A component Model for the ABS Language

Michaël Lienhardt

M. Bravetti, I. Lanese, D. Sangiorgi, J. Schäfer, Y. Welsh, G. Zavattaro

Motivation

Why Components?

Motivation

Two keywords of the HATS project

at runtime

- Adaptability ← Adapt to the environment

- Evolvability ← Enable modification

Adaptability = Evolvability + operations

Motivation

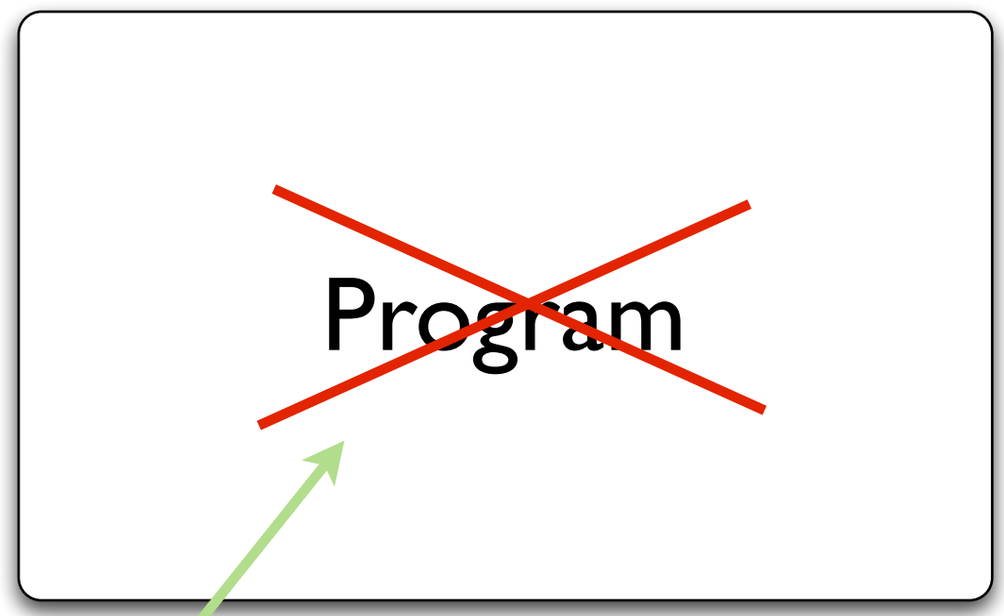
Typically:

We have a Program to update

Motivation

Typically:

Usually

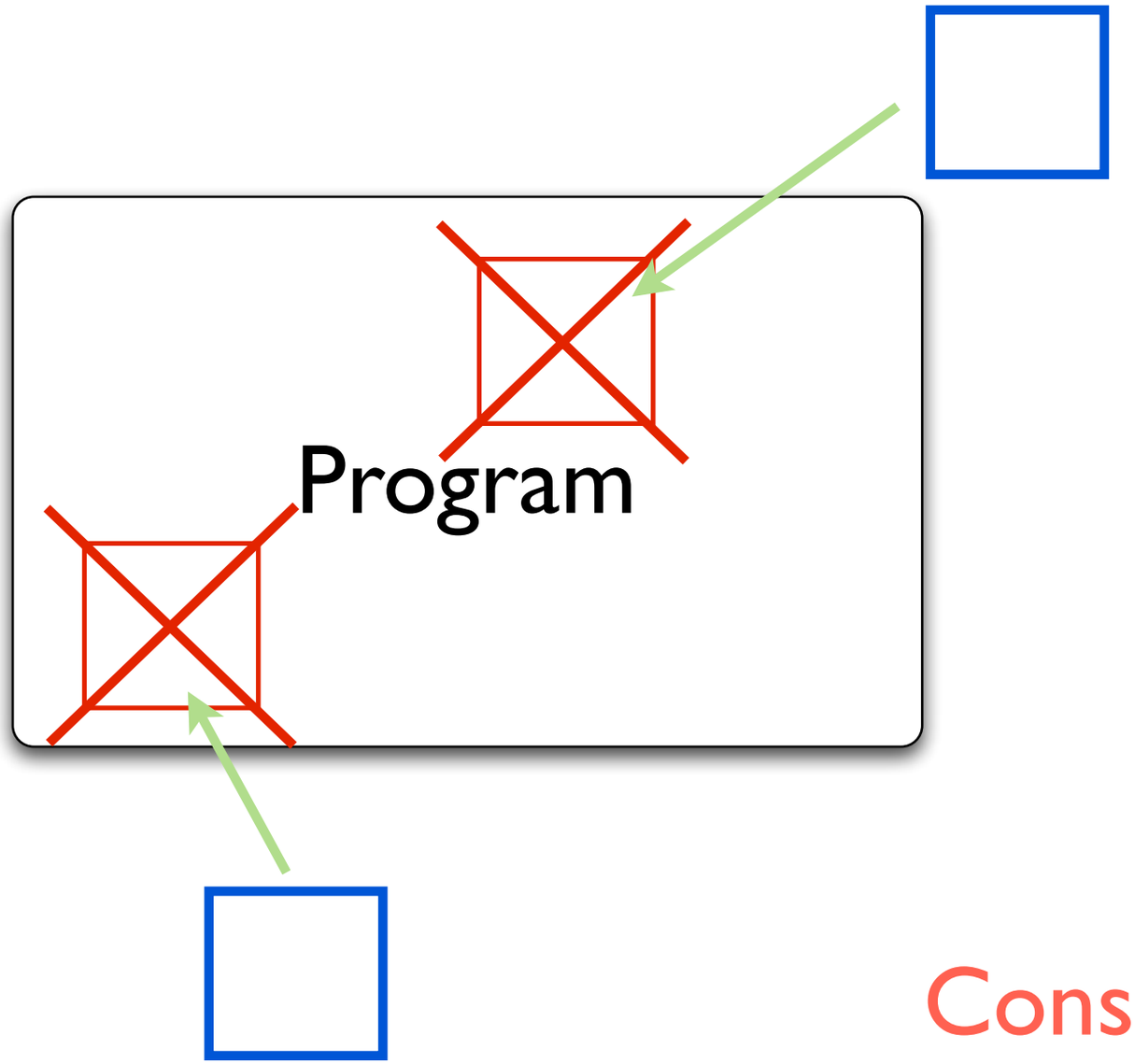


Very costly

Motivation

Typically:

Change only what's necessary



Consistency

Motivation

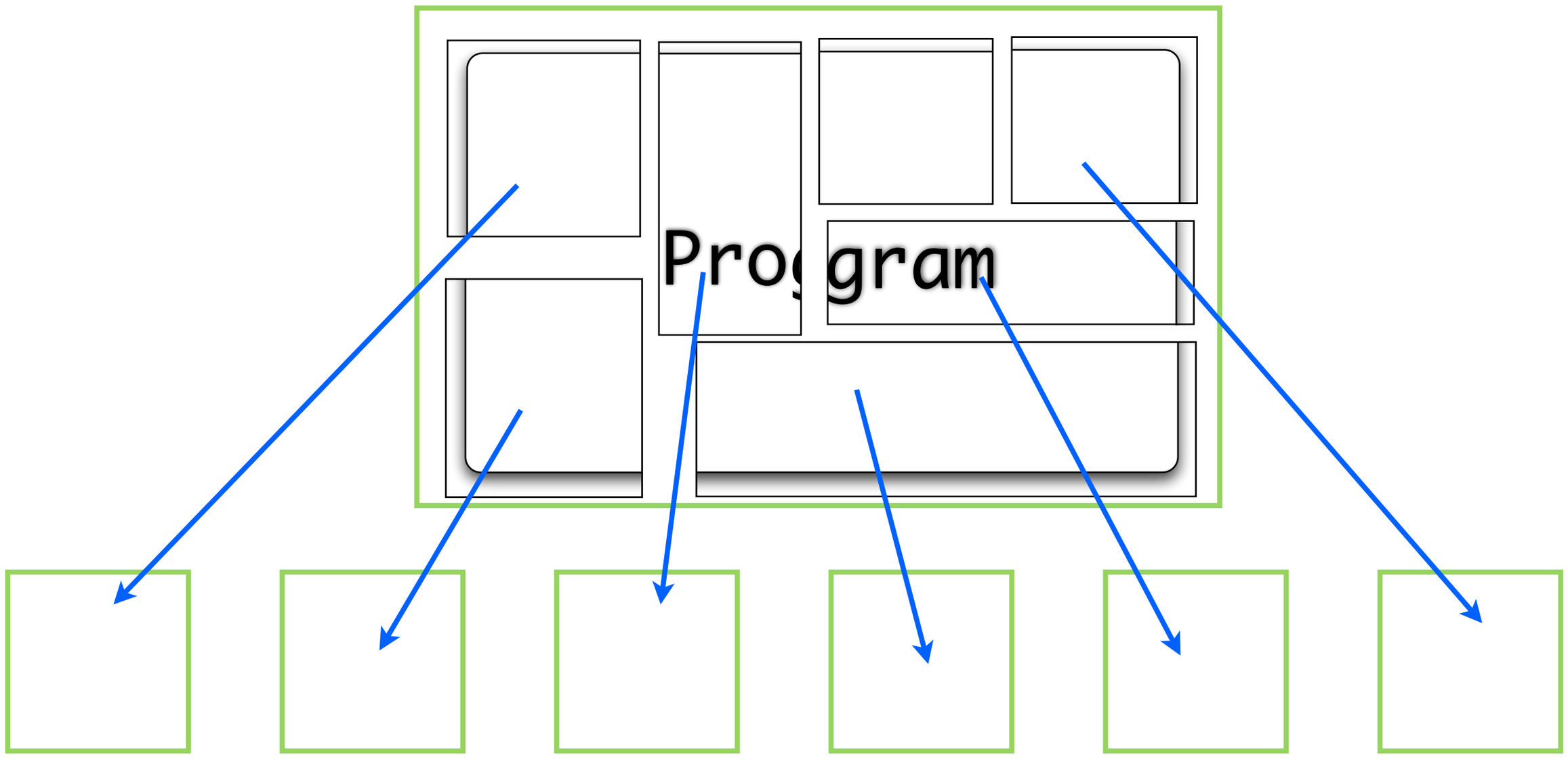
Typically:

We have a Program to restructure

For instance, new sites are available

Motivation

Typically:



Motivation

Hence, we need:

- Programs as sets of talkative boxes
- Isolation
- Mobility

Motivation

A classical approach to structure programs into boxes is

Components

Classic Components

- Boxes

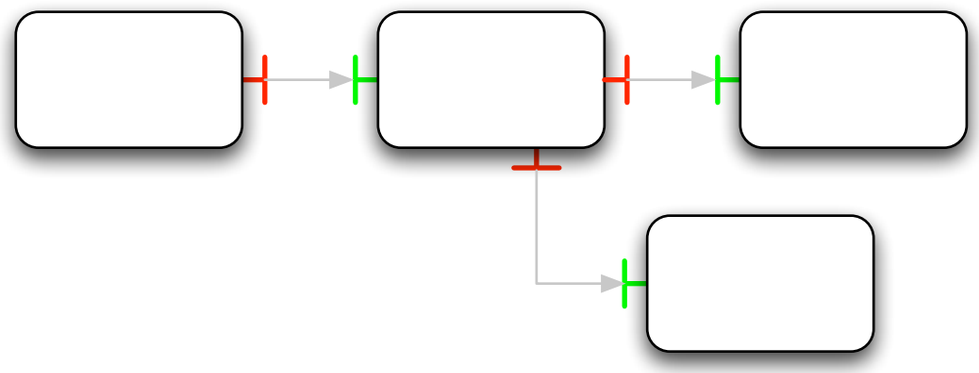
Classic Components

- Boxes with ports

Classic Components

- Boxes with ports

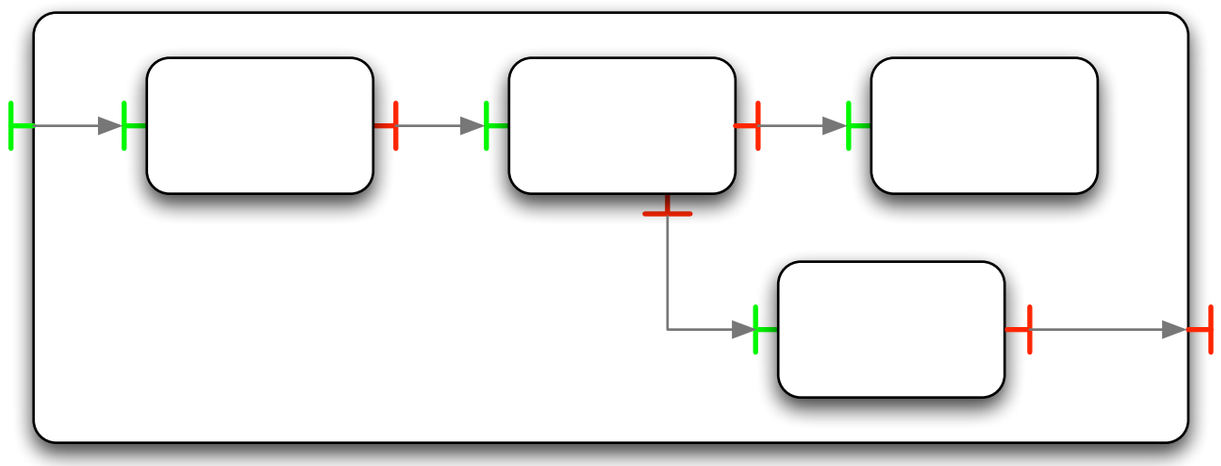
  that can be assembled



Classic Components

- Boxes with ports

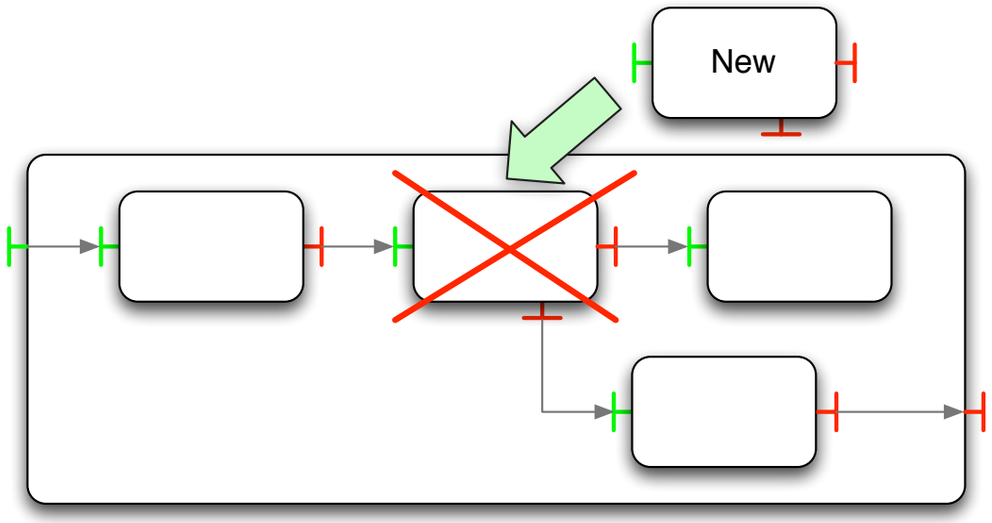
  that can be assembled in hierarchy

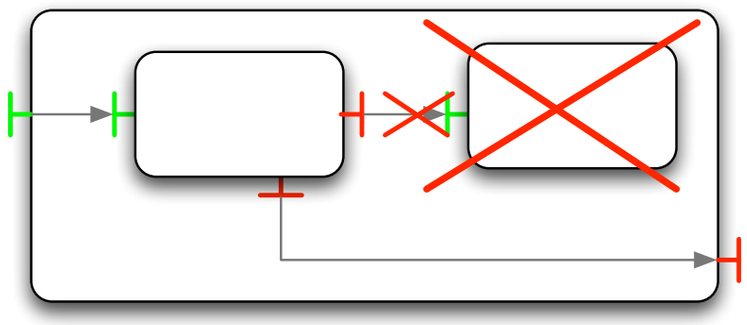


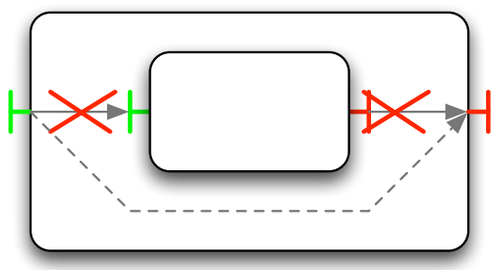
Classic Components

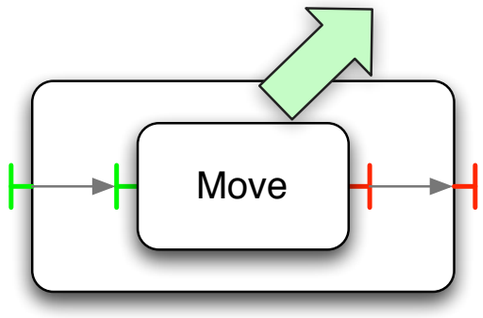
- Boxes with ports

  so we can manipulate their structure









Classic Components

But

Why Yet Another Component Model?

(there's already Fractal, OSGi, Ensemble, Appia, darwin,...)

Classic Components

What we want to do

Formal model

That interacts with Objects

That can easily express Adaptability

Classic Components

BUT

Common Approaches:

- ADL Based
- Just a Model
- Informal, Complex

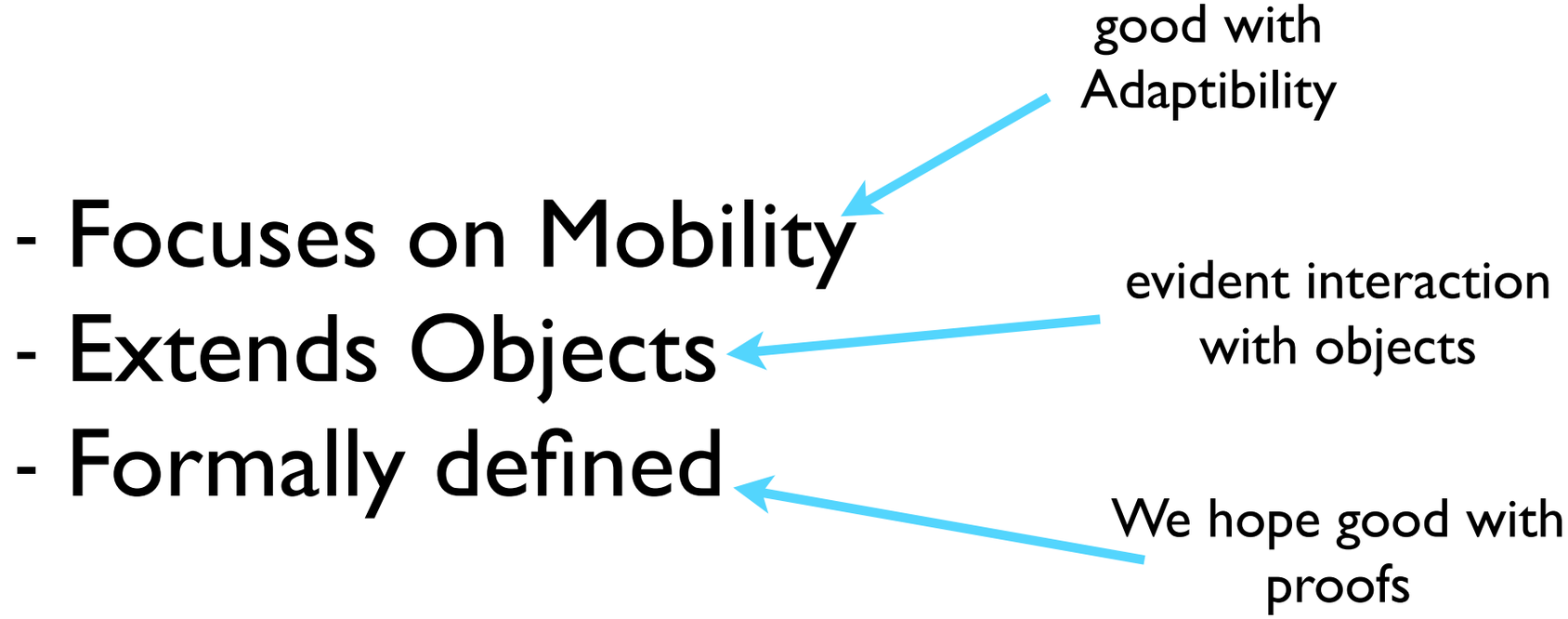
Cannot express runtime modification

no interaction with objects

difficult to prove properties

Our Component Model

- Focuses on Mobility
- Extends Objects
- Formally defined



Our Component Model

$$
\begin{array}{rcl}
P & ::= & 0 \mid x \mid vaP \mid P \mid P \mid !P \mid a(S)R[A\}[P] \\
A & ::= & a(x) \mid a\langle P \rangle \mid \textbf{open } S \mid \textbf{close } S \\
 & \mid & a \textbf{ in } b \mid a \textbf{ out } b \mid a\_m\langle P \rangle \\
 & & \\
M & ::= & 0 \mid m(x)P \mid M \mid M
\end{array}
$$

Our Component Model

Process

Action

$$P ::= 0 \mid x \mid \nu a P \mid P \mid P \mid A.P \mid a(S)\{M\}[P]$$

$$A ::= a(x) \mid a\langle P\rangle \mid \textbf{open } S \mid \textbf{close } S \mid a \textbf{ in } b \mid a \textbf{ out } b \mid a\_m\langle P\rangle$$

$$M ::= 0 \mid m(x)P \mid M \mid M$$

Methods

Our Component Model

≅ π-Calculus

$$
\begin{array}{lcl}
P & ::= & 0 \mid x \mid \nu a P \mid P \mid P \mid A.P \mid a(S)\{M\}[P] \\
A & ::= & a(x) \mid a\langle P\rangle \mid \textbf{open } S \mid \textbf{close } S \\
 & \mid & a \textbf{ in } b \mid a \textbf{ out } b \mid a\_m\langle P\rangle \\
M & ::= & 0 \mid m(x)P \mid M \mid M
\end{array}
$$

Our Component Model

Component

$$P \quad ::= \quad 0 \quad | \quad x \quad | \quad \nu a P \quad | \quad P \,|\, P \quad | \quad A.P \quad | \quad a(S)\{M\}[P]$$

$$A \quad ::= \quad a(x) \quad | \quad a\langle P \rangle \quad | \quad \textbf{open } S \quad | \quad \textbf{close } S$$
$$| \quad a \textbf{ in } b \quad | \quad a \textbf{ out } b \quad | \quad a\_m\langle P \rangle$$

$$M \quad ::= \quad 0 \quad | \quad m(x)P \quad | \quad M \,|\, M$$

Our Component Model

$$P \quad ::= \quad 0 \quad | \quad x \quad | \quad \nu a P \quad | \quad P \,|\, P \quad | \quad A.P \quad | \quad a(S)\{M\}[P]$$

$$A \quad ::= \quad a(x) \quad | \quad a\langle P \rangle \quad | \quad \textbf{open}\ S \quad | \quad \textbf{close}\ S$$
$$| \quad a\ \textbf{in}\ b \quad | \quad a\ \textbf{out}\ b \quad | \quad a\_m\langle P \rangle$$

$$M \quad ::= \quad 0 \quad | \quad m(x)P \quad | \quad M \,|\, M$$

Method invocation

Method definition

Our Component Model

$$P \quad ::= \quad 0 \quad | \quad x \quad | \quad \nu a P \quad | \quad P \,|\, P \quad | \quad A.P \quad | \quad a(S)\{M\}[P]$$

$$A \quad ::= \quad a(x) \quad | \quad a\langle P \rangle \quad | \quad \textbf{open } S \quad | \quad \textbf{close } S$$
$$| \quad a \textbf{ in } b \quad | \quad a \textbf{ out } b \quad | \quad a\_m\langle P \rangle$$

$$M \quad ::= \quad 0 \quad | \quad m(x)P \quad | \quad M \,|\, M$$

Sub Components and tasks

Our Component Model

$$P \quad ::= \quad 0 \quad | \quad x \quad | \quad \nu a P \quad | \quad P \,|\, P \quad | \quad A.P \quad | \quad a(S)\{M\}[P]$$

$$A \quad ::= \quad a(x) \quad | \quad a\langle P \rangle \quad | \quad \textbf{open } S \quad | \quad \textbf{close } S$$
$$| \quad a \textbf{ in } b \quad | \quad a \textbf{ out } b \quad | \quad a\_m\langle P \rangle$$

$$M \quad ::= \quad 0 \quad | \quad m(x)P \quad | \quad M \,|\, M$$

Channels

To return values

Our Component Model

$$P \quad ::= \quad 0 \quad | \quad x \quad | \quad \nu a P \quad | \quad P \,|\, P \quad | \quad A.P \quad | \quad a(S)\{M\}[P]$$

$$A \quad ::= \quad a(x) \quad | \quad a\langle P \rangle \quad | \quad \mathbf{open}\ S \quad | \quad \mathbf{close}\ S$$
$$| \quad a\ \mathbf{in}\ b \quad | \quad a\ \mathbf{out}\ b \quad | \quad a\_m\langle P \rangle$$

$$M \quad ::= \quad 0 \quad | \quad m(x)P \quad | \quad M \,|\, M$$

Visibility

to control communications
and encode the wrapping

Our Component Model

$$P \quad ::= \quad 0 \quad | \quad x \quad | \quad \nu a P \quad | \quad P \mid P \quad | \quad A.P \quad | \quad a(S)\{M\}[P]$$

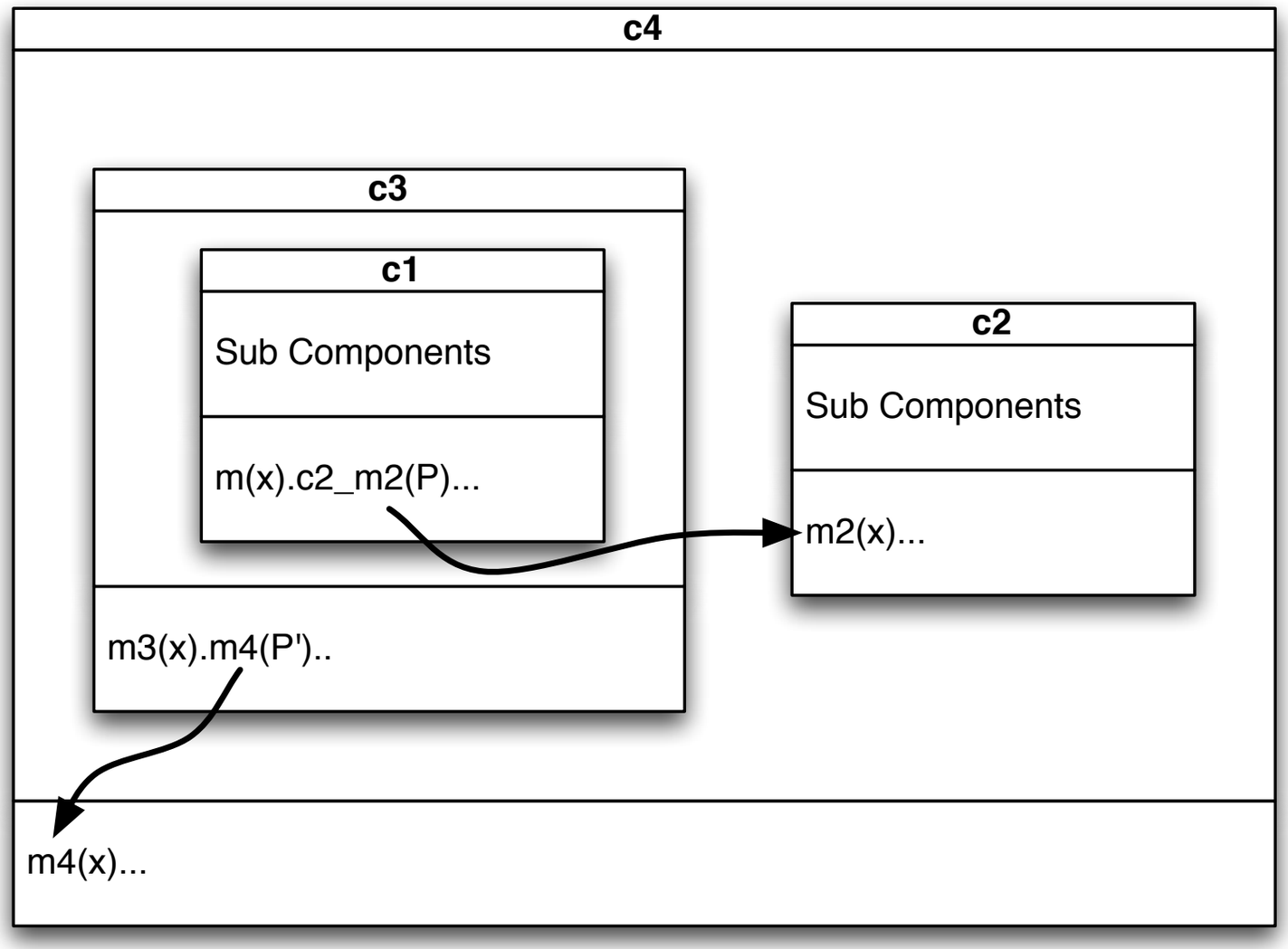
$$A \quad ::= \quad a(x) \quad | \quad a\langle P \rangle \quad | \quad \textbf{open } S \quad | \quad \textbf{close } S$$
$$| \quad a \textbf{ in } b \quad | \quad a \textbf{ out } b \quad | \quad a\_m\langle P \rangle$$

$$M \quad ::= \quad 0 \quad | \quad m(x)P \quad | \quad M \mid M$$

Manipulation

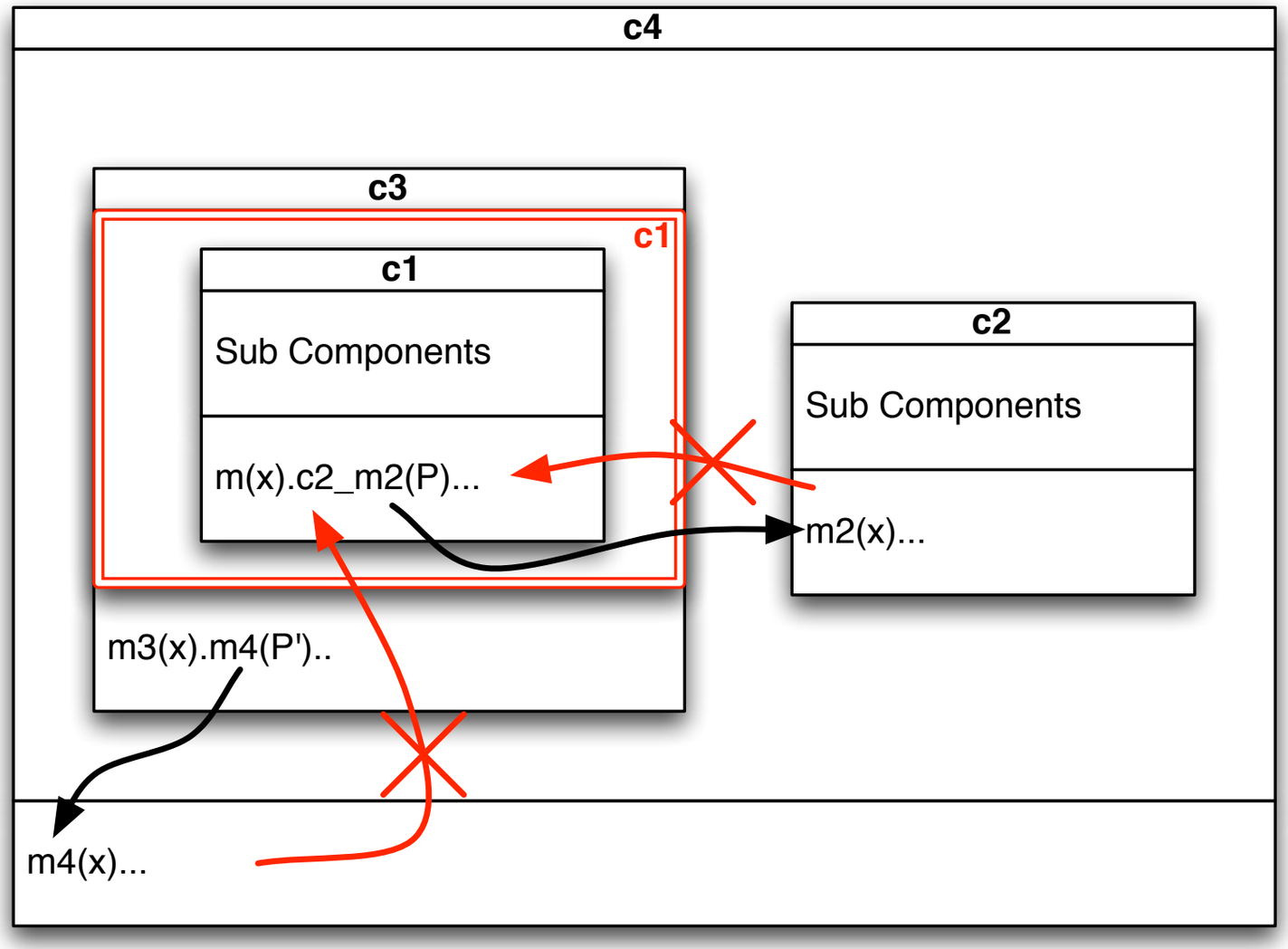
to modify components

Components as Objects



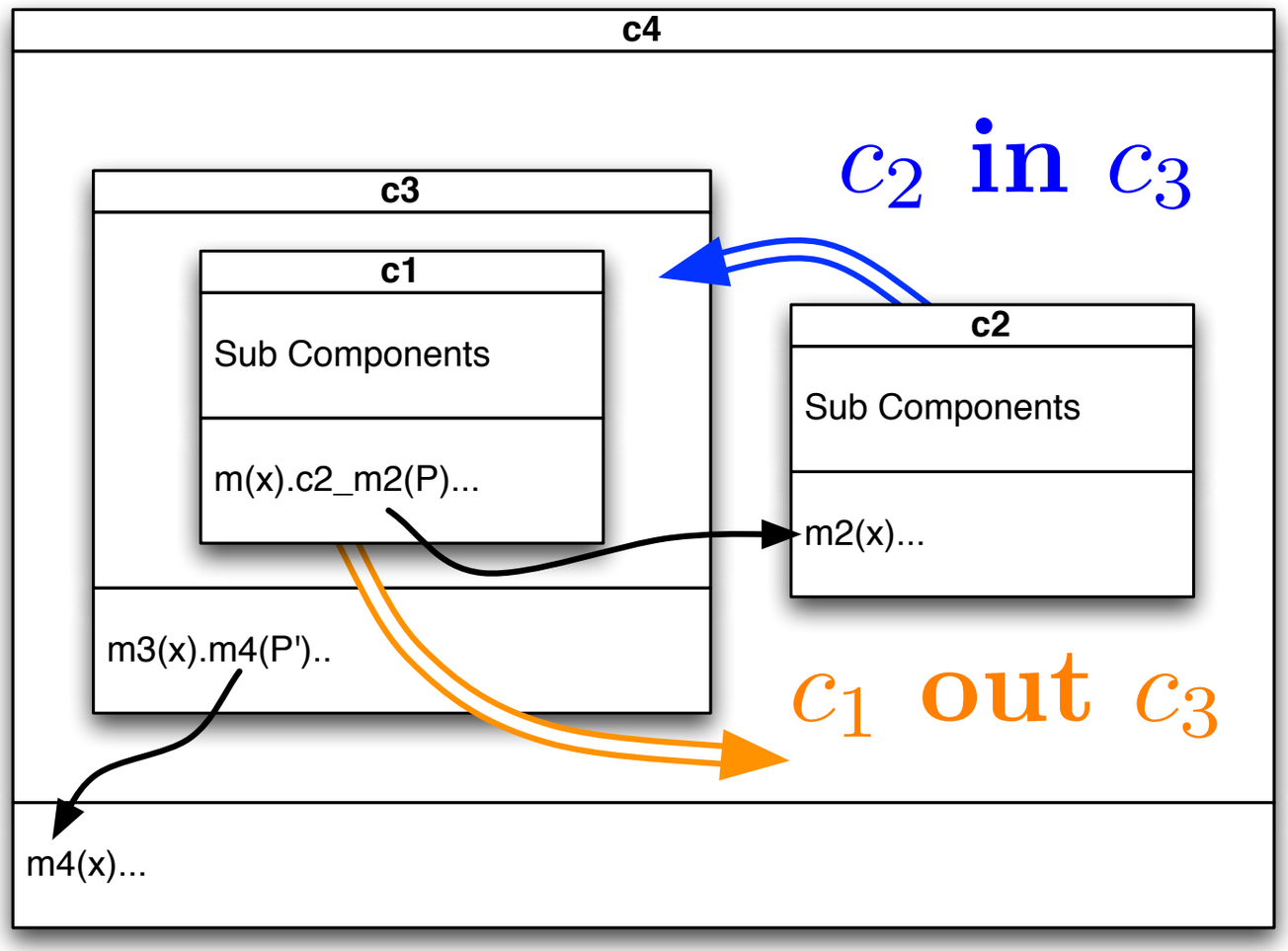
- Inner component = is a field of
- Tree structure
- Graph communication

Components as Isolation Boxes



- $c_3(c_1)\{\ldots\}[\ldots]$
- Control over communications
- Only affect method calls
- Controlled by $\mathbf{open}\ S$ / $\mathbf{close}\ S$

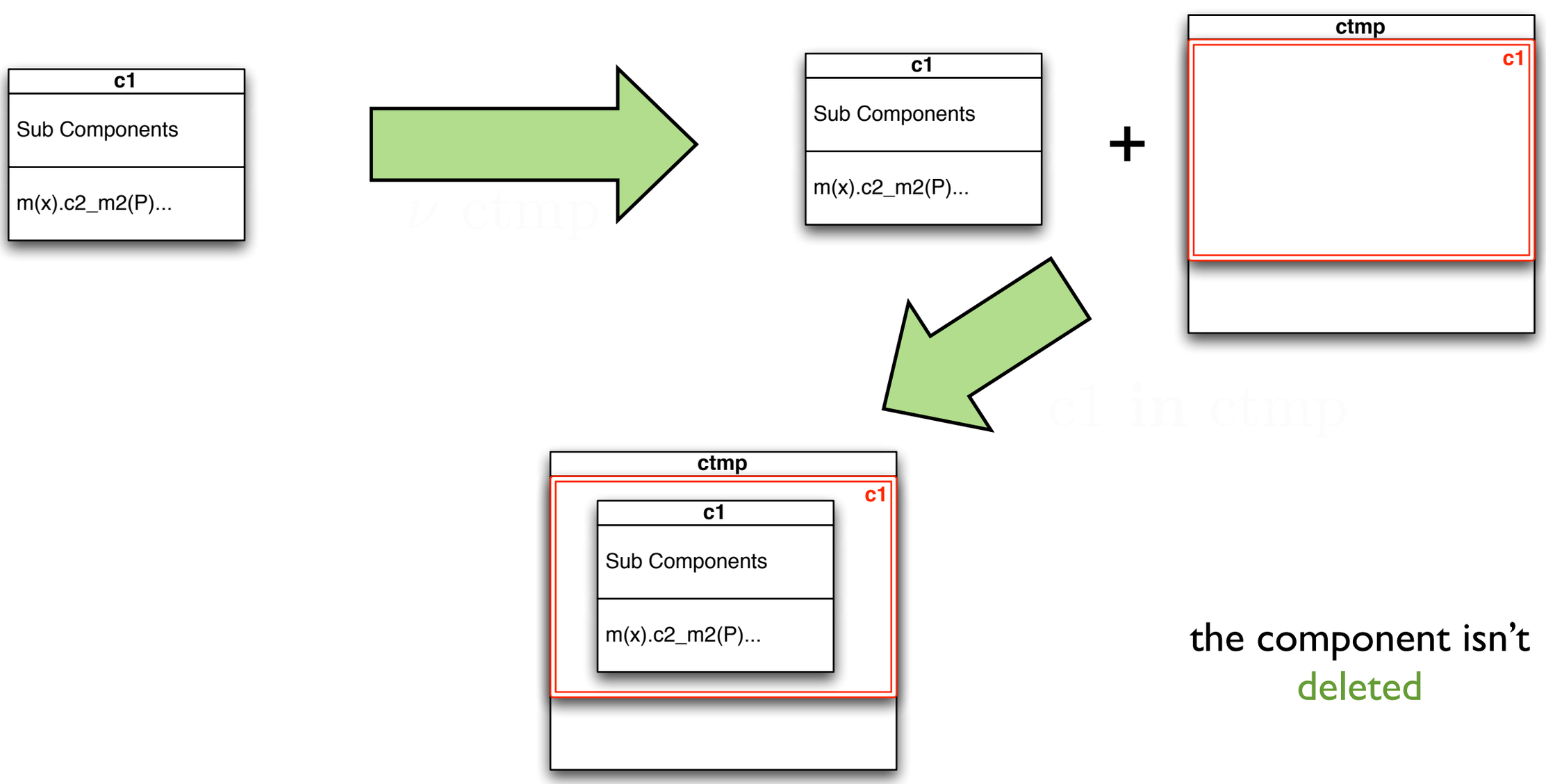
Components as Mobility Basis



We then encode the other Adaptability operators

Components as Adaptibility Basis

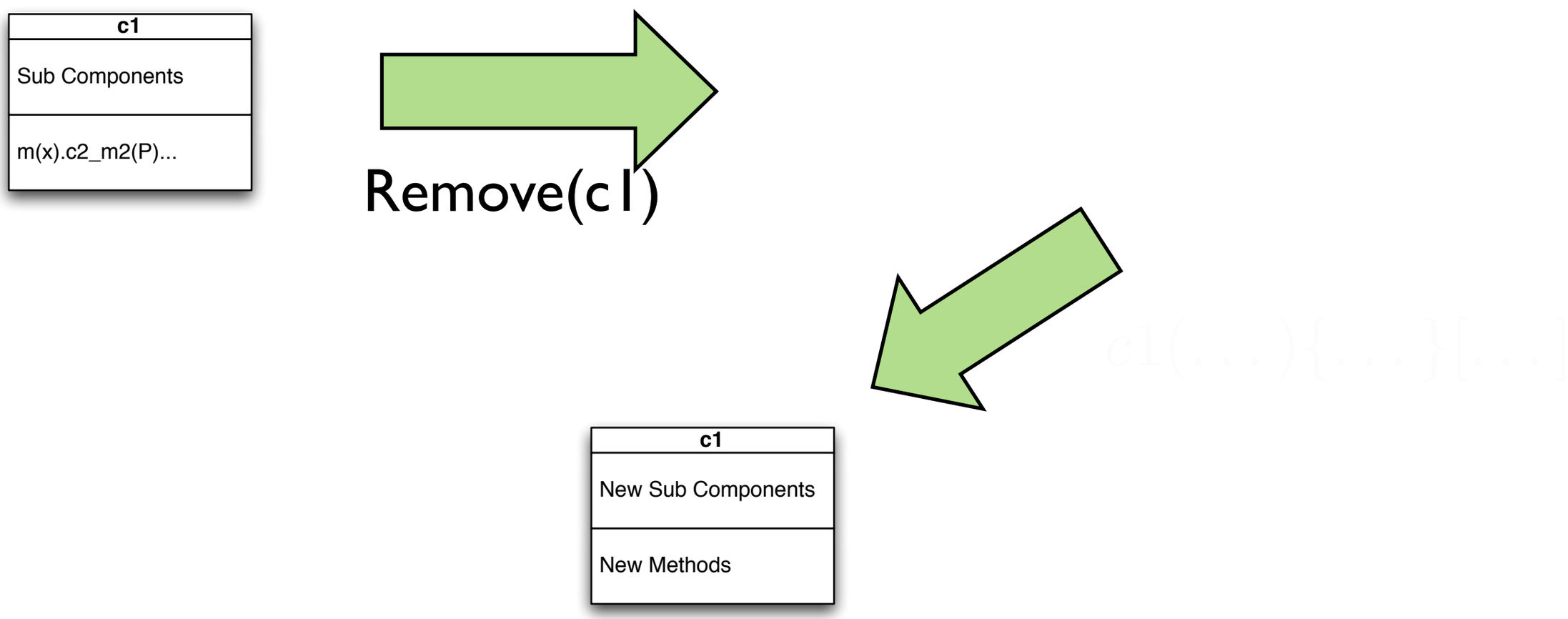
(1/3) Remove(c1)



the component isn't deleted

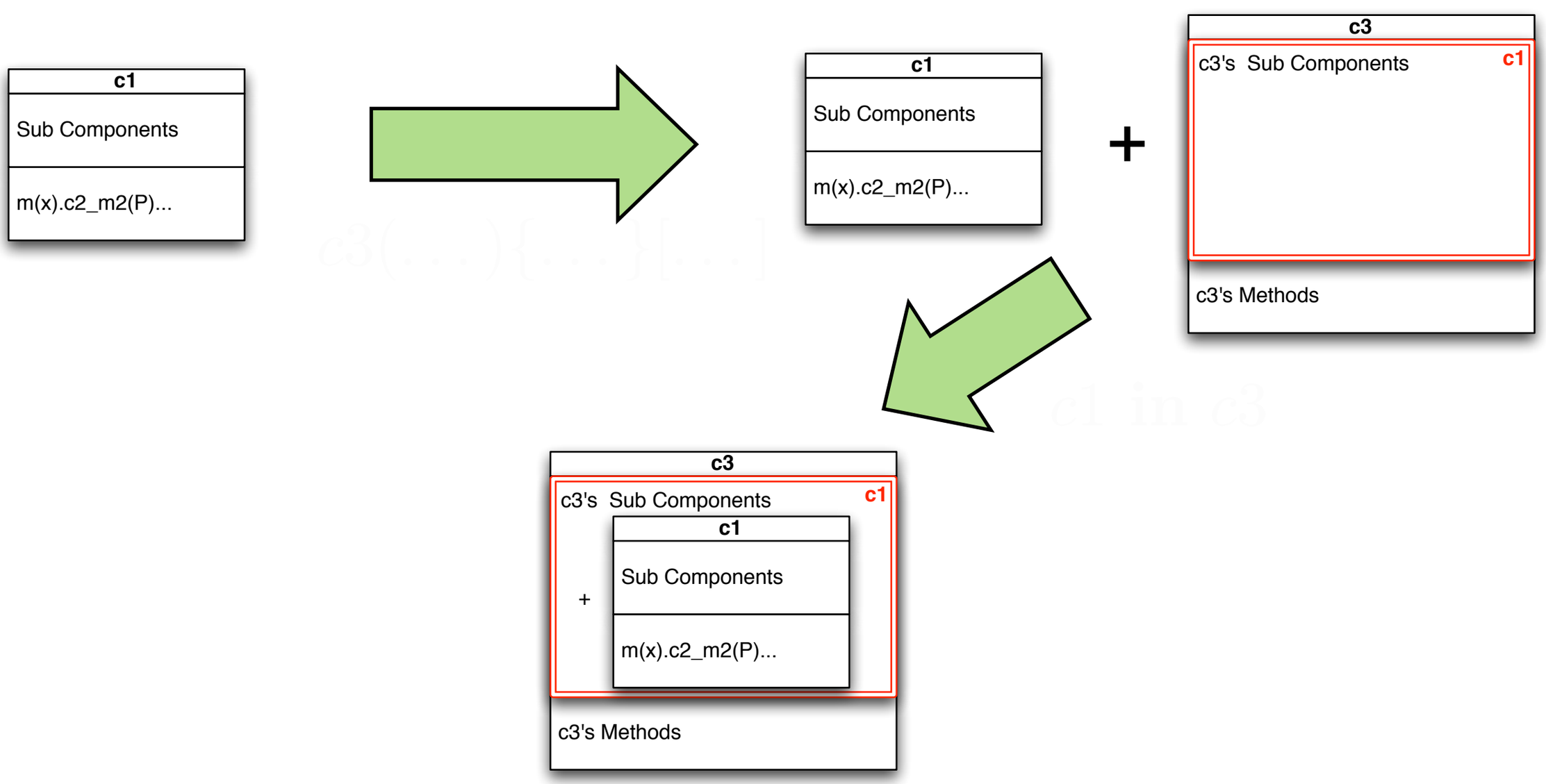
Components as Adaptibility Basis

(2/3) Update(c1)



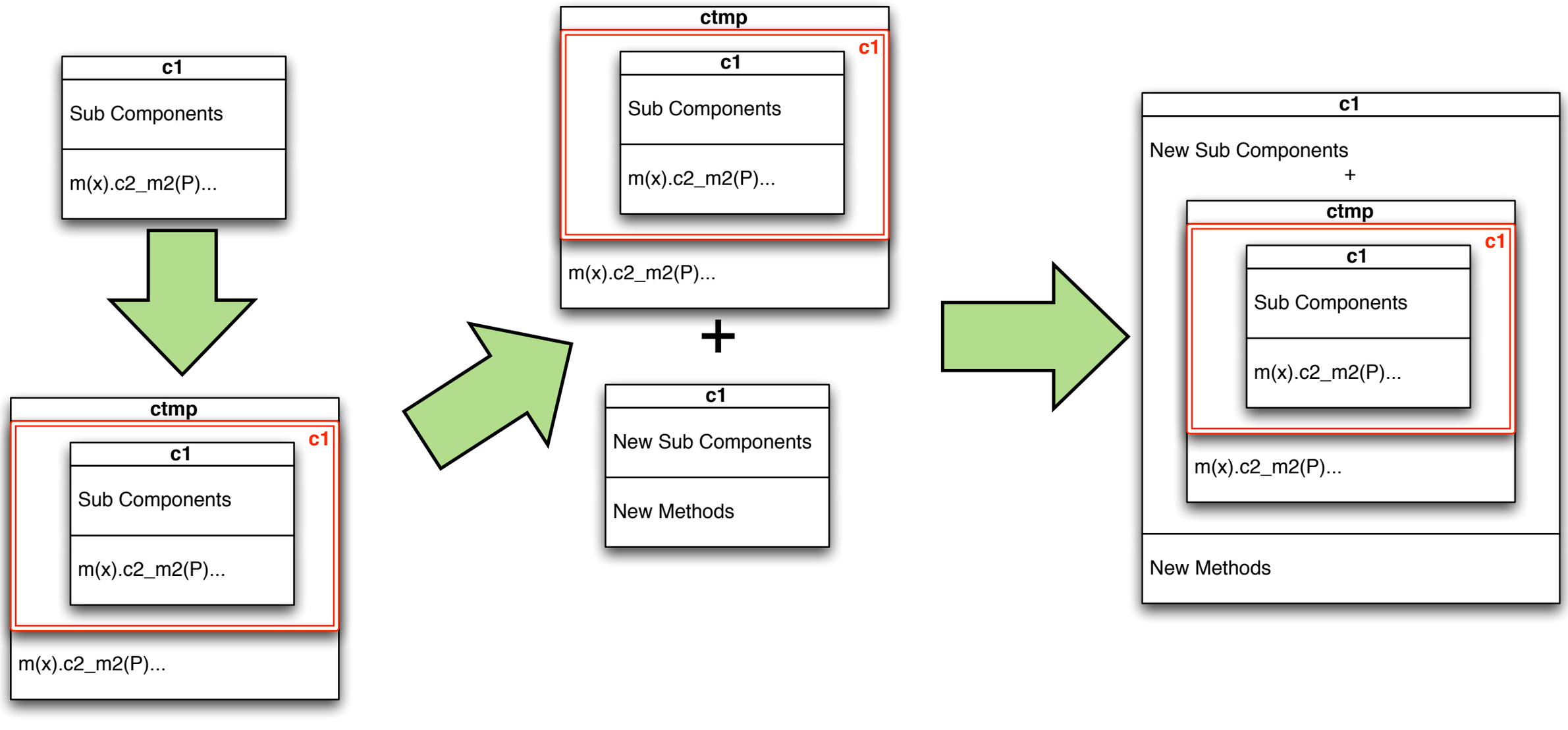
Components as Adaptibility Basis

(3/3) Wrap(c1,c3)



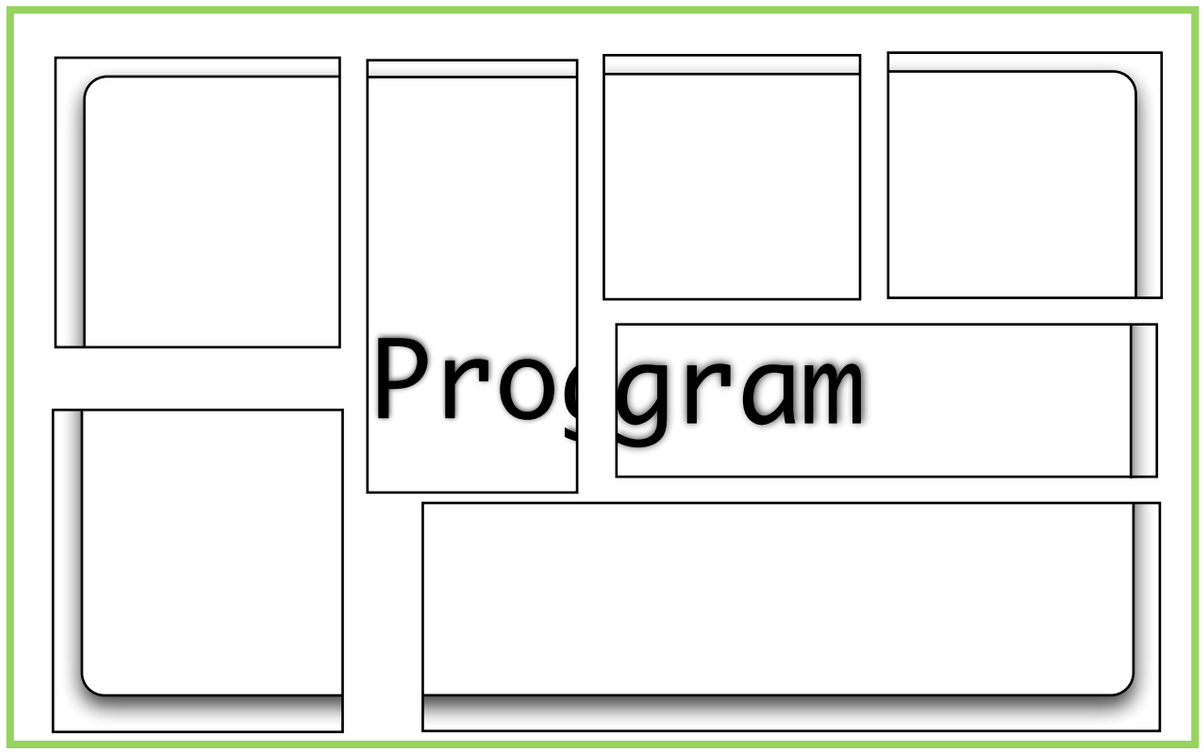
Components as Adaptibility

(1/2) Wrap(c1)



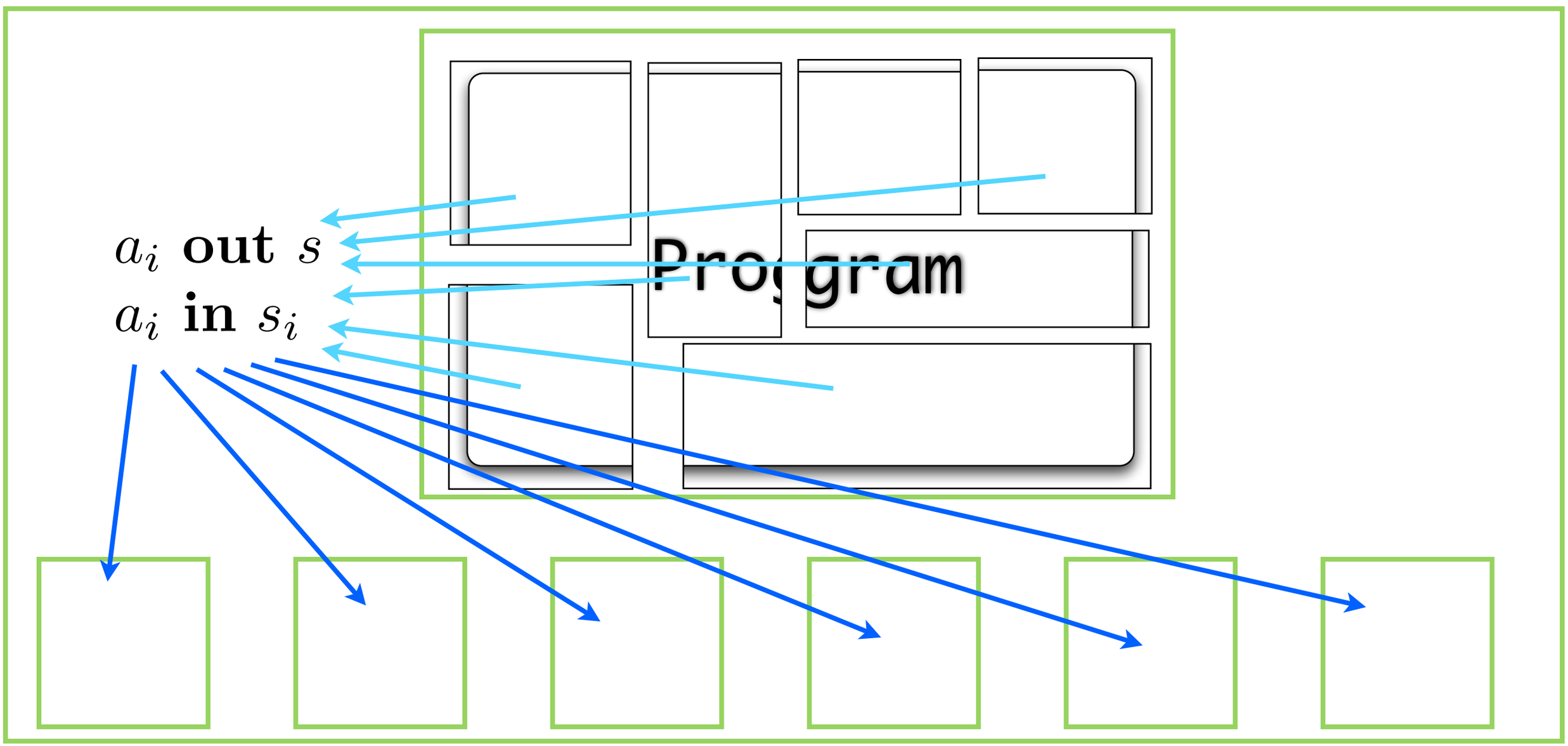
Components as Adaptibility

(2/2) deploy



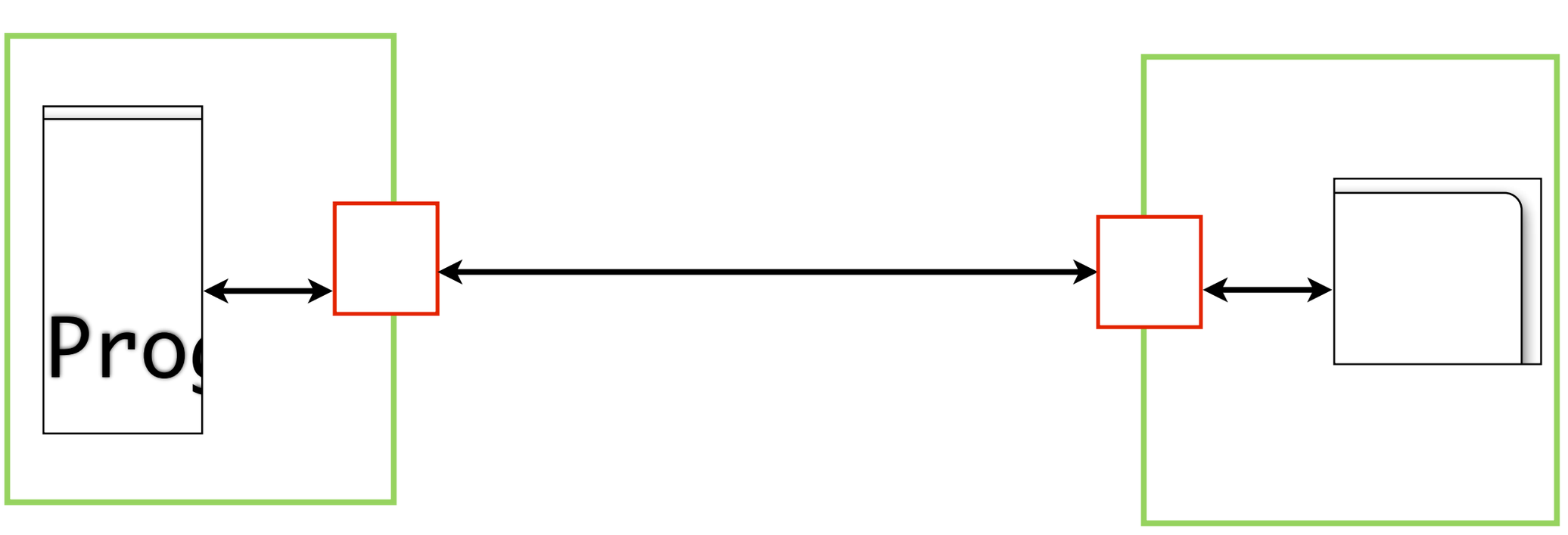
Components as Adaptibility

(2/2) deploy



Components as Adaptibility

(2/2) deploy



Conclusion

- Our component Model
  - Is Formal
  - Capture the notion of Object
  - Has a relatively simple semantic
  - Can encode ‘safe’ modifications

Conclusion

- In comparison to other Model
  - Does not have a remove operator
  - Does not have links

Conclusion

- Our Model may still need improvements
  - For the deploy
  - For message forwarding
  - To manage errors (real deletion)
  - To manage sessions

Conclusion

Integration to ABS needs
to be addressed

Thank You
for your attention