



Guided Architecture Trade Space Exploration

Fusing Model Based Engineering and Design by Shopping

Sam Procter (sprocter@sei.cmu.edu)

Lutz Wrage

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

Copyright 2019 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

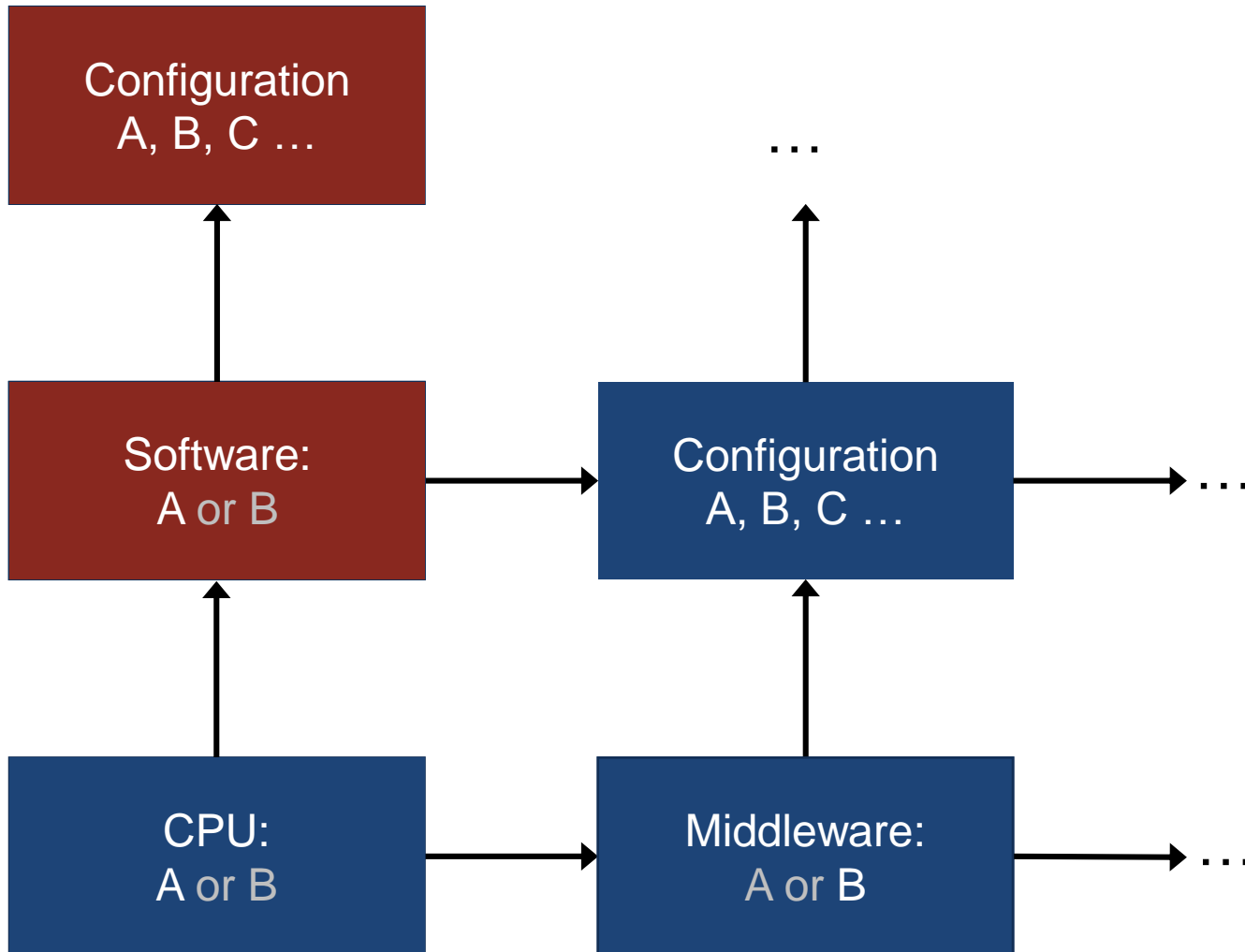
NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM19-0924

More components, more complexity



What matters:

- Satisfy functional properties
- Achieve non-functional objectives

But that's not actually how it all works.

System designers rely on their *expertise* and *intuition* instead

- Model-Based System Engineering (MBSE) supports that intuition, but has some drawbacks at large scale.
- Design Space Exploration works well at scale, but has some usability issues and rarely uses multipurpose system models

So, we created and evaluated the *Guided Architecture Trade Space Explorer*, which supports designers' intuition by integrating:

- A standardized MBSE language and tool
- An established DSE tool

Outline

A Wheel-Braking System

Designing by Shopping

Guided Architecture Trade Space Exploration

Outline

A Wheel-Braking System

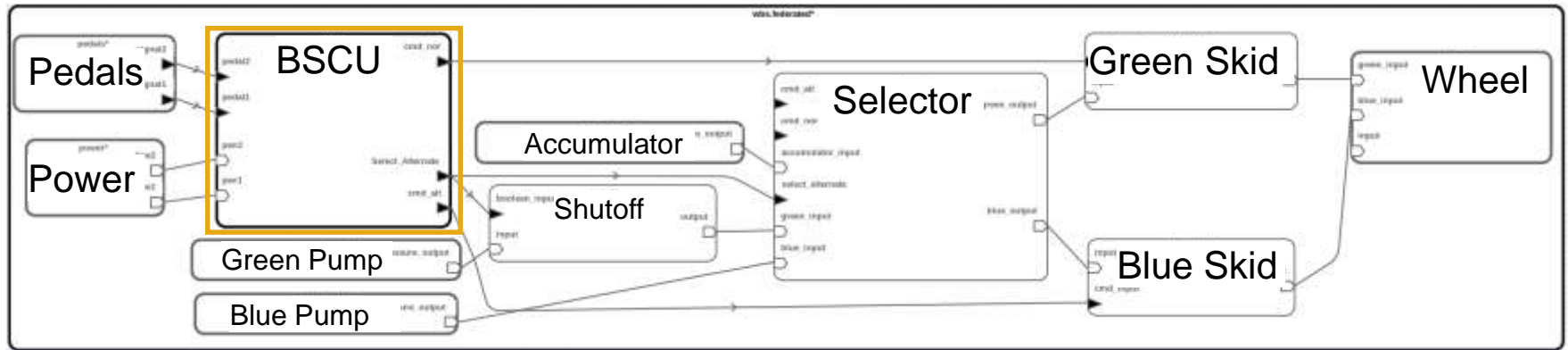
Designing by Shopping

Guided Architecture Trade Space Exploration

The wheel brake system



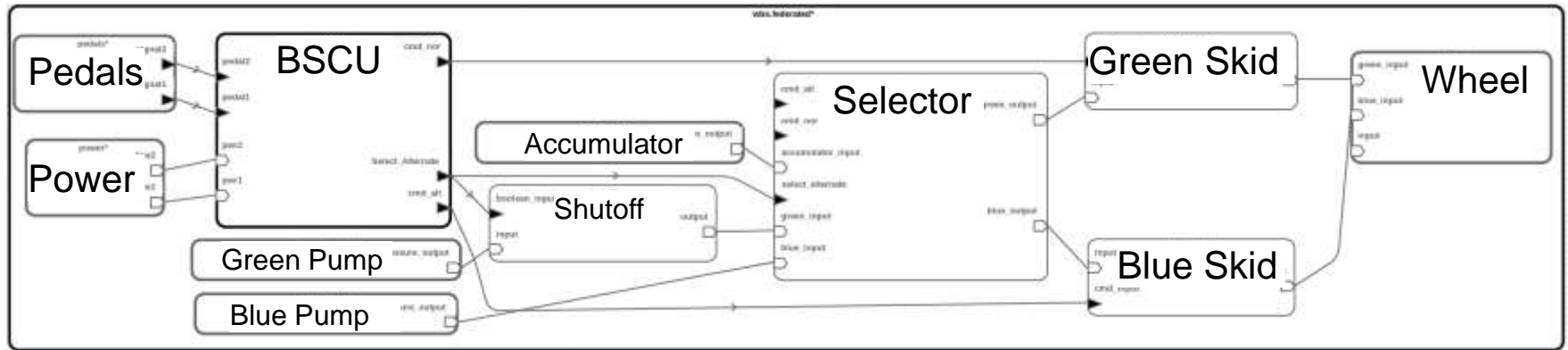
The wheel brake system



Two subsystems (command and monitor) + common platform

- Two monitor implementations, two command implementations
 - Platform varies in power budget, wiring gauge, CPU architecture
 - Multiple CPUs must have the same architecture
 - Power required by CPUs must match platform provisions
- ... and that's just one component!

Architecture Analysis and Design Language



This is AADL's graphical syntax (textual syntax on... next slide).

International standard (SAE AS5506C)

Used in academia, industry, government in the US, EU, China

<https://aadl.info>

Architecture Analysis and Design Language

```
system implementation wbs.generic
subcomponents
  -- Pedal subsystem
  pedals      : system impl::pedals::pedals.generic;

  -- Power subsystem
  power       : system impl::power::power.generic;

  -- The two pumps at the top of the diagram
  blue_pump   : system impl::pump::pump.generic;
  green_pump  : system impl::pump::pump.generic;

  -- The accumulator pump
  accumulator : system impl::pump::pump.generic;

  -- The selector subsystem
  selector    : system impl::valves::selector;
  bscu        : system impl::bscu::bscu.generic;

  wheel       : system impl::wheel::wheel.i;

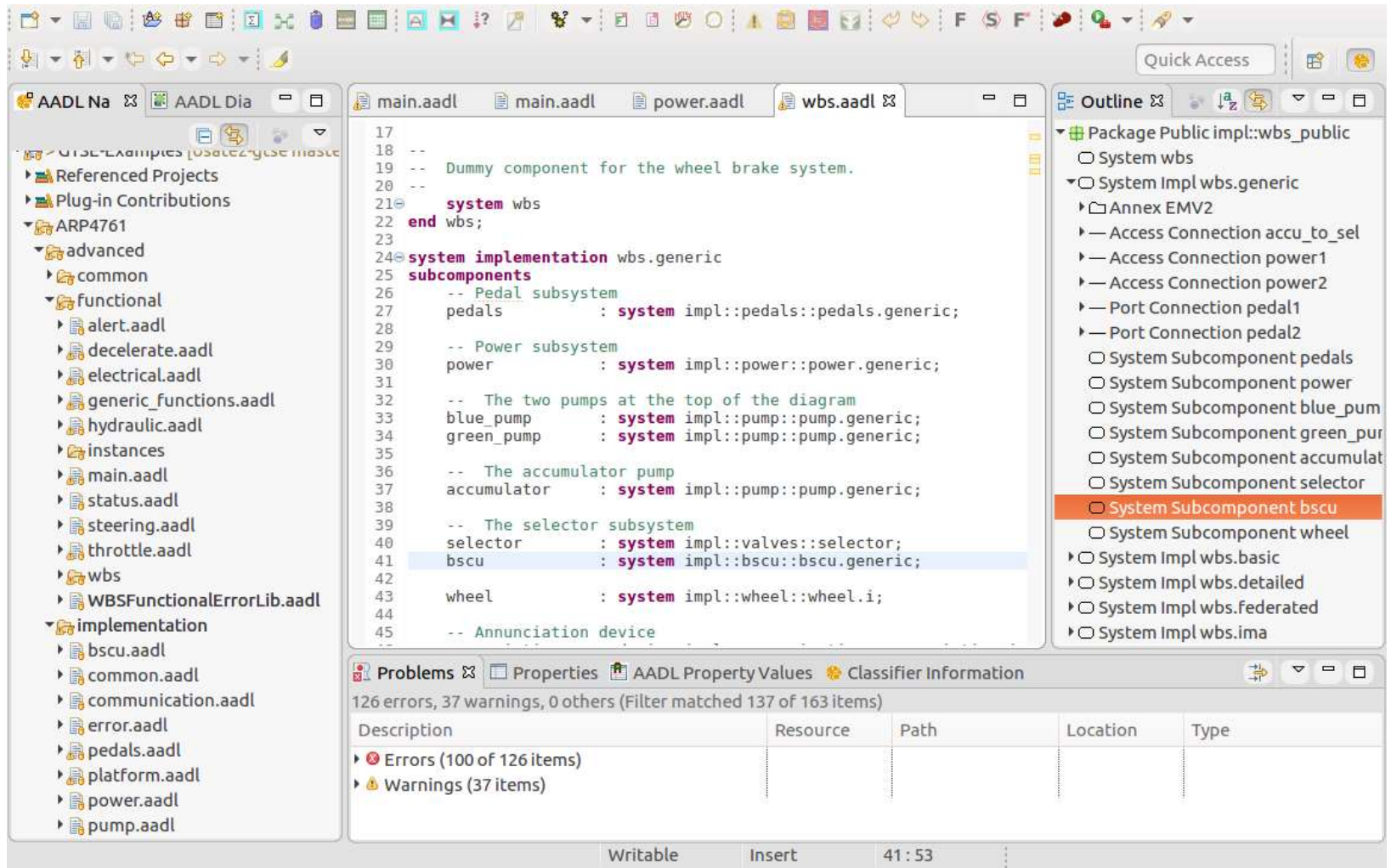
  -- Annunciation device
  -- annunciation : device impl::communication::annunciation.i;
connections
  accu_to_sel : bus access selector.accumulator_input <-> accumulator.pressure_output;
  power1      : bus access bscu.pwr1 <-> power.line1;
  power2      : bus access power.line2 <-> bscu.pwr2;
  pedal1      : port pedals.signal1 -> bscu.pedal1;
  pedal2      : port pedals.signal2 -> bscu.pedal2;
properties
  SEI::WeightLimit => 50.0 kg;
```

```
device implementation powersource.large
properties
  SEI::Price => 1000.00;
  SEI::NetWeight => 7.5 kg;
  SEI::PowerCapacity => 300.0 w;
end powersource.large;
```

Textual syntax is better for properties / computer scientists

Graphical syntax is better for structure / system engineers

Open Source Architecture Tool Environment



OSATE is open source & SEI maintained

<https://osate.org>

OSATE is a system analysis toolbench

In addition to expected IDE functionality, OSATE supports:

- Latency analysis
- Power consumption / budgeting
- Scheduling analysis
- Much more (safety, security, etc.)

... more are being added by the SEI and external researchers.

Example Domain-Specific Plugin

```
private double calcBrakingPower(ComponentInstance ci) {
    double power = 0.0;
    /* Recurse into subcomponents */
    EList<ComponentInstance> cil = ci.getComponentInstances();
    for (ComponentInstance subi : cil) {
        power += calcBrakingPower(subi);
    }
    power += PropertyUtils.getRealValue(ci,
        GetProperties.lookupPropertyDefinition(ci,
            "DemoProperties", "BrakingPower"), 0.0);
    return power;
}
```

Outline

A Wheel-Braking System

Designing by Shopping

Guided Architecture Trade Space Exploration

Designing by Shopping (Balling)

What's wrong with optimization?

- “A priori articulation of preference” (Hwang and Masud) is hard.

How do we fix it?

- Provide a range of options so users can intuitively understand tradeoffs
 - Options should be *pareto optimal*

Think of buying a shirt on Amazon...

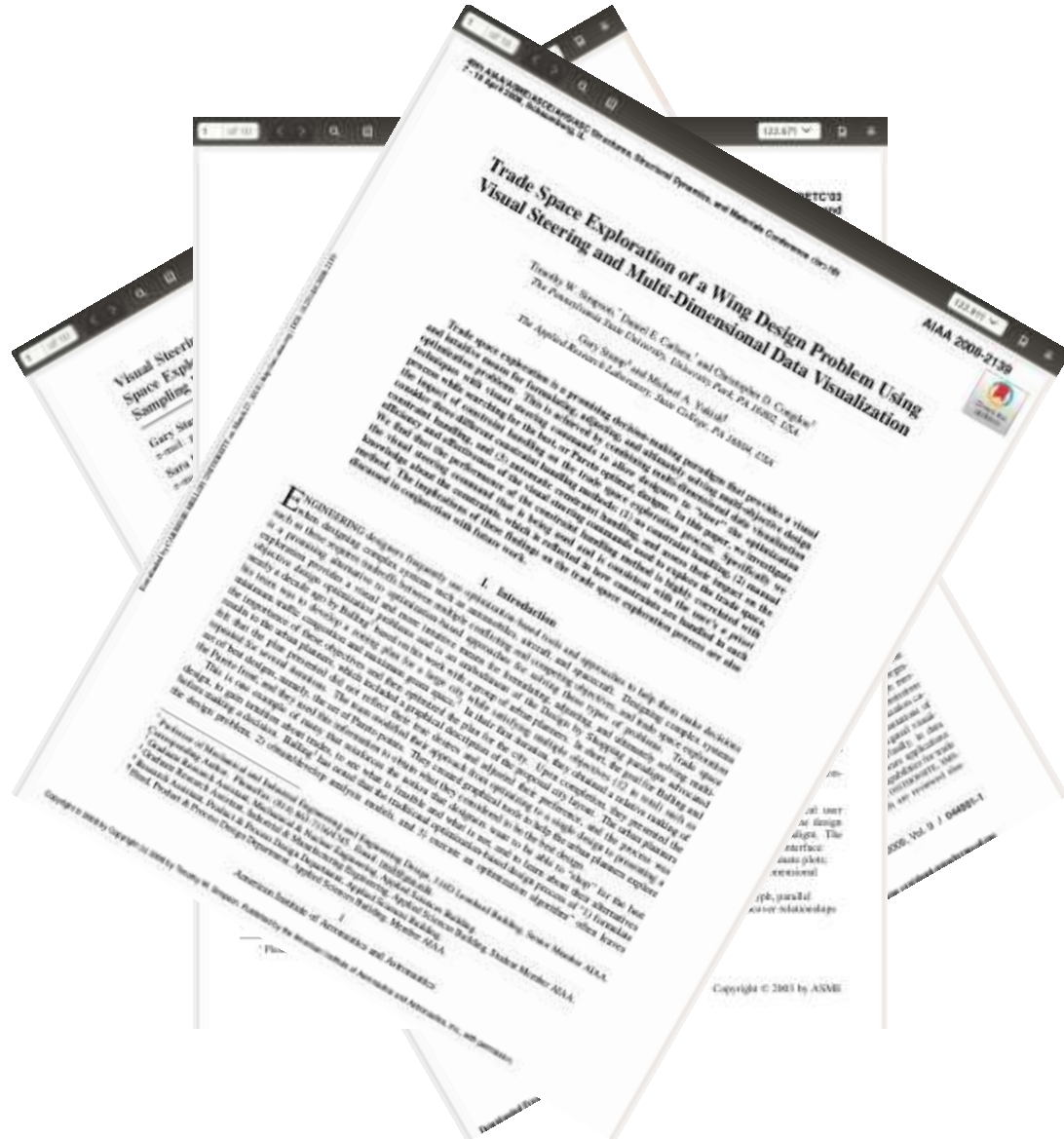
- It's hard to envision the perfect shirt without seeing any examples
 - And even if you do, what are the odds it exists?
- Look at some examples (yellow vs blue shirts) then refine your search
 - Repeat until you're satisfied

Penn State's ARL Trade Space Visualizer

Java based software for design-by-shopping.

Includes both a range of evolutionary algorithms and a variety of visualizations.

Evaluated in aeronautics and aerospace domains.



Outline

A Wheel-Braking System

Designing by Shopping

Guided Architecture Trade Space Exploration

GATSE: What is it?

Plugin for OSATE, three main elements:

- Support for new *configuration* language (more detail coming)
- Modifies OSATE's instantiation and analysis logic
 - To make it headless
 - To support “skeleton” architectures
- Creates ATSV-connection artifacts

A Configuration Language for AADL

An AADL Model

```
package P

  system S
  end S;

  system implementation S.i
    subcomponents
      sub: processor Intel;
    end S;

  processor Intel
  end Intel;

  processor implementation Intel.i3
  end Intel.i3;

  processor implementation Intel.i5
  end Intel.i5;

end P;
```

Assign a component implementation
and a property value

```
configuration C1 extends S.i {
  sub => Intel.i3;
  #SEI::Weight => 0.2 kg;
}
```

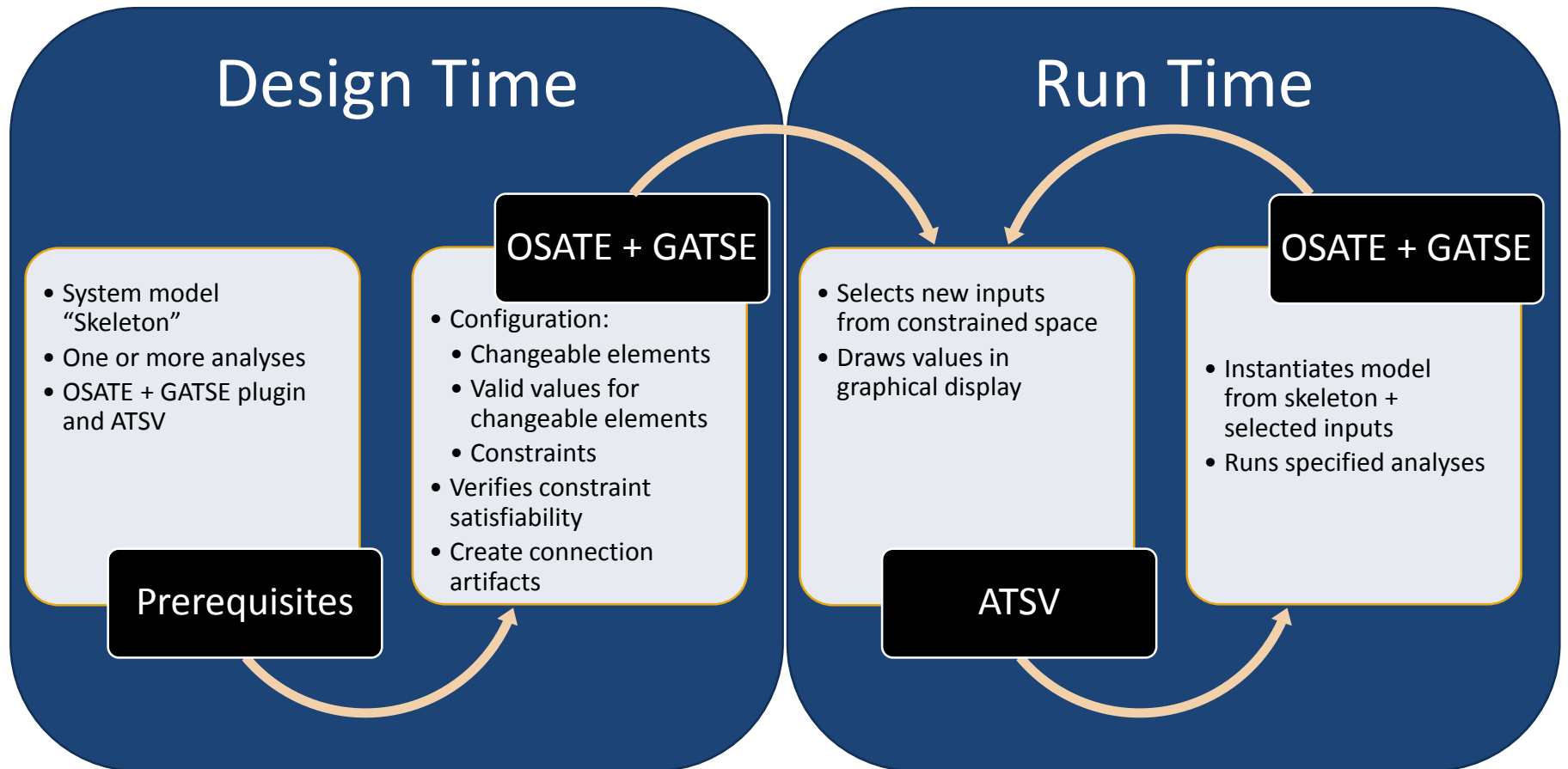
Extend a configuration and override an assignment
Assign a property in a nested configuration

```
configuration C2 extends S.i with C1 {
  sub => Intel.i5 {
    #SEI::MIPSCapacity => 1500 MIPS;
  } }
}
```

Parameterized configuration
with list of valid choices

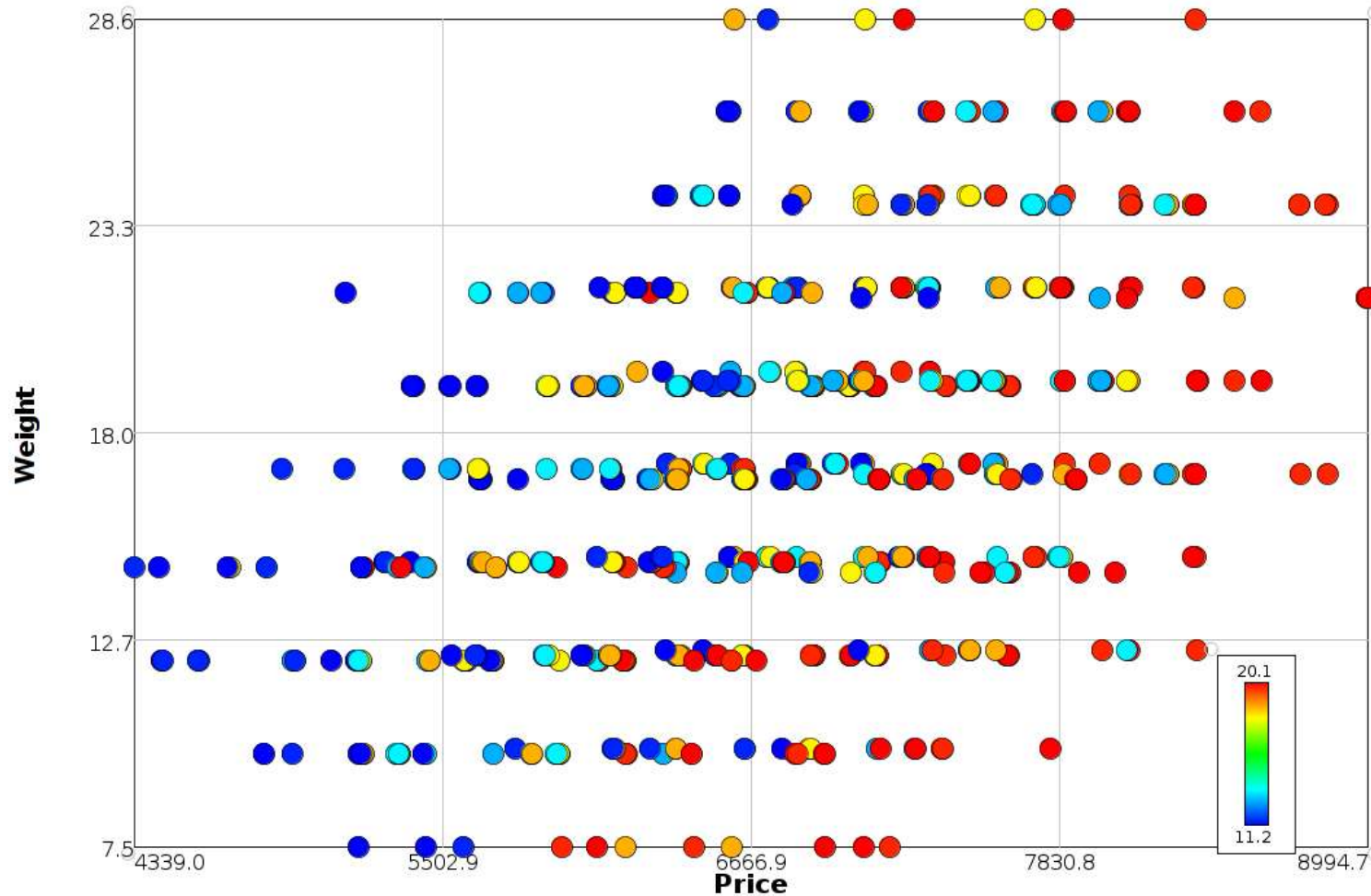
```
configuration C3 (
  proc: processor Intel
    from (Intel.i3, Intel.i5)
) extends S.i {
  sub => proc;
  #SEI::MIPSCapacity => 1000MIPS;
}
```

GATSE: How do you use it?



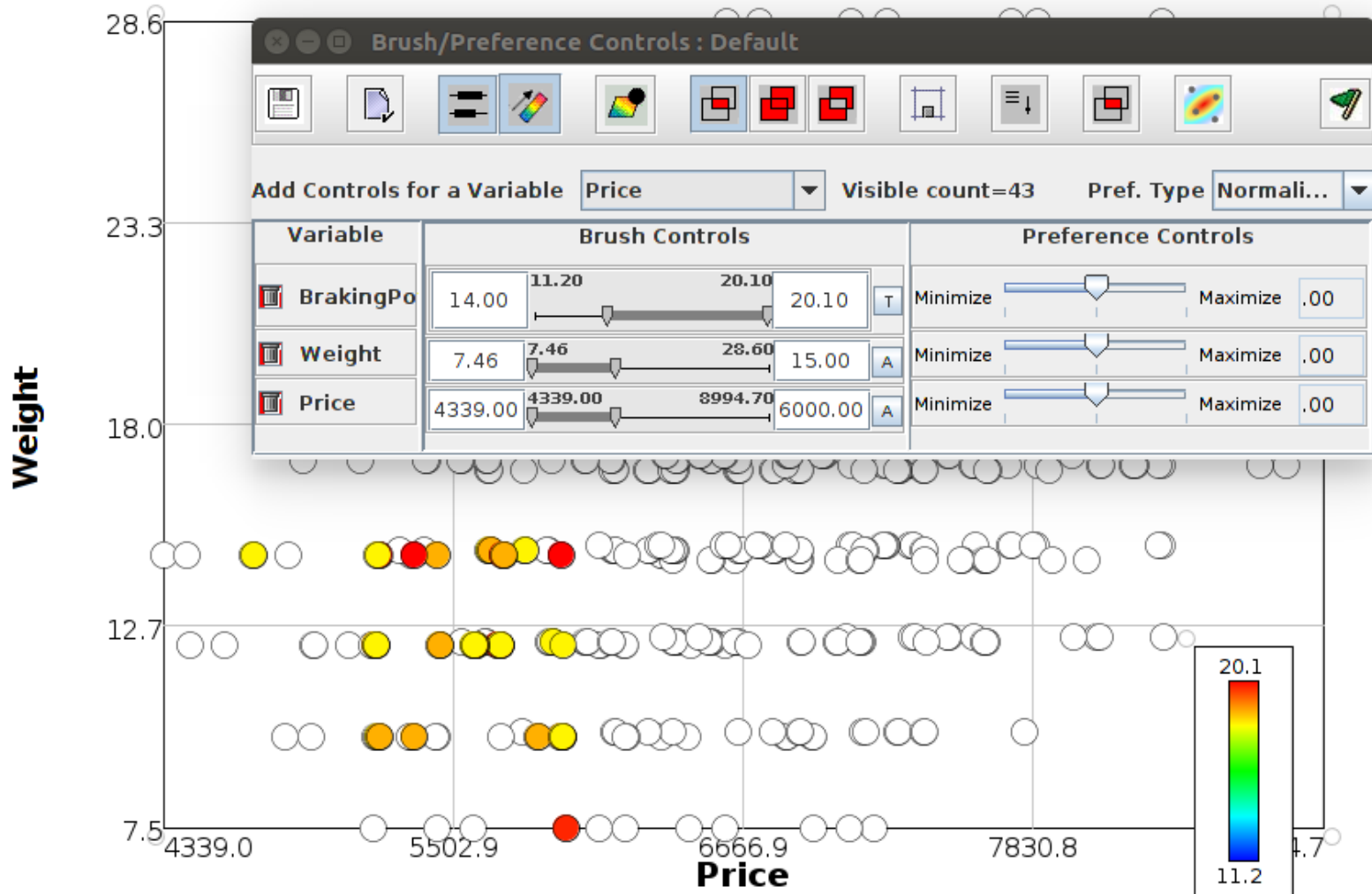
GATSE (ATSV): In action – Viewing

Weight vs. Price vs. Braking Power

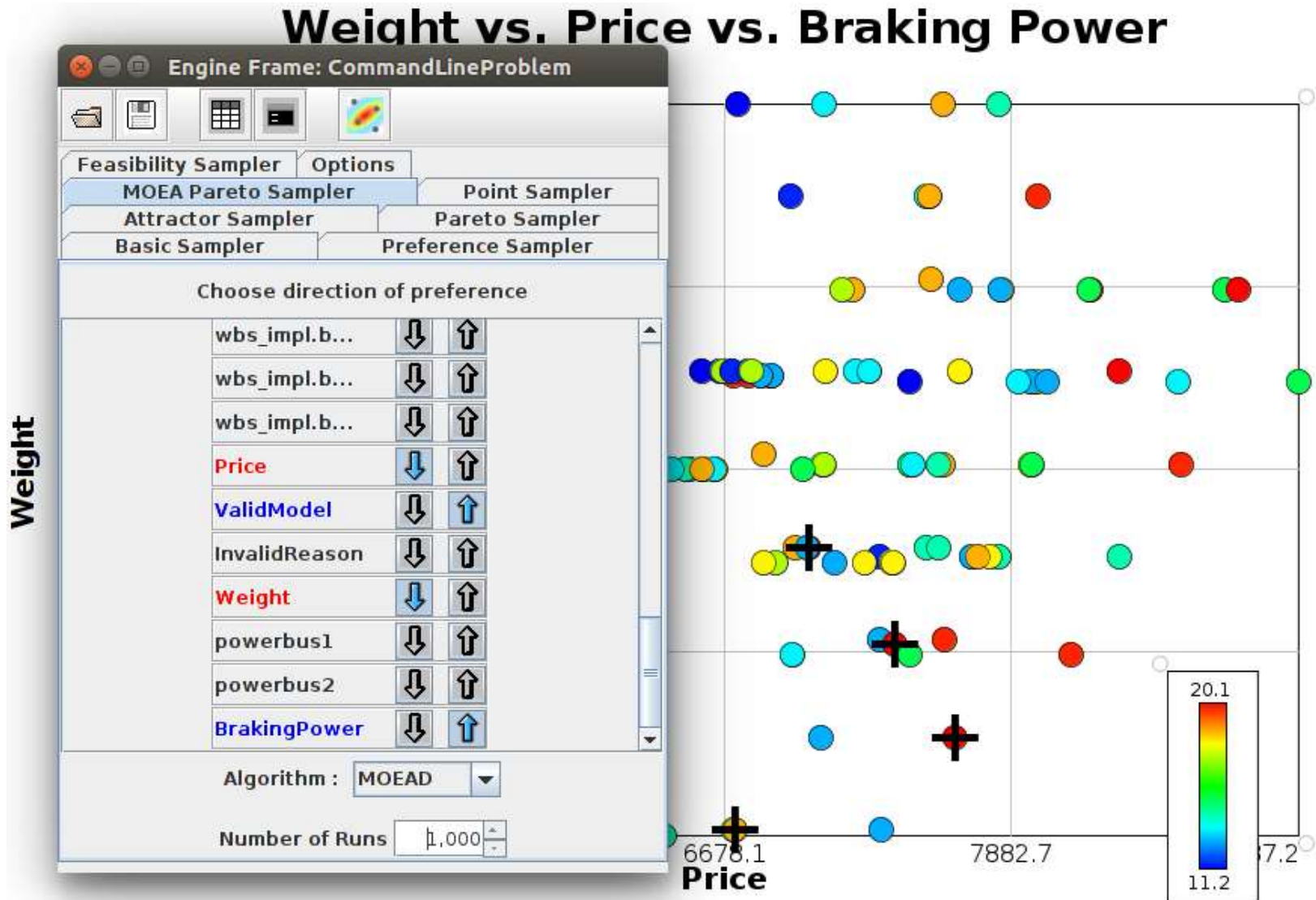


GATSE (ATSV): In action – Filtering

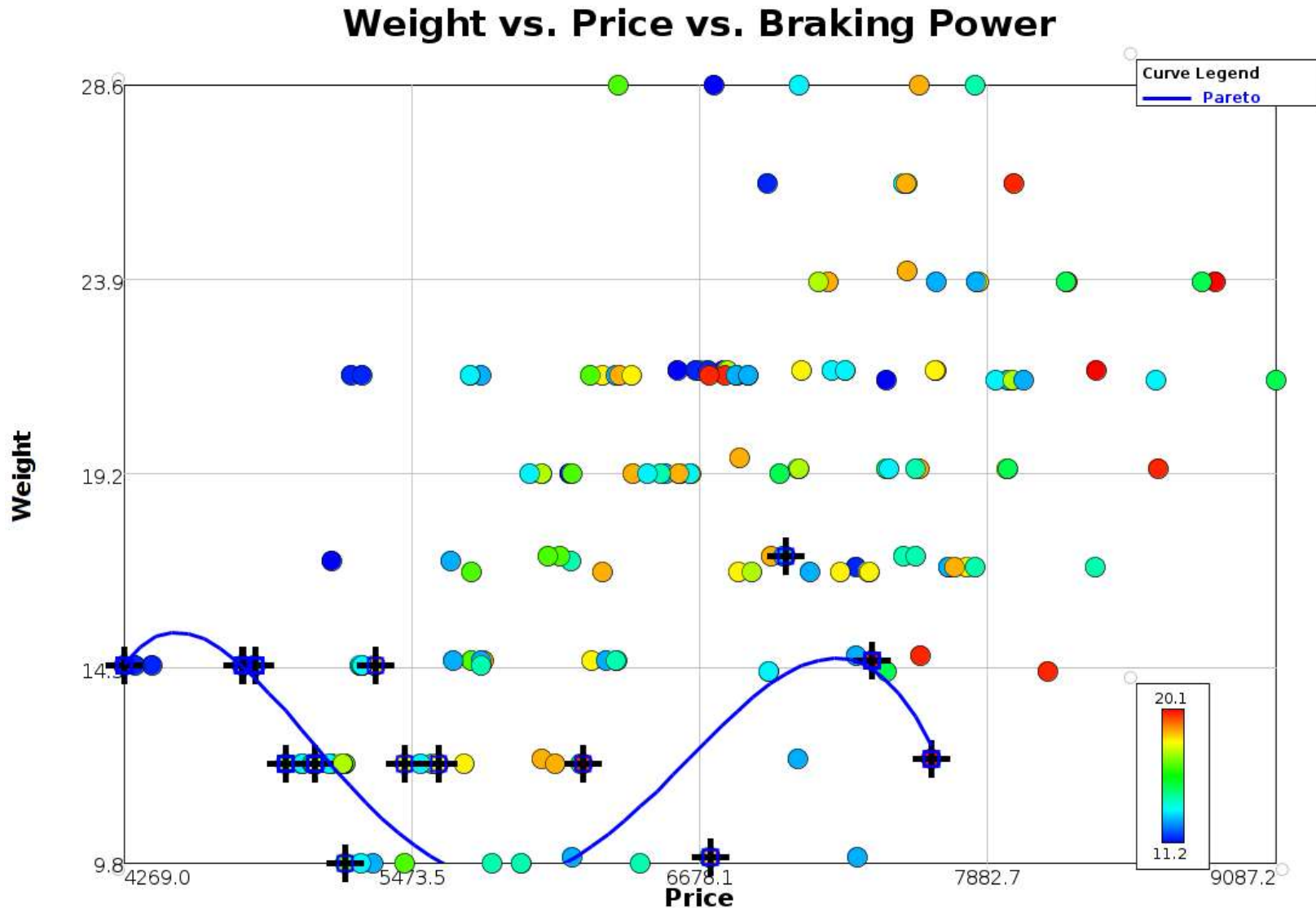
Weight vs. Price vs. Braking Power



GATSE (ATSV): In action – Tailoring

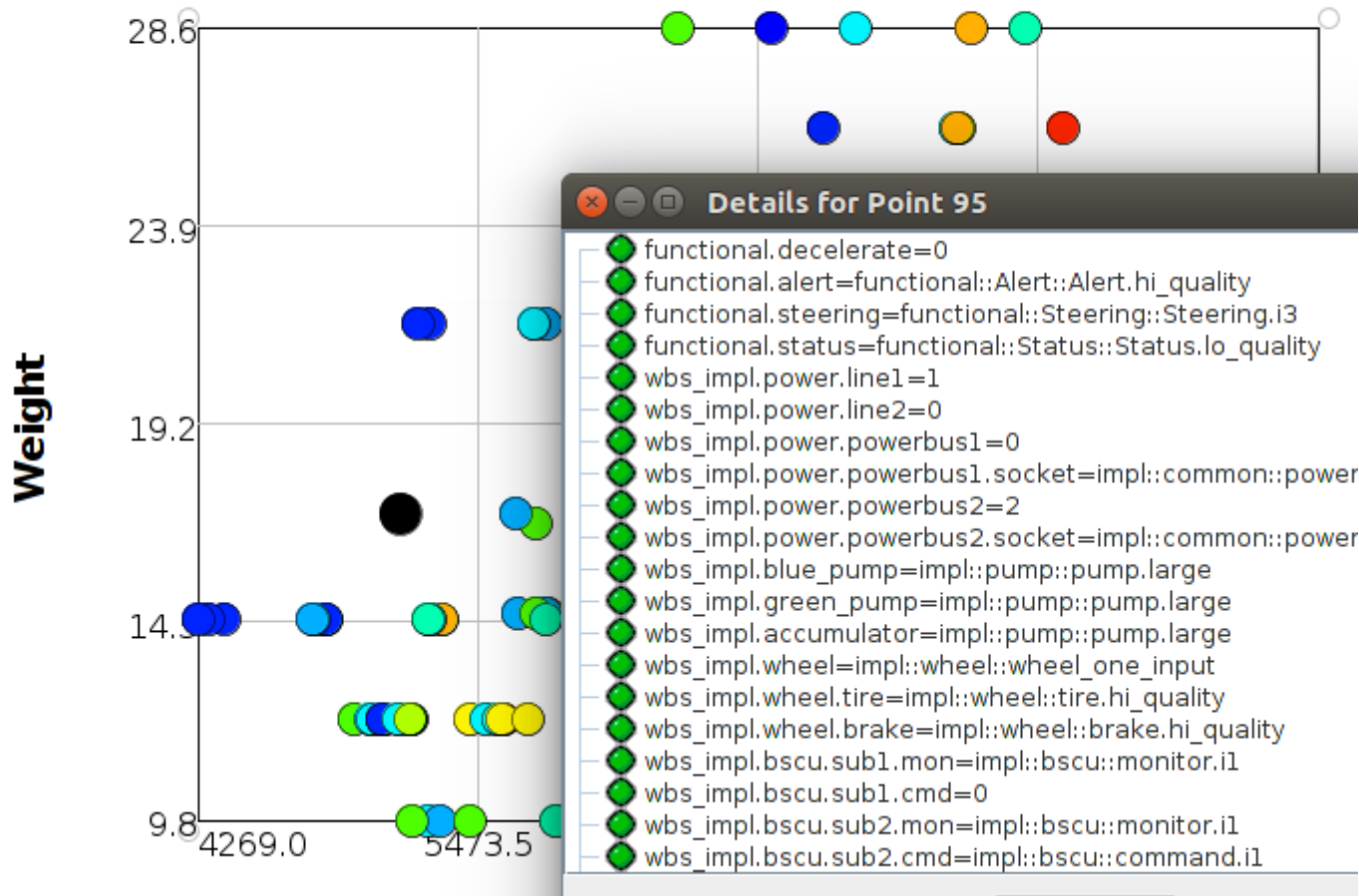


GATSE (ATSV): In action – Pareto



GATSE (ATSV): In action – Detail

Weight vs. Price vs. Braking Power



Example Domain-Specific Plugin

```
public class BrakingPower extends AbstractAnalysis {  
    @Override public void runAnalysis(SystemInstance instance,  
        SystemOperationMode som, AnalysisErrorReporterManager errMgr,  
        IProgressMonitor progressMonitor, Response resp) {  
        resp.addVariable("BrakingPower", ATSVVariableType.FLOAT,  
            String.valueOf(calcBrakingPower(instance)));  
    }  
  
    private double calcBrakingPower(ComponentInstance ci) {  
        double power = 0.0;  
        /* Recurse into subcomponents */  
        EList<ComponentInstance> cil = ci.getComponentInstances();  
        for (ComponentInstance subi : cil) {  
            power += calcBrakingPower(subi);  
        }  
        power += PropertyUtils.getRealValue(ci,  
            GetProperties.lookupPropertyDefinition(ci,  
                "DemoProperties", "BrakingPower"), 0.0);  
        return power;  
    }  
}
```

The GATSE Vision

AADL: Custom Properties

```

device implementation tire.hi_quality
properties
-- Built-in properties supporting
-- cost and weight analyses
SEI::Price => 1000.0;
SEI::NetWeight => 3.5 kg;

-- Custom property supporting domain-
-- specific analysis, potentially
-- derived from other analysis /
-- modeling tools
DemoProperties::BrakingPower => 10.0;
end tire.hi_quality;
    
```

OSATE: Custom Analyses

```

public class BrakingPower extends AbstractAnalysis {
@Override public void runAnalysis(SystemInstance instance,
SystemOperationMode som, AnalysisErrorReporterManager errMgr,
IProgressMonitor progressMonitor, Response resp) {
    resp.addVariable("BrakingPower", ATSVVariableType.FLOAT,
String.valueOf(calcBrakingPower(instance)));
}

private double calcBrakingPower(ComponentInstance ci) {
double power = 0.0;
// RECURSE INTO subcomponents
EList<ComponentInstance> cil = ci.getComponentInstances();
for (ComponentInstance subi : cil) {
    power += calcBrakingPower(subi);
}
power += PropertyUtils.getRealValue(ci,
GetProperties.lookupPropertyDefinition(ci,
"DemoProperties", "BrakingPower", 0.0));
return power;
}
}
    
```

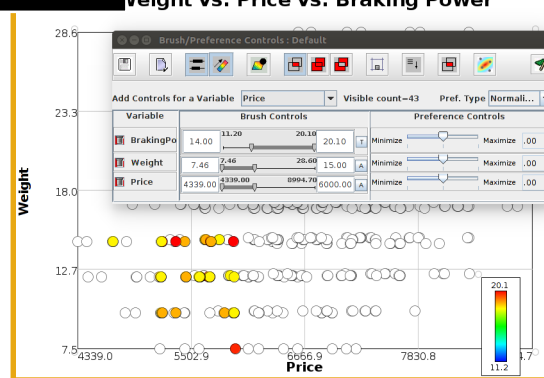


Enables

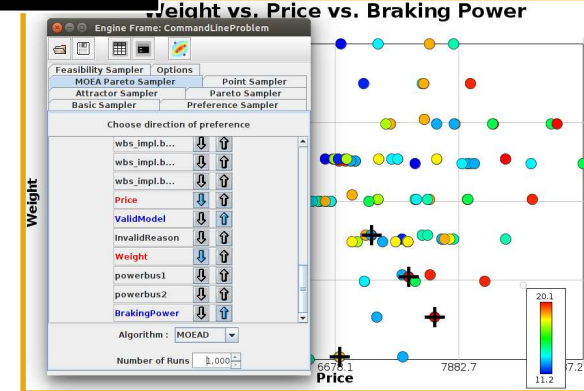
Viewing



Filtering



Tailoring





Guided Architecture Trade Space Exploration

Fusing Model Based Engineering and Design by Shopping

Sam Procter (sprocter@sei.cmu.edu)

Lutz Wrage

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213