Synthesizing Multi-View Models of Software Systems

Lambeau Bernard

ICTEAM Institute
Université catholique de Louvain
November 2011

Outline

- Introduction
  - A step-by-step explanation of the thesis title
- Model synthesis as a promising approach for reasoning about software systems
  - Interactive synthesis of state machines from scenarios
- Conclusion

Synthesizing Multi-view Models of Software Systems

Synthesizing Multi-view Models of Software Systems

Synthesizing Multi-view Models of **Software Systems**

Synthesizing Multi-view Models of **Software Systems**

with courtesy of quechoisir.fr

Synthesizing Multi-view Models of Software Systems

- A *system* is a set of active components, called *agents*, that interact so as to fulfill goals
- Agents restrict their behavior to meet the goals they are responsible for [Fea87, Avl09]
- Some agents are *software* components, *i.e.* automated agents
  - Others are human beings, electronic devices, etc.

Running example

Building software systems is hard

The solution is highly technical

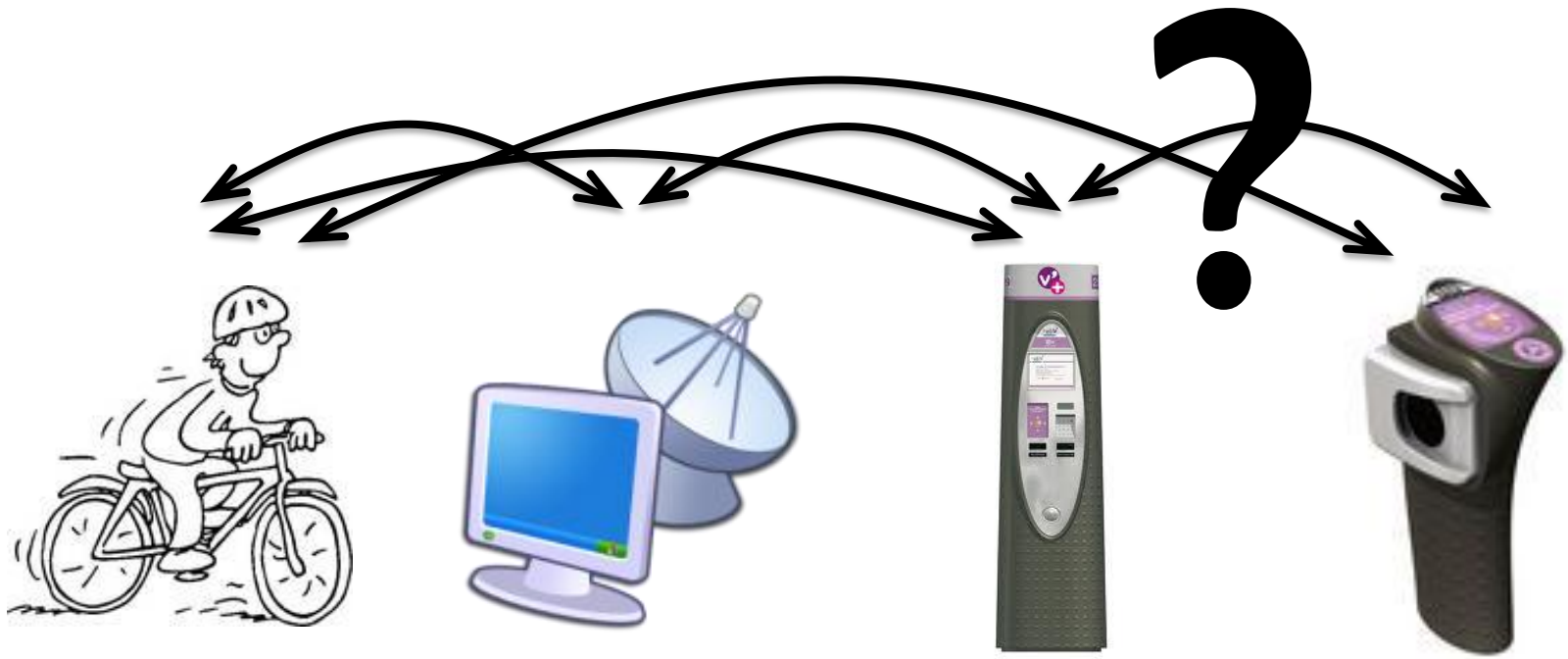
website

internet, mobile communications

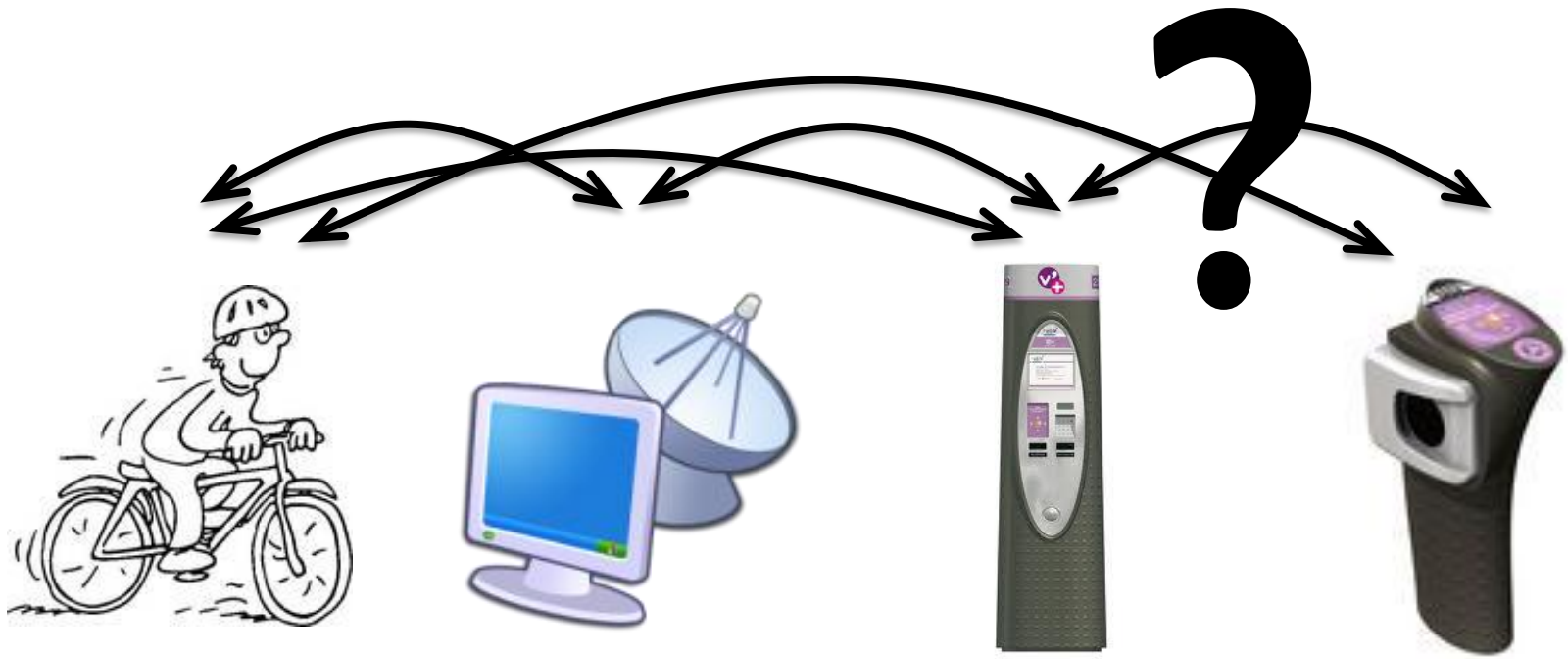
electro-mechanical devices

smart card, touch screen

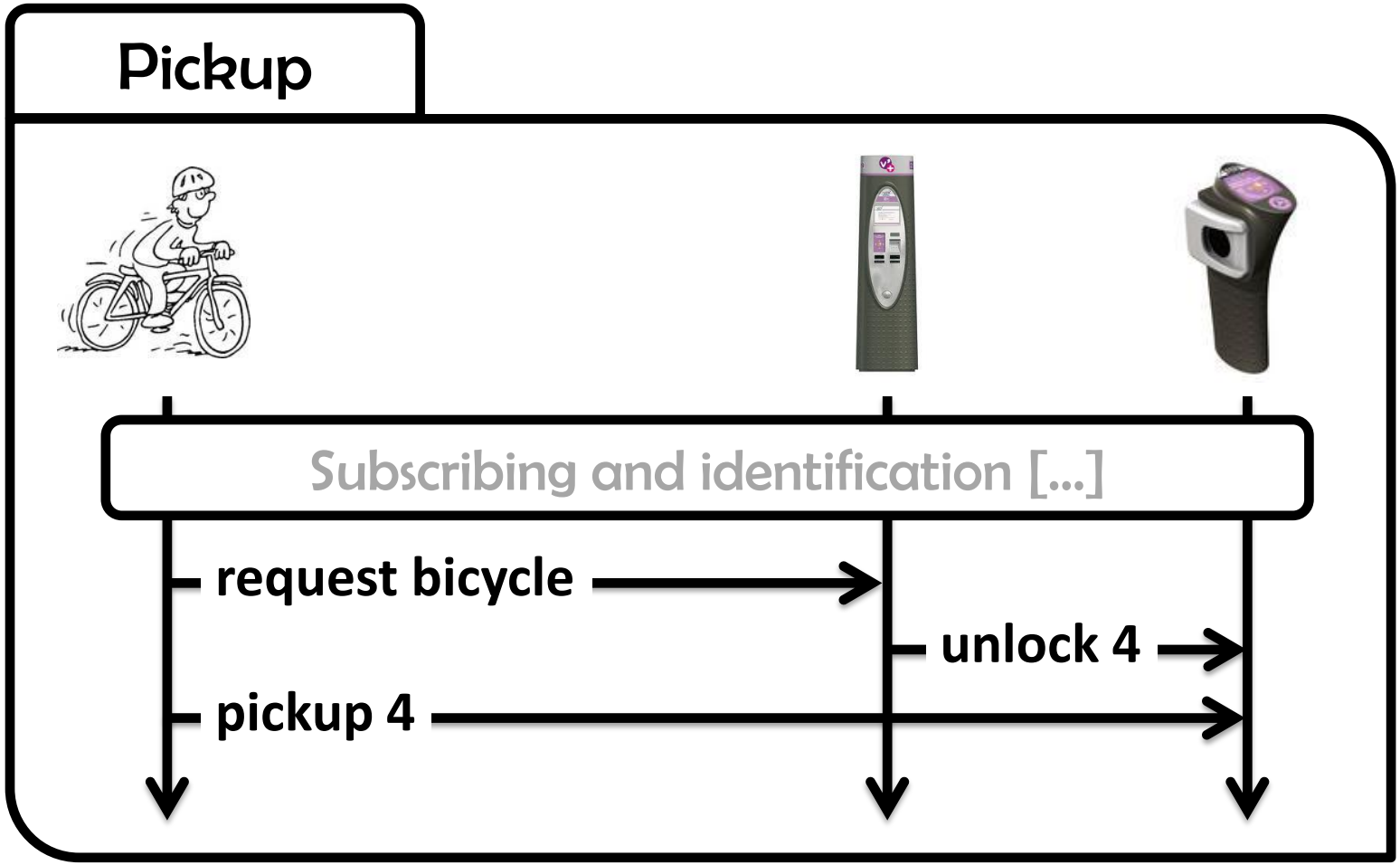
What about the problem?



The hardest part of software development is determining what the system should do [Bro87]



Is this interaction right?

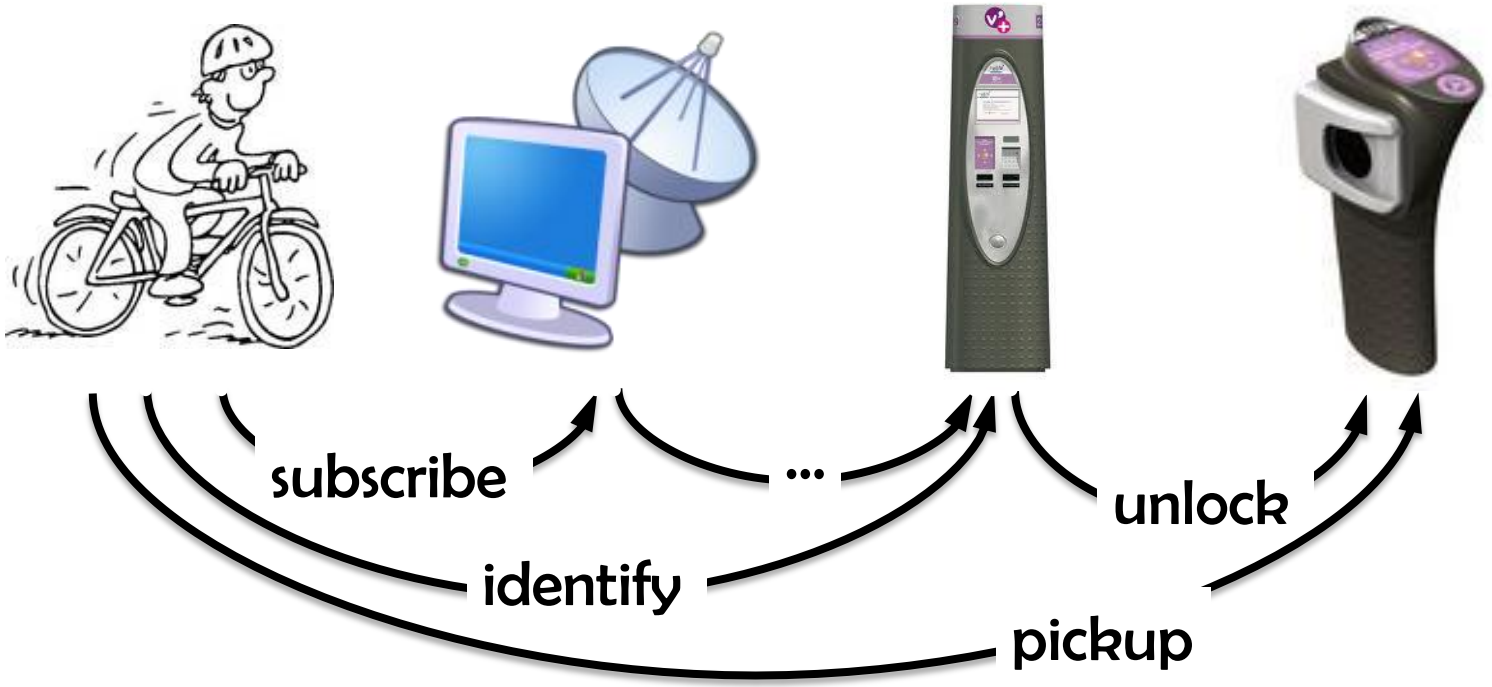


Not necessarily…

The hardest part of software development is determining what the system should (not) do

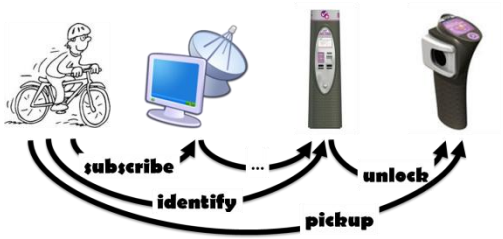
Synthesizing Multi-view Models of Software Systems

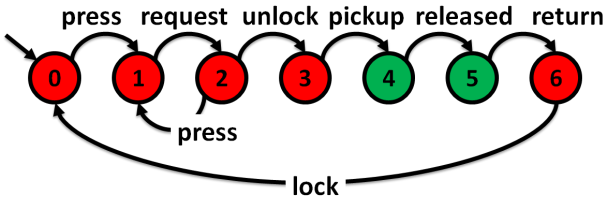
- Model
  - An abstract representation of the target system and its intent

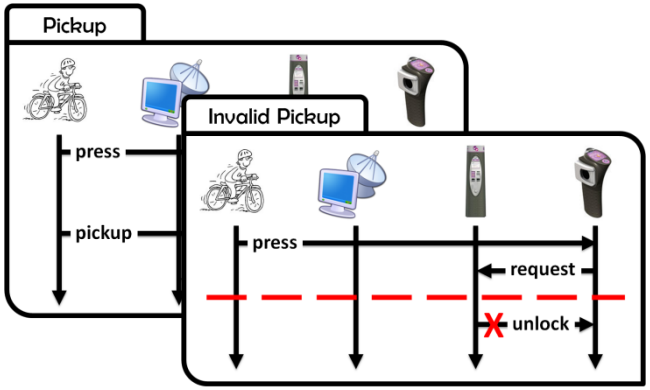


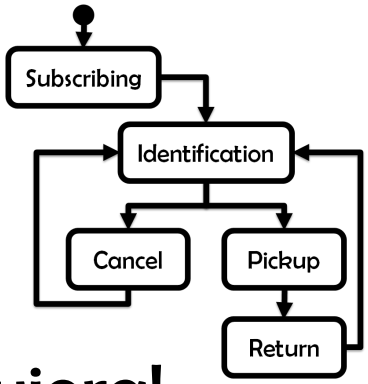
Synthesizing Multi-view Models of Software Systems

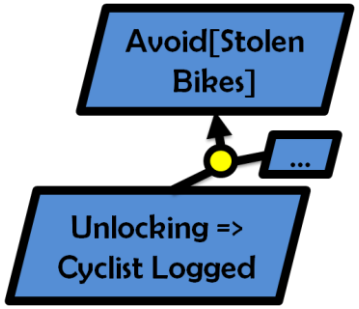
Different models cover complementary but overlapping system aspects











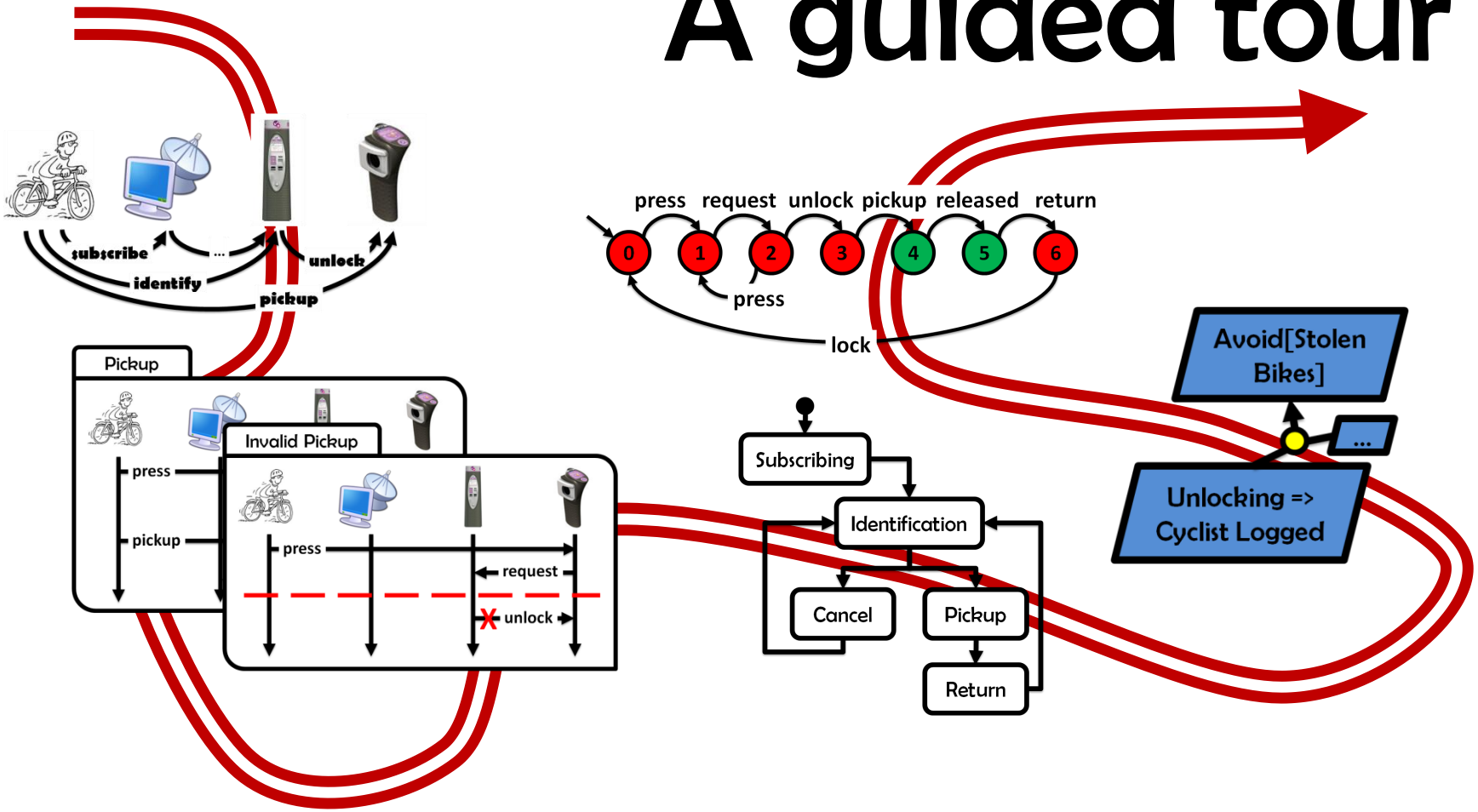
Structural
what? who?

Behavioral
how?

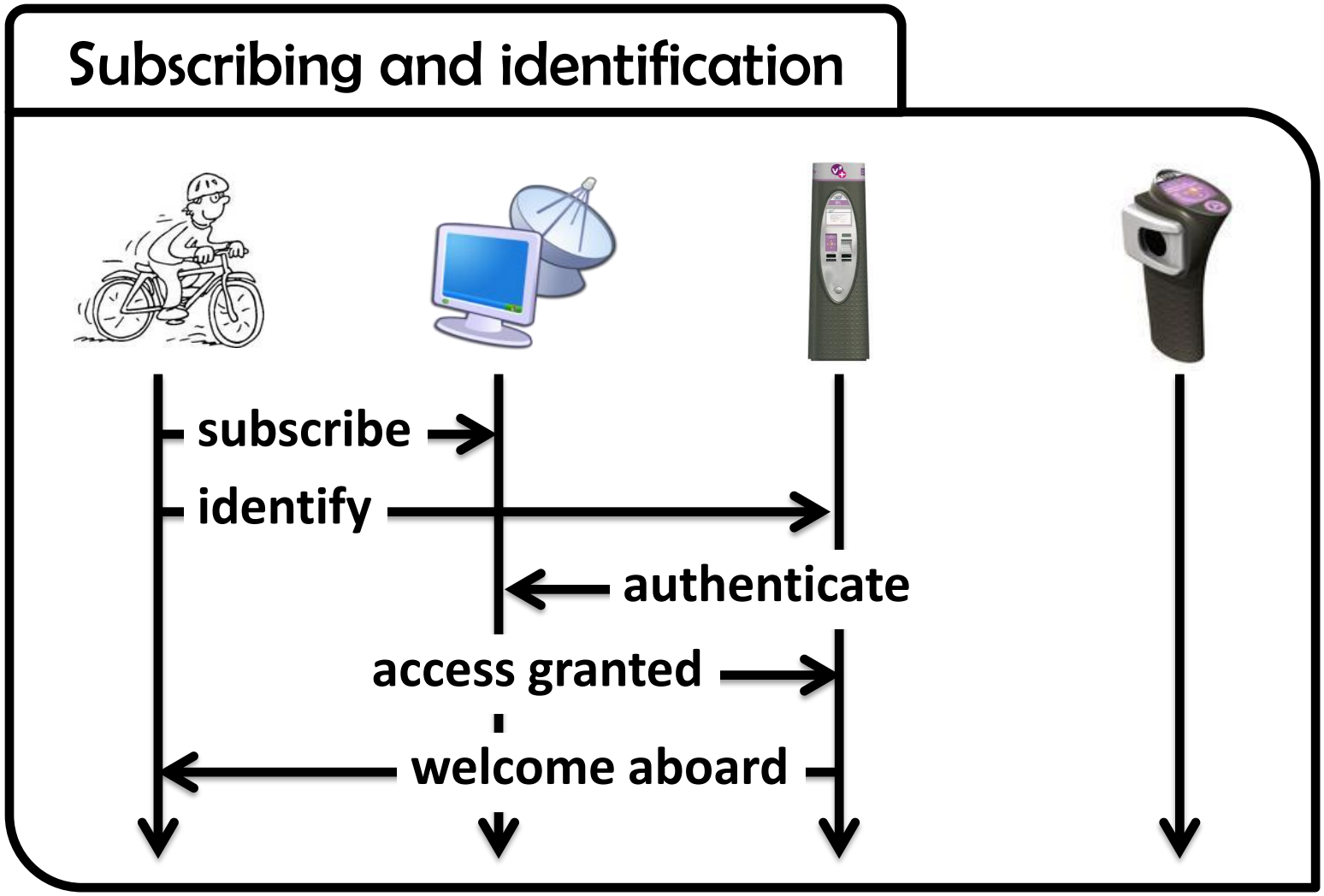
Intentional
why?

Multi-view models

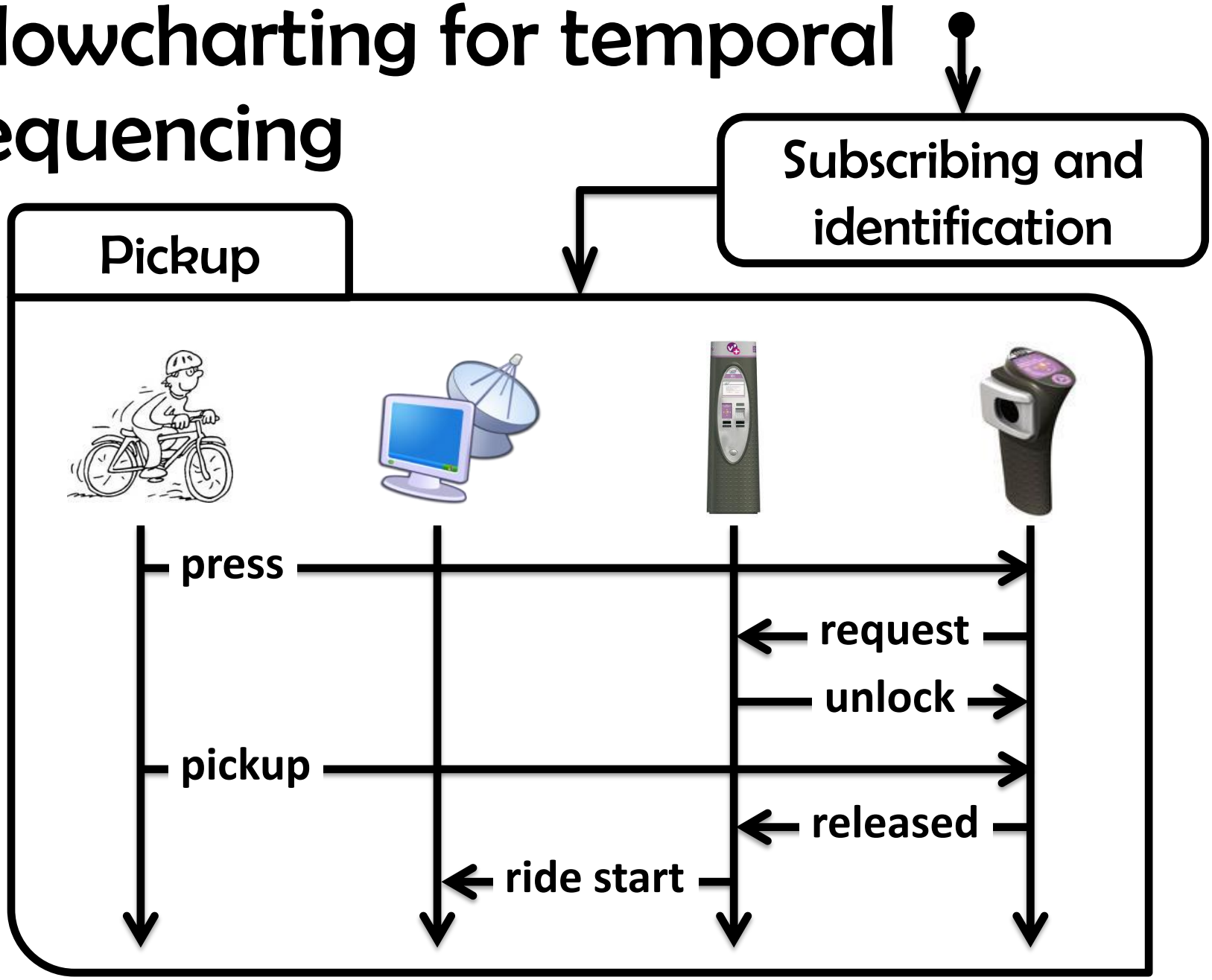
A guided tour



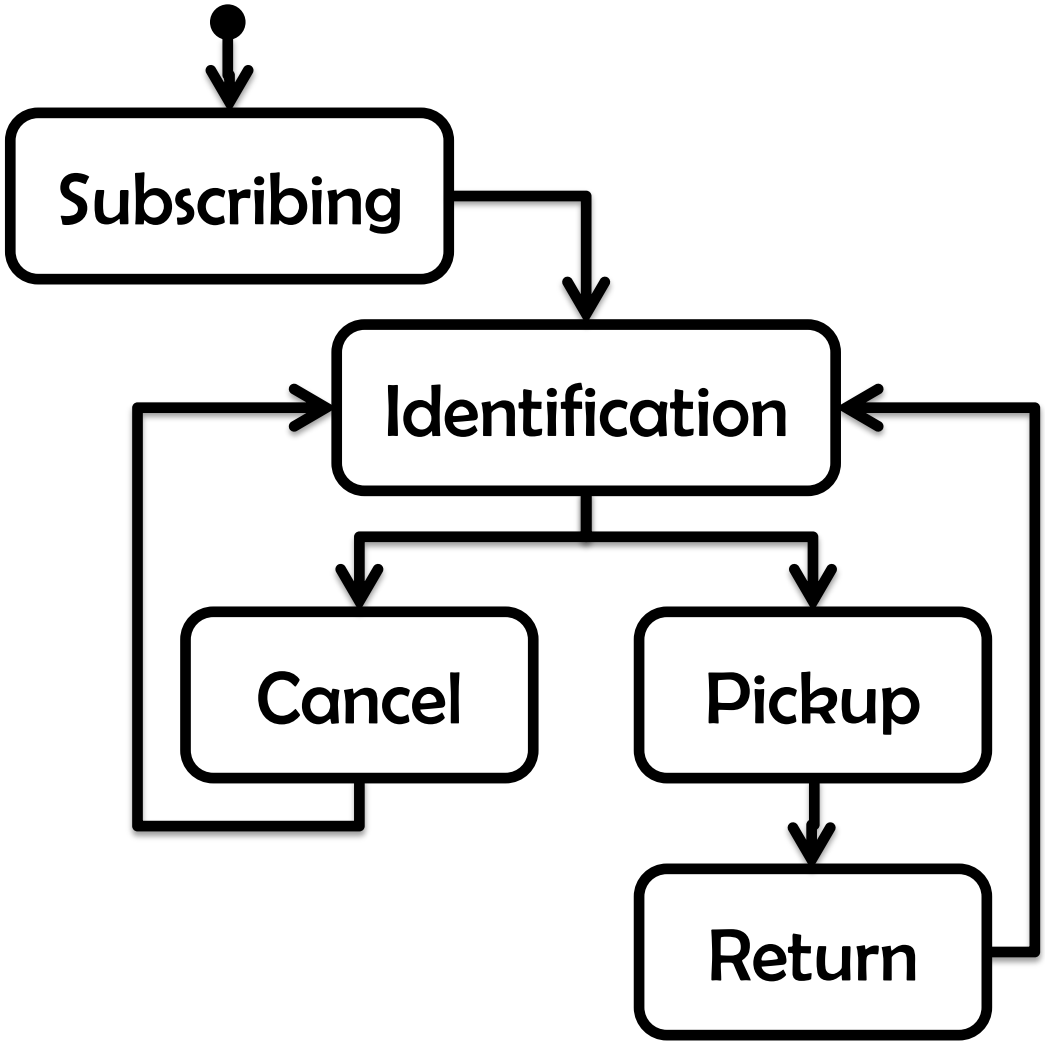
Scenarios capture examples of agent interactions



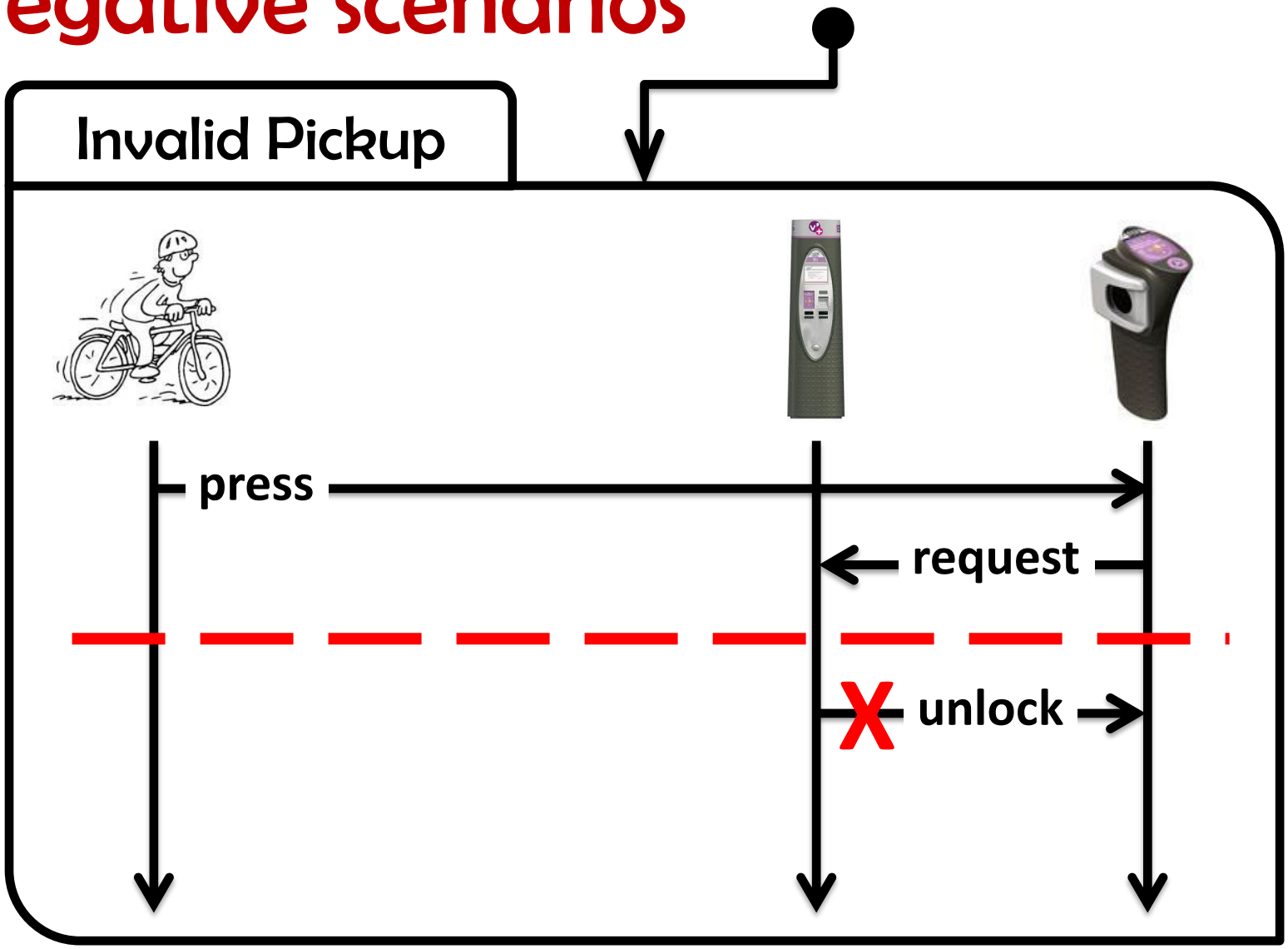
Flowcharting for temporal sequencing



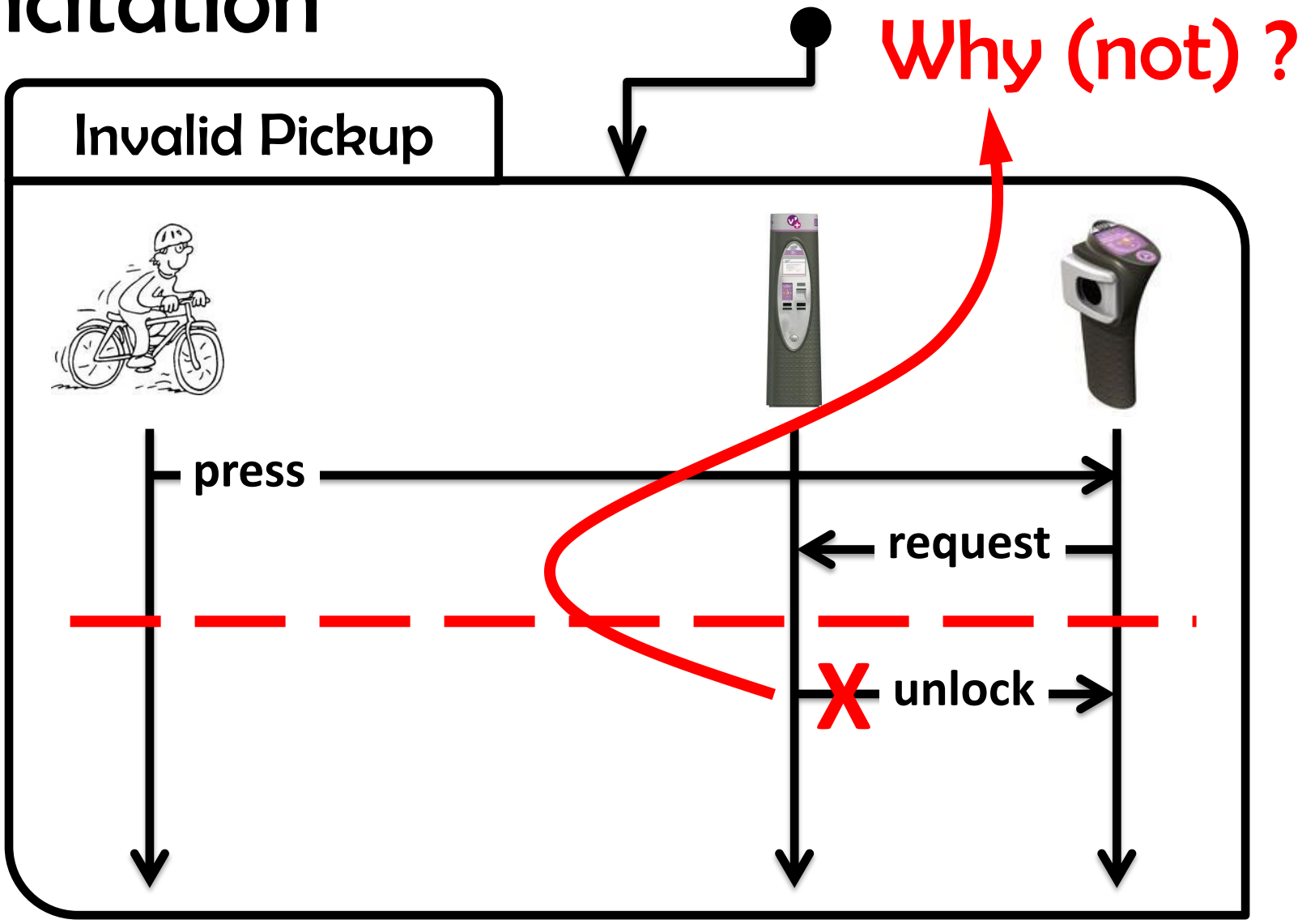
High-level scenarios also support loops and alternatives



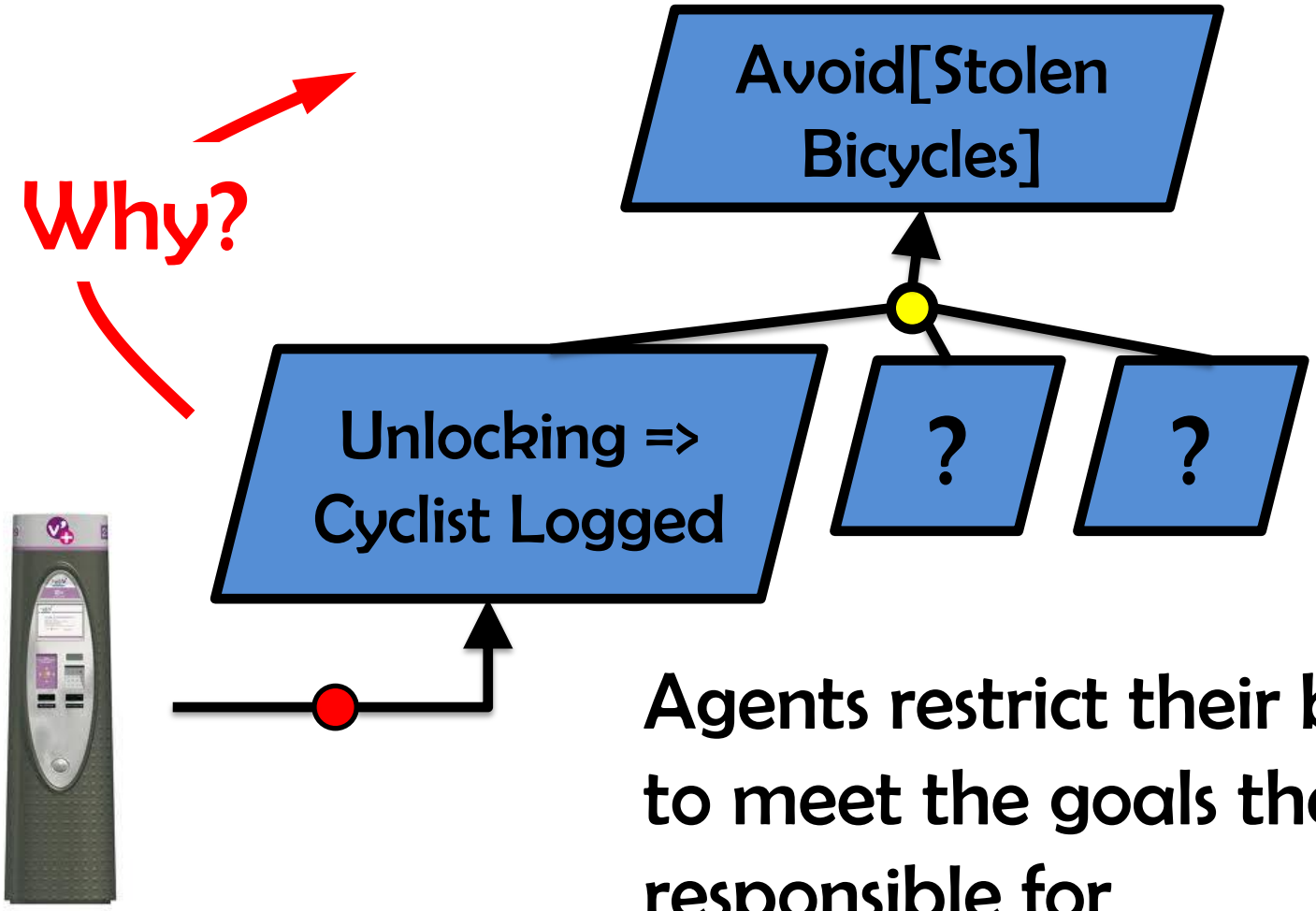
Counterexamples through Negative scenarios



Ask " why? " for effective model elicitation

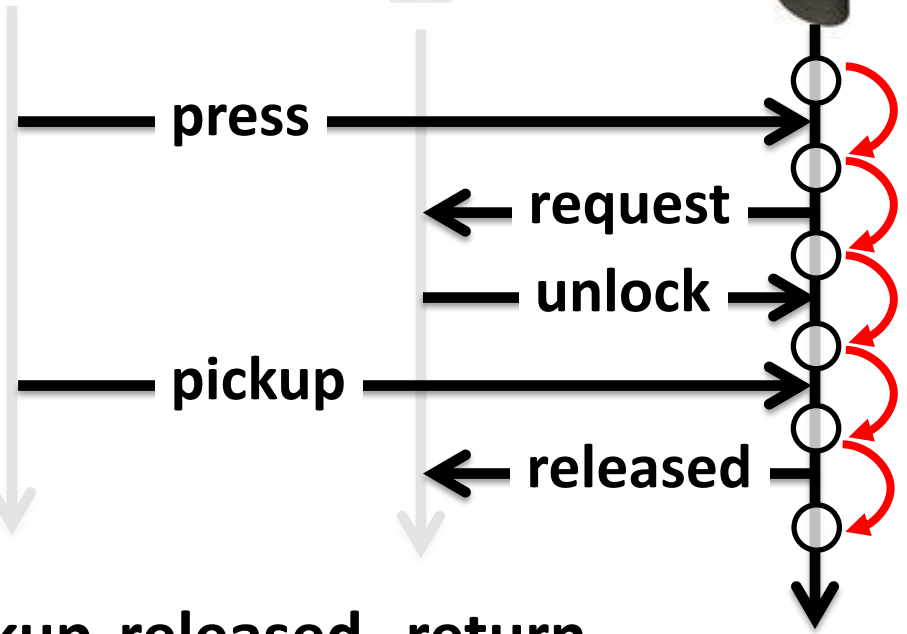


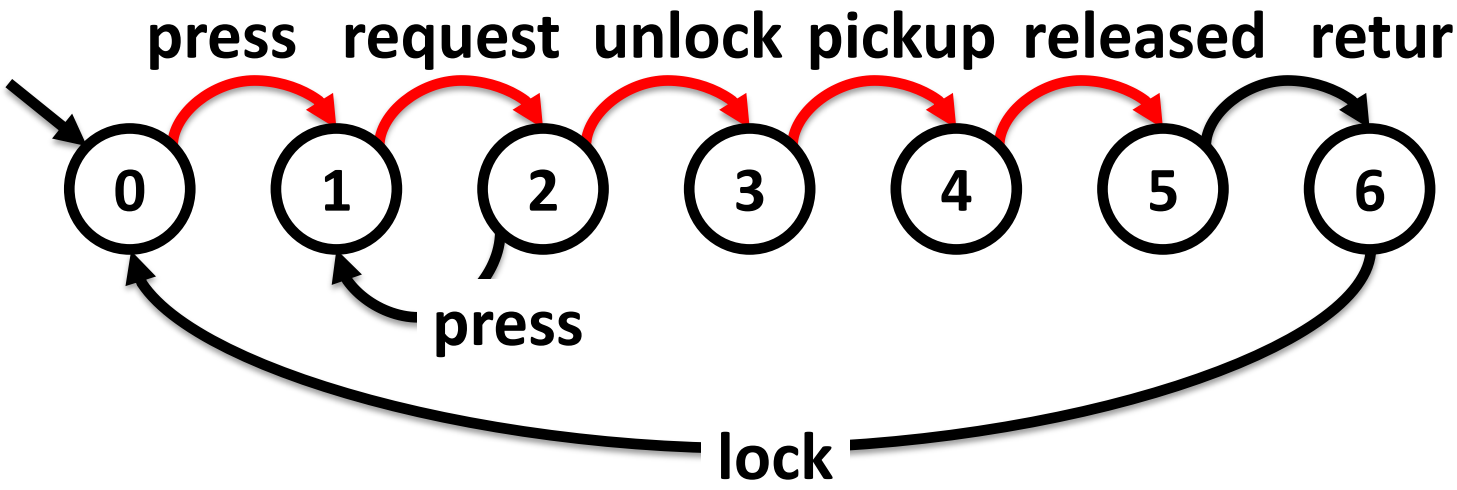
Goals are prescriptive statements of intent



Agents restrict their behavior to meet the goals they are responsible for

Agent behaviors through State machines



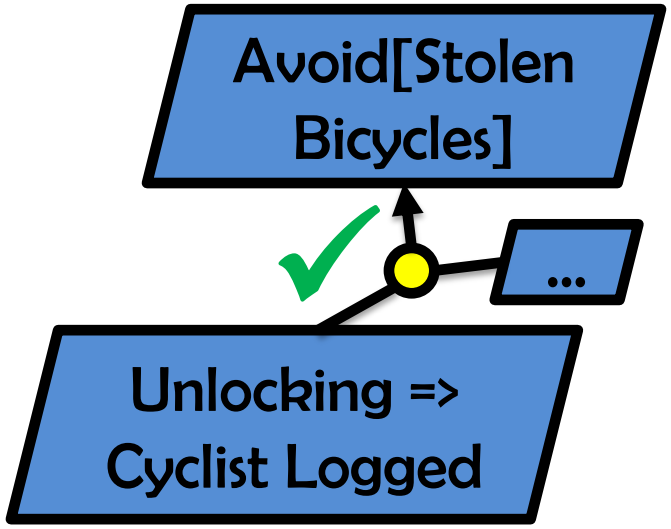


System behavior through state machine composition

Parallel composition of agent behaviors [Hoa85]

- Agents behave asynchronously but synchronize on shared events [Mil89, Mag99]
- The composition is captured through a state machine

Consistency links between state machines and goals



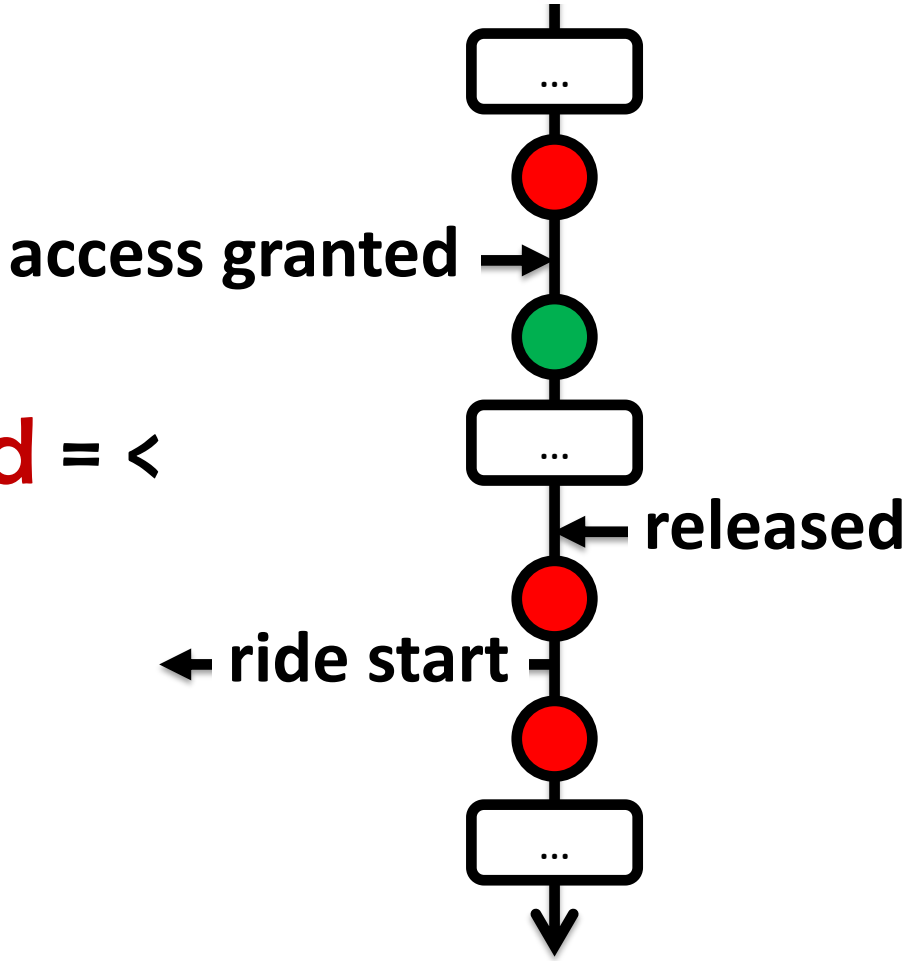
The composed system should meet high-level goals

Agents restrict their behavior to meet their requirements

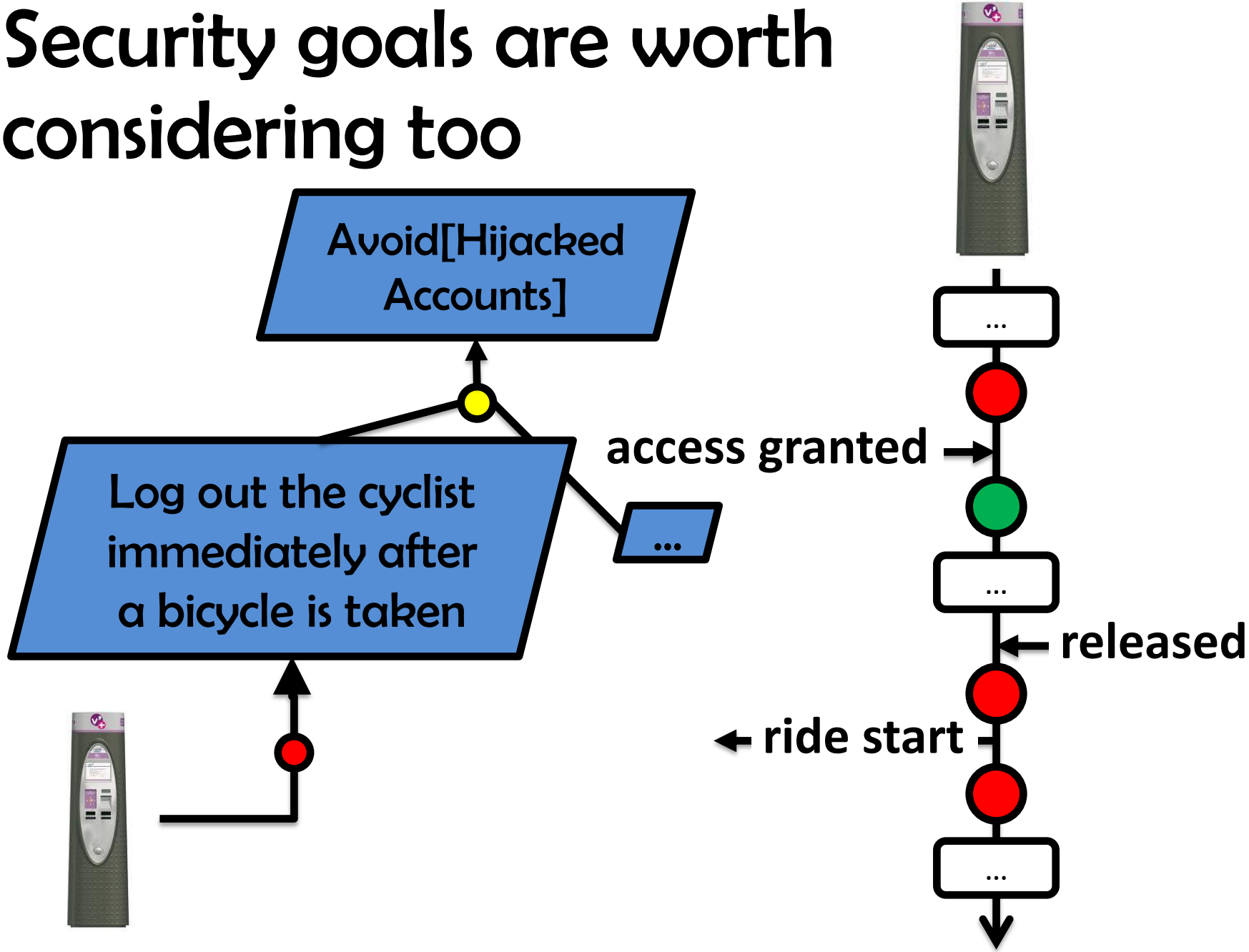
Goal predicates in terms of State variables

Unlocking =>
Cyclist Logged

fluent Cyclist Logged = <
{ granted },
{ released } >
initially false



Security goals are worth considering too

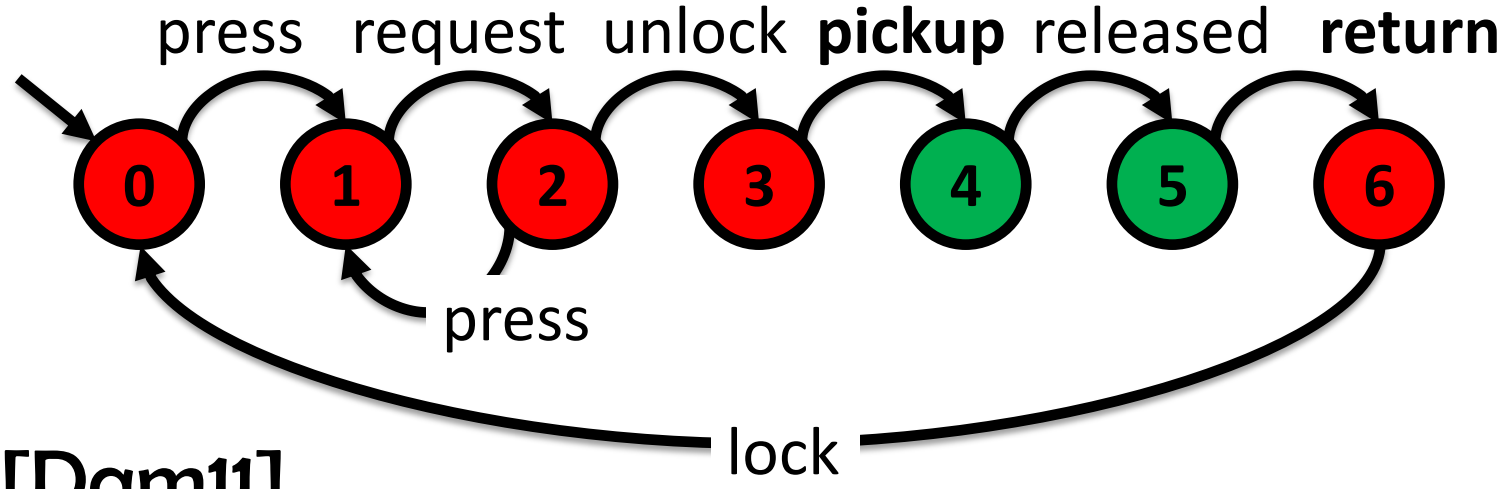


State variables may decorate state machines

fluent Free = <

{ pickup },

{ return } > initially false



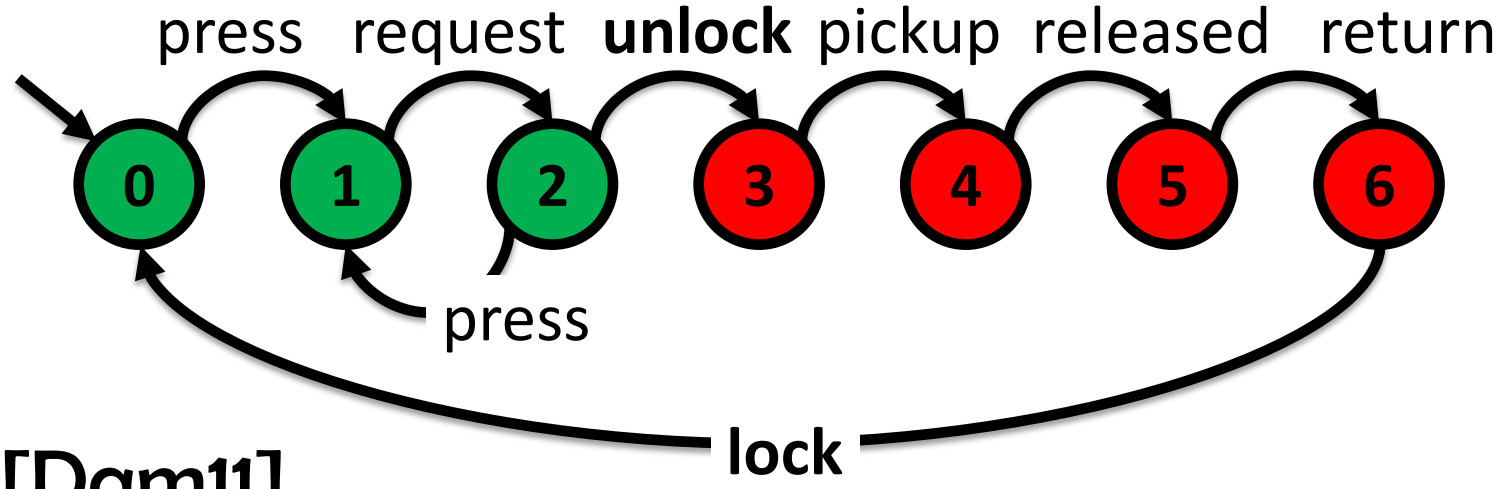
Cfr. [Dam11]

State variables may decorate state machines

fluent Locked = <

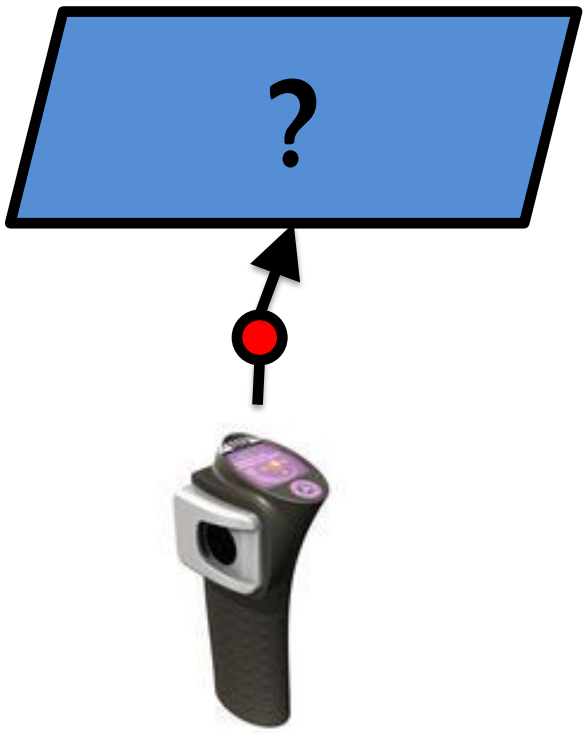
{ lock },

{ unlock } > initially **true**

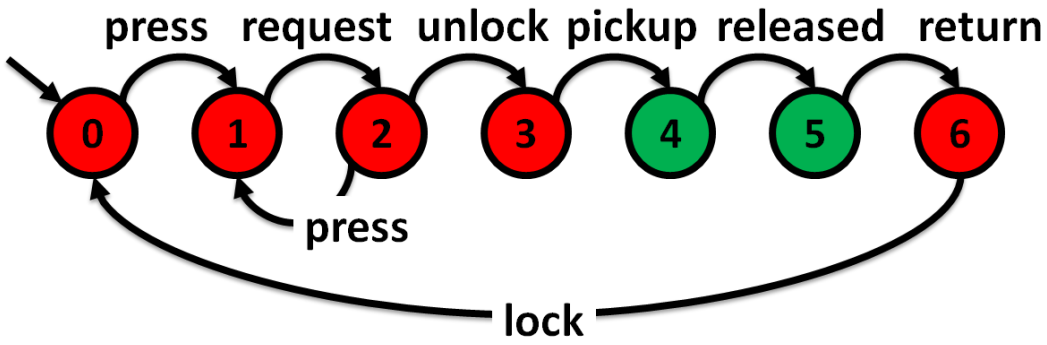


Cfr. [Dam11]

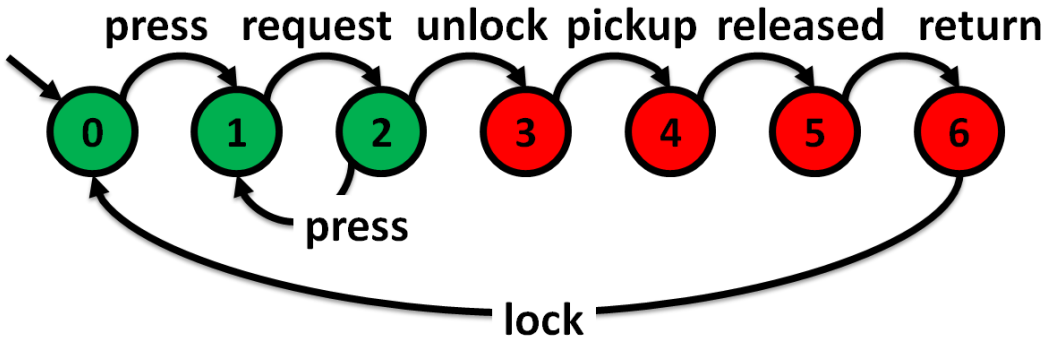
Some requirements are hidden behind variable assignments



Free

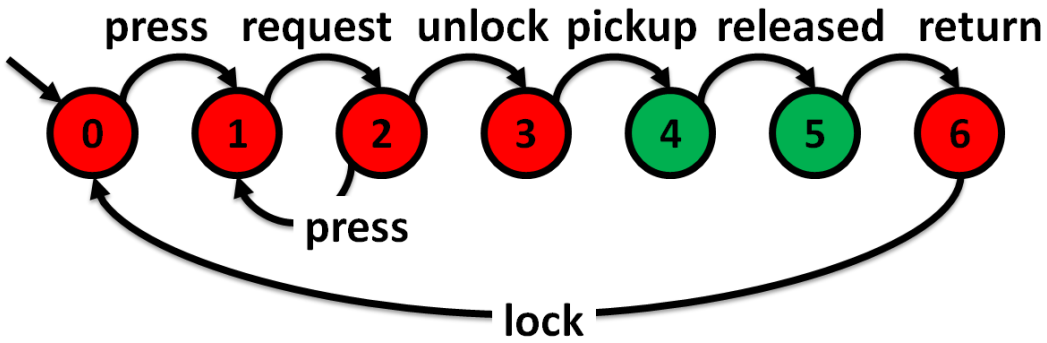


Locked

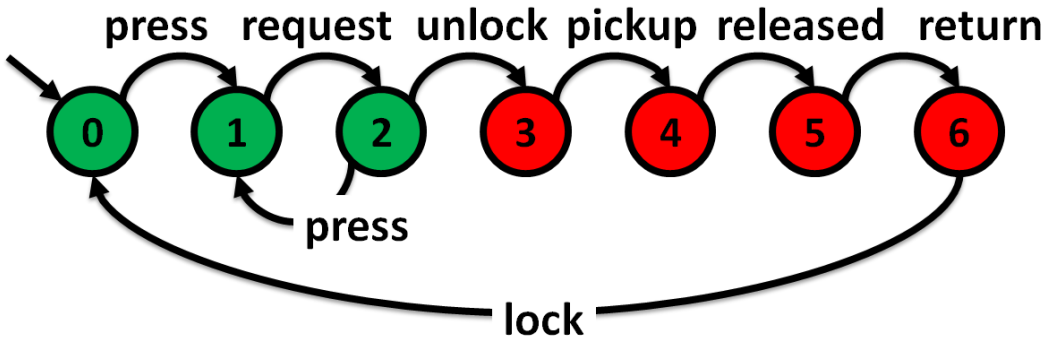


Some requirements are hidden behind variable assignments

Free



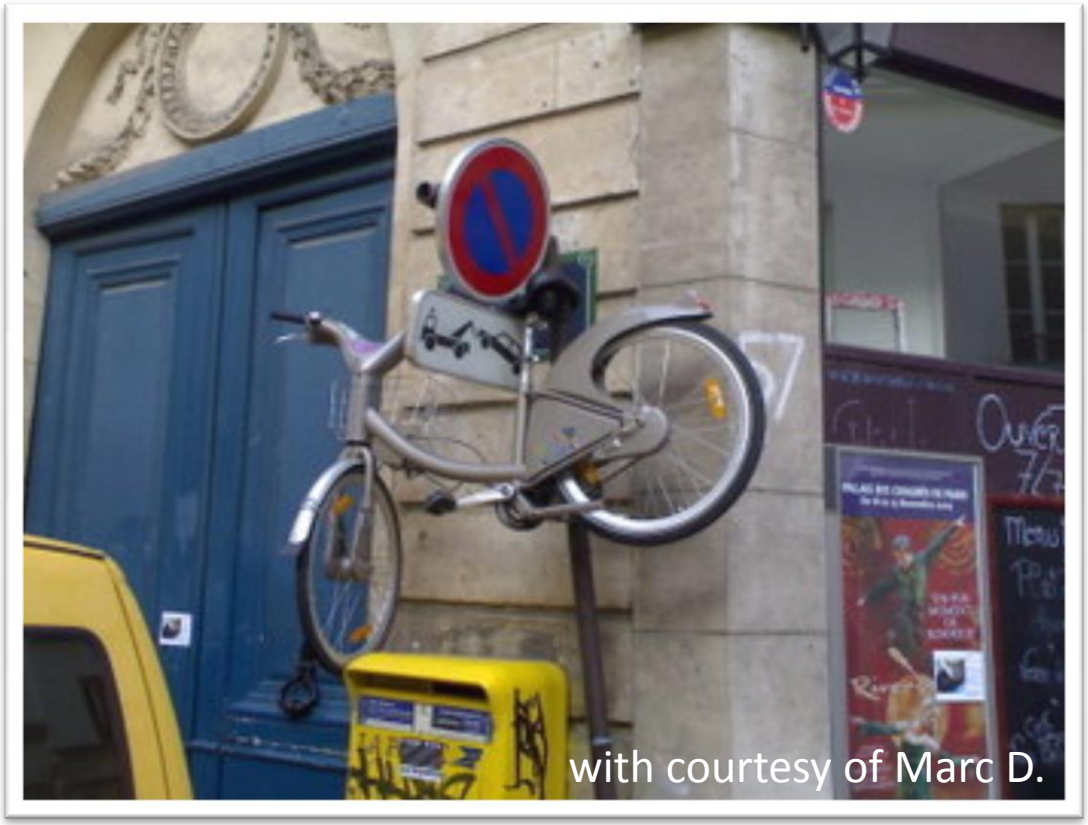
Locked



Avoid annoying people...!

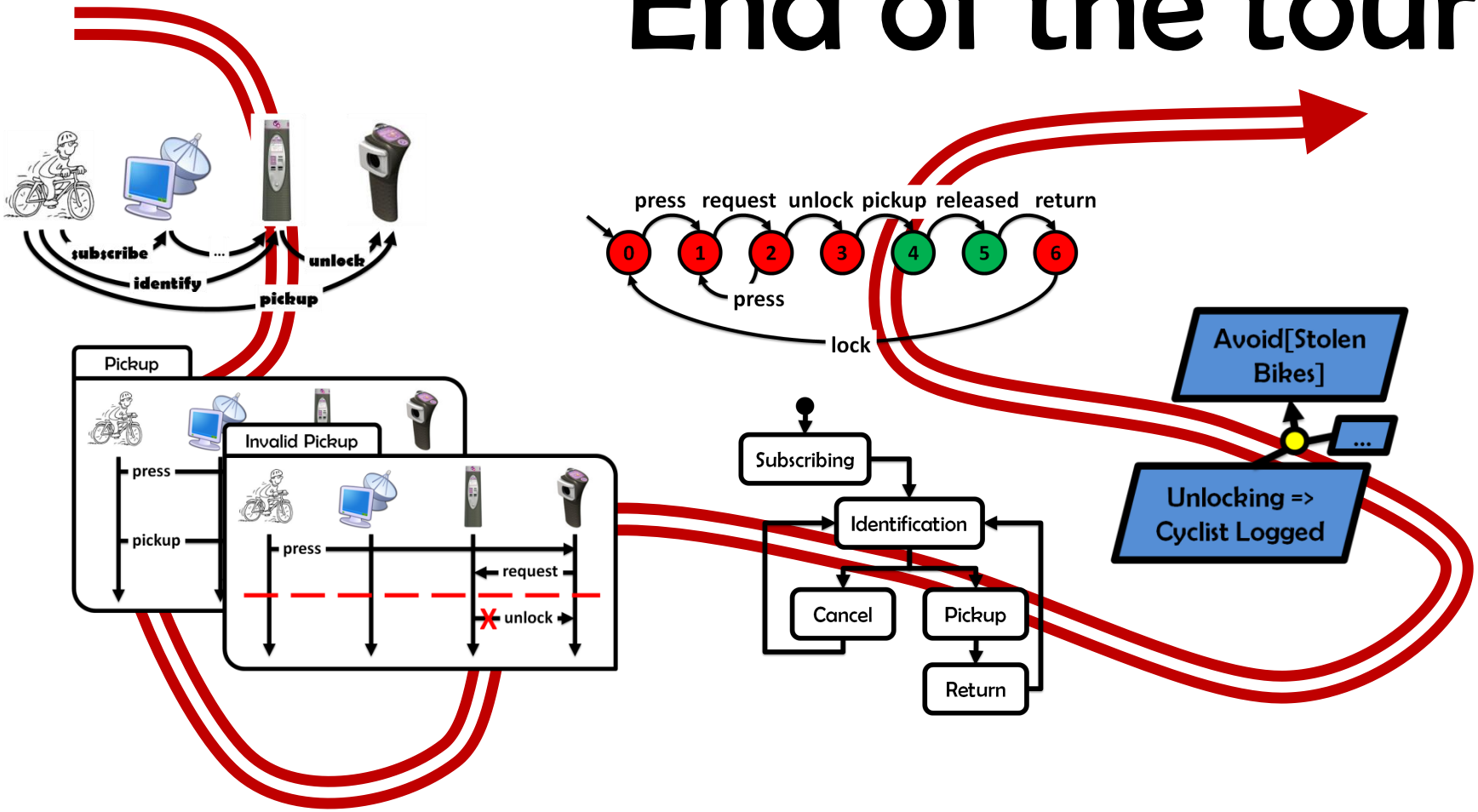
Why?

Avoid[Free and Locked]

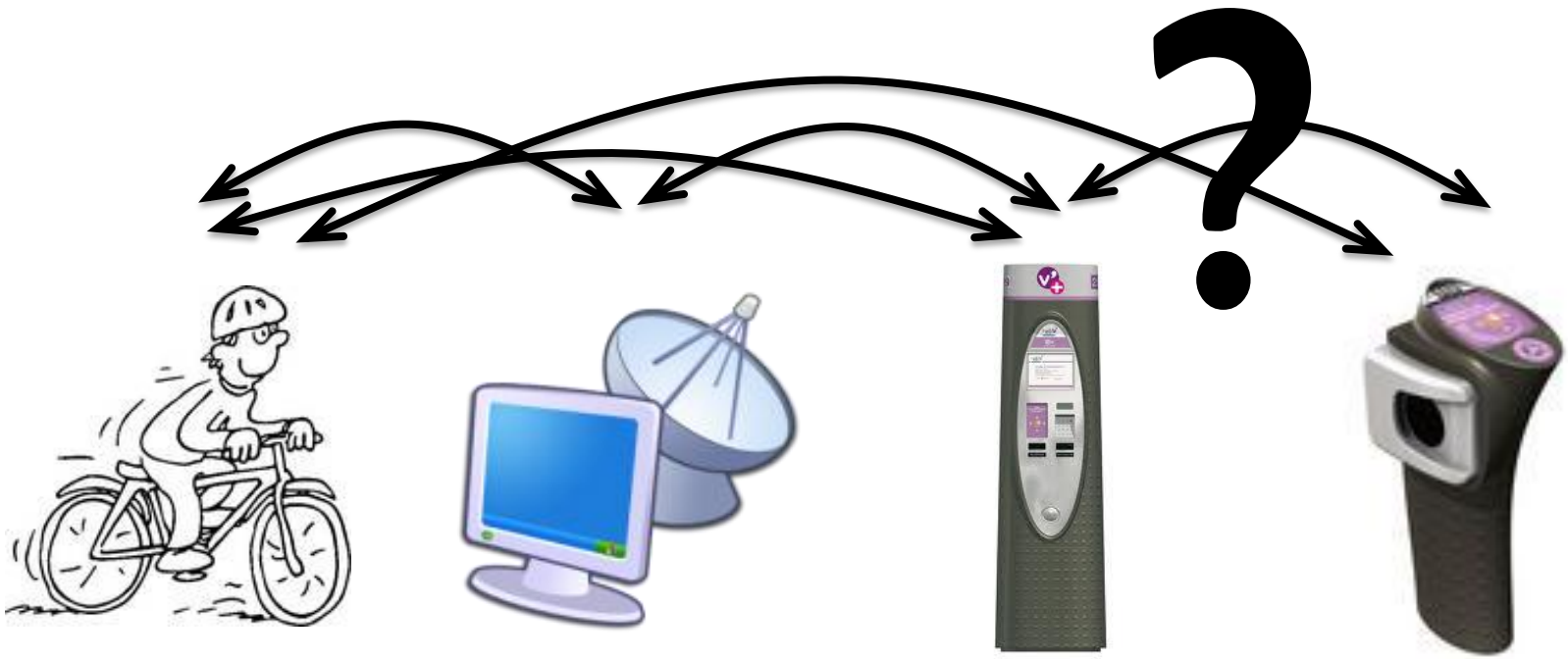


Multi-view models

End of the tour

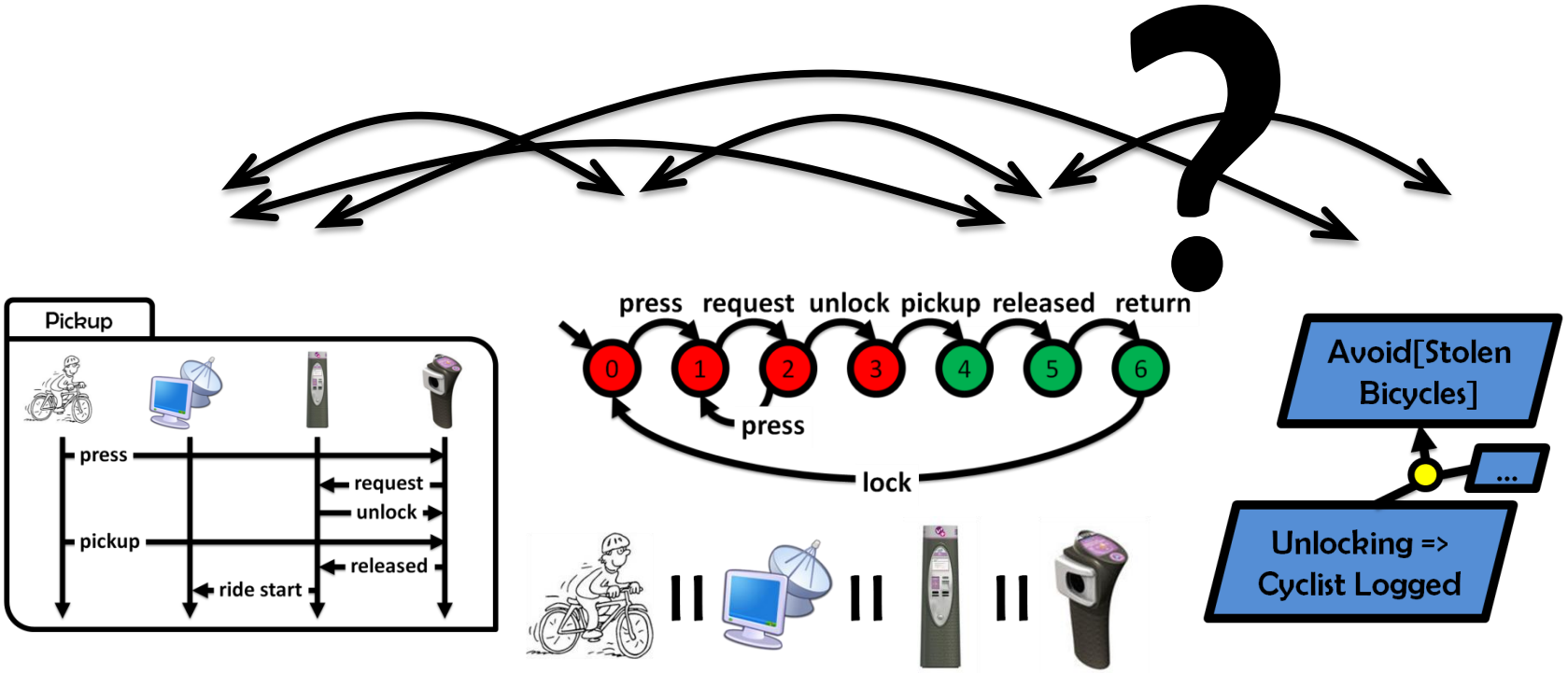


Do you remember this slide?



The hardest part of software development is determining what the system should do [Bro87]

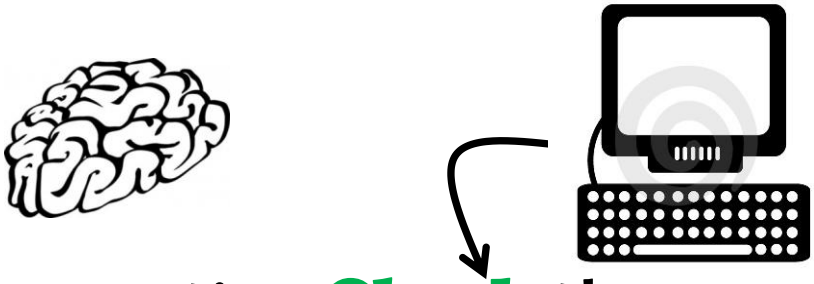
Modeling software systems could hardly be simpler....



High-quality models should be adequate, complete, consistent, precise, analyzable, comprehensible [Avl09]

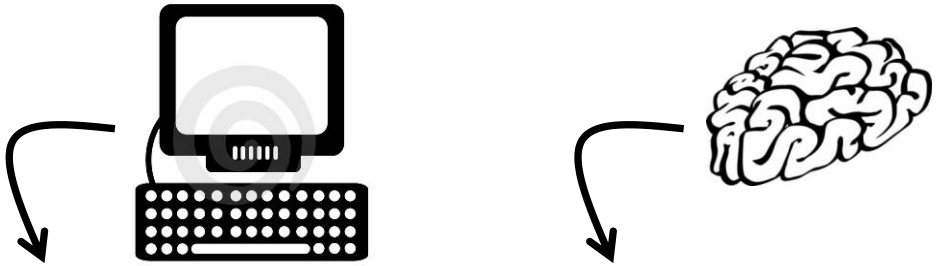
Automated support is needed for system modeling towards...

Better consistency



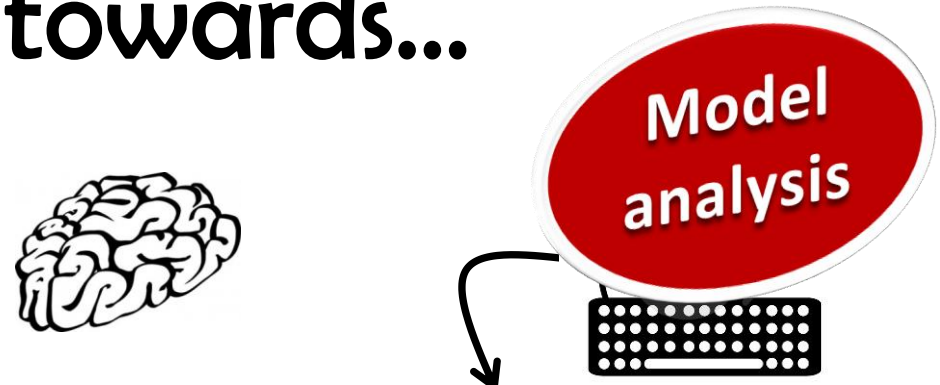
– Build models, **Claim** properties, **Check** them,
Correct the model and/or the properties...

Better completion



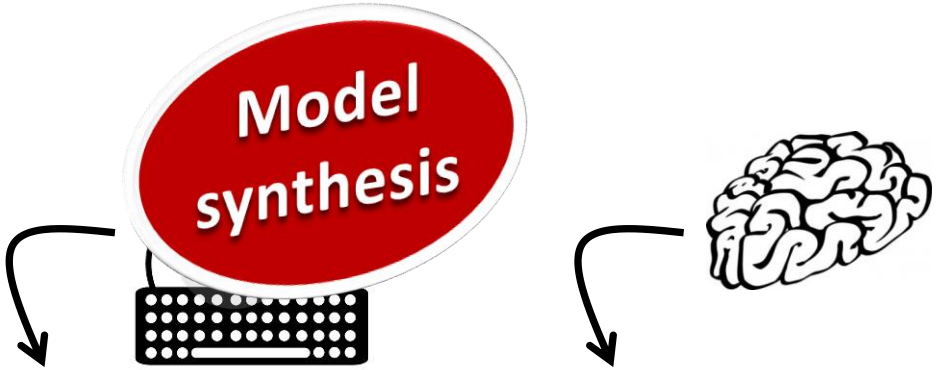
– Build models, **Infer** properties, **Validate** them,
Complete the model and/or the properties...

Automated support is needed for system modeling towards...



Better consistency

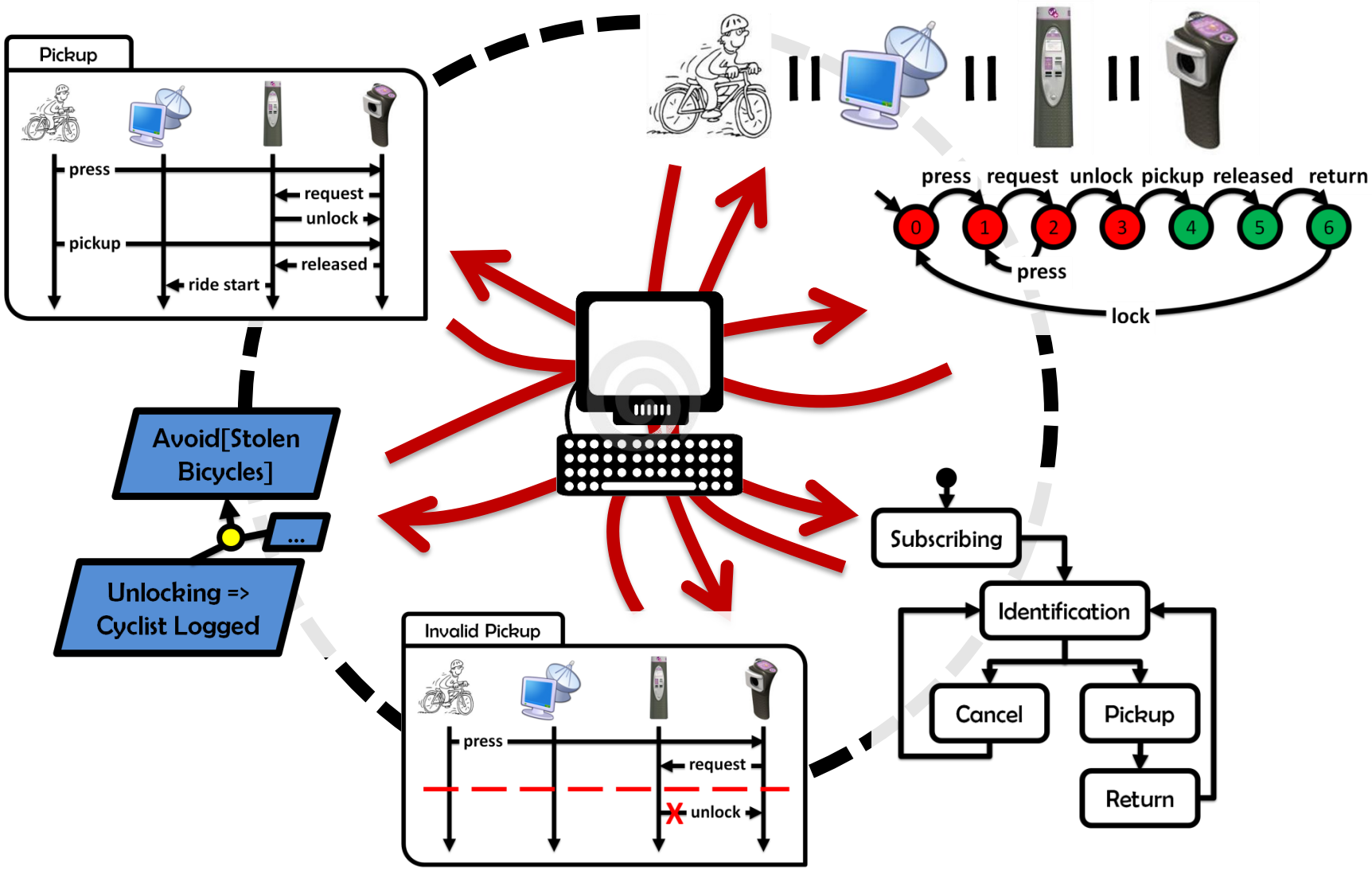
– Build models, **Claim** properties, **Check** them,
Correct the model and/or the properties...



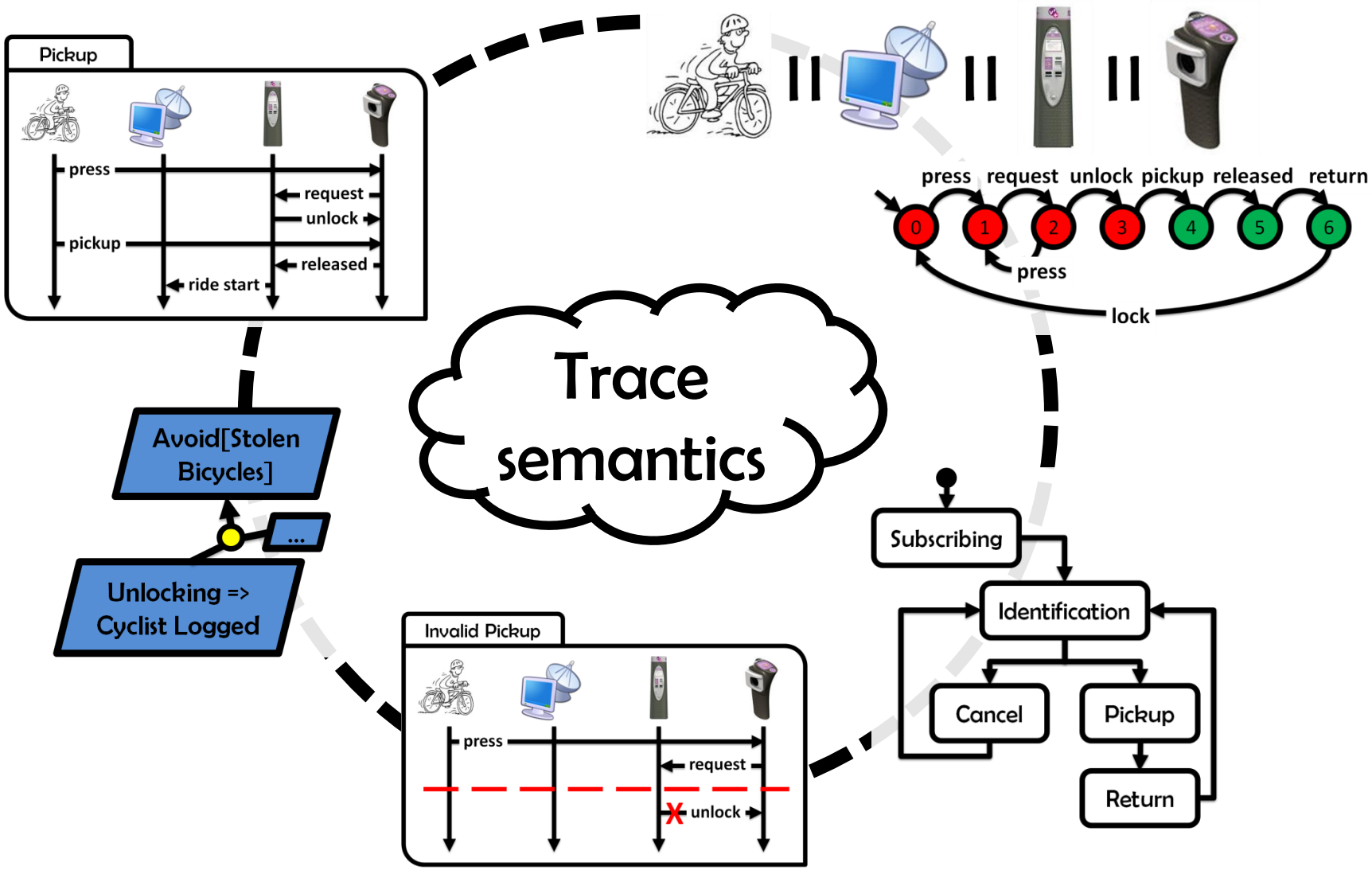
Better completion

– Build models, **Infer** properties, **Validate** them,
Complete the model and/or the properties...

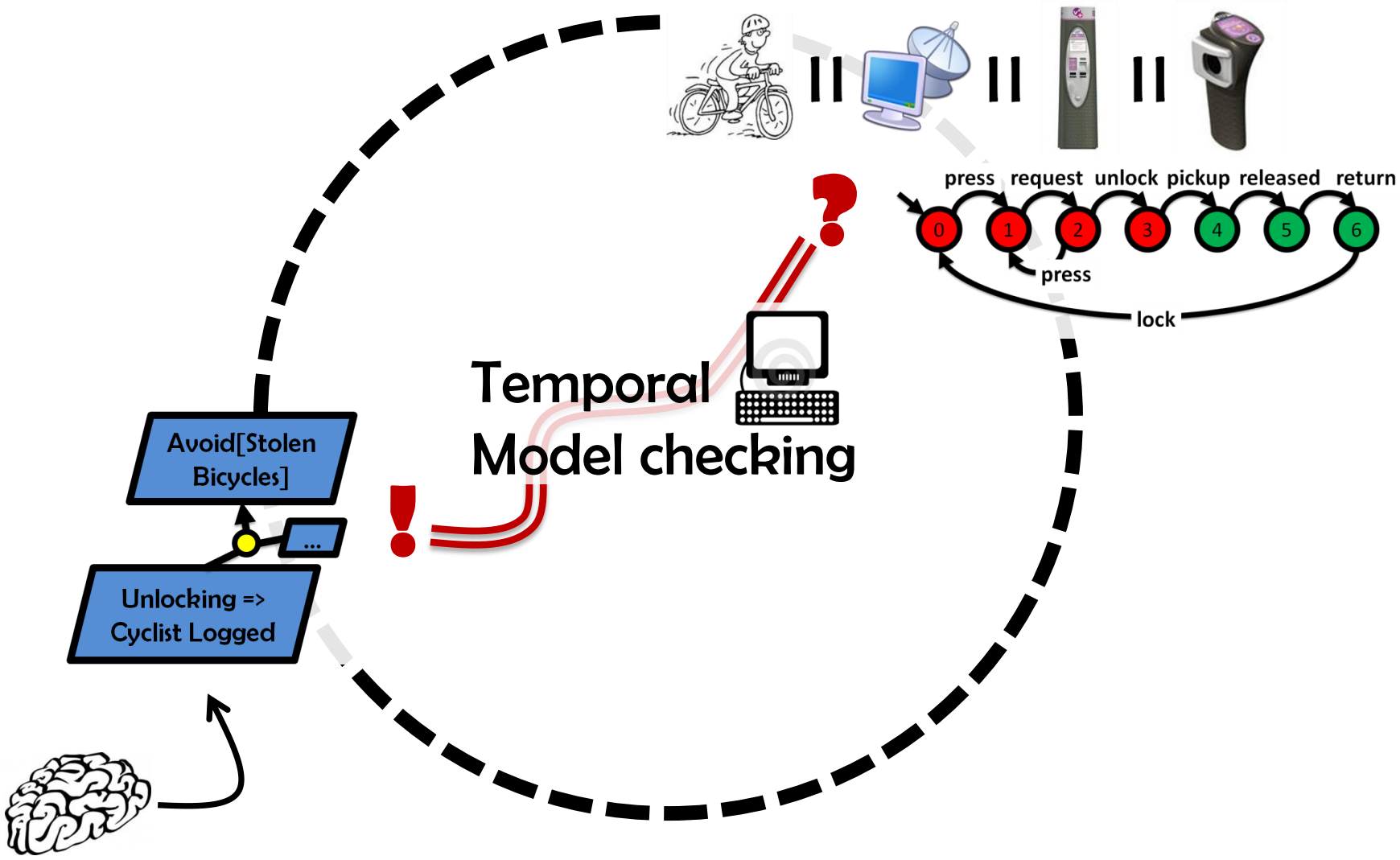
Synthesizing Multi-view Models of Software Systems



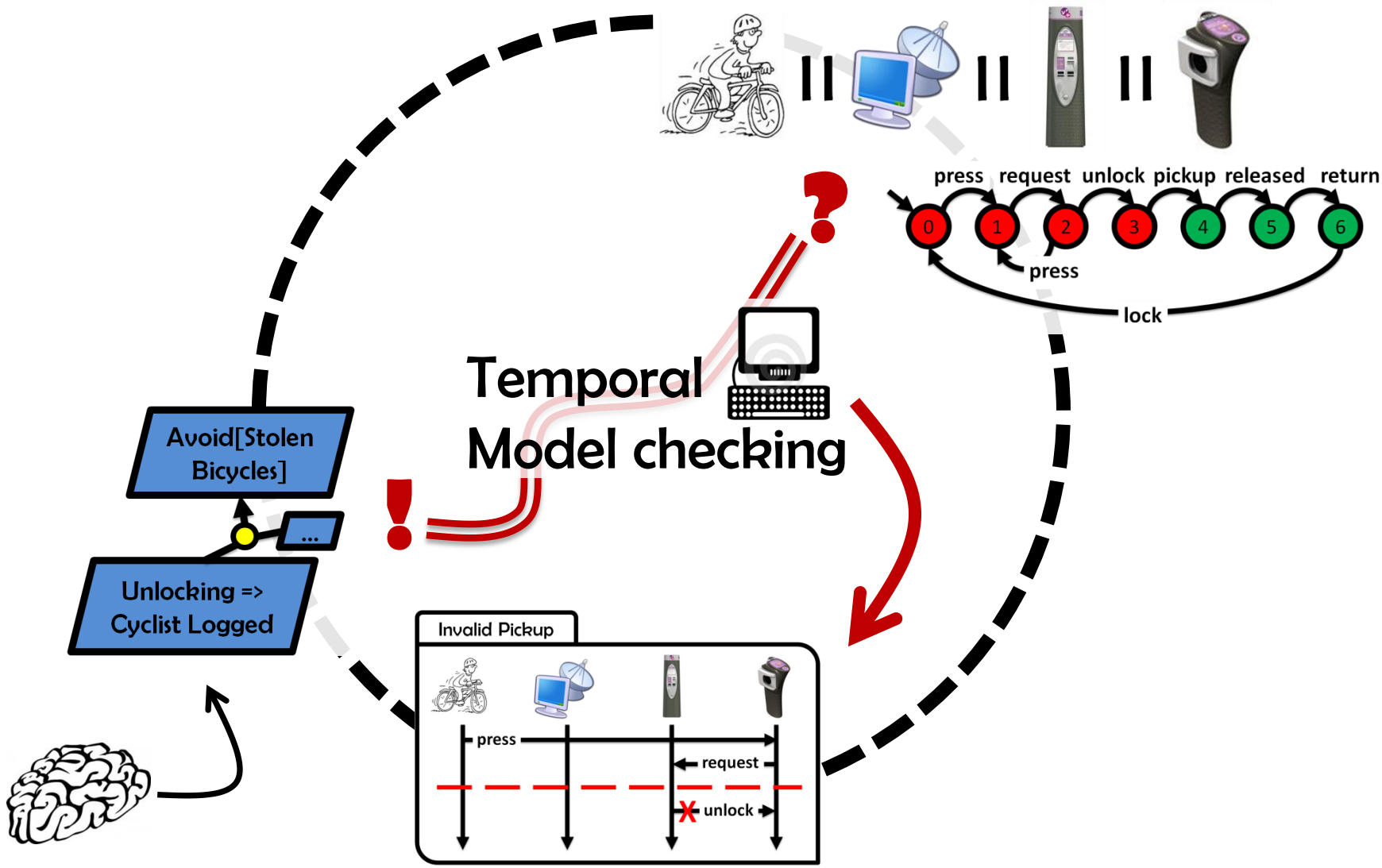
A formal framework for system modeling



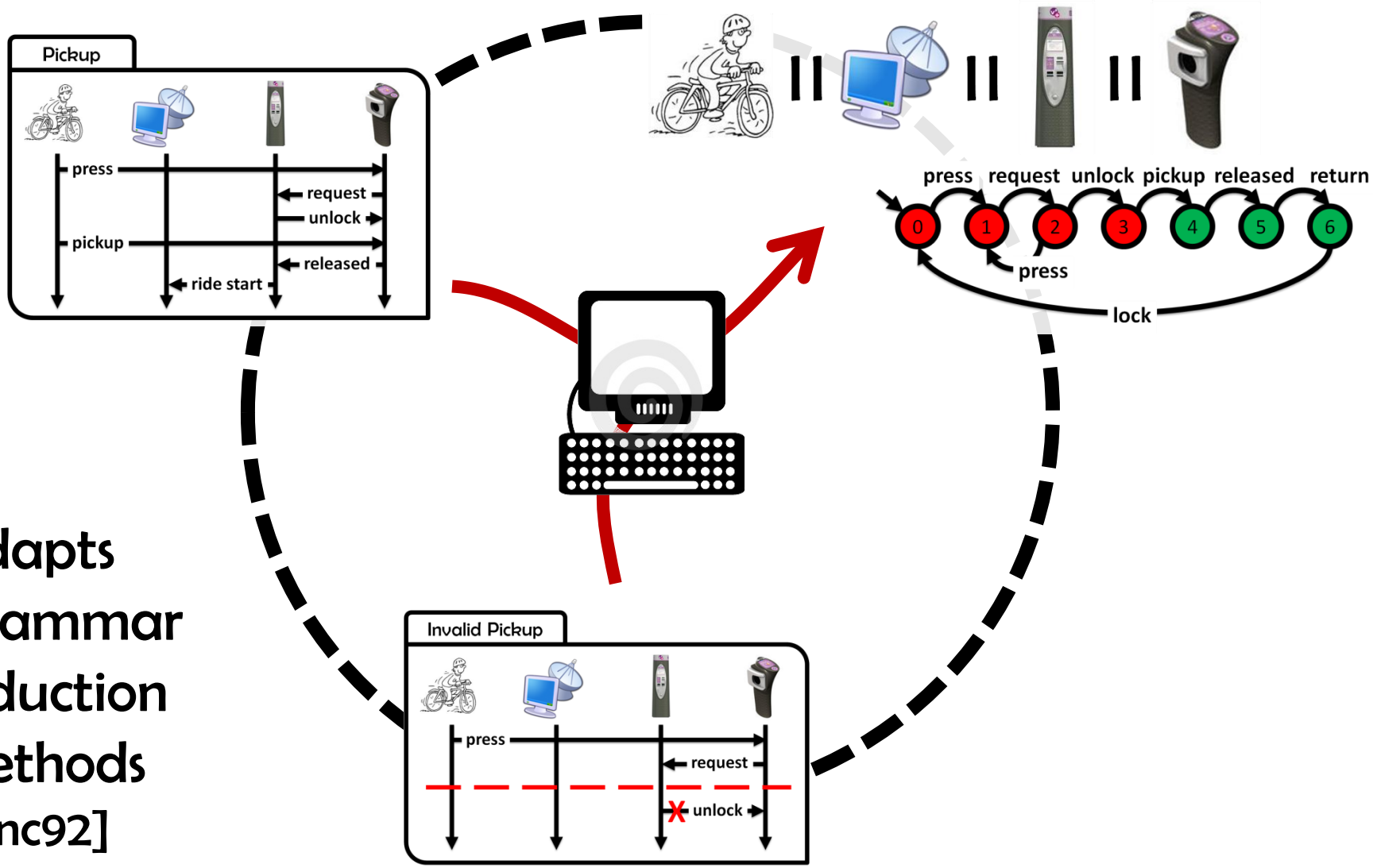
“Claim and check” consistency analysis



"Claim and check" consistency analysis

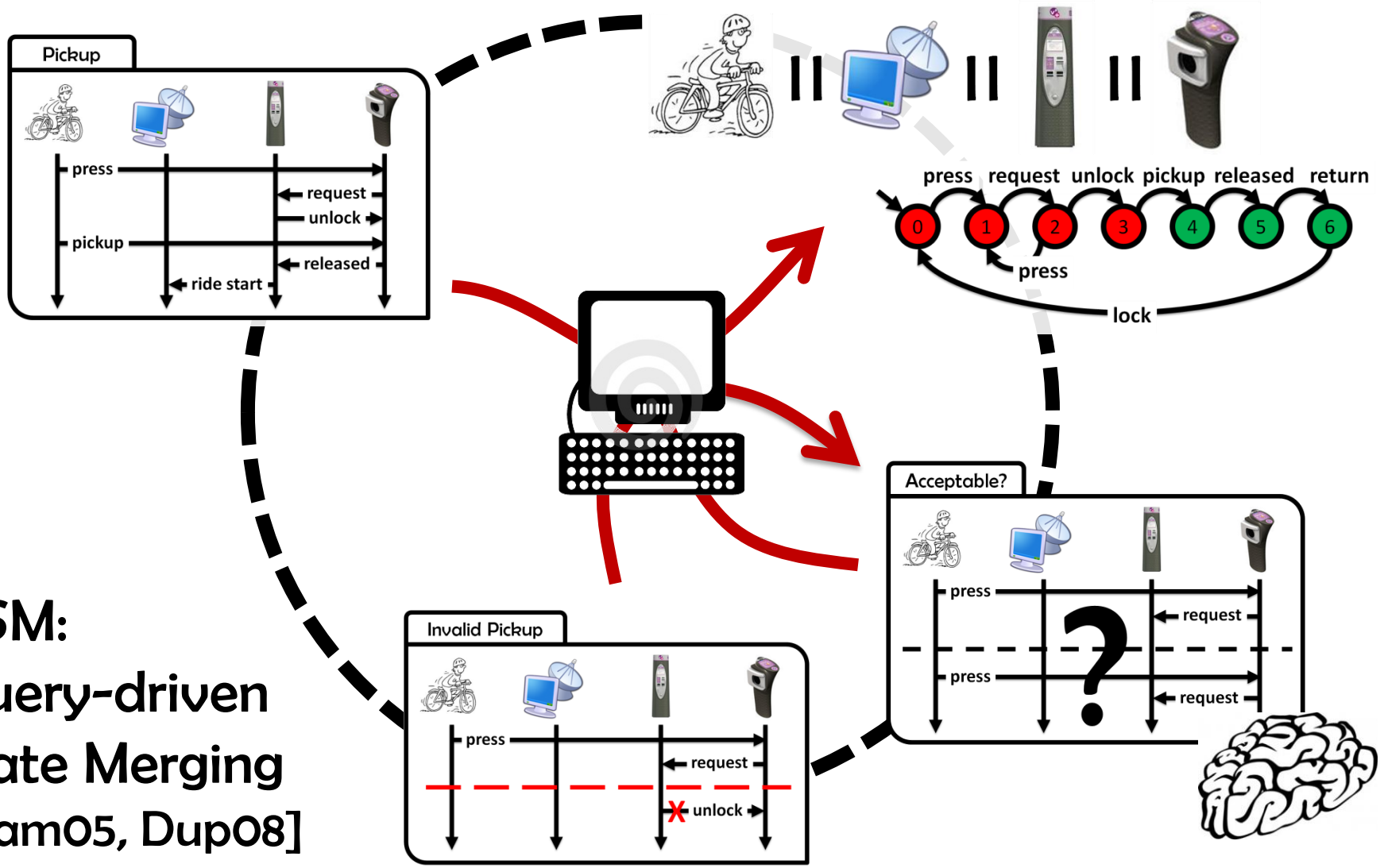


State machine induction from scenarios



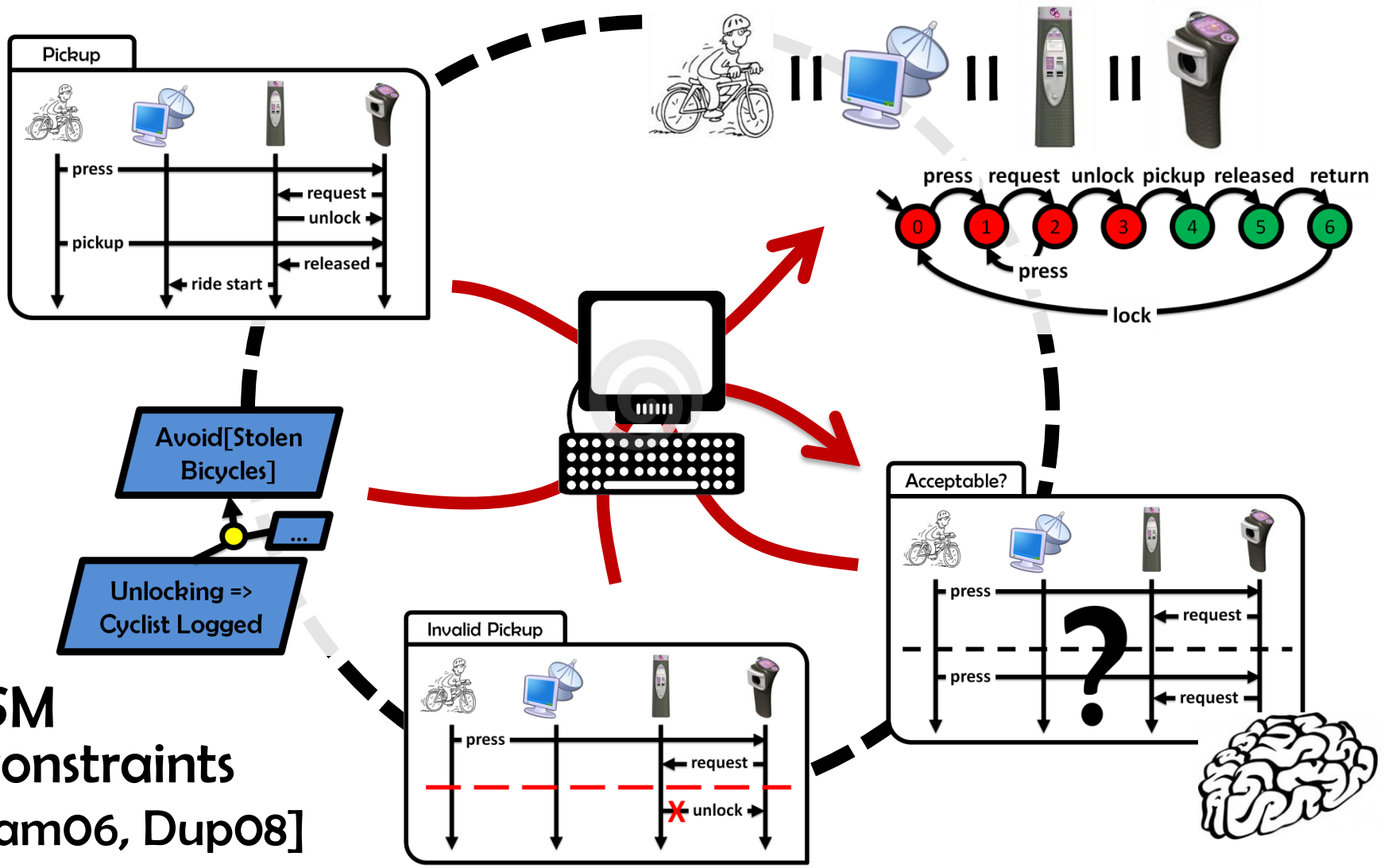
Adapts Grammar Induction Methods [Onc92]

Interactive state machine induction from scenarios



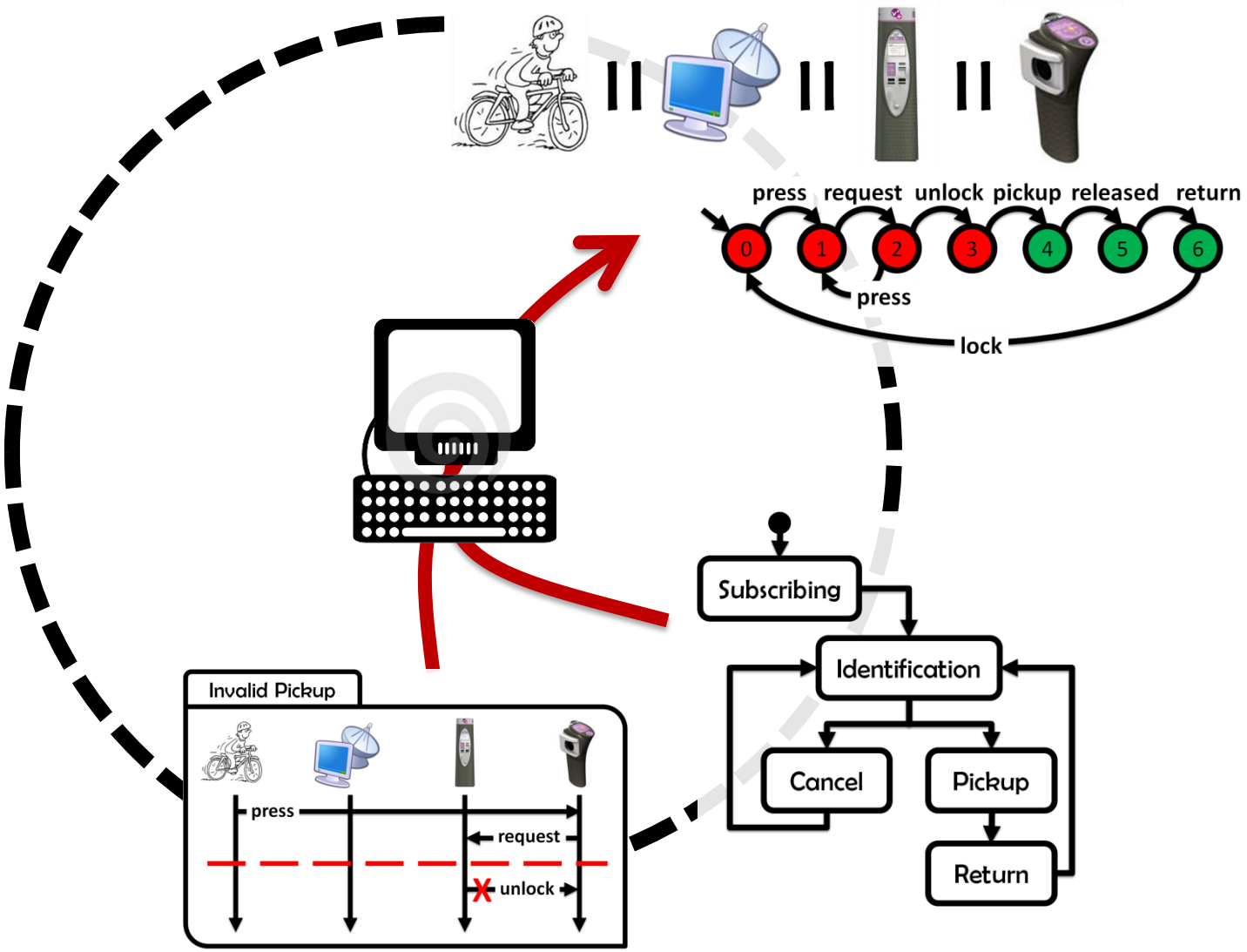
QSM:
Query-driven
State Merging
[Dam05, Dup08]

Interactive state machine induction from scenarios and goals



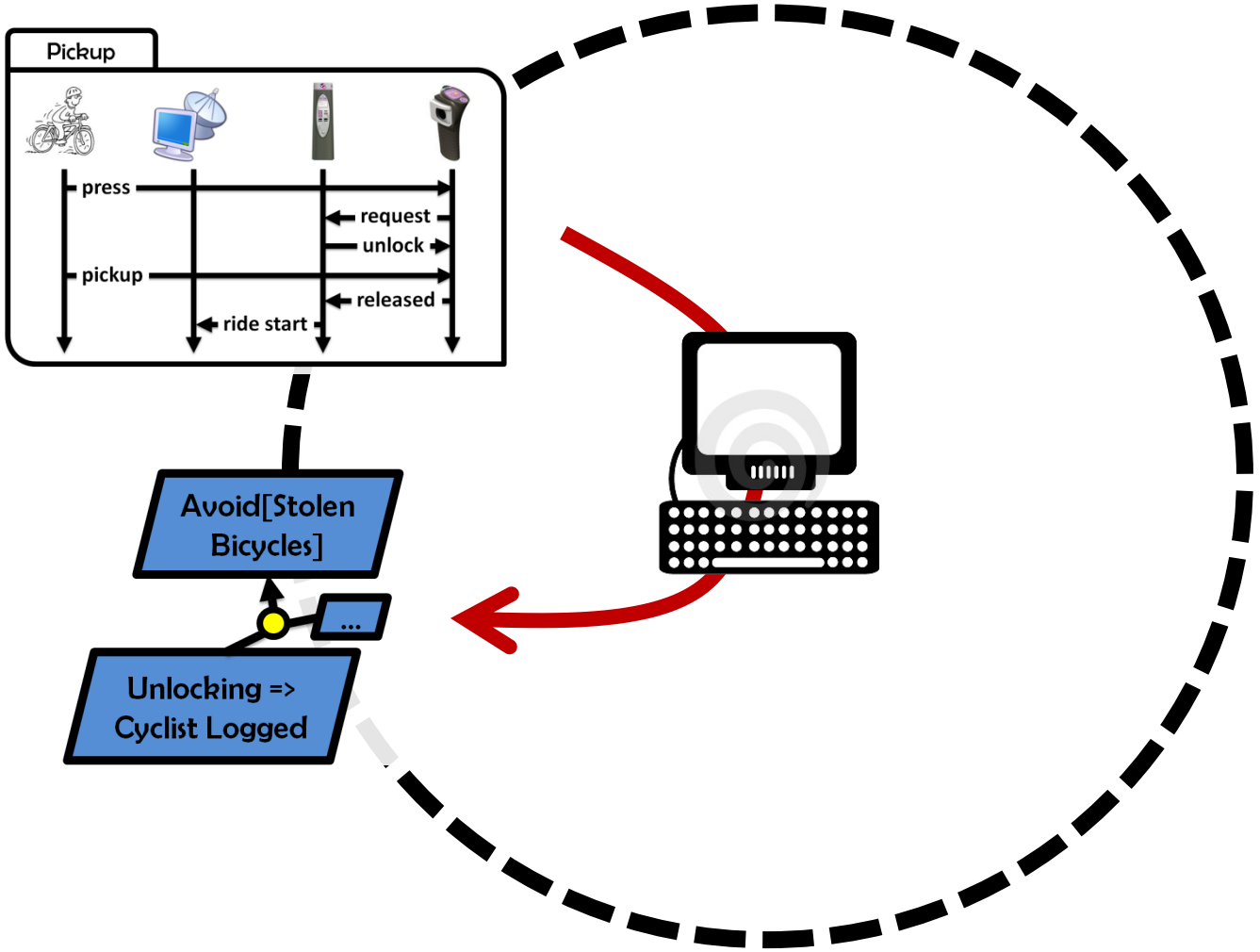
QSM
+ constraints
[Dam06, Dup08]

State machine induction from high-level scenarios



ASM:
Automaton
State
Merging
[Lam08]

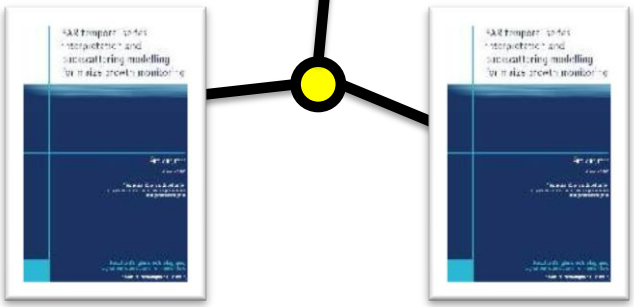
Extra goodness: goal inference from scenarios [Dam06, Dam11]



Why are they GREAT synthesis techniques?

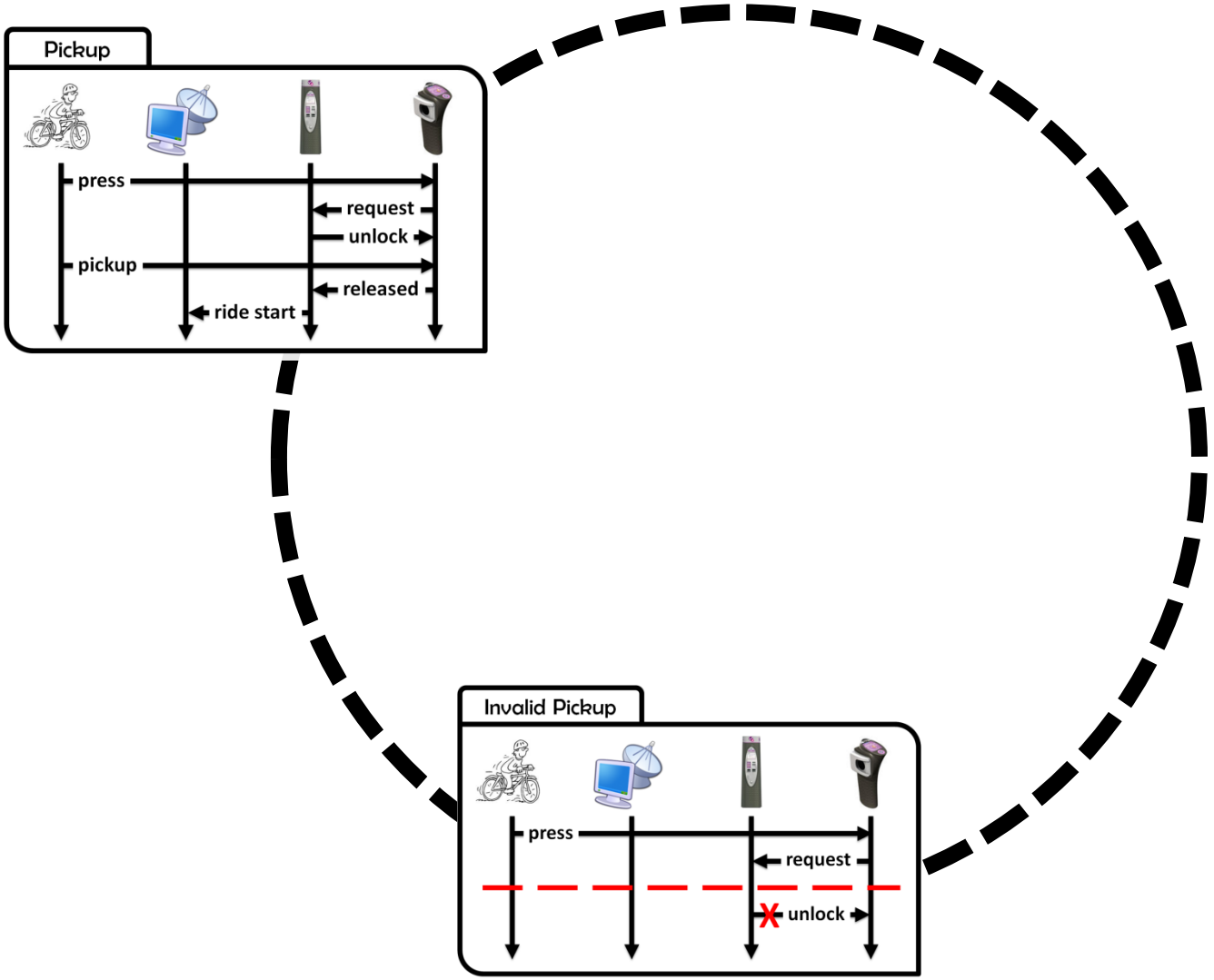
Effective support for system building

Models should be **adequate, consistent, complete, comprehensible, analyzable**

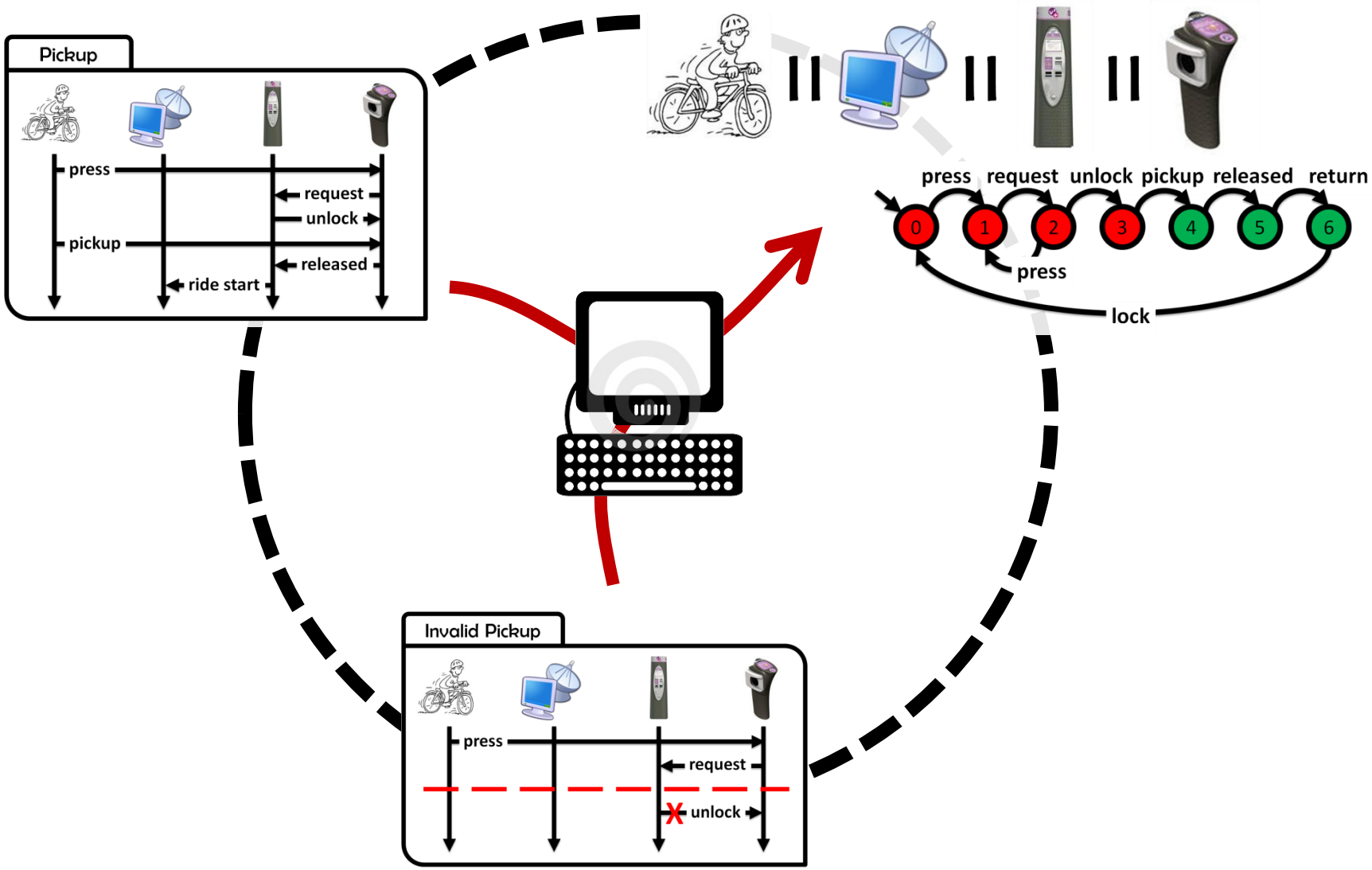


From models to software [...]

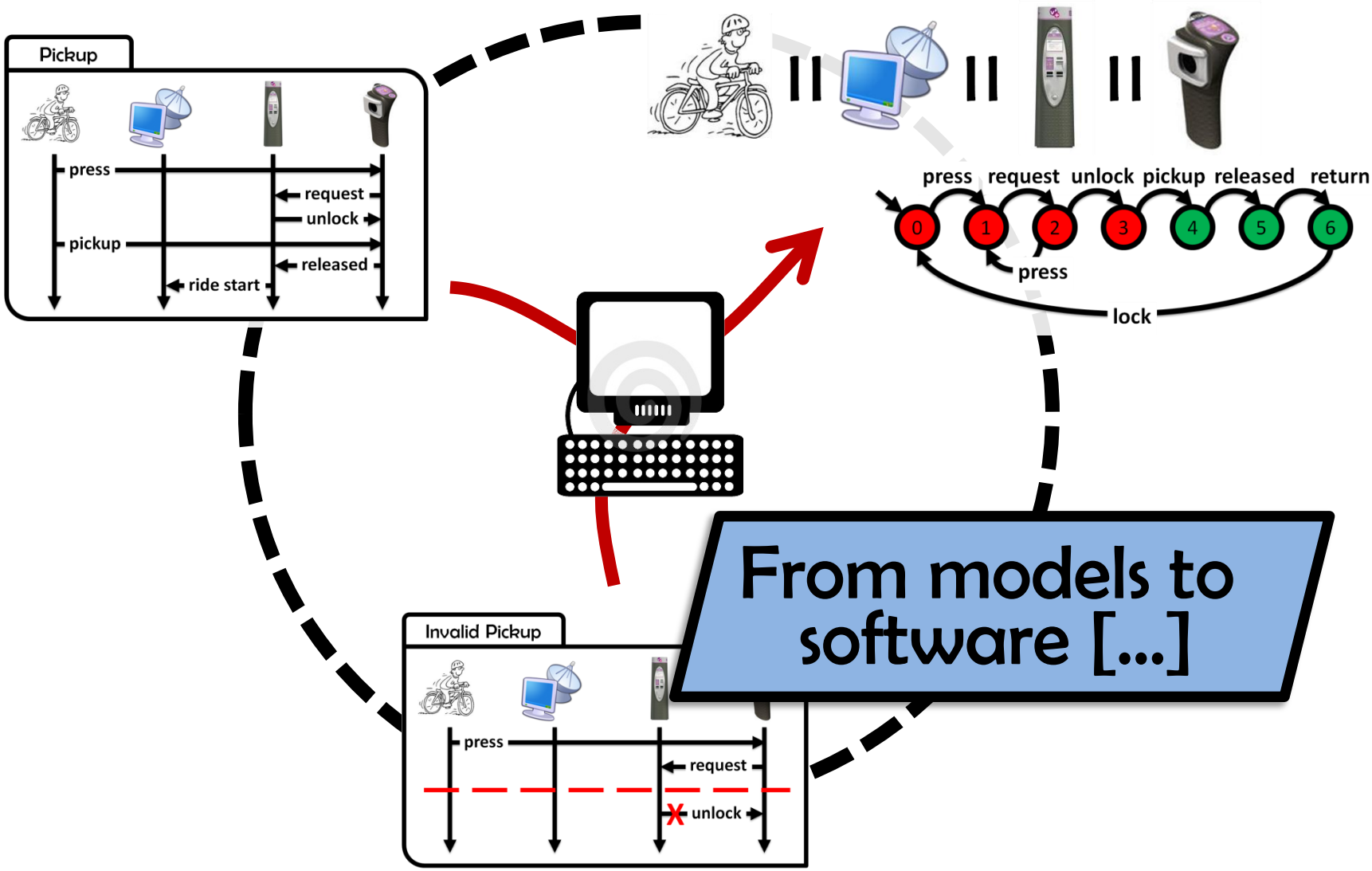
Input scenarios are comprehensible and easy to draw



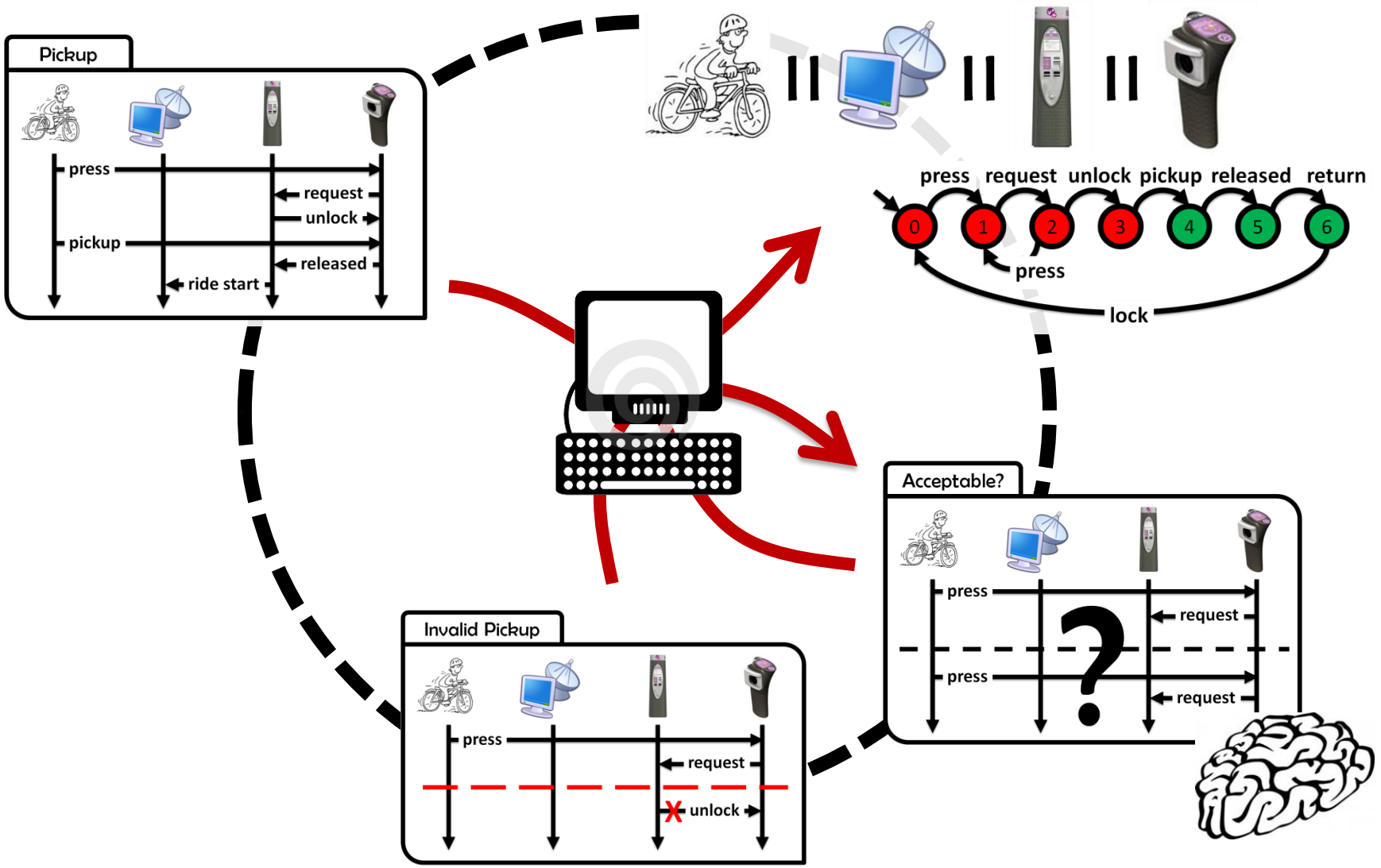
Our techniques synthesize consistent state machines



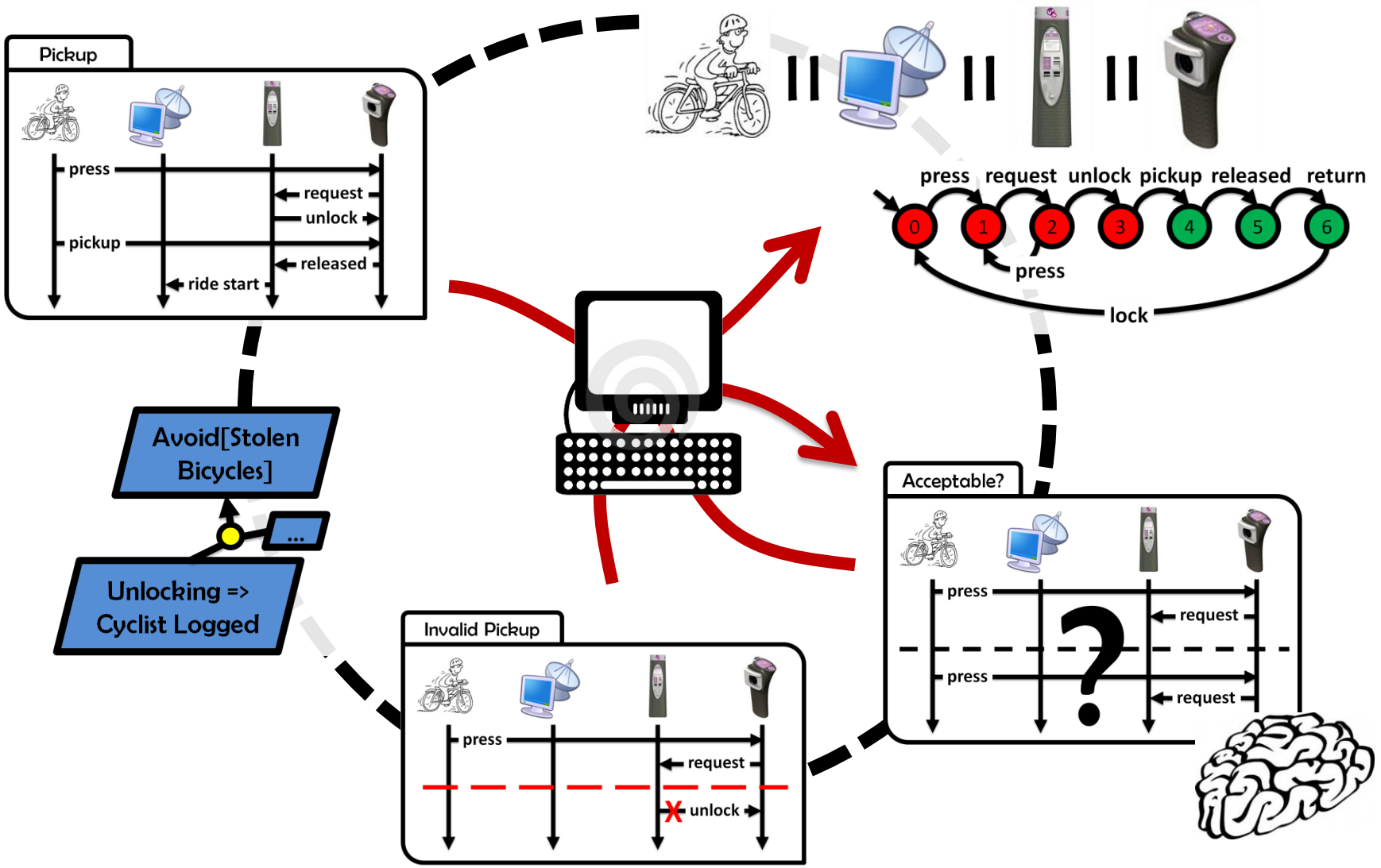
State machines are analyzable and close to the source code



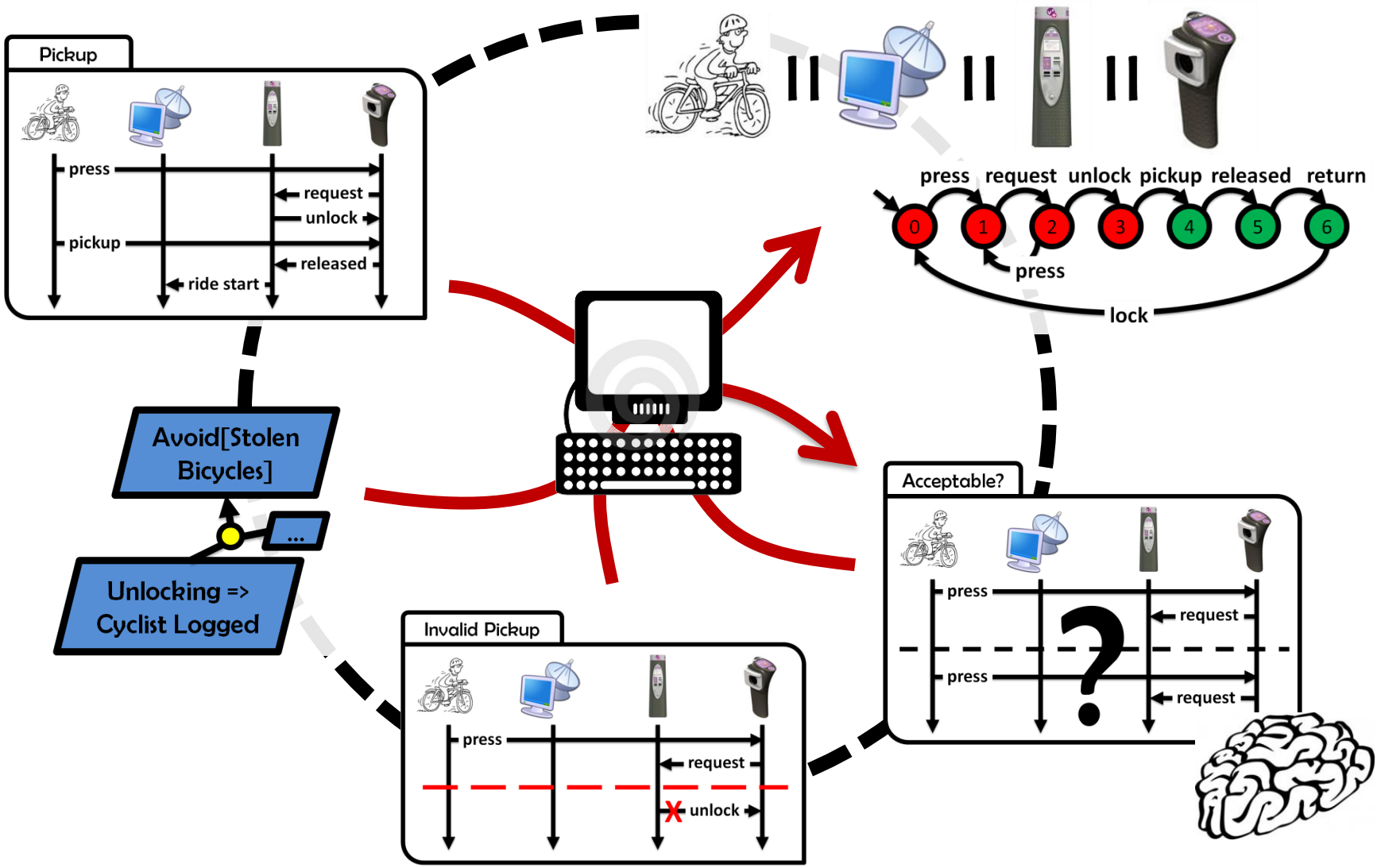
Thought provoking for better adequacy and completeness



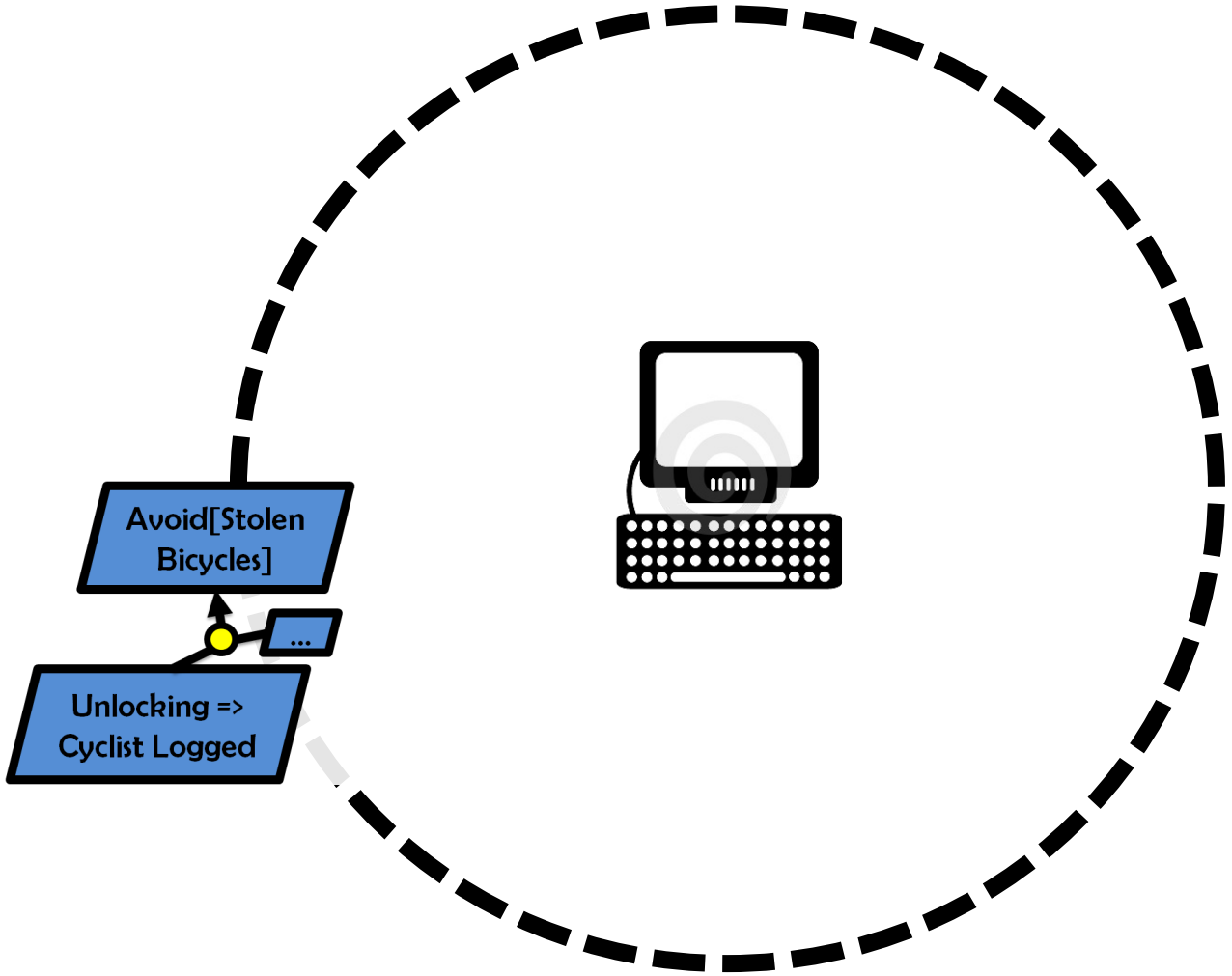
Knowing why makes you a much smarter software engineer



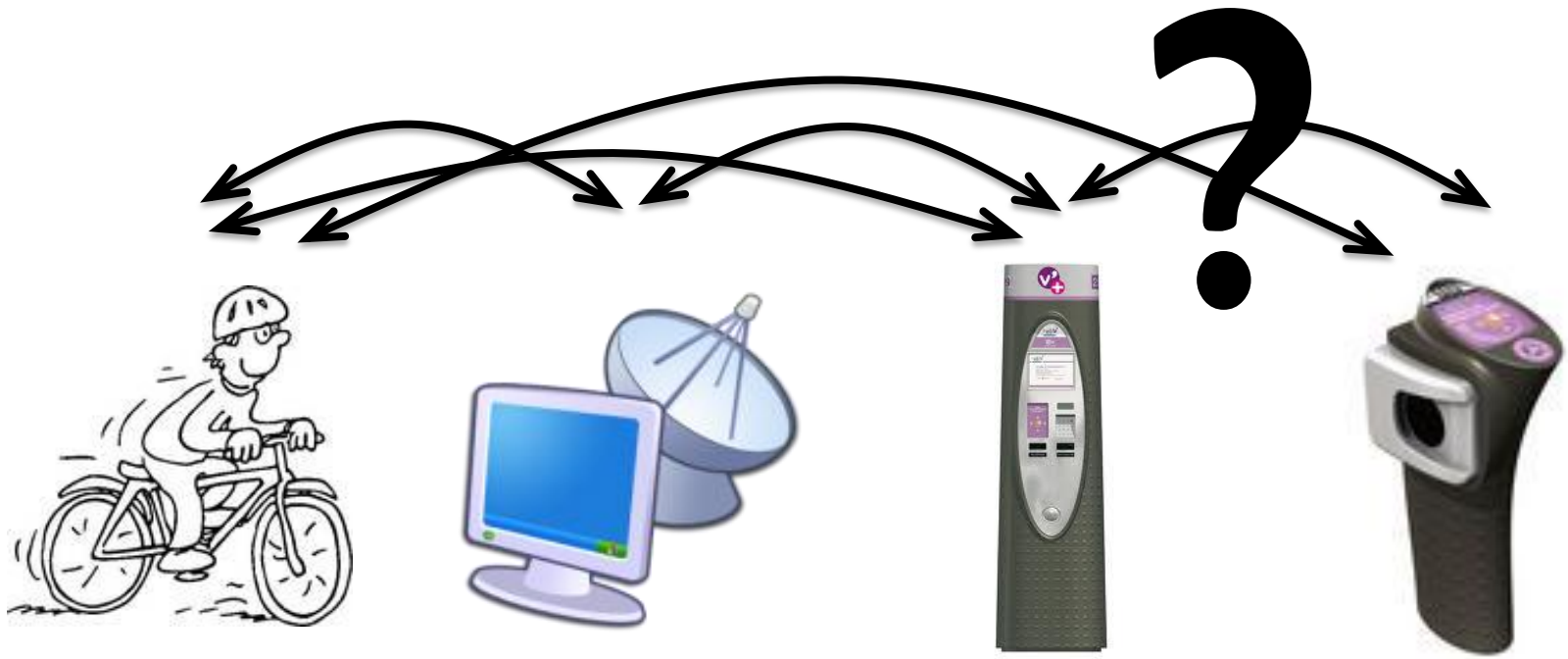
Synthesized state machines are consistent with known goals



The Operational changes, the Intentional stays

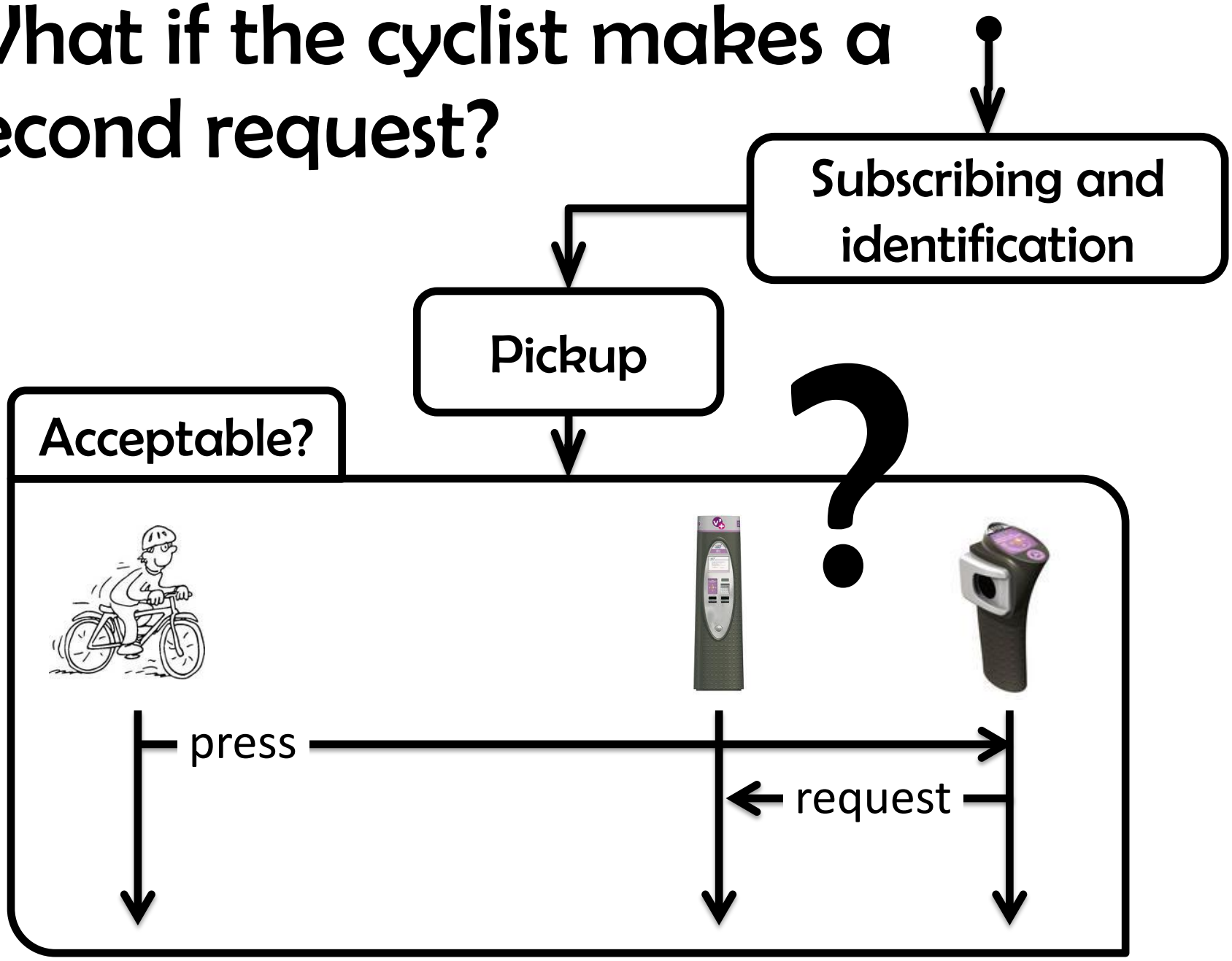


Scenario questions are thought provoking

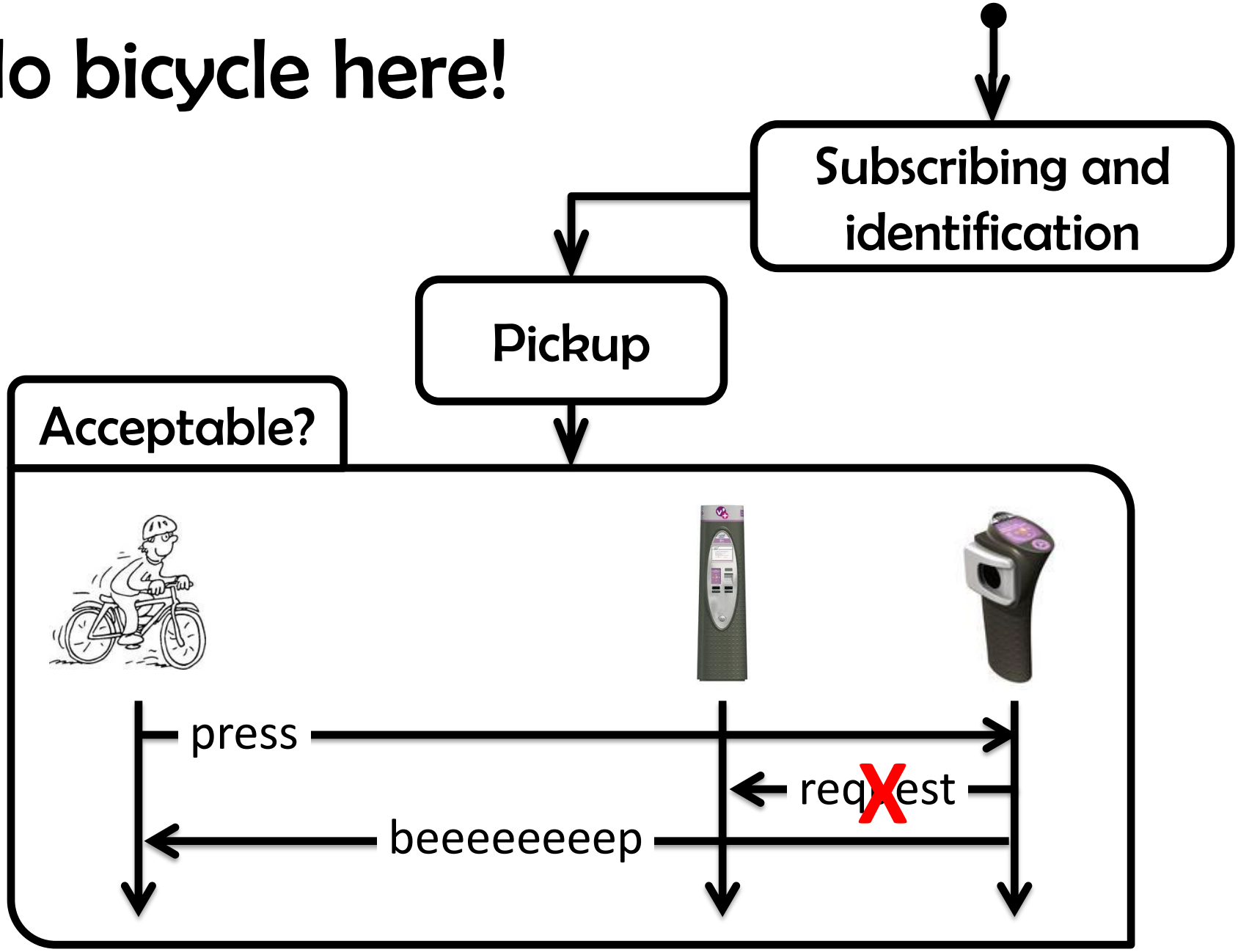


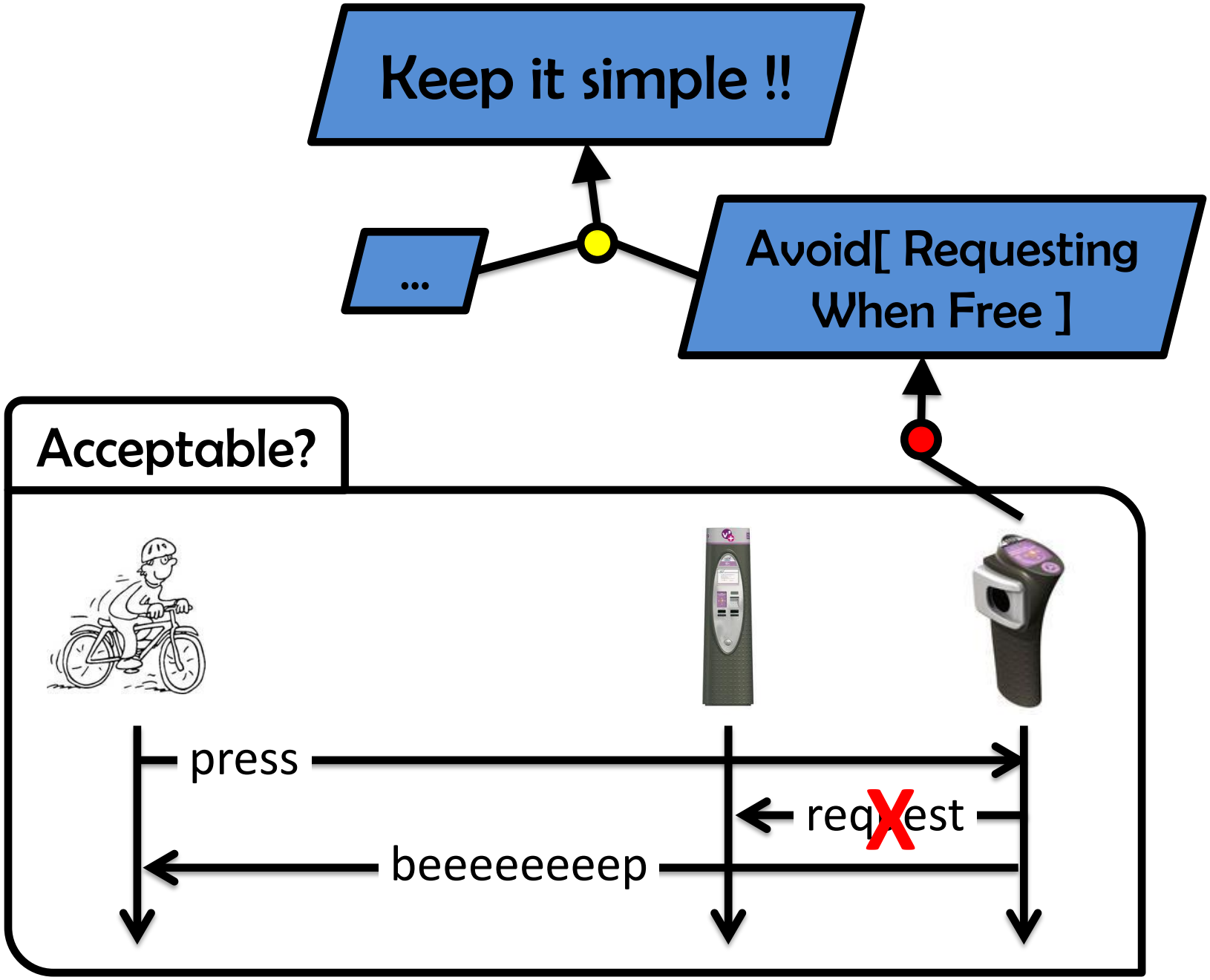
Some examples…

What if the cyclist makes a second request?



No bicycle here!




Keep it simple !!
...
Avoid[Requesting When Free]
Acceptable?
press
request
beeeeeeeeep

Subscribing and identification

Pickup

press

request

unlock

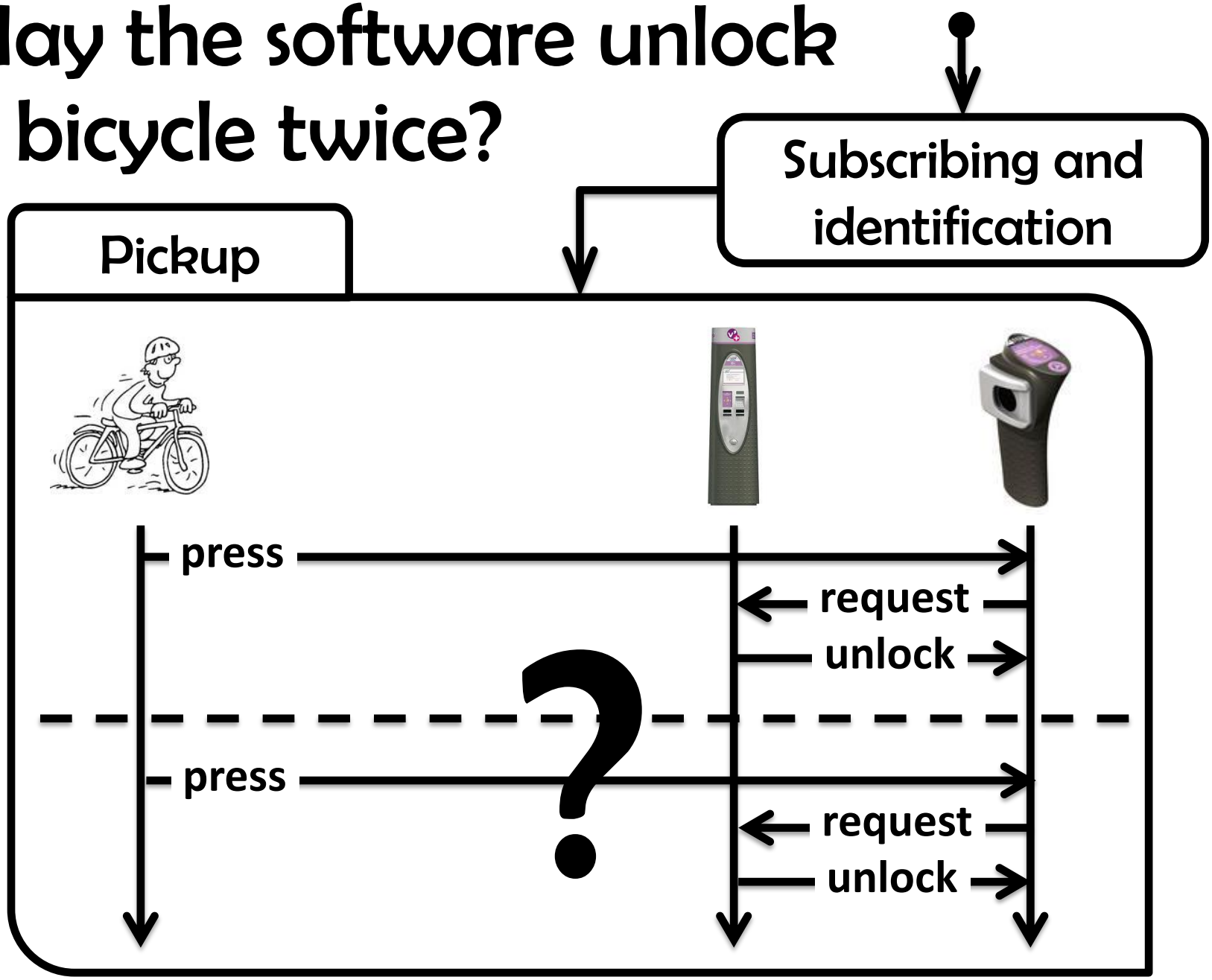
?

press

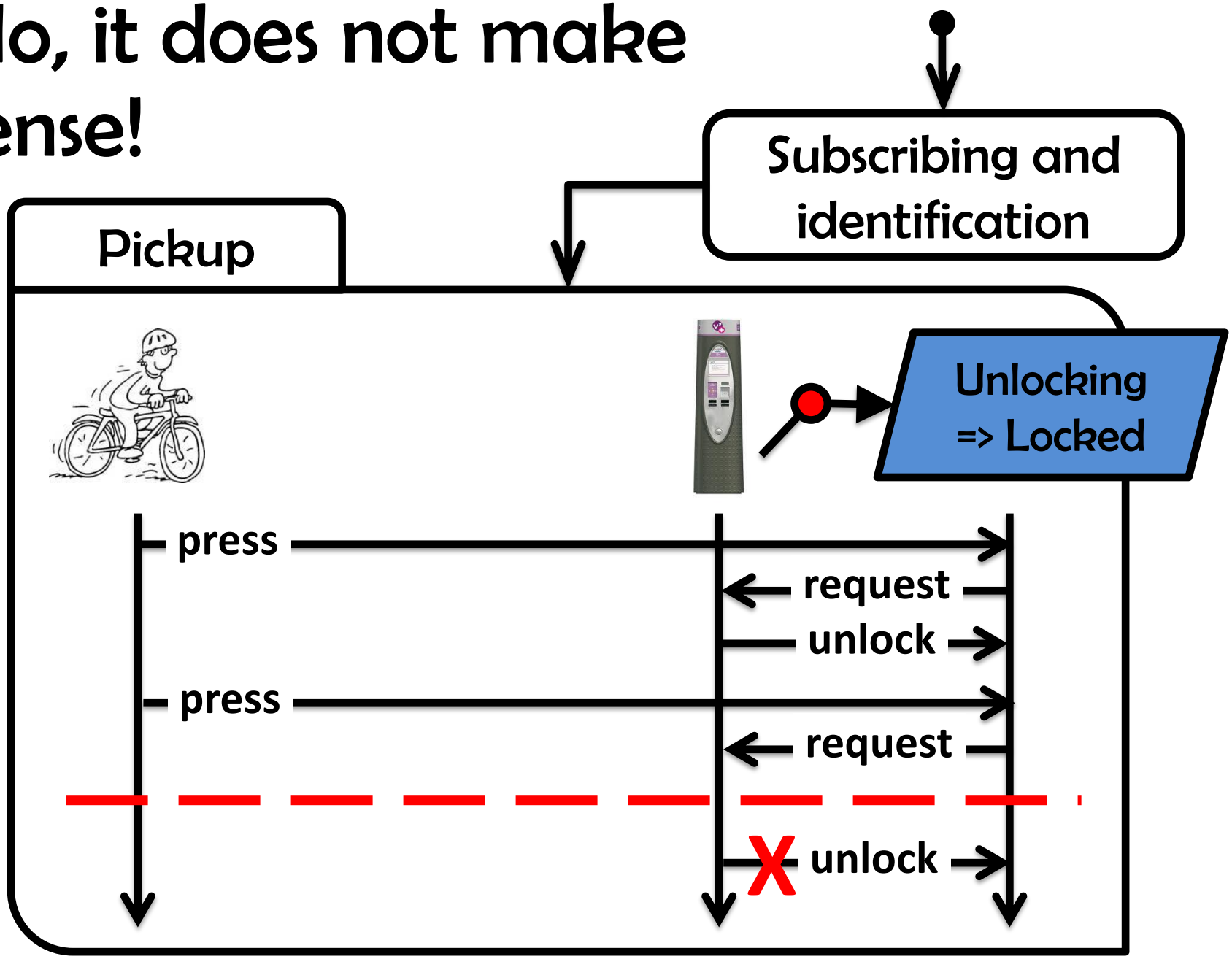
request

unlock

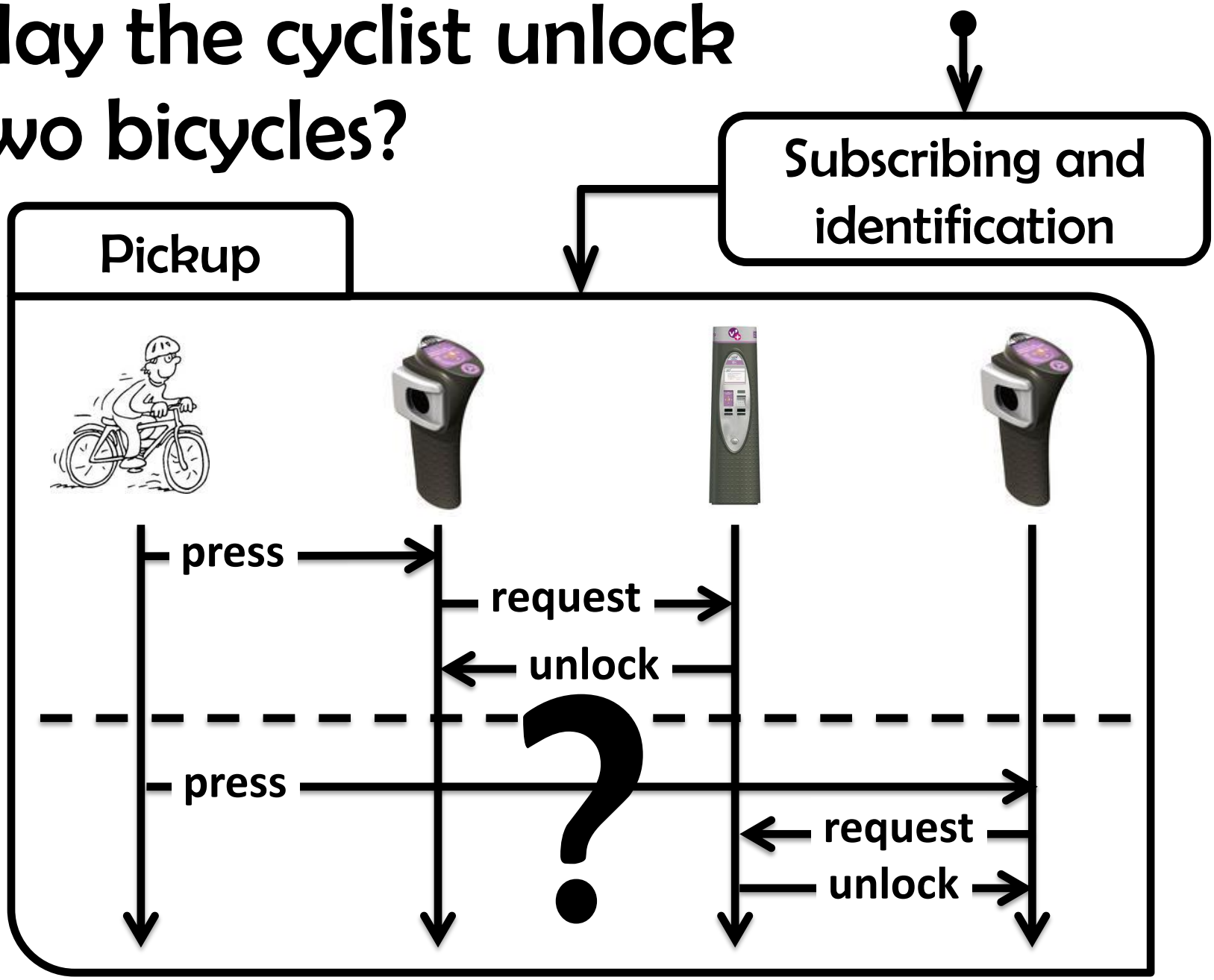
May the software unlock a bicycle twice?



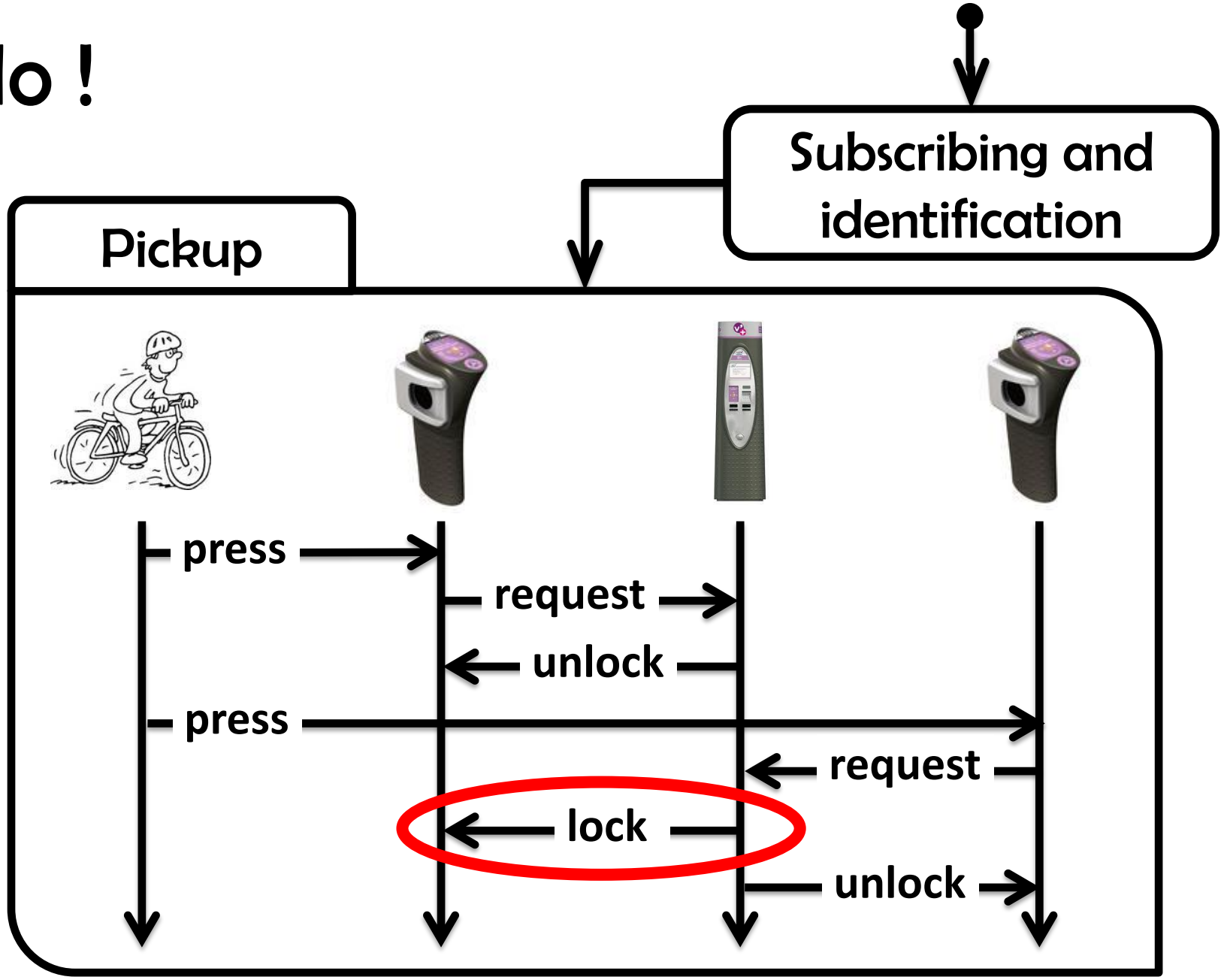
No, it does not make sense!



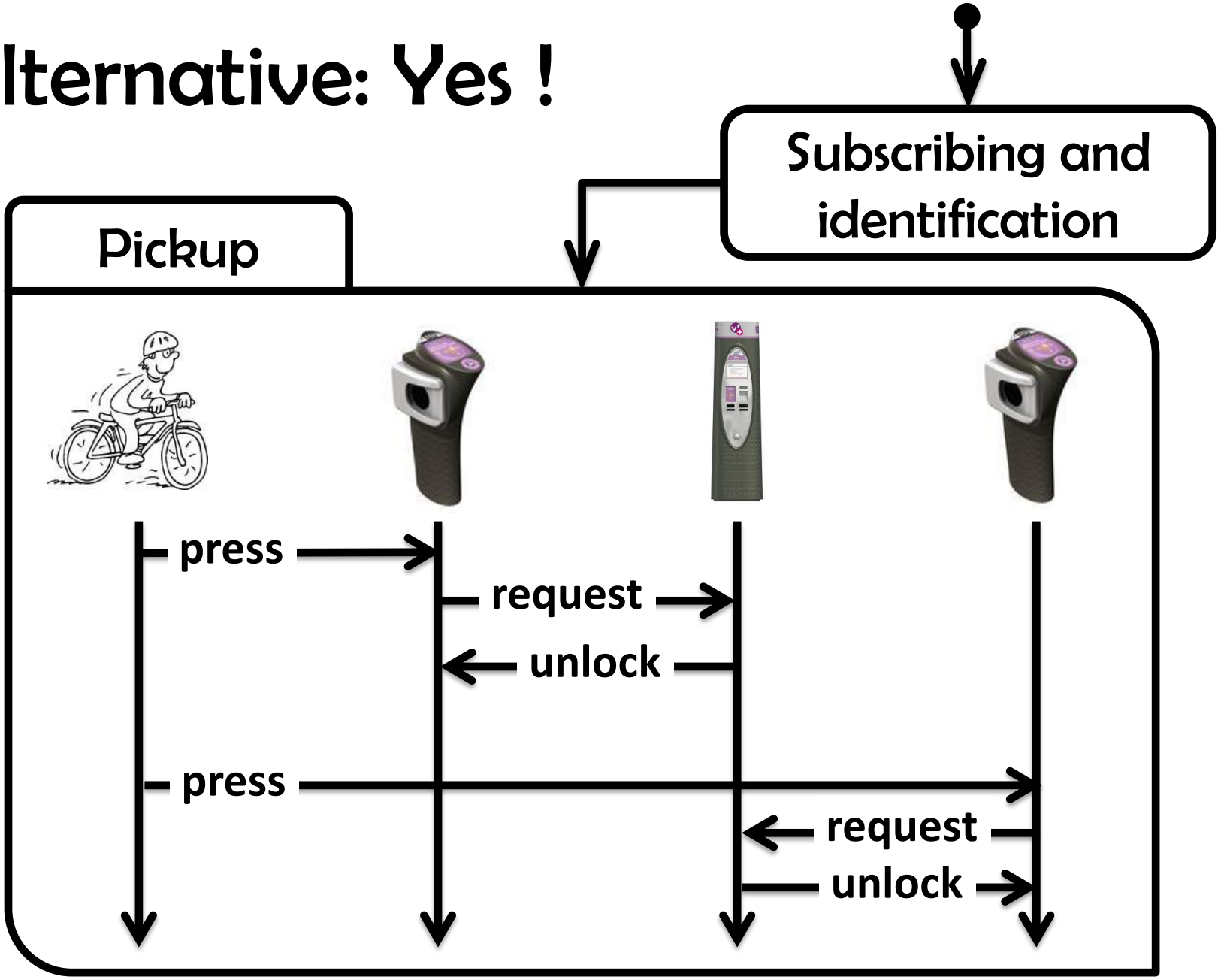
May the cyclist unlock two bicycles?



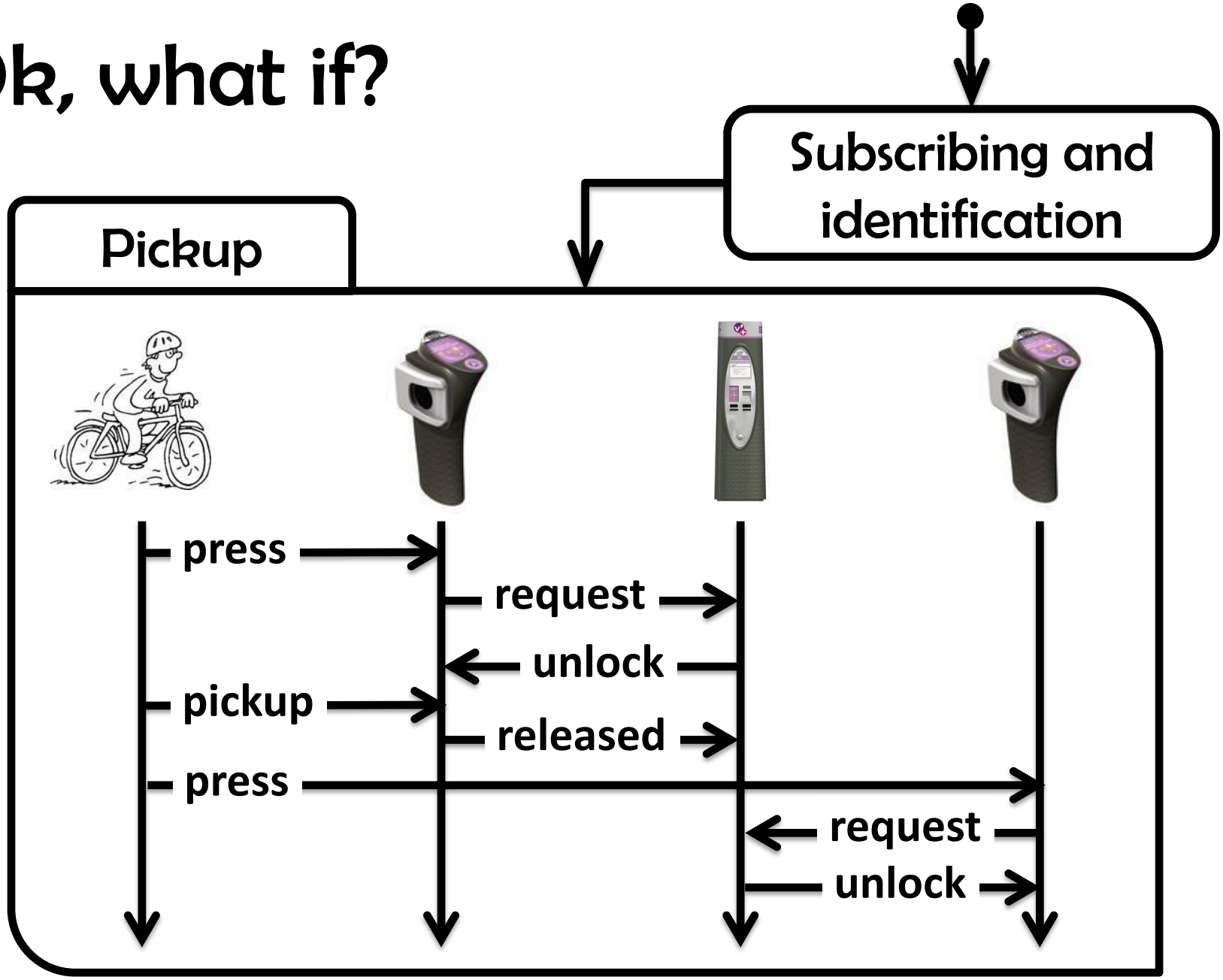
No !


Subscribing and identification
Pickup
press
request
unlock
press
request
lock
unlock

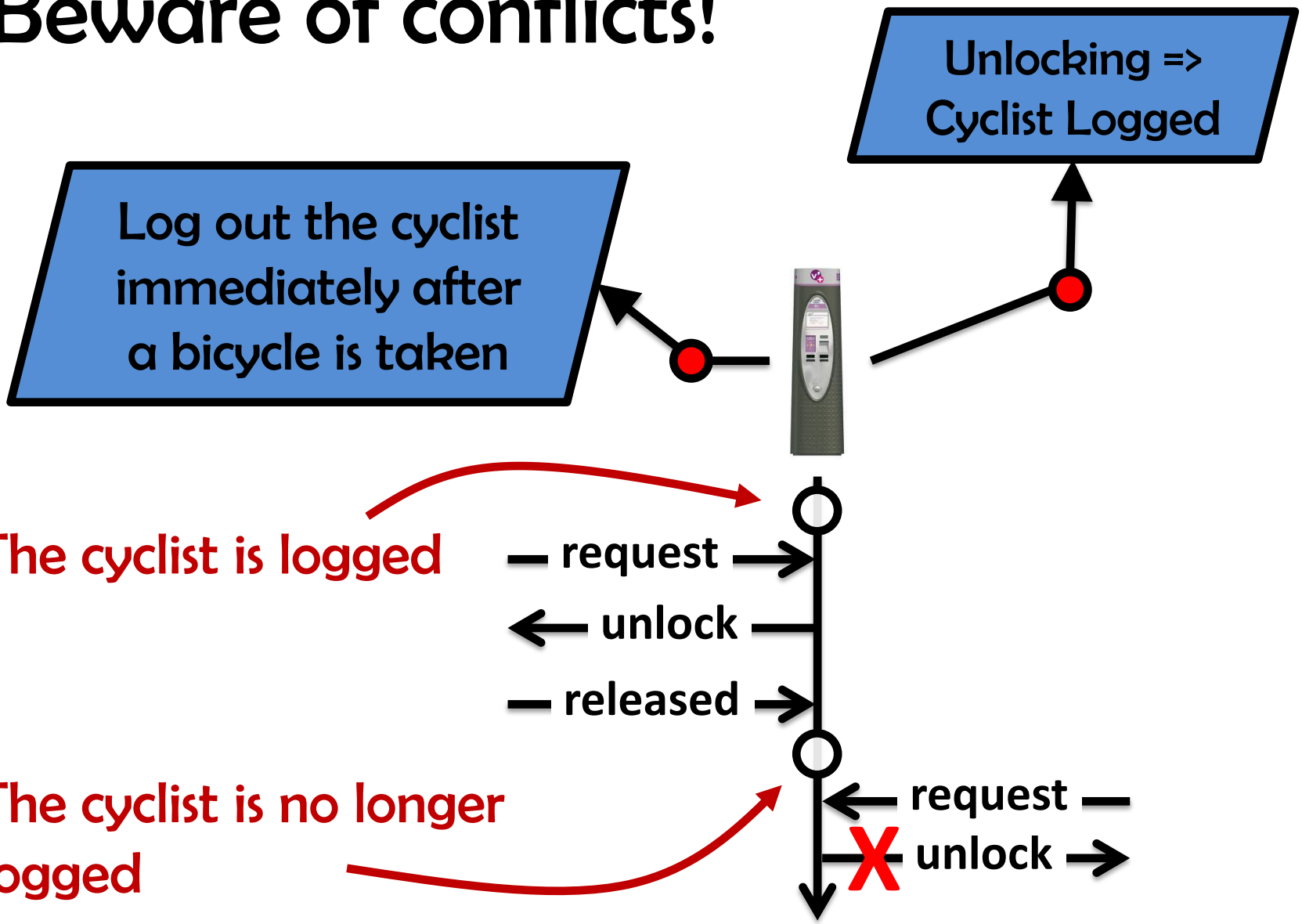
Alternative: Yes !



Ok, what if?

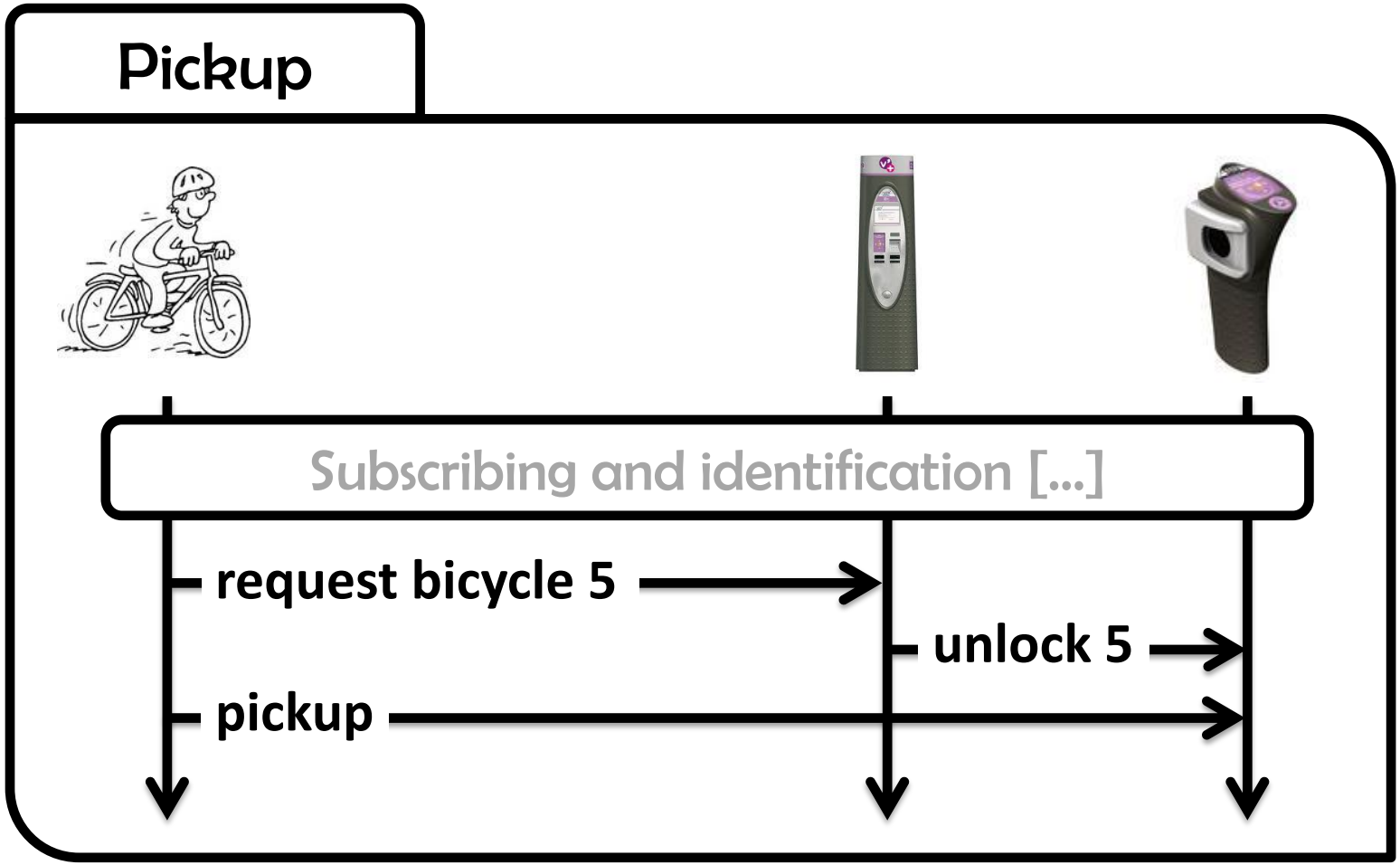

Subscribing and identification
Pickup
press
request
unlock
pickup
released
press
request
unlock

Beware of conflicts!

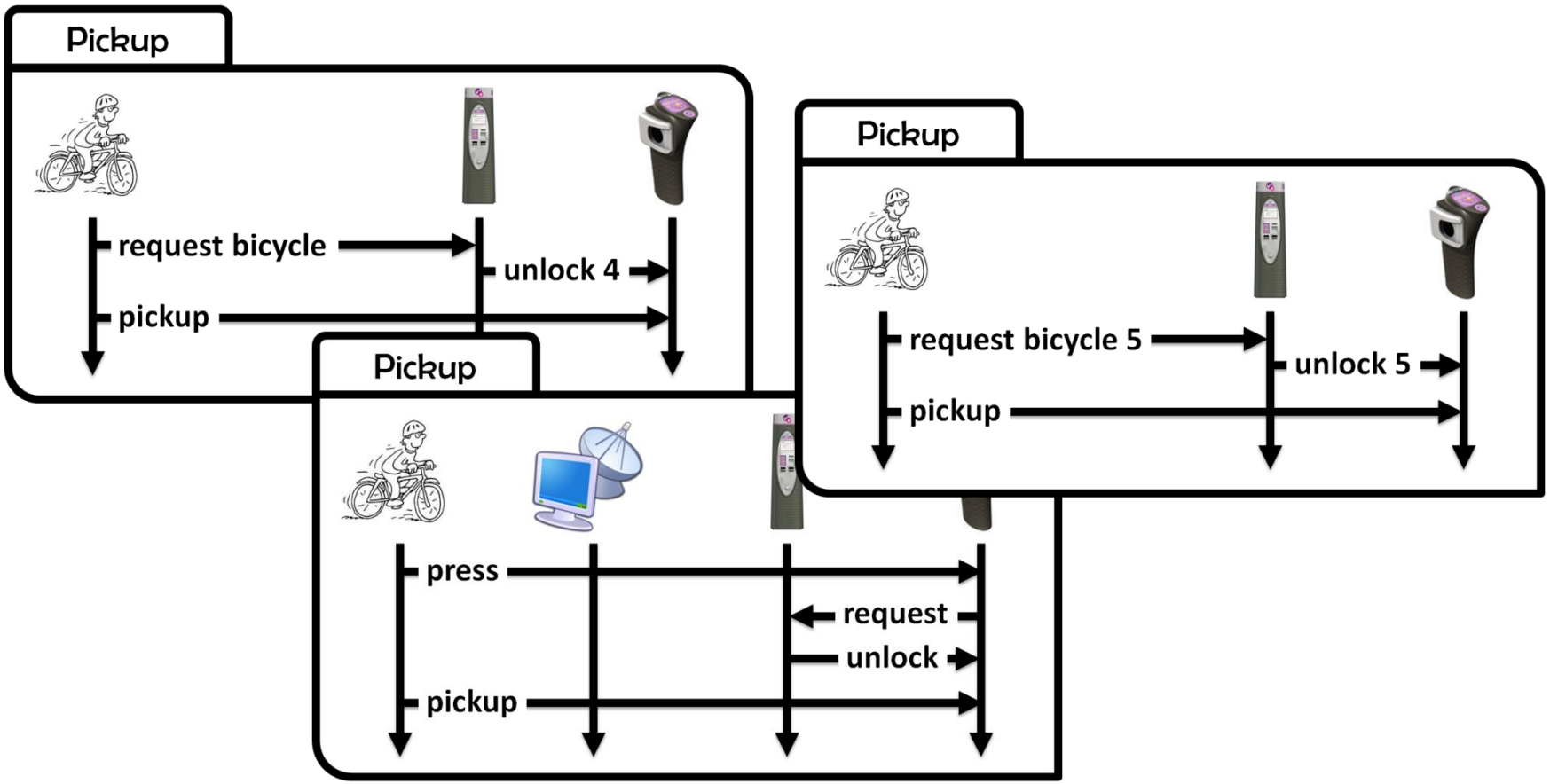


Conclusion

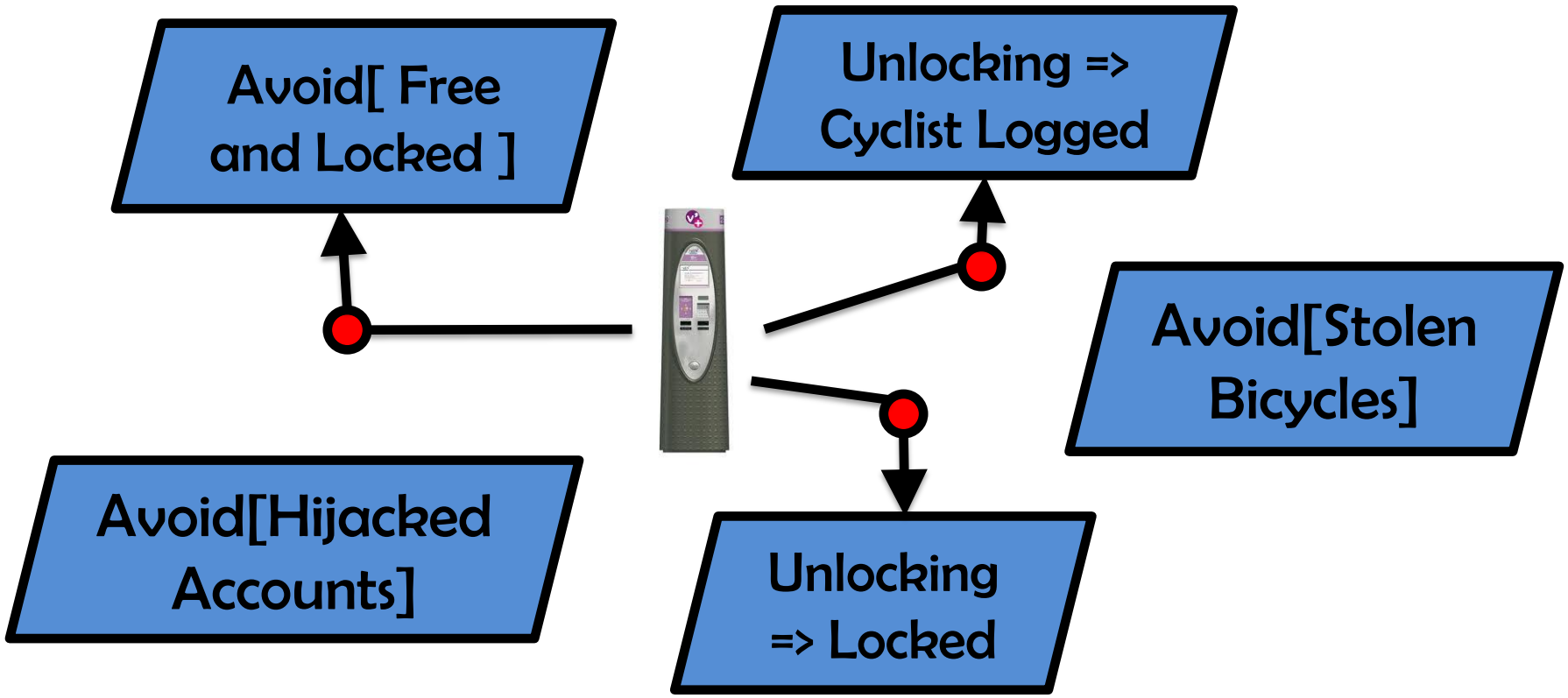
Why not this design?



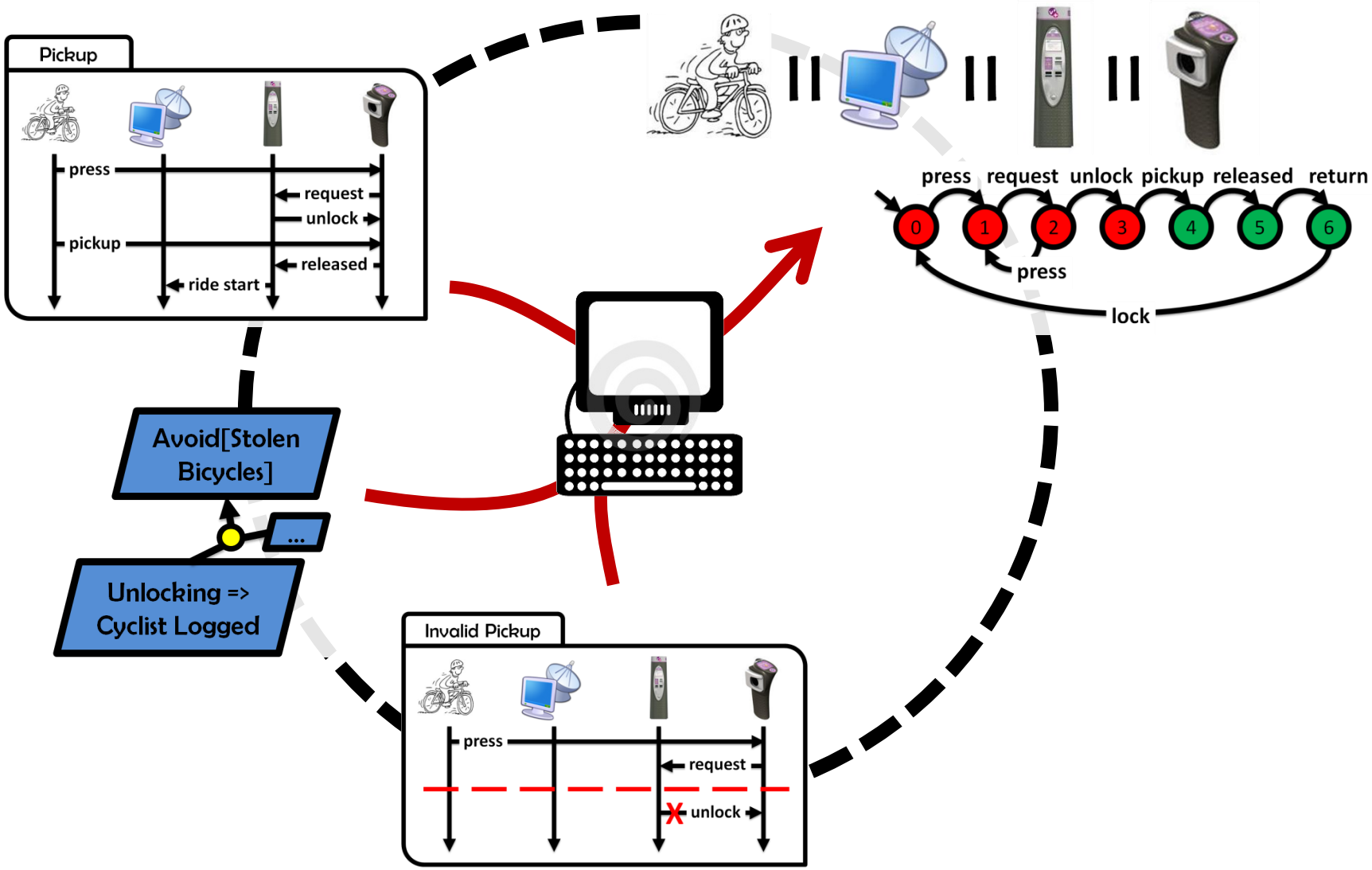
Models play a significant role for identifying requirements and exploring designs of software systems

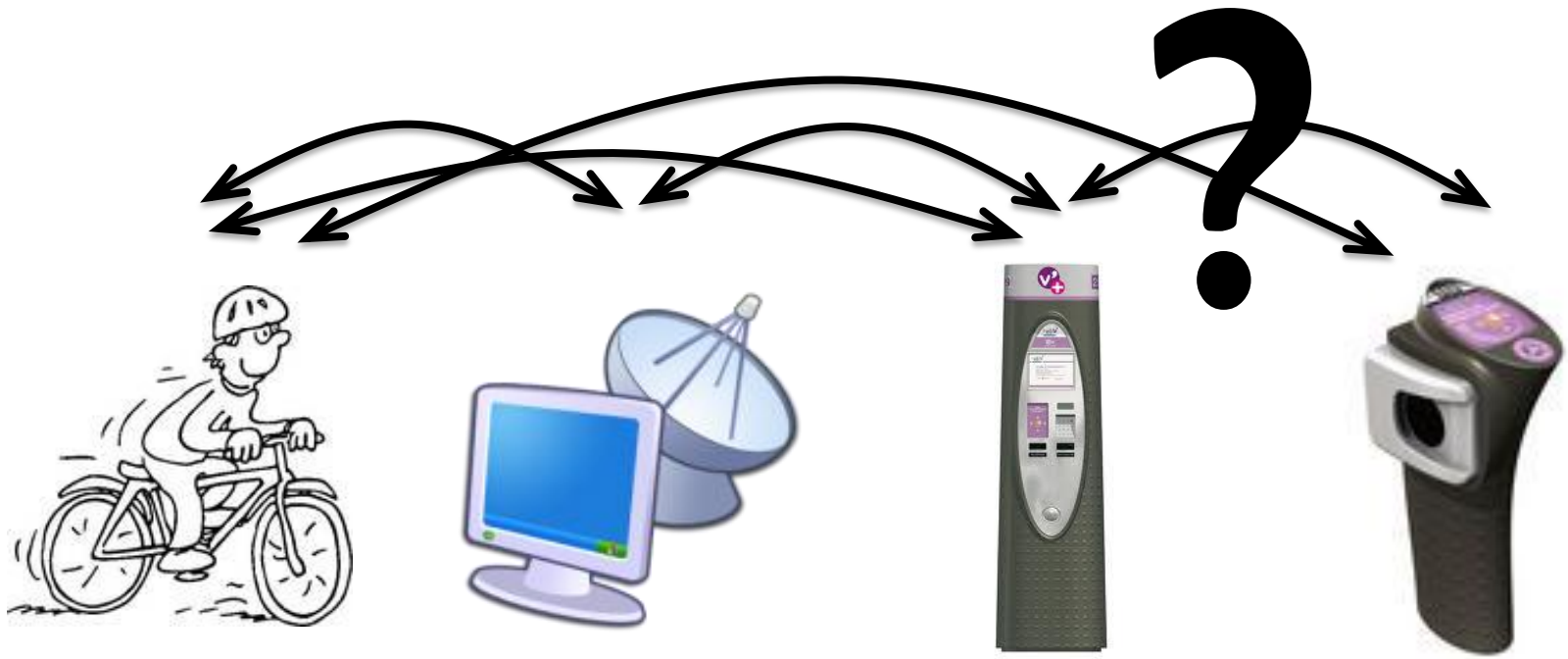


The Operational changes, the Intentional stays



Goals are a safety belt for the next modeling iteration

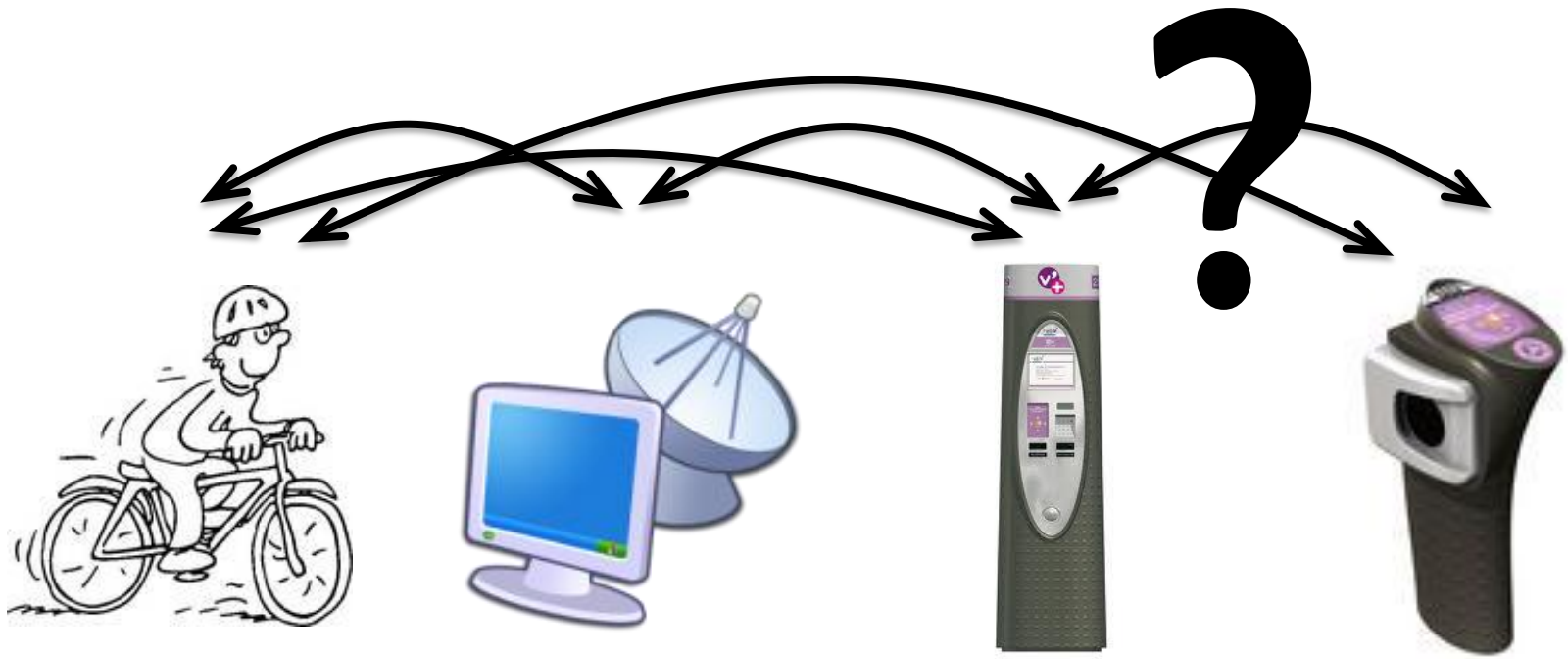




How do you feel ?

- About the models
  - You are confused about the different models and their relationships
- About the target system
  - You forgot some features in the first place
  - You think about some features I haven't considered
  - You agree with some of the features but disagree with others
  - You are confused about the system and what should be done now

Keep calm !



The hardest part of software development is determining what the system should do [Bro87]

But there is support for this !

Thank you !

References

[Avl09] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Wiley, March 2009.

[Bro87] F.P. Brooks, *No silver bullet: Essence and accidents of software engineering*. IEEE Computer, 20(4):10-19, April 1987.

[Dam05] C. Damas, B. Lambeau, P. Dupont and A. van Lamsweerde, Generating annotated behavior models from end-user scenarios. IEEE Transactions on Software Engineering, 31(12):1056-1073, 2005.

[Dam06] C. Damas, B. Lambeau and A. van Lamsweerde, *Scenarios, goals, and state machines: a win-win partnership for model synthesis.* In International ACM Symposium on the Foundations of Software Engineering, pages 197{207, Portland, Oregon, November 2006.

[Dam11] C. Damas, *Analyzing Multi-View Models of Software Systems*, PhD thesis, Université catholique de Louvain, Louvain-la-Neuve, 2011.

[Dup08] P. Dupont, B. Lambeau, C. Damas and A. van Lamsweerde, *The QSM algorithm and its application to software behavior model induction.* Applied Artificial Intelligence, 22:77-115, 2008.

[Fea87] M.S. Feather, *Language support for the specification and development of composite systems.* ACM Transactions on Programming Languages and Systems, 9:198-234, March 1987.

References

[Fin92] A. Finkelstein, J. Kramer, B. Nuseibeh, A. Finkelstein and M. Goedicke, *Viewpoints: A Framework for Integrating Multiple Perspectives in System Development*. 2:31-57+, 1992.

[Hoa85] C.A.R. Hoare, *Communicating Sequential Processes,* Prentice Hall International Series in Computing Science, Prentice Hall, April 1985.

[Lam08] B. Lambeau, C. Damas and P. Dupont, *State-merging dfa induction algorithms with mandatory merge constraints.* In ICGI'08: Proceedings of the 9th international colloquium on Grammatical Inference, pages 139{153, Berlin, Heidelberg, 2008. Springer-Verlag.

[Mil89] R. Milner, *Communication and concurrency*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[Mag99] J. Magee and J. Kramer. *Concurrency: State Models and Java Programs,* Wiley, 1999.

[Onc92] J. Oncina and P. Garcia, *Inferring regular languages in polynomial update time.* In N. Pierez de la Blanca, A. Sanfeliu, and E.Vidal, editors, Pattern Recognition and Image Analysis, volume 1 of Series in Machine Perception and Artificial Intelligence, pages 49-61. World Scientific, Singapore, 1992.