

RESEARCH @

ECLIPSE
FOUNDATION

Open Research Webinars

OW2

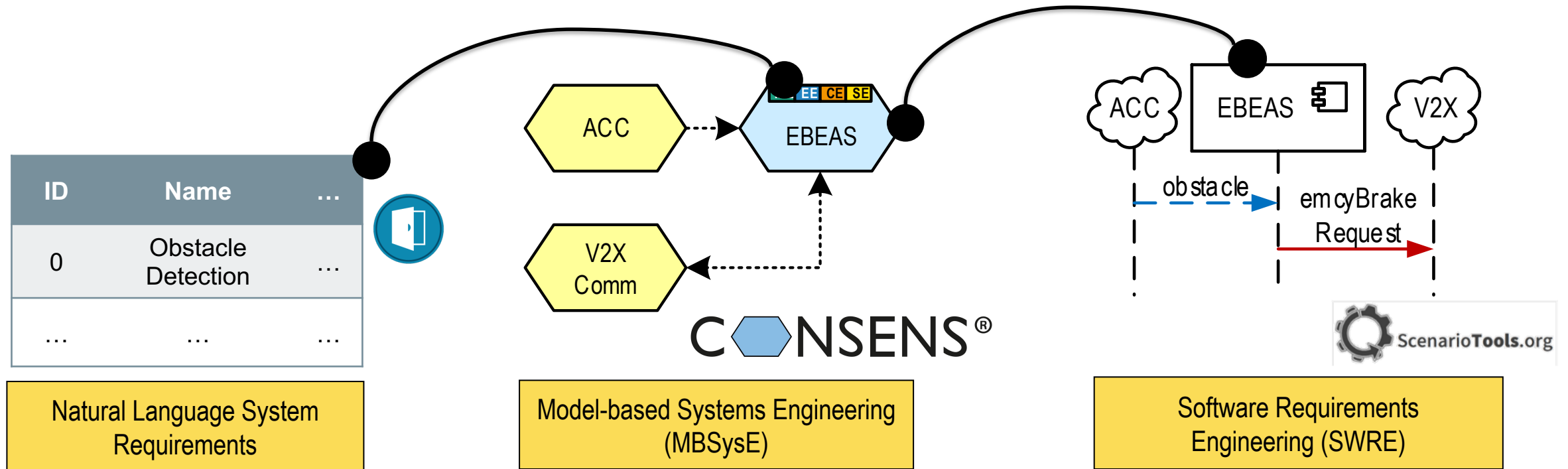
Extensible Traceability Management with Eclipse Capra

Jan-Philipp Steghöfer

Chalmers University of Technology | University of Gothenburg

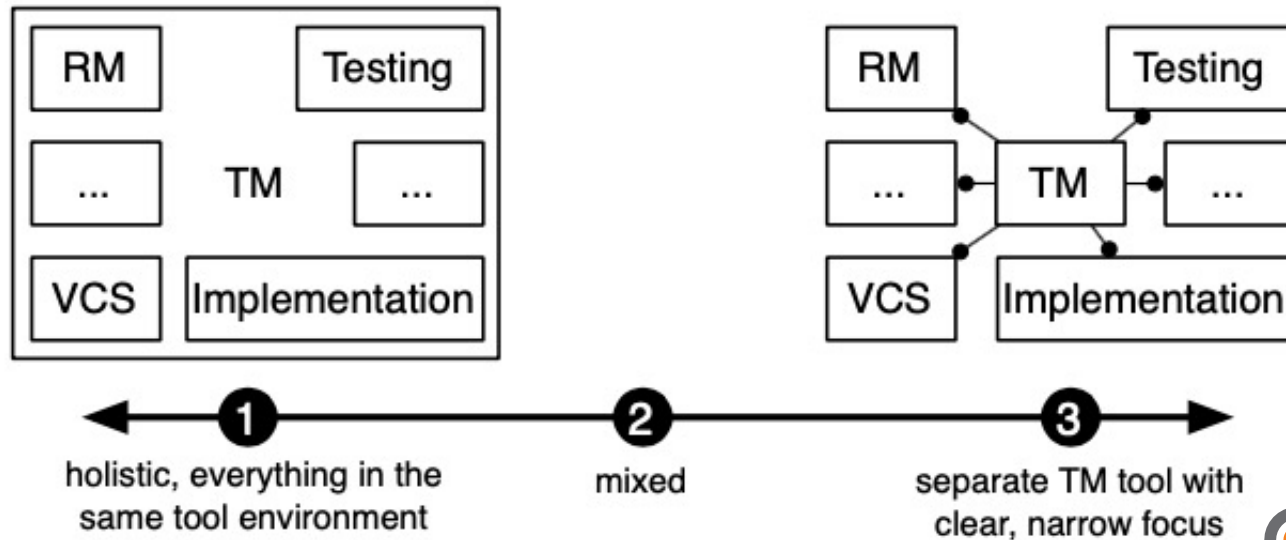


Traceability



Example by David Schmelter, Fraunhofer IEM, Paderborn

Tool Solutions



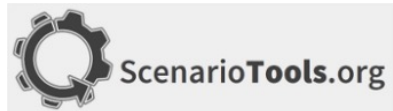
- Holistic solution: full control over all artifacts and trace models; simplifies maintenance; almost impossible to achieve in practice
- Separate TM tool: more effort in maintenance required, easier to extend, easier to use in existing heterogeneous environments



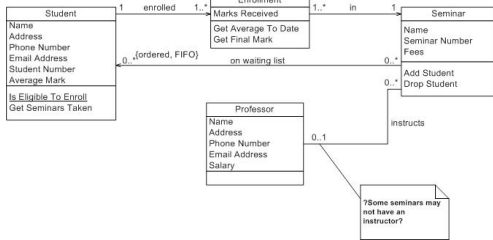
Salome Maro, Anthony Anjorin, Rebekka Wohlrab, Jan-Philipp Steghöfer:
Traceability maintenance: factors and guidelines. ASE 2016: 414-425

Eclipse Capra

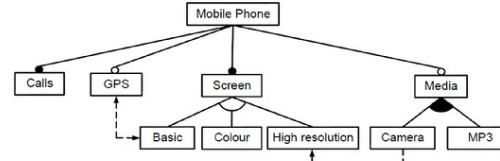
Requirements



Design



Variability



Tests



Grails Plugins					
S	W	Job	Last Success	Last Failure	Last Duration
●	☀	grails-joda-time	55 min (#10)	7 hr 44 min (#8)	36 sec
●	☀	grails-selenium-rc	11 hr (#23)	23 hr (#18)	3 min 54 sec
●	☀	grails-session-temp-files	11 hr (#6)	23 hr (#1)	23 sec
●	☀	grails-springcache	11 hr (#36)	1 day 0 hr (#31)	3 min 18 sec
●	☀	grails-tellurium	N/A	11 hr (#8)	2 min 16 sec



Source Code

```
public void setLocaleToCookie(HttpServletResponse request, String locale) {
    cookieGenerator.addCookie(request, locale);
}

public String getLocaleFromCookie(HttpServletRequest request) {
    Cookie cookie = WebUtils.getCookie(request, REQUEST_ATTRIBUTE_LANGUAGE);
    return cookie != null ? cookie.getValue() : null;
}

@Override
public void doInit() throws ServletException {
    cookieGenerator.setCookieName(REQUEST_ATTRIBUTE_LANGUAGE);
}
```



Heterogeneous systems

AMALTHEA Contents Tree

AMALTHEA_Democar_MappingExample_with

This section enables the contents of this element

- Labels (71)
- Tasks (6)
 - Task_PP3
 - Task_PP0
 - Task_PP2

Eclipse Capra

- Driven by requirements from industrial partners in a number of research projects
- Only (known, active) universal open source traceability tool
- Goals:
 - Test-bed for new traceability approaches and ideas
 - Enable studies and research on traceability in practical settings
 - Industrial strength: should be usable in production



11

**Research
Papers**

15

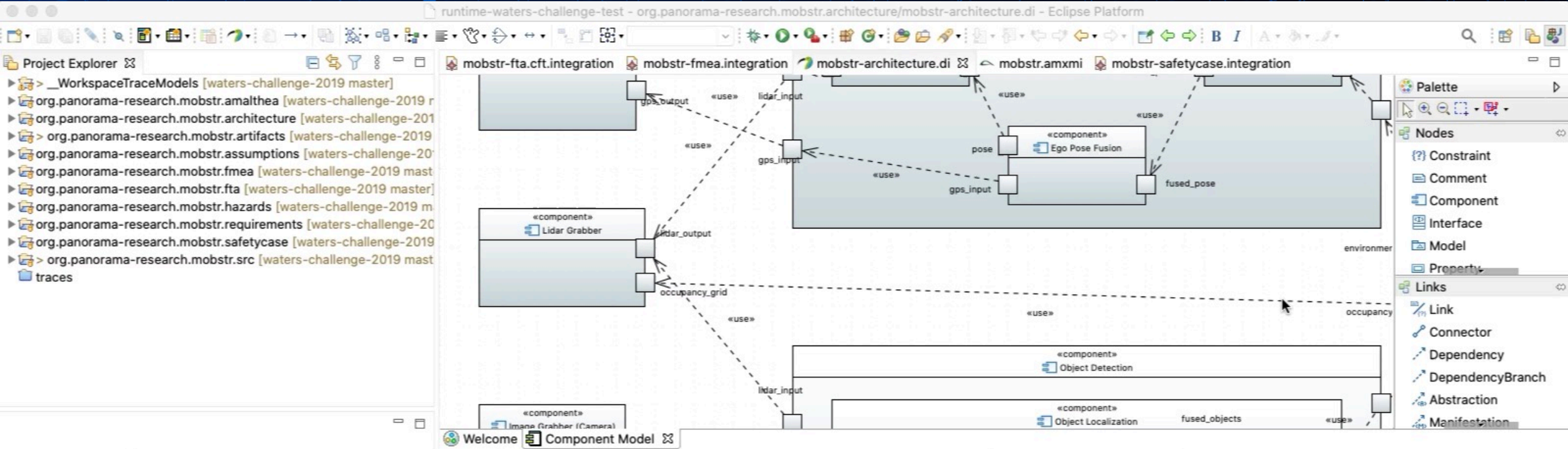
**Thesis
Projects**

5

**Research
Projects**

Performing common traceability tasks with Eclipse Capra

Create trace links



runtime-waters-challenge-test - org.panorama-research.mobstr.architecture/mobstr-architecture.di - Eclipse Platform



-
- Lidar Grabber : Component ->
No point cloud provided by the lidar grabber : Failure; : HasFailureMode
- occupancy_grid : Port (Go to)
- lidar_output : Port (Go to)
- Lidar Grabber : Component ->
No point cloud provided by the lidar grabber : Failure;
: RelatedTo
- No point cloud
- Lidar Grabber : Component ->
Erroneous point cloud provided by the lidar grabber (false negative) : Failure; : HasFailureMode
- Erroneous point cloud provided by the lidar grabber (false negative)

The screenshot displays the Eclipse Platform interface with the Mobstr architecture diagram open. The diagram shows a central component model with various components and their interactions. The components include:

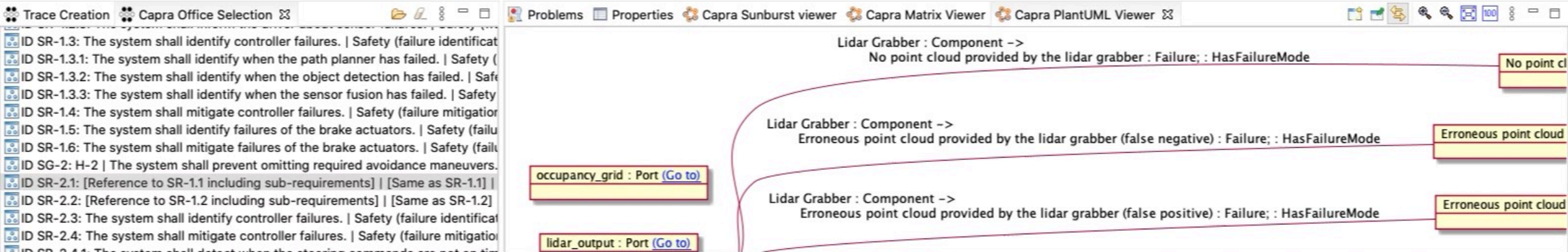
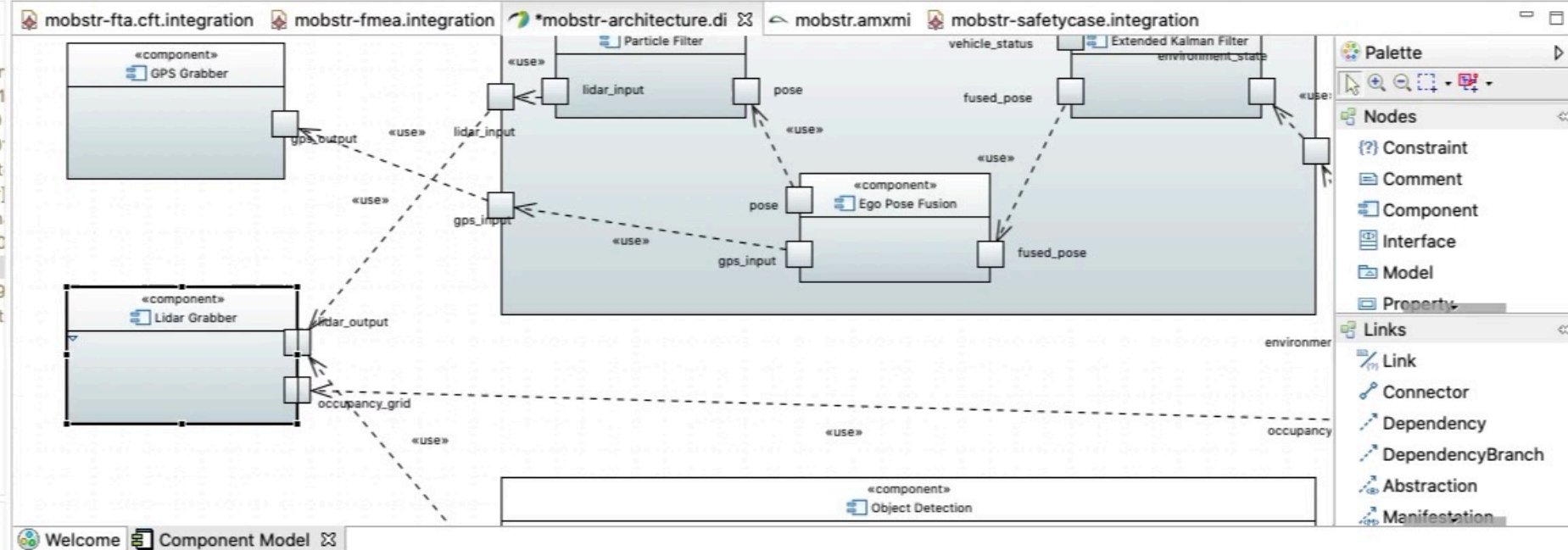
- GPS Grabber**: A component that provides `gps_output` and `gps_input` ports.
- Lidar Grabber**: A component that provides `lidar_output` and `occupancy_grid` ports.
- Particle Filter**: A component that uses `lidar_input` and `pose` to provide `pose` and `gps_input`.
- Ego Pose Fusion**: A component that uses `pose` and `gps_input` to provide `pose` and `gps_input`.
- Extended Kalman Filter**: A component that uses `vehicle_status` and `environment_state` to provide `environment_state` and `environment`.
- Object Detection**: A component that uses `occupancy_grid` to provide `occupancy`.

The diagram also shows various use cases and dependencies between these components. For example, the `GPS Grabber` is used by the `Particle Filter` and `Ego Pose Fusion`. The `Lidar Grabber` is used by the `Particle Filter` and `Ego Pose Fusion`. The `Particle Filter` is used by the `Ego Pose Fusion`. The `Ego Pose Fusion` is used by the `Extended Kalman Filter`. The `Extended Kalman Filter` is used by the `Object Detection`.

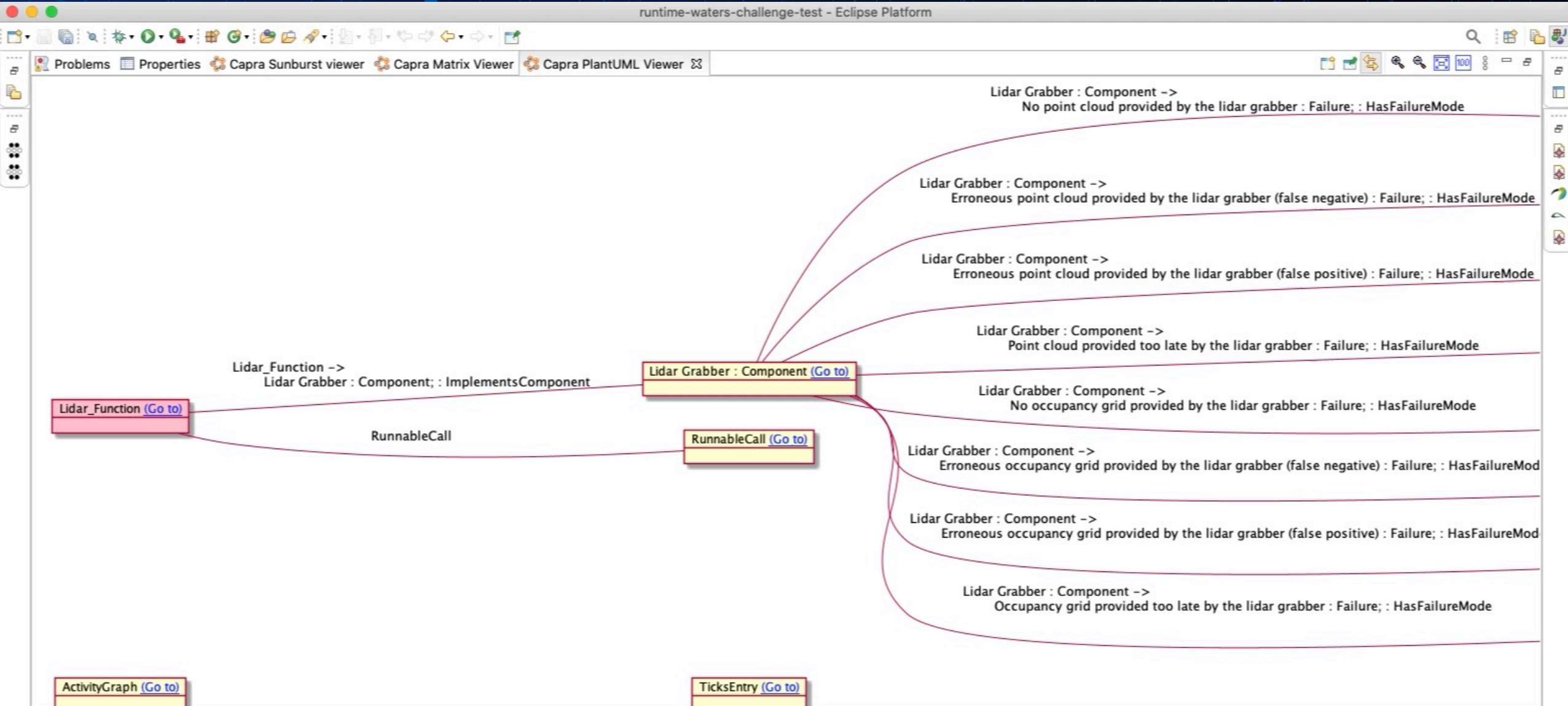
Below the diagram, the **Trace Creation** tab is active, showing a list of requirements (ID SR-1.3, ID SR-1.3.1, ID SR-1.3.2, ID SR-1.3.3, ID SR-1.4, ID SR-1.5, ID SR-1.6, ID SG-2: H-2, ID SR-2.1, ID SR-2.2, ID SR-2.3, ID SR-2.4, ID SR-2.4.1) and their associated failure modes. The requirements are grouped into three categories:

- Failure; : HasFailureMode**: This category includes requirements ID SR-1.3, ID SR-1.3.1, ID SR-1.3.2, ID SR-1.3.3, ID SR-1.4, ID SR-1.5, ID SR-1.6, ID SG-2: H-2, ID SR-2.1, ID SR-2.2, ID SR-2.3, ID SR-2.4, and ID SR-2.4.1.
- Failure; : HasFailureMode**: This category includes requirements ID SR-1.3.1, ID SR-1.3.2, ID SR-1.3.3, ID SR-1.4, ID SR-1.5, ID SR-1.6, ID SG-2: H-2, ID SR-2.1, ID SR-2.2, ID SR-2.3, ID SR-2.4, and ID SR-2.4.1.
- Failure; : HasFailureMode**: This category includes requirements ID SR-1.3.1, ID SR-1.3.2, ID SR-1.3.3, ID SR-1.4, ID SR-1.5, ID SR-1.6, ID SG-2: H-2, ID SR-2.1, ID SR-2.2, ID SR-2.3, ID SR-2.4, and ID SR-2.4.1.

The requirements are also linked to specific failure modes, such as "No point cloud provided by the lidar grabber", "Erroneous point cloud provided by the lidar grabber (false negative)", and "Erroneous point cloud provided by the lidar grabber (false positive)".



Coverage Analysis



Extending Eclipse Capra

Defining a custom TIM

```

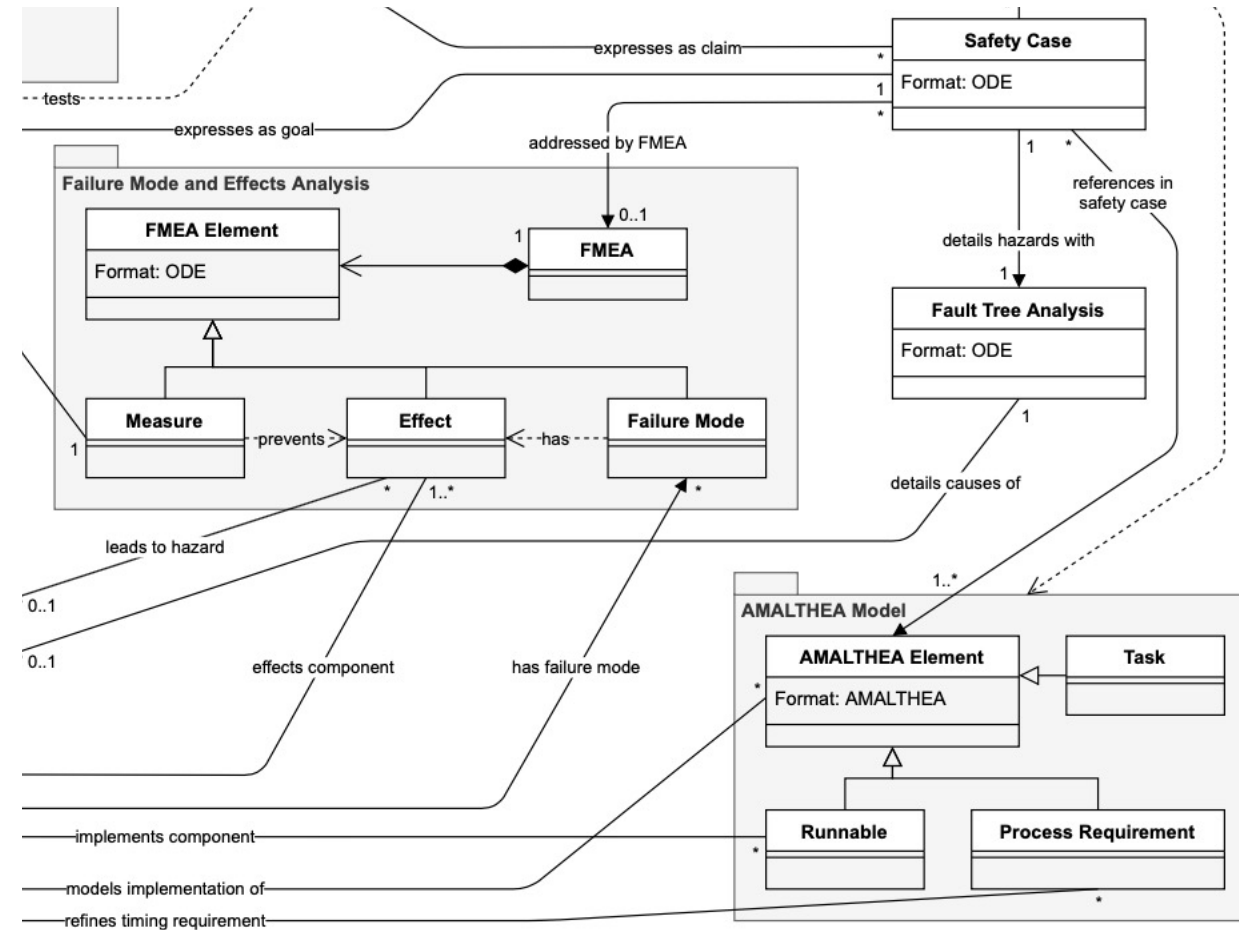
/*
 * Links Safety Case to FMEA
 */
class AddressedByFMEA extends DirectedTraceLink {
  refers Element [1] source // Safety Case
  refers FMEA [1] target // FMEA
}

/*
 * Links Safety Case to Amalthea Element
 */
class ReferencesInEvidences extends DirectedTraceLink {
  refers Element [1] source // Element -- Safety Case
  refers INamed [1..*] target // Amalthea Element
}

/*
 * Links the Fault Tree Analysis to a Hazard
 */
class DetailsCausesOf extends DirectedTraceLink {
  refers Element [1] source // Element -- Fault Tree Analysis
  refers Element [1..*] target // Element -- Hazard
}

/*
 * Links the Safety Case to the Fault Tree Analysis
 */
class DetailsHazardWith extends DirectedTraceLink {
  refers Element [1] source // Element -- Safety Case
  refers Element [1] target // Element -- Fault Tree Analysis
}

```



Handlers for non-EMF artifacts

```
public class CDTHandler extends AbstractArtifactHandler<ICElement> implements IAnnotateArtifact {

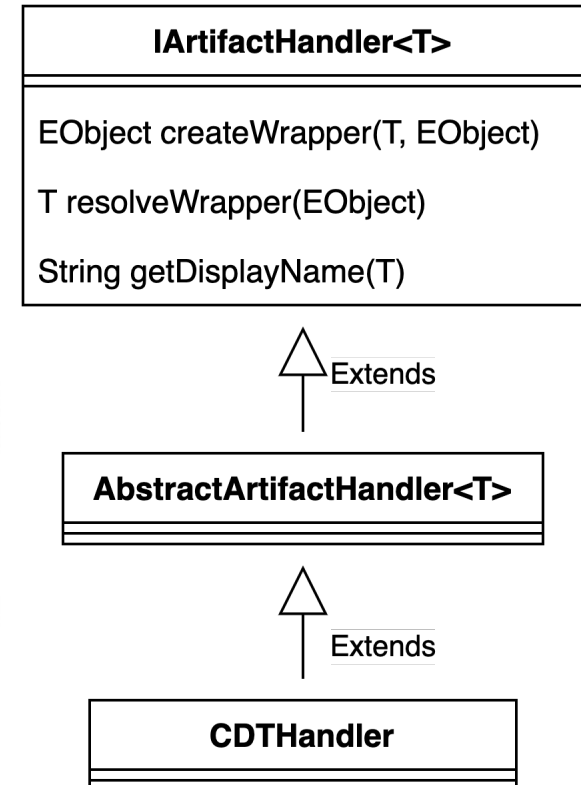
    @Override
    public EObject createWrapper(ICElement element, EObject artifactModel) {
        ICompositeType type = (ICompositeType) element.getParent().getAncestor(ICElement.C_CLASS);
        if (type == null)
            type = (ICompositeType) element.getParent().getAncestor(ICElement.C_STRUCT);
        if (type == null)
            type = (ICompositeType) element.getParent().getAncestor(ICElement.C_UNION);

        String typePrefix = type == null ? "" : type.getName() + "/";
        String uri = new URIBuilder().setScheme("platform").setPath("/resource" + element.getPath())
            .setFragment(typePrefix + element.getElementName()).toString();

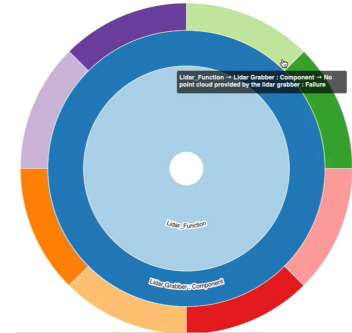
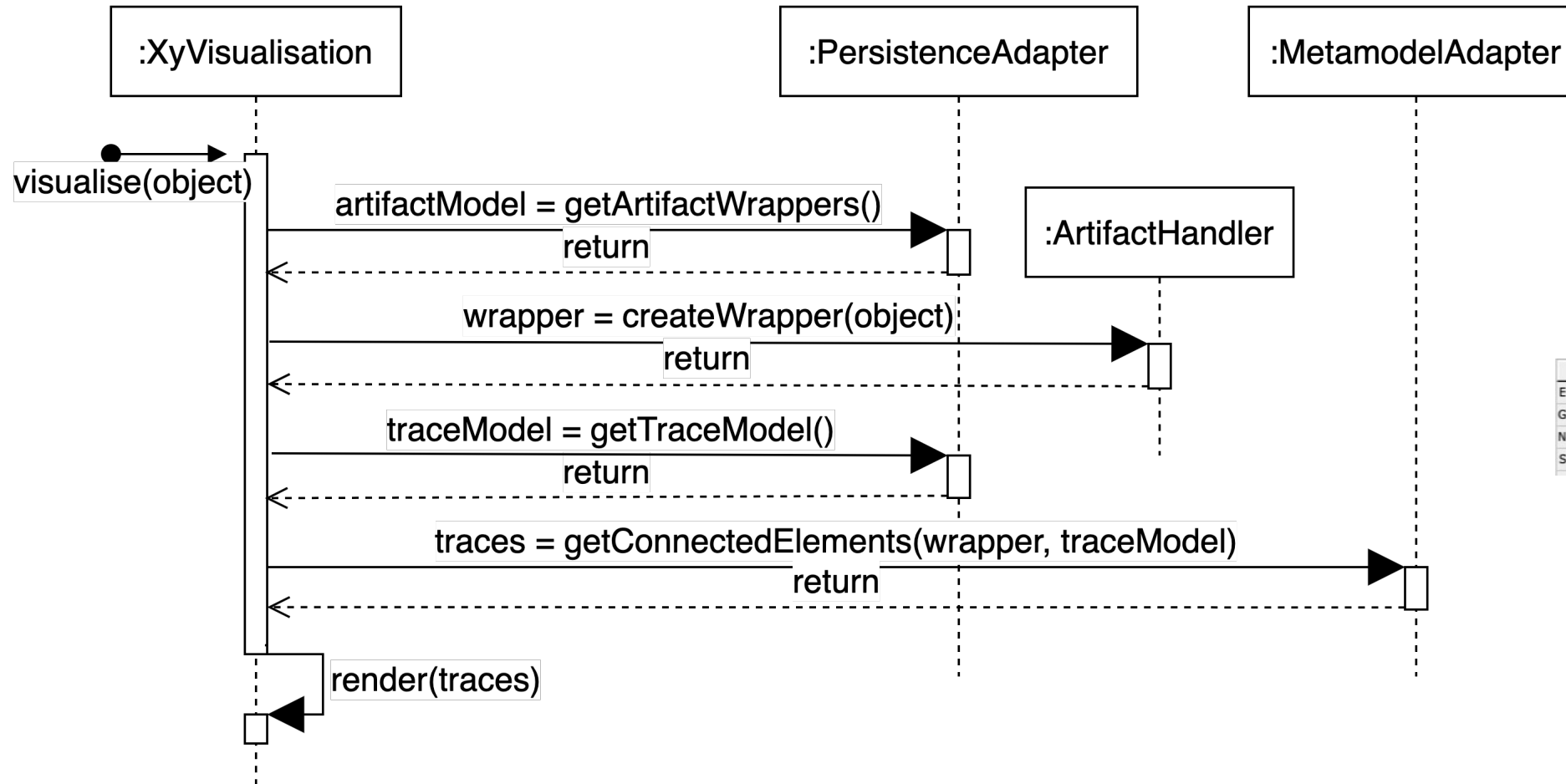
        ArtifactMetaModelAdapter adapter = ExtensionPointHelper.getArtifactWrapperMetaModelAdapter().orElseThrow();
        return adapter.createArtifact(artifactModel, this.getClass().getName(), uri, element.getHandleIdentifier(),
            element.getElementName(), element.getPath().toString());
    }

    @Override
    public ICElement resolveWrapper(EObject wrapper) {
        ArtifactMetaModelAdapter adapter = ExtensionPointHelper.getArtifactWrapperMetaModelAdapter().orElseThrow();
        return CoreModel.create(adapter.getArtifactIdentifier(wrapper));
    }

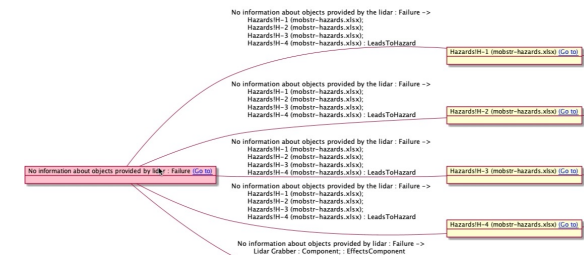
    @Override
    public String getDisplayName(ICElement element) {
        return element.getElementName();
    }
}
```



Additional visualisations/analysis



	Erroneous posi...	GPS Grabber : C...	No position infor...	Stuck-at failure ...
Errone...		HasFailureMode		
GPS Gr...				
No pos...		HasFailureMode		
Stuck-...		HasFailureMode		



Eclipse Capra – Traceability management for the Eclipse platform

- Extensible and flexible – easy to adapt to your own needs
 - Completely open source
 - Compatible with many common artifacts out of the box
 - Driven by research – driving research
 - We are looking for additional committers!
-
- <https://eclipse.org/capra>

