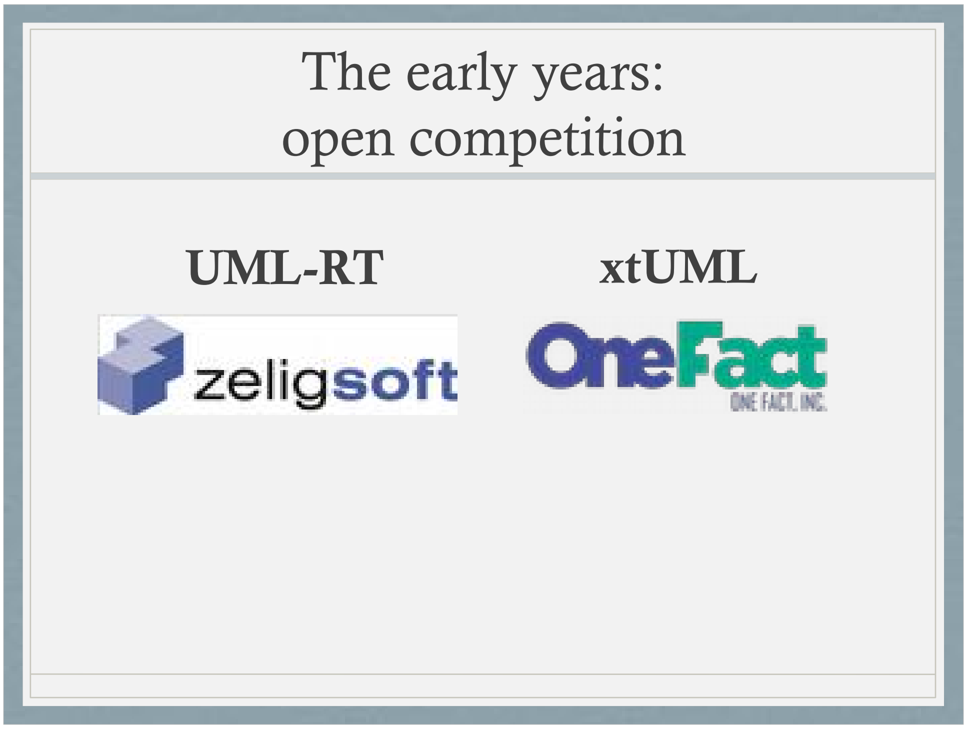
A Tale of Two Dialects

(in three acts)

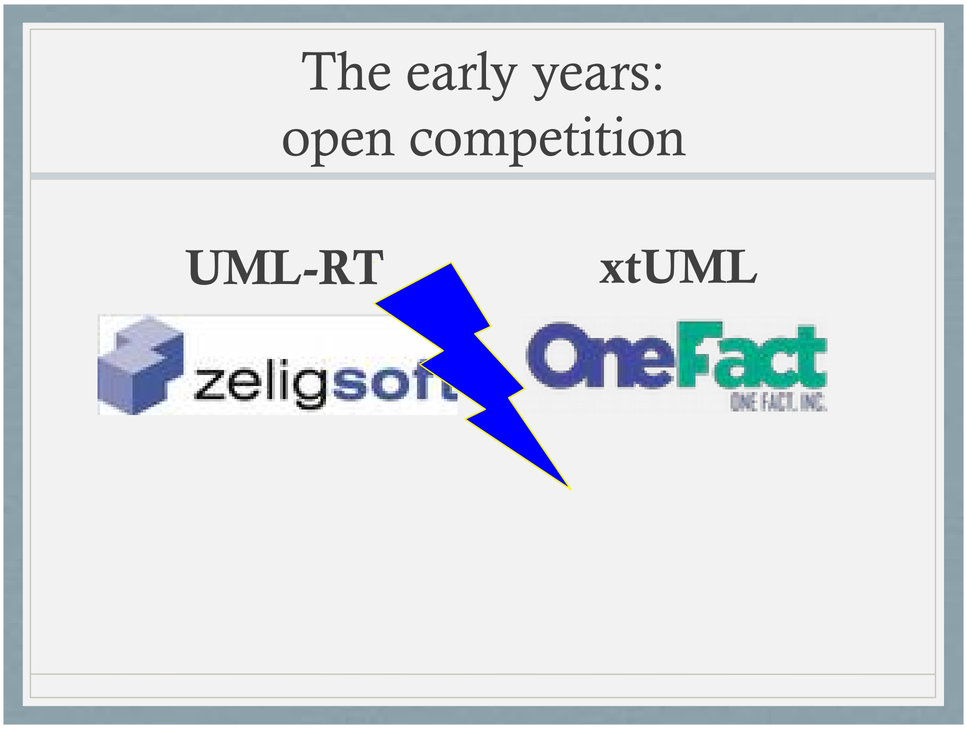


The early years: open competition

UML-RT **xtUML**

{Slide displays: UML-RT slide showing UML-RT elements}

R: UML-RT is a domain-specific language based on UML to simplify modeling so all other approaches suck!



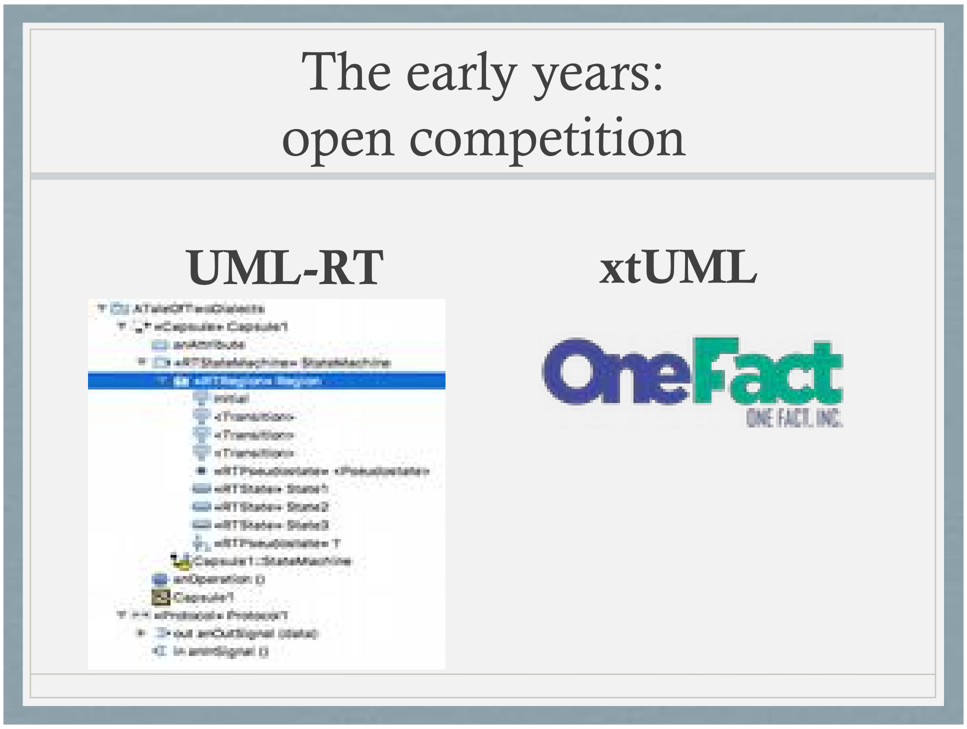
The early years: open competition

UML-RT

xtUML

{Slide displays: UML-RT slide showing UML-RT elements}

R: UML-RT is a domain-specific language based on UML to simplify modeling so all other approaches suck!



The early years: open competition

UML-RT

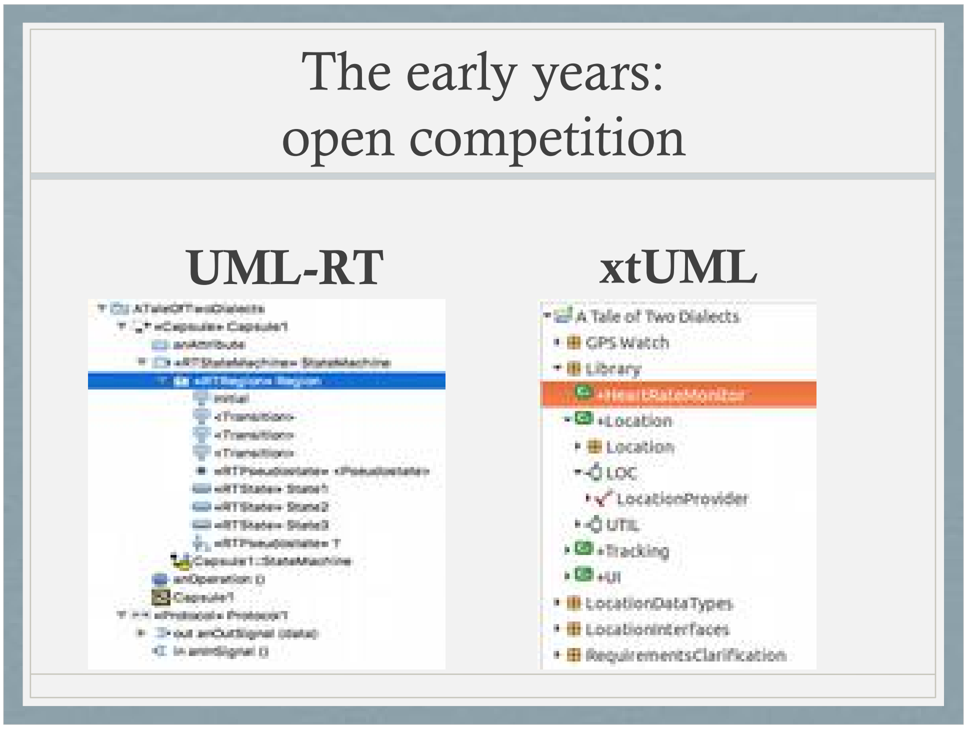
```
▾ ATaleOfTwoDialects
  ▾ «Capsule» Capsule1
      anAttribute
    ▾ «RTStateMachine» StateMachine
      ▾ «RTRegion» Region
          Initial
          <Transition>
          <Transition>
          <Transition>
          «RTPseudostate» <Pseudostate>
          «RTState» State1
          «RTState» State2
          «RTState» State3
          «RTPseudostate» T
        Capsule1::StateMachine
      anOperation ()
      Capsule1
  ▾ «Protocol» Protocol1
    ▸ out anOutSignal (data)
      in anInSignal ()
```

xtUML

OneFact
ONE FACT, INC.

{Slide displays: xtUML slide showing classes and components}

S: xtUML is based (almost) directly on UML, so yours is the one for sissies!



The early years: open competition

UML-RT

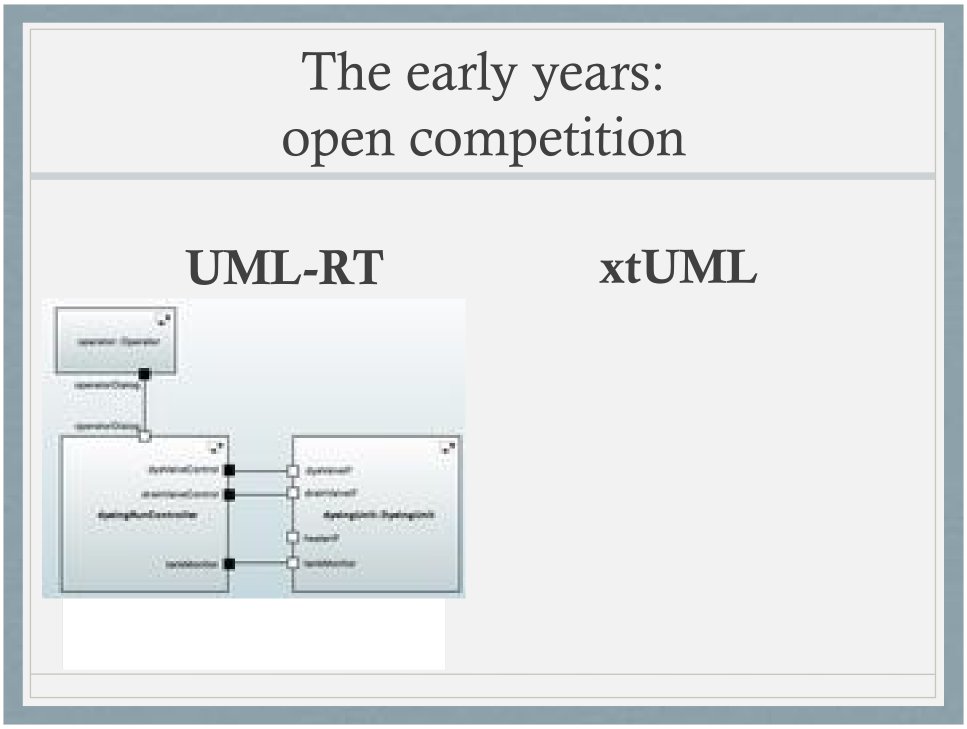
ATaleOfTwoDialects
«Capsule» Capsule1
anAttribute
«RTStateMachine» StateMachine
«RTRegion» Region
initial
<Transition>
<Transition>
<Transition>
«RTPseudostate» <Pseudostate>
«RTState» State1
«RTState» State2
«RTState» State3
«RTPseudostate» T
Capsule1::StateMachine
anOperation ()
Capsule1
«Protocol» Protocol1
out anOutSignal (data)
in anInSignal ()

xtUML

A Tale of Two Dialects
GPS Watch
Library
+HeartRateMonitor
+Location
Location
LOC
LocationProvider
UTIL
+Tracking
+UI
LocationDataTypes
LocationInterfaces
RequirementsClarification

{Slide displays: xtUML slide showing classes and components}

S: xtUML is based (almost) directly on UML, so yours is the one for sissies!



The early years: open competition

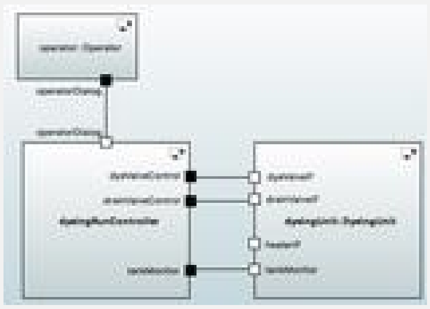
UML-RT **xtUML**

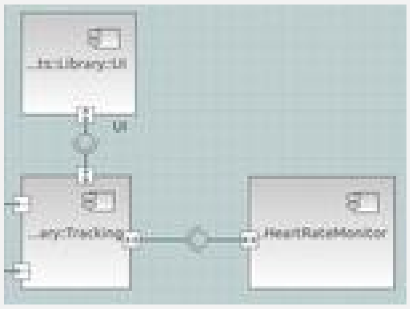
{Slide displays: UML-RT slide showing simple capsule diagram}

R: Well, my tool let’s you use capsules to represent the system hierarchy and protocols to carry messages - so, this tool is the best and the other one is lame.

The early years: open competition

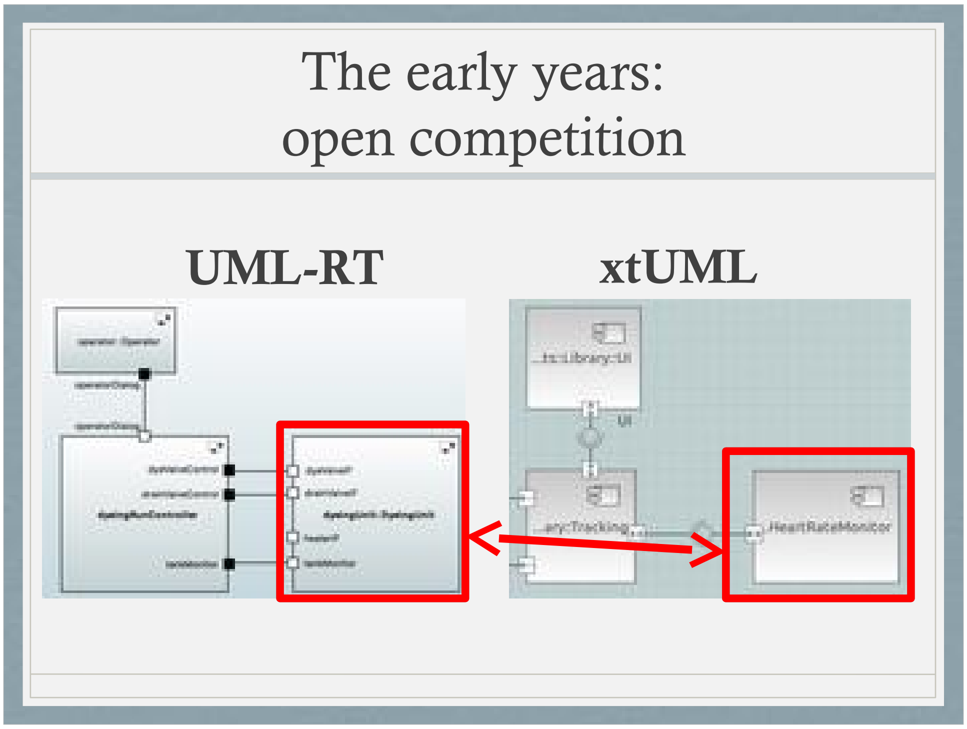
UML-RT **xtUML**





{Slide displays: xtUML slide showing classes and components}

S: Mine shows classes and components and provides formalism, so yours is the one that stinks.

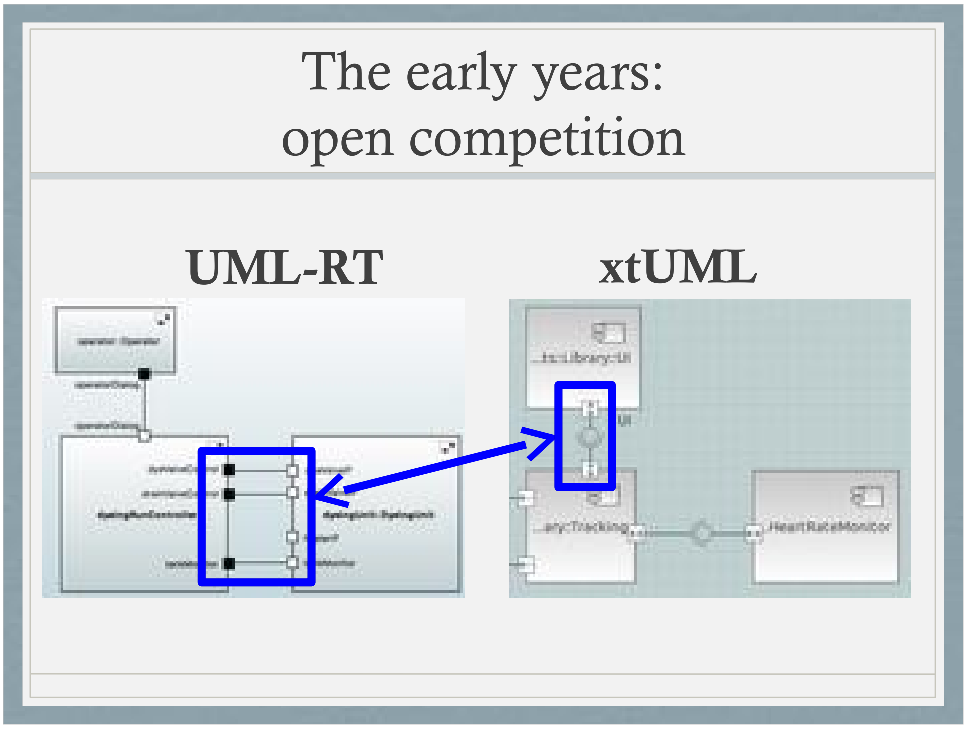


The early years: open competition

UML-RT

xtUML

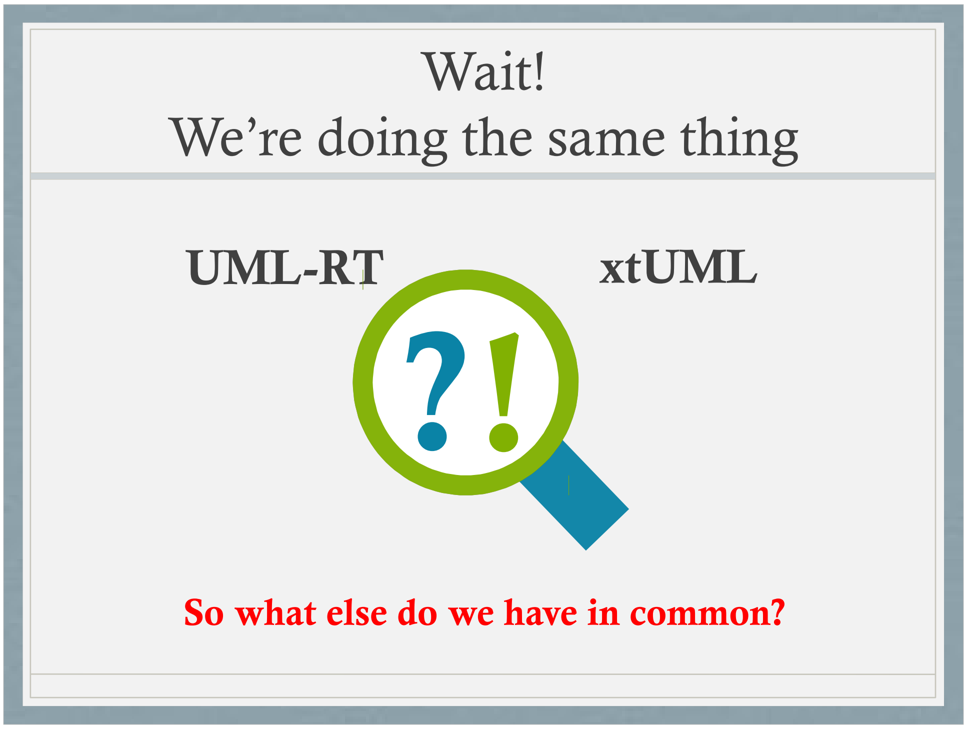
{Slide displays: highlight capsules and components,
add double arrow between them.}
R: Wait… My capsules look like your components!



The early years: open competition

UML-RT **xtUML**

{Slide displays: ~~Remove highlight for capsules and components, remove arrows;~~
highlight interfaces, dependencies, protocols and connectors;
add double arrows between similar elements}
S: And my interfaces and dependencies look like your protocols and connectors!



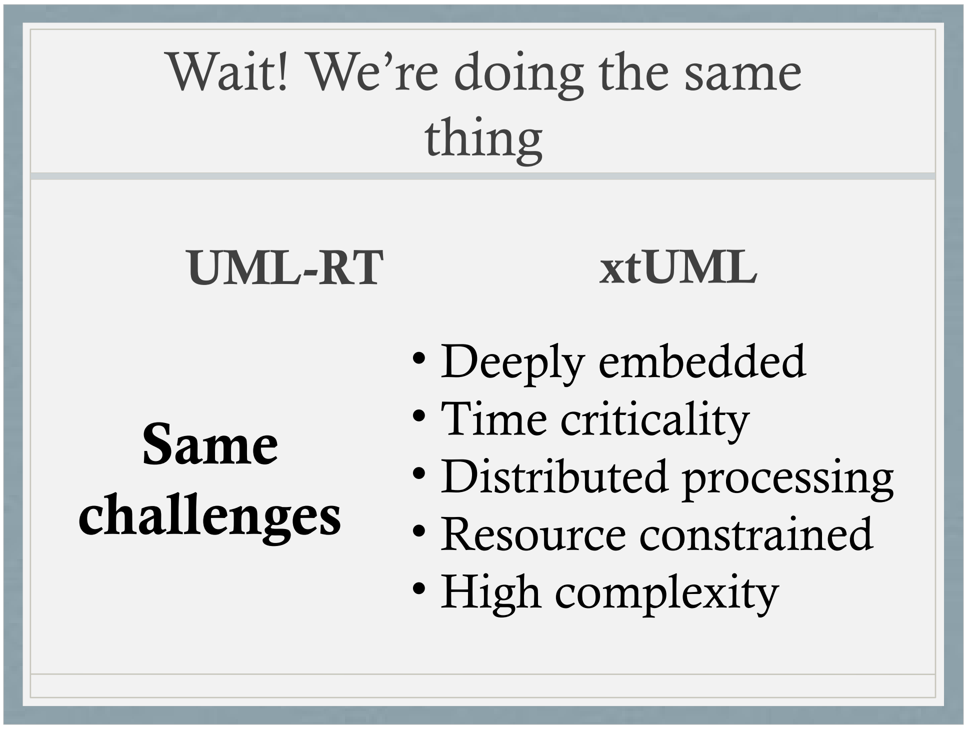
Wait!
We're doing the same thing

UML-RT **xtUML**

So what else do we have in common?

{Slide displays: add a "?" image in the middle}

R: Well then… what else do we have in common?



Wait! We're doing the same thing

UML-RT **xtUML**

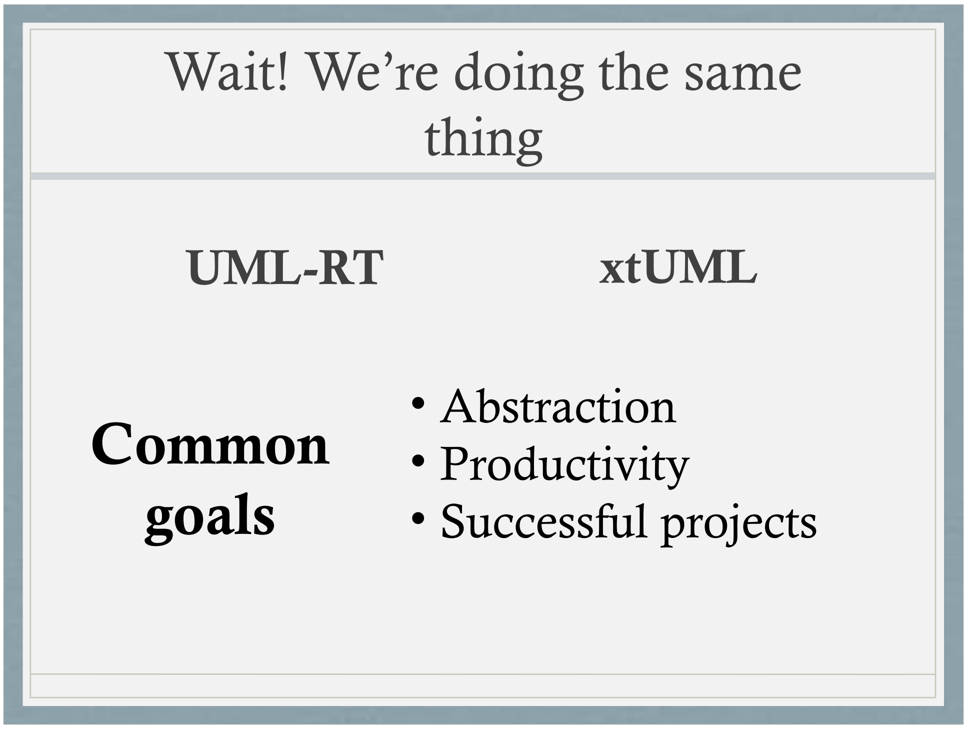
Same challenges

- Deeply embedded
- Time criticality
- Distributed processing
- Resource constrained
- High complexity

{Slide displays: “?” replaced with list from dialog}

S: Both dialects are facing the **same challenges** of

- Deeply embedded
- Time criticality
- Distributed processing
- Resource constrained
- High complexity



Wait! We're doing the same thing

UML-RT **xtUML**

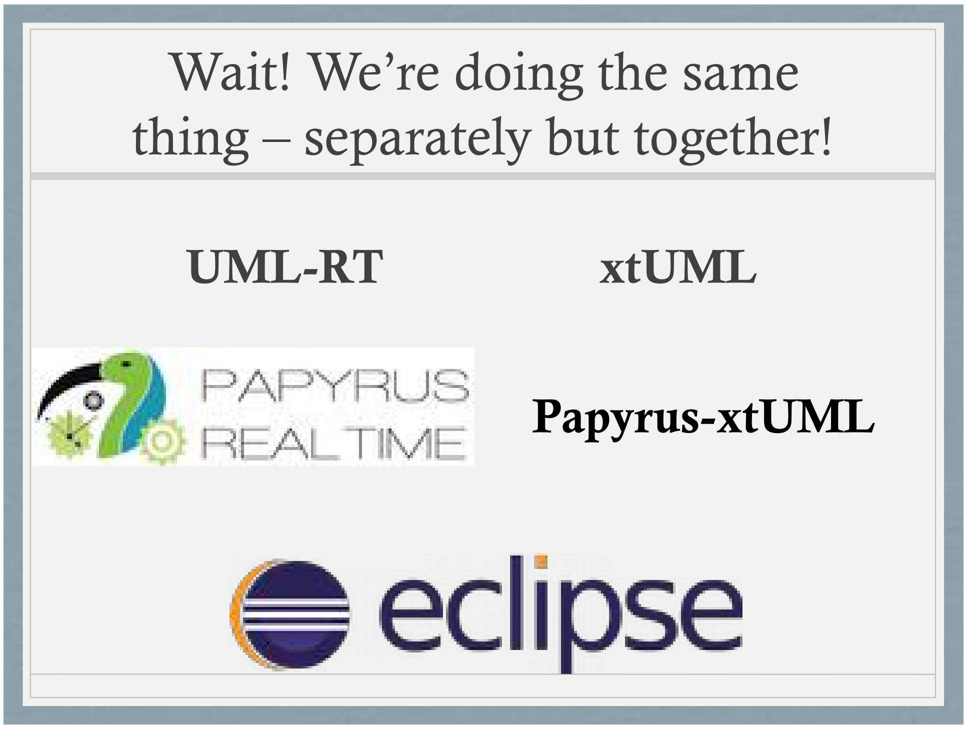
Common goals

- Abstraction
- Productivity
- Successful projects

{Slide displays: previous list replaced with new one - or added to new one?}

R: Yes, and the **goals are also common**, namely

- Abstraction
- Productivity
- Successful projects



Wait! We're doing the same thing – separately but together!

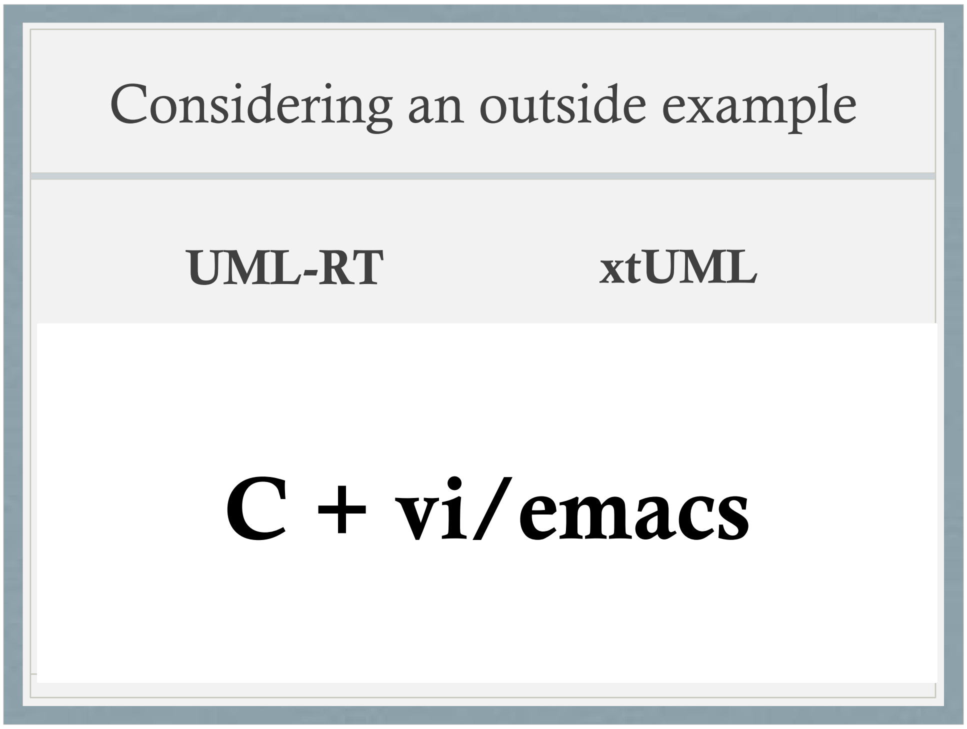
UML-RT **xtUML**

PAPYRUS REAL TIME **Papyrus-xtUML**

eclipse

{Slide displays: Replace list(s) with Eclipse + Papyrus-RT + Papyrus-xtUML}

S: And we both have projects in Eclipse as part of the Papyrus product line!


Considering an outside example

UML-RT **xtUML**

C + vi/emacs

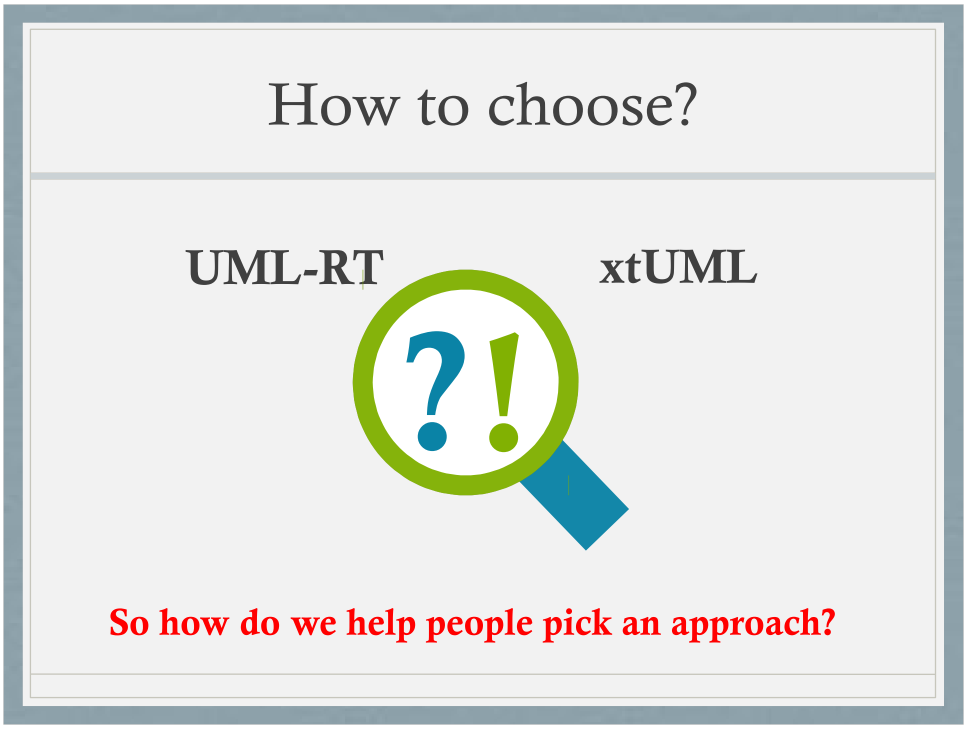
[considering an outside example]

{Slide displays: remove list and replace with “C+vi”}
R: Can you agree that C+vi

{Slide displays: add “/emacs” after “C+vi”}
S: ...or EMACS!

R: ...yes, O.K., can you agree that C+vi/emacs is the right language/tool choice for some particular tasks?

S: Yes, like modifying a print driver on Linux or illuminating an LED on my Arduino development board



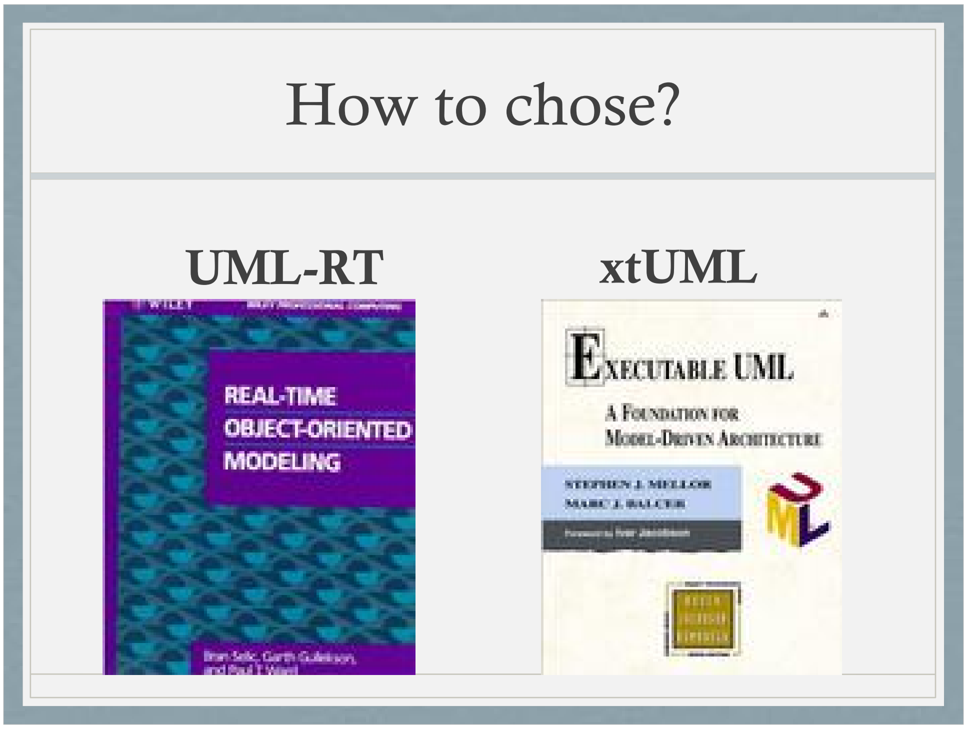
How to choose?

UML-RT

xtUML

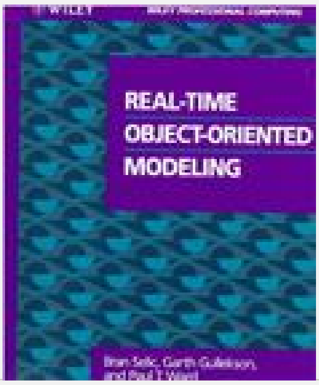
So how do we help people pick an approach?

[how to choose]
{Slide displays: remove “C+vi/emacs”; add “?”}
R: So how do we help people pick an approach?

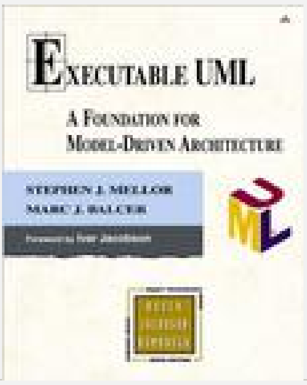


How to chose?

UML-RT


WILEY
REAL-TIME OBJECT-ORIENTED MODELING
Bran Selic, Garth Gullekson, and Paul T. Ward

xtUML



{Slide displays: show respective books on each side}
S: Research both methods by reading the first 3 chapters of the books.

R: Consider the place of action language in your requirements.

S: Consider the level of abstraction at which you want to model.

R: In some situations there will be a deciding factor like the need for interpretive execution or the existence of a large library of existing dialect models.

Already play together

{Slide displays: Show Eclipse, PolarSys, and the Papyrus Industry Consortium logos}

S: And, on top of that, we are already both members of the Papyrus Industry Consortium!

Ready to play together under Papyrus!

{Slide displays: Add Papyrus logo}

R: So we might as well recognize our differences and similarities and work together to make MBE and Papyrus successful!

I think this is the beginning of a beautiful friendship.