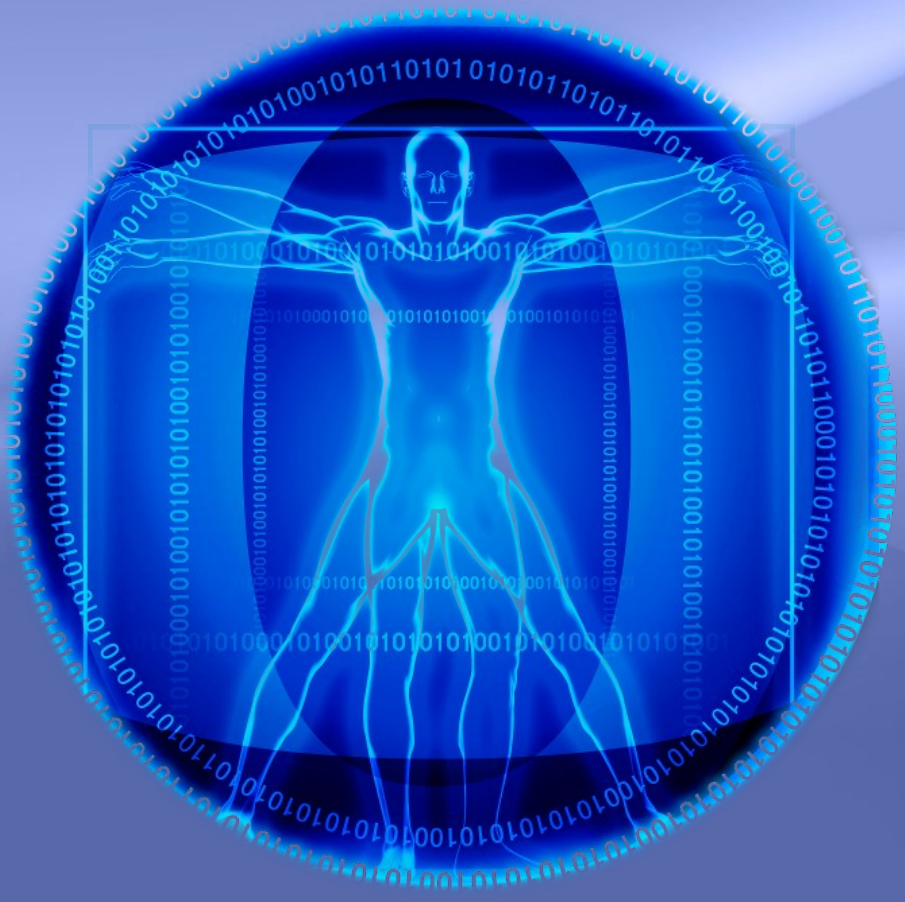
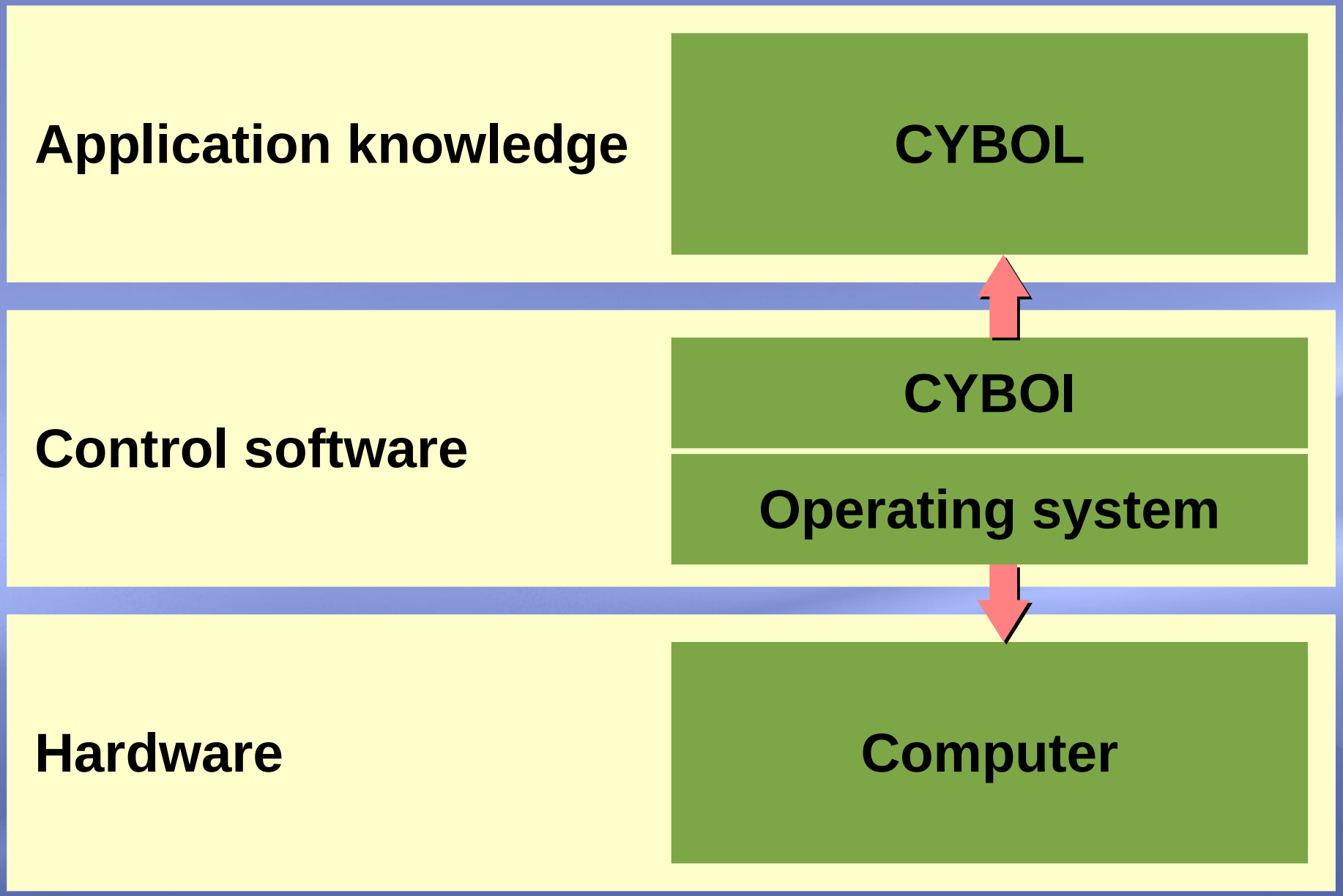
Cybernetics Oriented Programming (CYBOP)



GNU GPLv3

Christian Heller
<christian.heller@tuxtax.de>



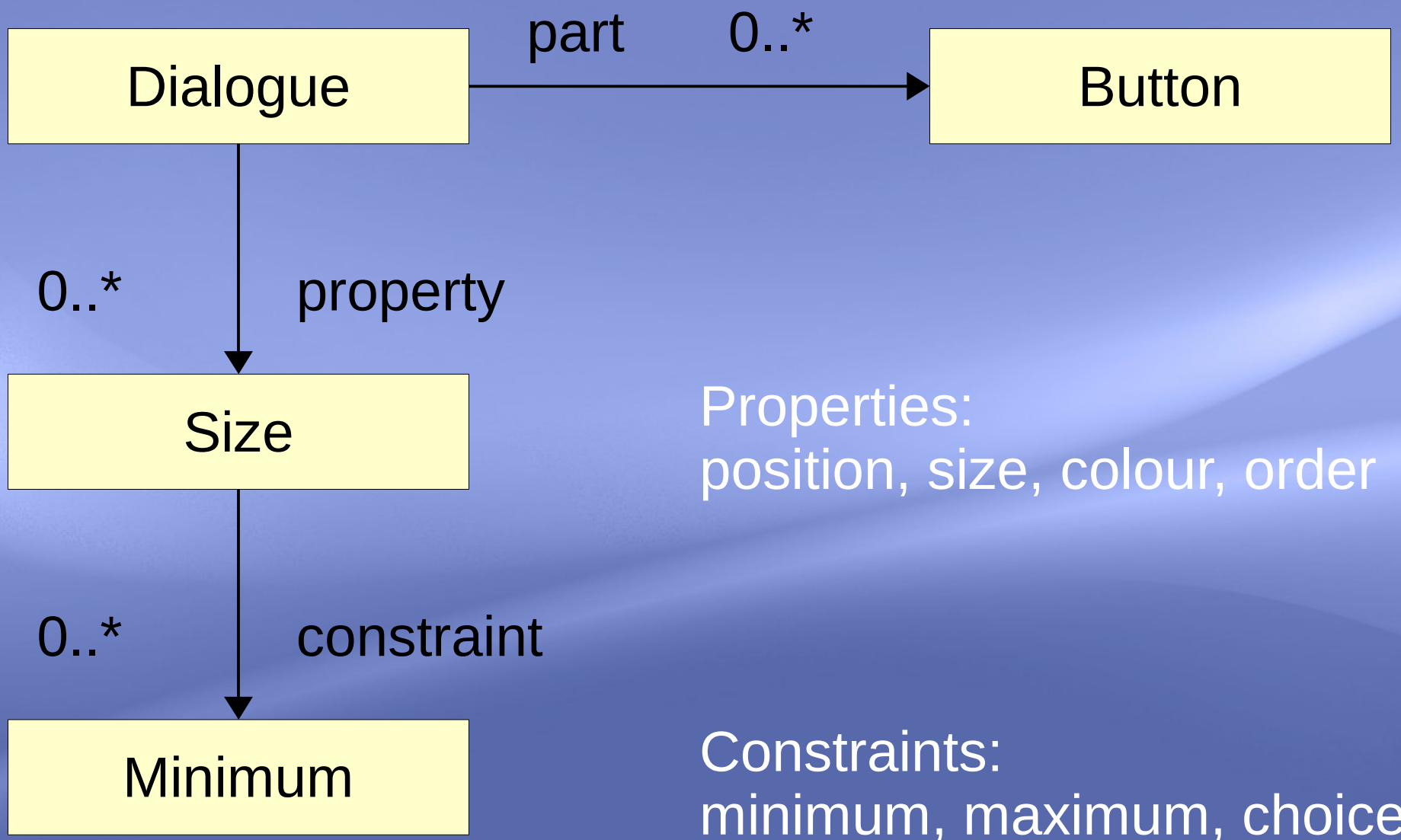

```
cyboi  app/run.cybol
```

Criterion	Java world	CYBOP world
Language	Java	CYBOL (XML-based)
Interpreter	Java Virtual Machine (JVM)	CYBOI (written in C)
Theory	Object Oriented Programming (OOP)	CYBOP (knowledge schema)

Java Class

```
public class Dialogue {

    // Structural attributes

    private String title;
    private Toolbar toolbar;
    private Button button;

    // Additional properties

    private Point position;
    private Dimension size;
    private Color background;

    // Methods

    public void initialise(Object o);
    public void process(Event e);
}
```

Parts make up a whole



Properties:
position, size, colour, order

Constraints:
minimum, maximum, choice

State Template "dialogue.cybol" (a structure in space)

```
<node>
    <node name="title" channel="inline" format="text/plain" model="Test"/>
    <node name="toolbar" channel="file" format="element/part" model="path/toolbar.cybol"/>
    <node name="button" channel="file" format="element/part" model="path/button.cybol">
        <node name="position" channel="inline" format="number/integer" model="5,5"/>
        <node name="size" channel="inline" format="number/integer" model="600,400"/>
        <node name="background" channel="inline" format="colour/terminal" model="red"/>
    </node>
</node>
```

Logic Template "initialise.cybol" (an algorithm in time)

```
<node>
  <node name="build_menu" channel="inline" format="flow/sequence" model="">
    <node name="model" channel="inline" format="path/knowledge" model=".app.logic.menu"/>
  </node>
  <node name="build_panel" channel="inline" format="flow/sequence" model="">
    <node name="model" channel="inline" format="path/knowledge" model=".app.logic.panel"/>
  </node>
  <node name="display" channel="inline" format="communicate/send" model="">
    <node name="channel" channel="inline" format="meta/channel" model="terminal"/>
    <node name="encoding" channel="inline" format="meta/encoding" model="utf-8"/>
    <node name="language" channel="inline" format="meta/language" model="message/tui"/>
    <node name="format" channel="inline" format="meta/format" model="element/part"/>
    <node name="message" channel="inline" format="path/knowledge" model=".app.dialogue"/>
    <node name="clear" channel="inline" format="logicvalue/boolean" model="true"/>
  </node>
</node>
```

CYBOL Features

language including knowledge tree

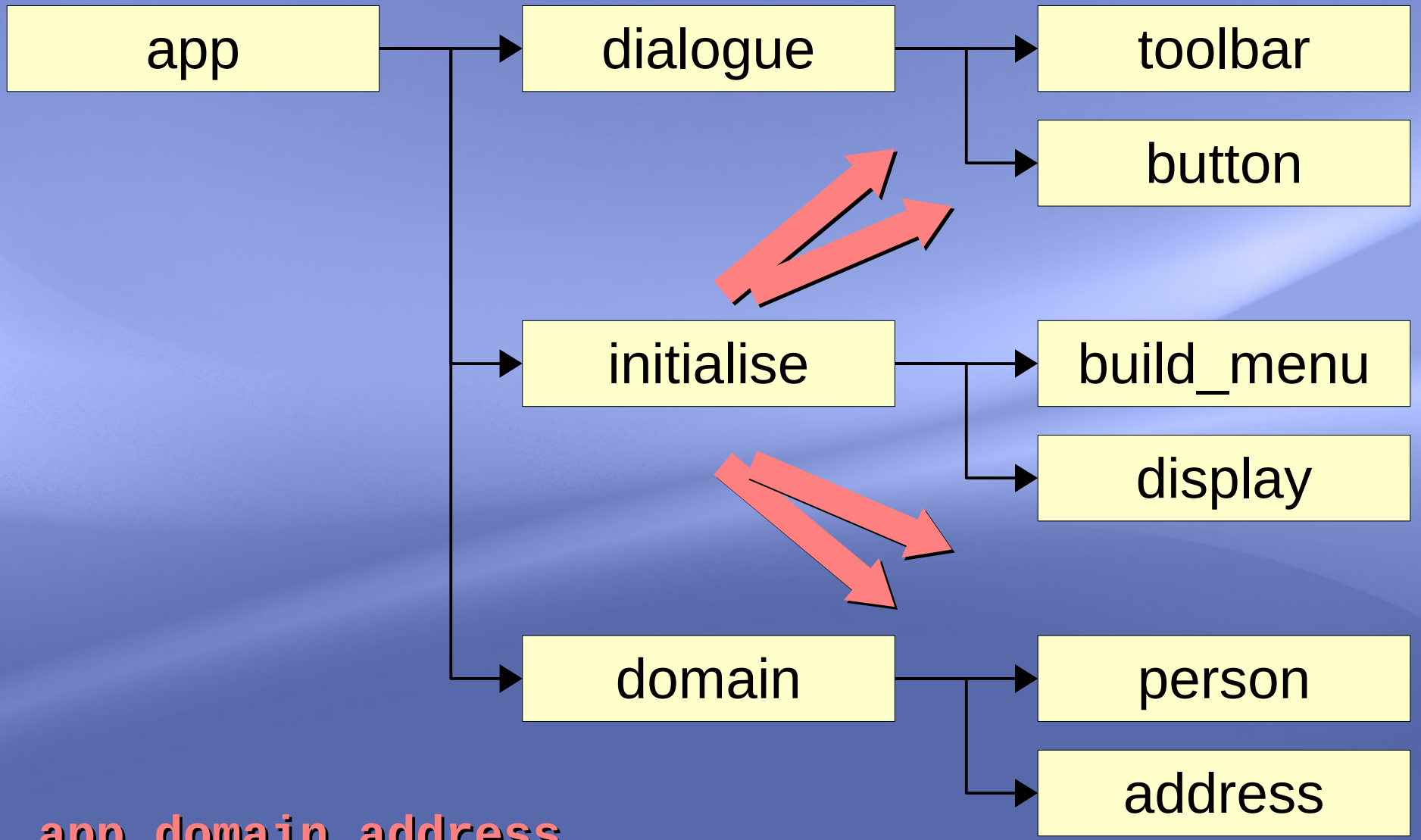
structure based on double-hierarchy pattern

communication using send/receive with cli/tui/gui/wui

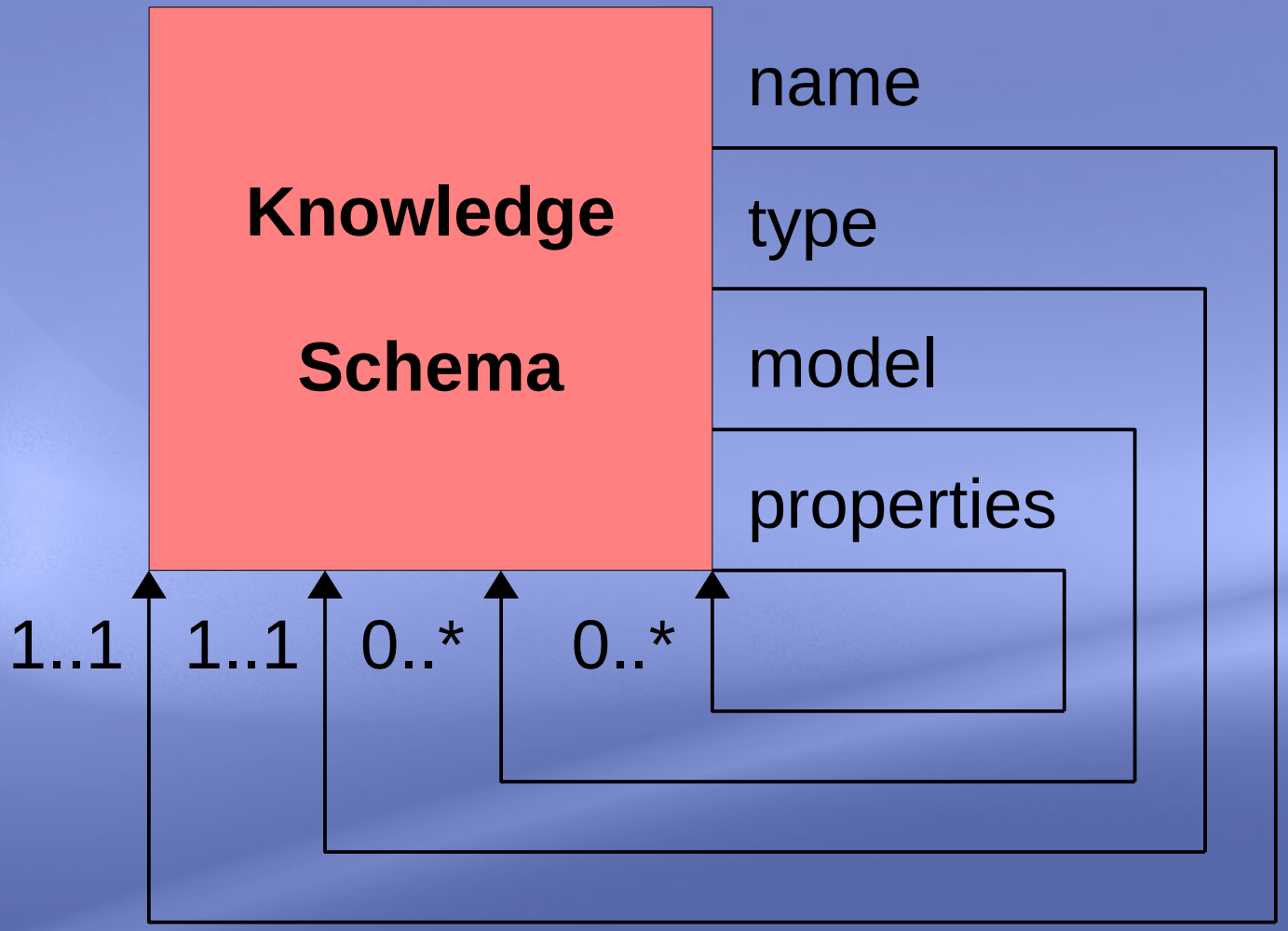
usage for standard applications and complex scripting

understanding possible for analysts/non-programmers

.app.dialogue.button#position



.app.domain.address

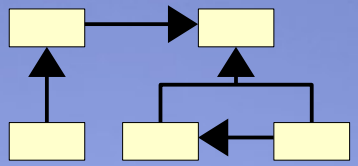


Item
+ Category
+ Compound
= Schema

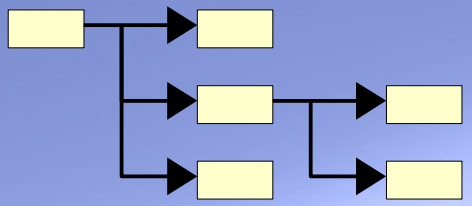
Unidirectional relations

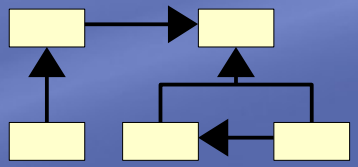
Traditional

CYBOP

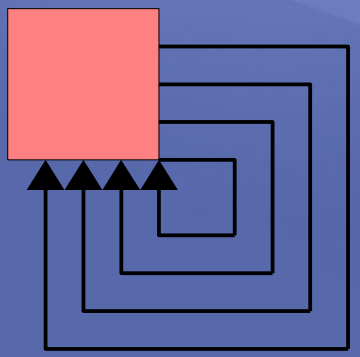


Programme Structure





Runtime Structure



CYBOI Features

one universal knowledge schema

representing the only unified container structure

one main event loop

many communication channels (terminal, xcb, win32, socket)

full i18n

Addendum

Inter-Disciplinary Ideas

Philosophy: Body & Mind = active system & passive data

Biology: Cell & DNA = system & application knowledge

Psychology: Thinking = item + category + compound

Physics: Space & Time = state-structure & logic-workflow

Solved Problems

falsified container content - due to container inheritance
fragile base class problem - due to polymorphism
untraceable data manipulation - due to global data access
circular references - due to bidirectional dependencies
bidirectional dependencies - due to reflective architectures
strong coupling - due to bundling of attributes and methods
difficult maintenance - due to high complexity

Characteristics

- long-life software system due to strict separation of application knowledge and system software
- less dependencies due to representation of structural- and meta data in form of a double hierarchy
- flexible architecture due to unified schema as the only remaining static memory structure
- clean knowledge model code due to tree structure with unidirectional relations
- reflexive meta programming due to changeable logic
- universal communication due to star-like (not layer-like) translation architecture
- easy handling due to one syntax describing structures (space) and algorithms (time) in the same way
- superfluous implementation phase due to direct interpretation of knowledge

Traditional

CYBOP

models suffer from complexity	one schema memory structure
strong coupling/dependencies	directed acyclic graph/tree
inflexible	easily extensible
difficult to maintain	long-life software system