \sigma)$$

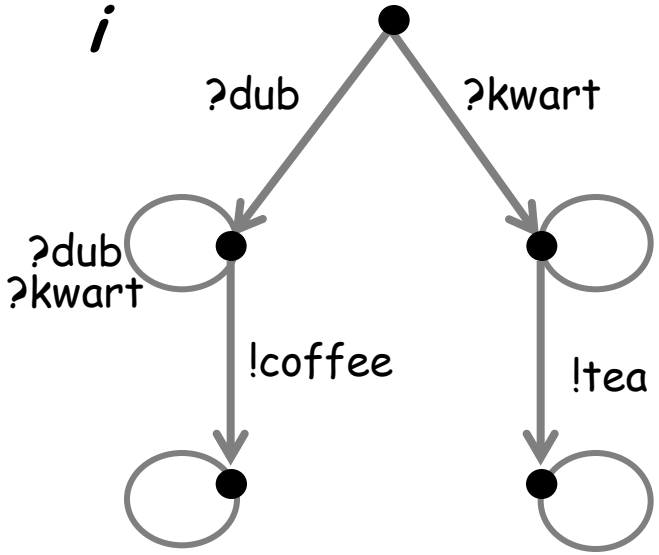


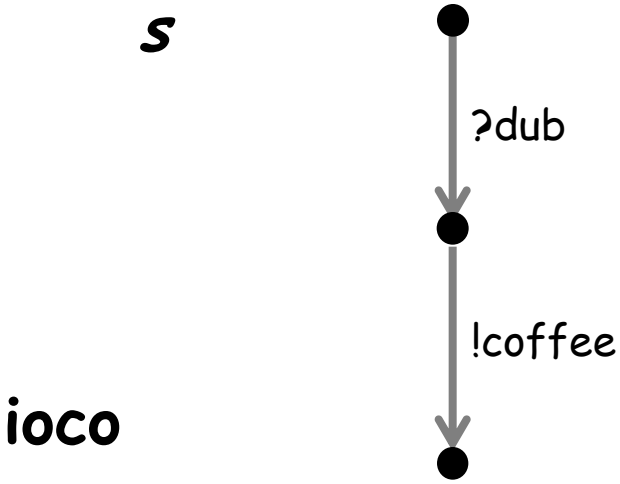
$\mathit{out}(i \textbf{ after } ?dub) = \{ !coffee, !tea \} \not\subseteq \mathit{out}(s \textbf{ after } ?dub) = \{ !coffee \}$

Implementation Relation ioco

axini

$$i \textbf{ ioco } s \;=_{def}\; \forall \sigma \in \mathit{Straces}(s) : \mathit{out}(i \textbf{ after } \sigma) \subseteq \mathit{out}(s \textbf{ after } \sigma)$$





ioco

out (***i*** **after** ?dub) = { !coffee }

out (***i*** **after** ?kwart) = { !tea }

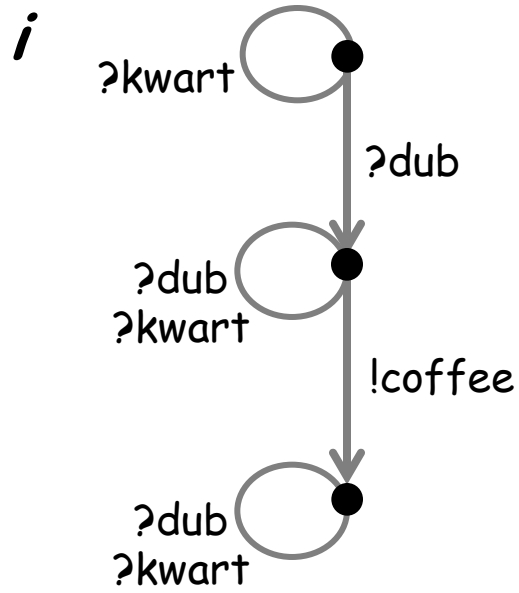
out (***s*** **after** ?dub) = { !coffee }

out (***s*** **after** ?kwart) = ∅

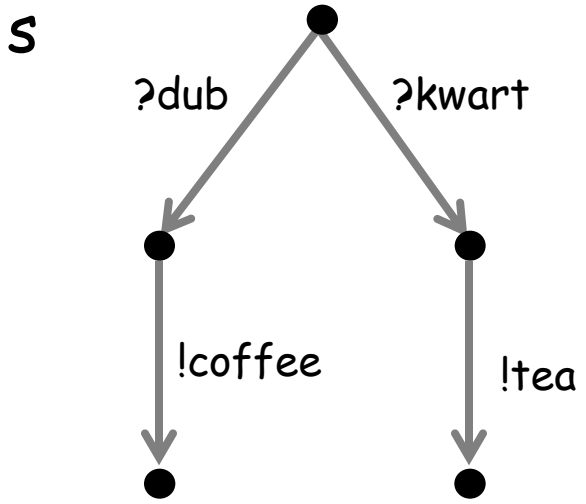
But ?kwart ∉ *Straces* (***s***)

Implementation Relation ioco

$$i \textbf{ ioco } s \;=_{def}\; \forall \sigma \in \mathit{Straces}(s) : \mathit{out}(i \textbf{ after } \sigma) \subseteq \mathit{out}(s \textbf{ after } \sigma)$$



ioco



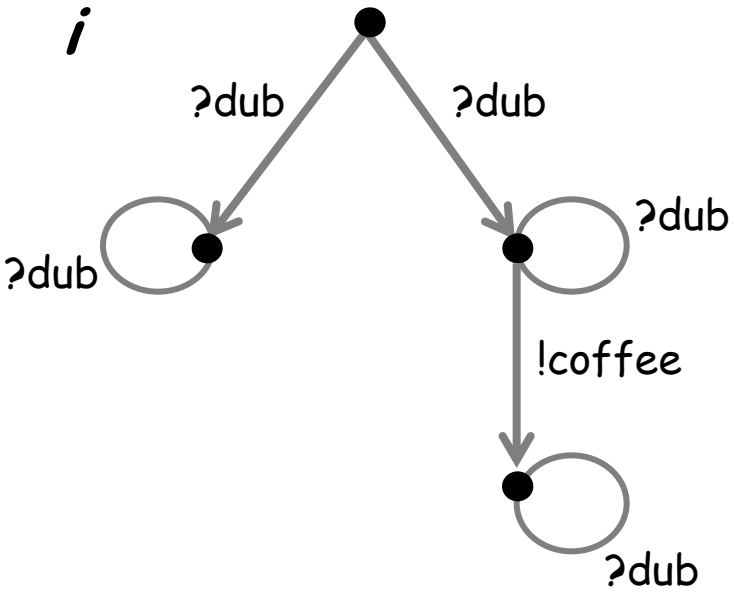
$\mathit{out}(i \textbf{ after } ?kwart) = \{ \delta \}$

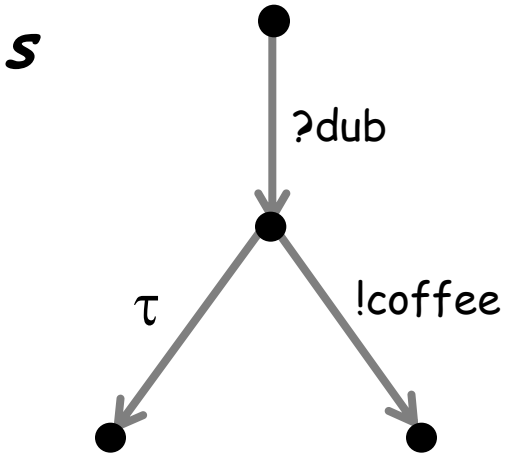
$\mathit{out}(s \textbf{ after } ?kwart) = \{ !tea \}$

Implementation Relation ioco

axini

$$i \textbf{ ioco } s \;=_{def}\; \forall \sigma \in \mathit{Straces}(s) : \mathit{out}(i \textbf{ after } \sigma) \subseteq \mathit{out}(s \textbf{ after } \sigma)$$





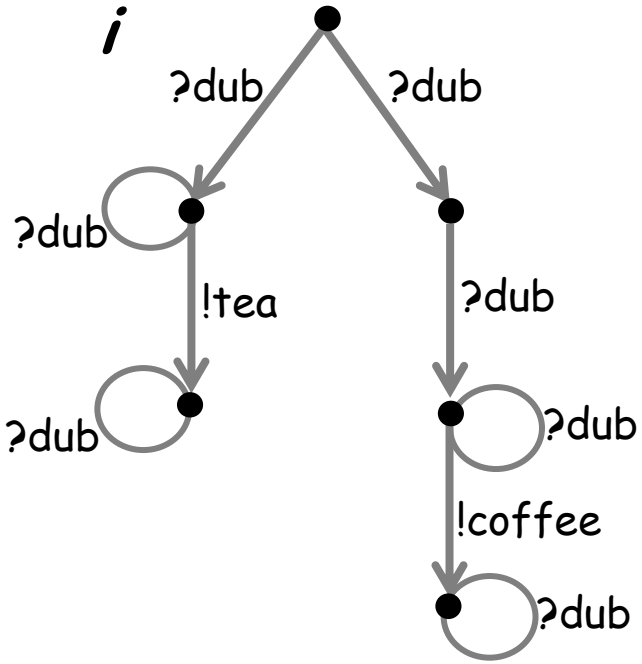
ioco

$\mathit{out}(\textbf{i after ?dub}) = \{ \delta, !\text{coffee} \}$

$\mathit{out}(\textbf{s after ?dub}) = \{ \delta, !\text{coffee} \}$

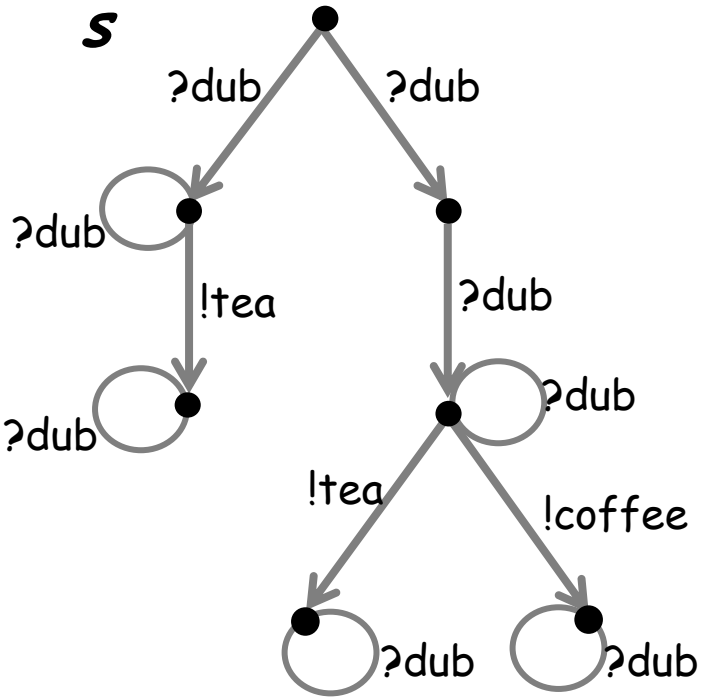
Implementation Relation ioco

i **ioco** s $=_{def}$ $\forall \sigma \in Straces(s) : out(i \textbf{ after } \sigma) \subseteq out(s \textbf{ after } \sigma)$



i **ioco** *s*

s **ioco** *i*



out (*i* **after** ?dub.?dub) = *out* (*s* **after** ?dub.?dub) = { !tea, !coffee }

out (*i* **after** ?dub.δ.?dub) = { !coffee } ≠ *out* (*s* **after** ?dub.δ.?dub) = { !tea, !coffee }

Our toolset

axini

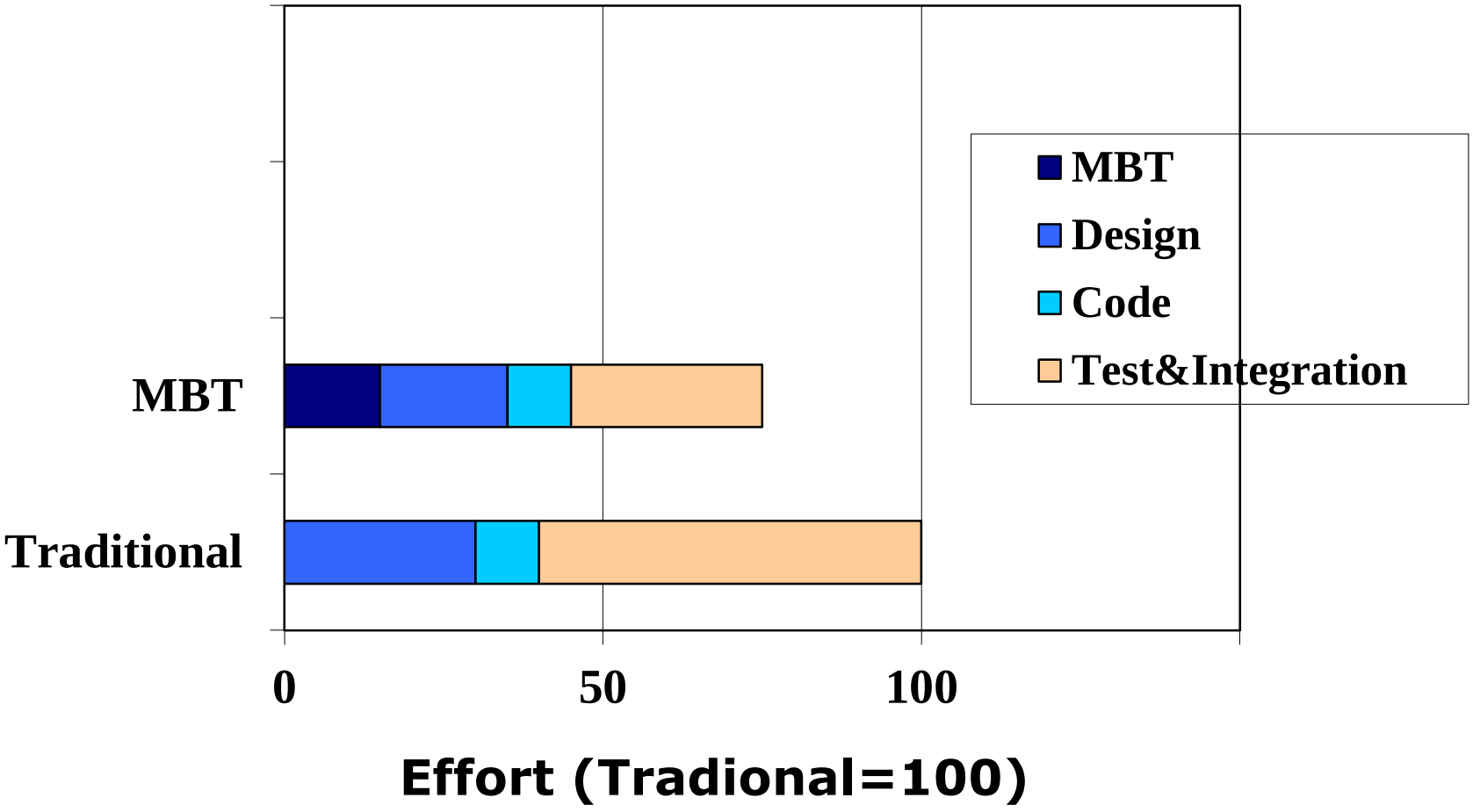
- Data
- Time
- Functions
- Parallelism
- Non-determinism
- On the fly and off line test-generation
- Test-generation strategies
- Model-checking/validation
- Test-case analysis

Questions?

axini

MBT: Effect on development

axini

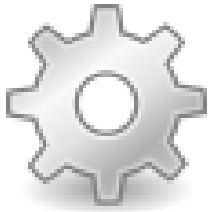


With MBT you find more bugs

axini

illustrative

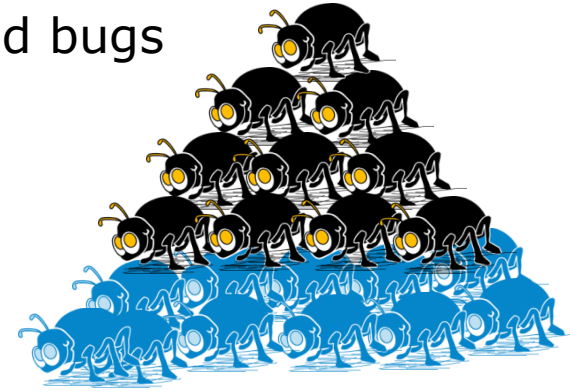
Development



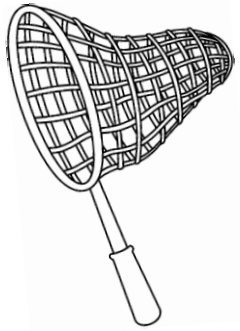
System test

Acceptance test

Production



Found bugs

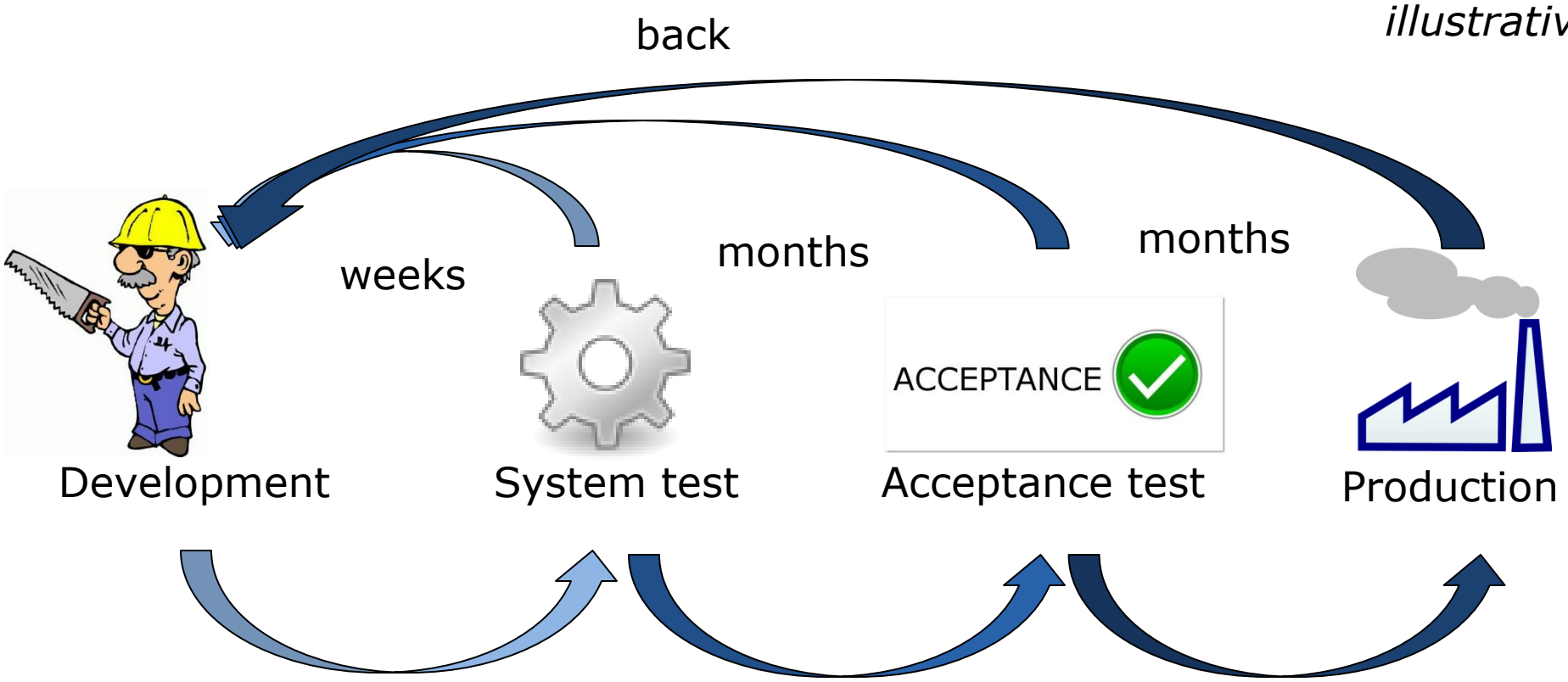


Bugs not found, but can be found with MBT

Without MBT: long lead time

axini

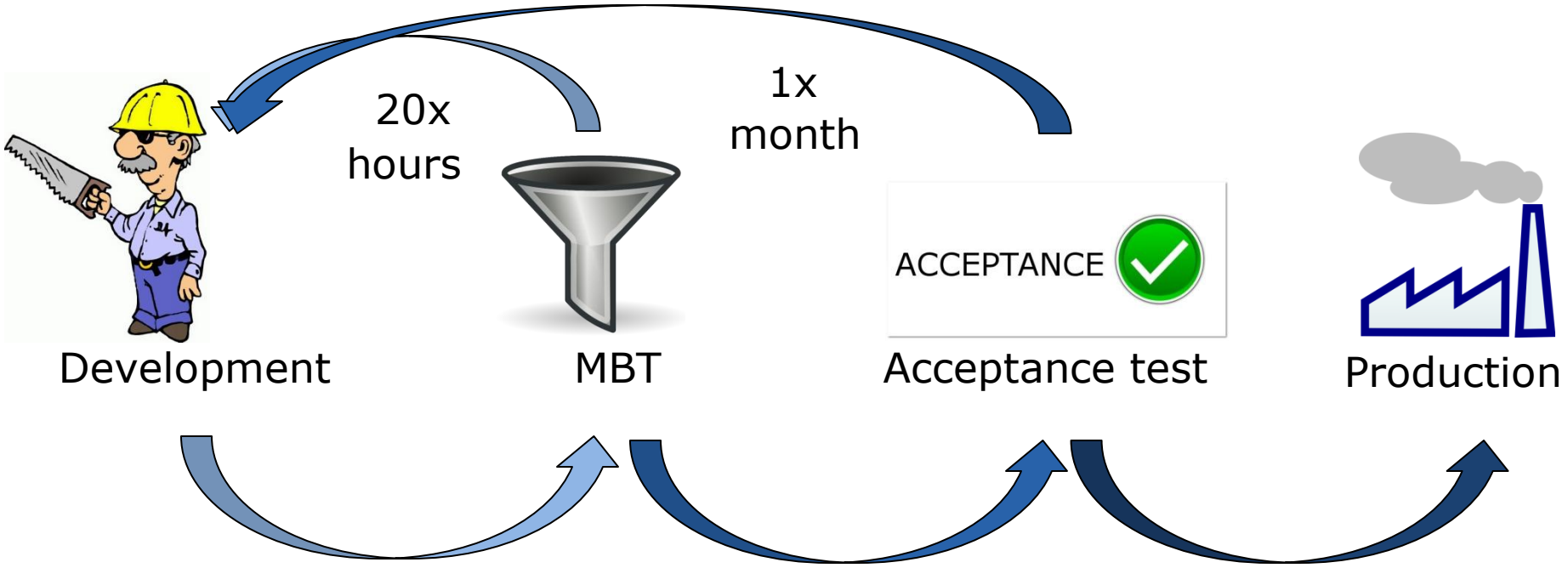
illustrative

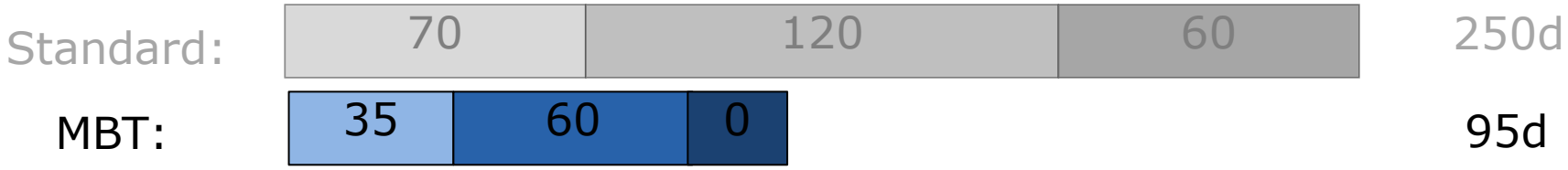


With MBT: short cycles, less rework

axini

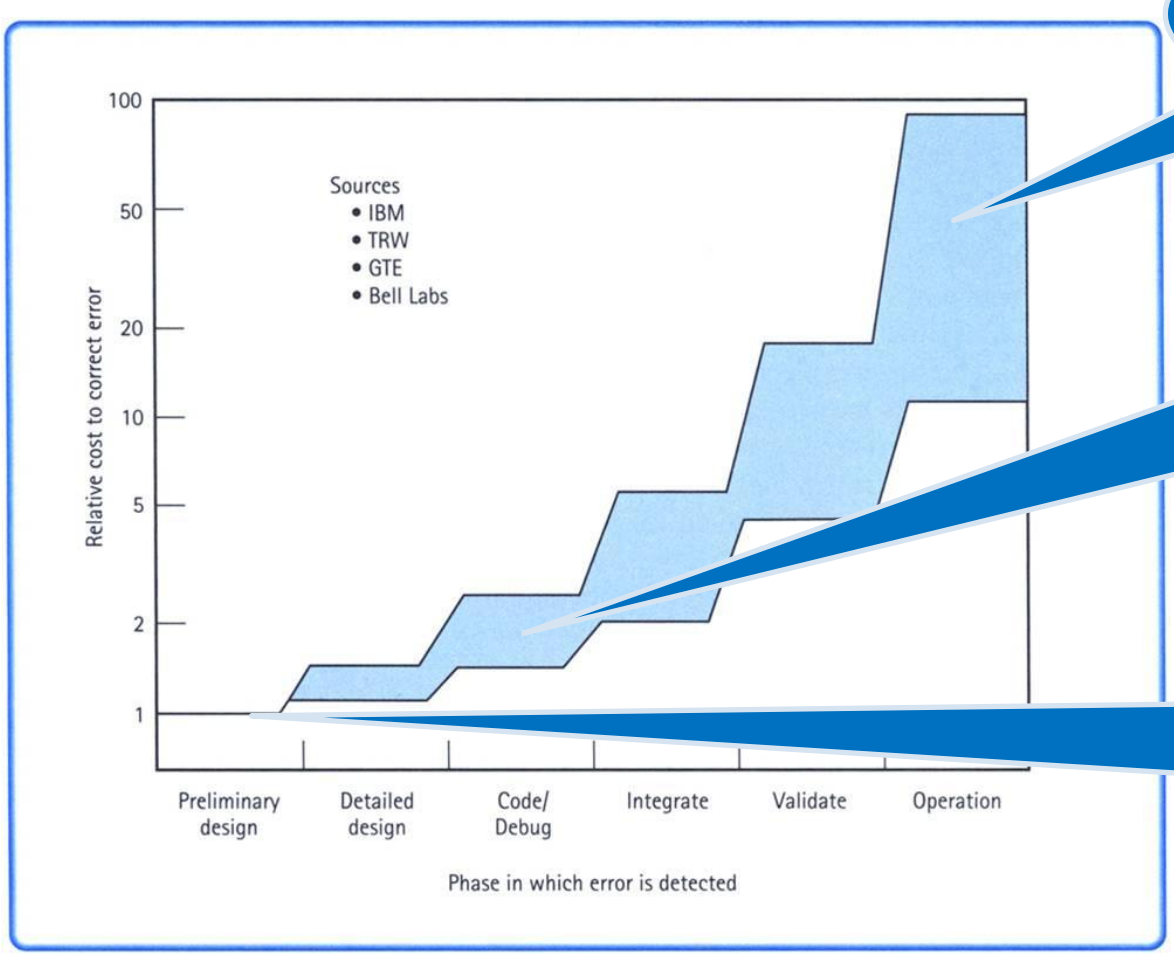
illustrative





Boehm: cost of errors

axini



Example: Production problems are most expensive to fix

Example: faults during coding phase are cheap (relatively) to fix

Example: faults in design are cheapest to fix

Effect of MBT

axini

- Early fault detection in specification
  - Modeling
  - Inspection
  - Simulation
- Early fault detection in implementation
  - Fast and thorough testing
- Ideal for
  - Agile testing, regression testing
  - Mission critical systems
  - Certification

Questions?

axini