View Inheritance as an Extension of the Normalization Ontology Design Pattern

Bene Rodriguez-Castro, Hugh Glaser and Ian Millard
University of Southampton, Southampton, UK
{b.rodriguez, hg, icm}@ecs.soton.ac.uk

University of Southampton
School of Electronics and Computer Science

Motivation

- The ReSIST (Resilience for Survivability in Information Society Technologies) project features a semantic web portal in the field of resilient and dependable computing: The ReSIST Knowledge Base Explorer (RKB Explorer) <www.rkbexplorer.com/explorer/>
- The representation of the concept **`Fault`** (Figure 1) in the ontology built for the RKB Explorer is **difficult to model** due to:
  - The complexity of its **definition**.
  - The number of **roles** that it fulfils in the ontology.
  - The number and different types of **relationships** that it participates in.
- The representation of the **`Fault`** domain concept has also to support:
  - Classifying occurrences of actual faults in real world systems.
  - Providing a keyword index for: subjects of publications, research interest areas of projects, institutions or people, and support of resilient mechanisms.
- The representation of multiple alternative criteria (views) to classify the abstractions of a certain domain concept, such as **`Fault`**, motivated the development of the View Inheritance ODP.

Structure: Elements and Relationships

- **TargetDomainConcept** (Figure 2): This class represents the ontology domain concept being defined for which multiple alternative abstraction criteria exist.
  - Figure 3: Fault.
- **Criterion1, Criterion2, …, Criterion_i** (Figure 2): These classes represent each one of the alternative abstraction criteria of the TargetDomainConcept. The list of classes may not be exhaustive or pairwise disjoint.
  - Figure 3: BasicViewPointFault, MajorGroupFault, NamedClassFault, NamedCombinedFault.
- **C1_Class1, …, C2_Class1, …, Ci_Class_x** (Figure 2): These classes refine each abstraction criteria class. The list of classes may not be exhaustive or pairwise disjoint.
  - Figure 3: Subclasses of BasicViewPointFault, MajorGroupFault, NamedClassFault, NamedCombinedFault.
- **C1_Class1Class2** or any **Ci_Class_xClass_y** (Figure 2): These classes participate in multiple inheritance relationships combining different refinements from ***the same*** abstraction criteria class.
- **C1Class3_C2Class2,** or any **CiClass_x_CjClass_y** (Figure 2): These classes participate in multiple inheritance relationships combining different refinements from ***different*** alternative abstraction criteria classes.
  - Figure 3: FaultType1, FaultType2, …, FaultType32

Inter- and Intra-criterion Multiple Inheritance

- **Inter-criterion:** when the parent classes involved in the multiple inheritance relation are subclasses of different abstraction criteria. The class **C1Class3_C2Class2** in Figure 2 is an example of this type of inheritance because one of its parent classes, C1Class3, is a refining concept of Criterion1 and the other parent class, C2Class2, is a refining concept of Criterion2.
- **Intra-criterion:** when the parent classes involved in the multiple inheritance relation are subclasses of the same abstraction criterion. The class **C1_Class1Class2** is an example of this type of inheritance because all of its parents classes, C1Class1 and C1Class2, are refining concepts of the same criterion, Criterion1.
- **Intra- and inter-criterion:** when there are at least two parents involved in the relation that are subclasses of the same abstraction criterion and there is at least one more different parent that is a subclass of a different abstraction criterion. An example of this type of inheritance is trivial to extrapolate from the composition of the previous two.

Conclusions

- A survey of the current ontology building techniques was carried out. The **Normalization ODP** seemed a viable option, yet the pattern did not fully address the definition of **`Fault`** and the application requirements of the **ReSIST project**.
- To bridge this gap, the **View Inheritance ODP** is put forward as an extension to the Normalization ODP, combining the latter with the notion of View Inheritance originated in the O-O software design.
- View Inheritance revealed two basic types of likely relations that could take place in the structure of the pattern: **Inter- or Intra-criterion Multiple Inheritance**.
- These contributions, while not solving all the modelling challenges of the ontology module for ReSIST, do provide additional awareness to be considered in the development process.

Figure 1

Definition of Fault used in the ontology for ReSIST

Avizienis et al. (2005). Basic Concepts and Taxonomy of Dependable and Secure Computing.

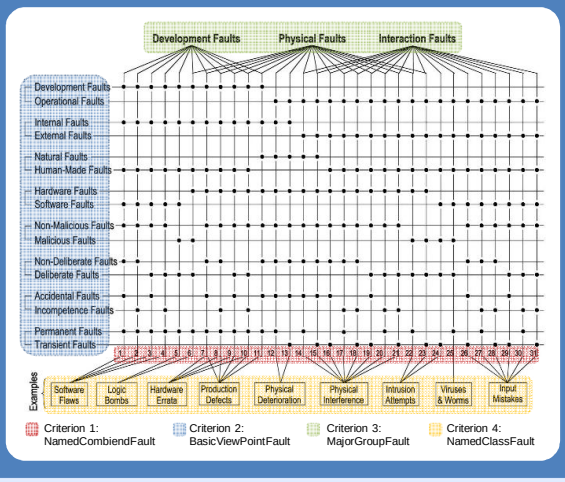


Figure 2

Structure of a generic use case of the View Inheritance ODP

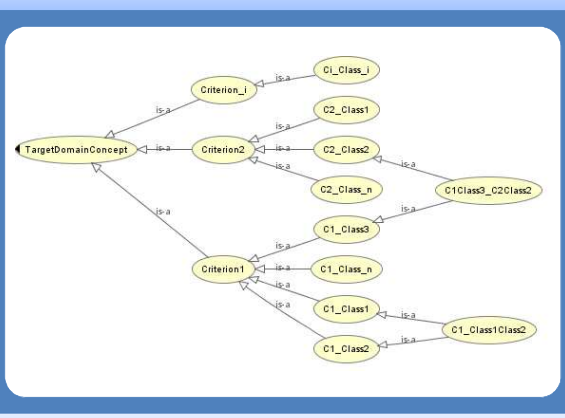
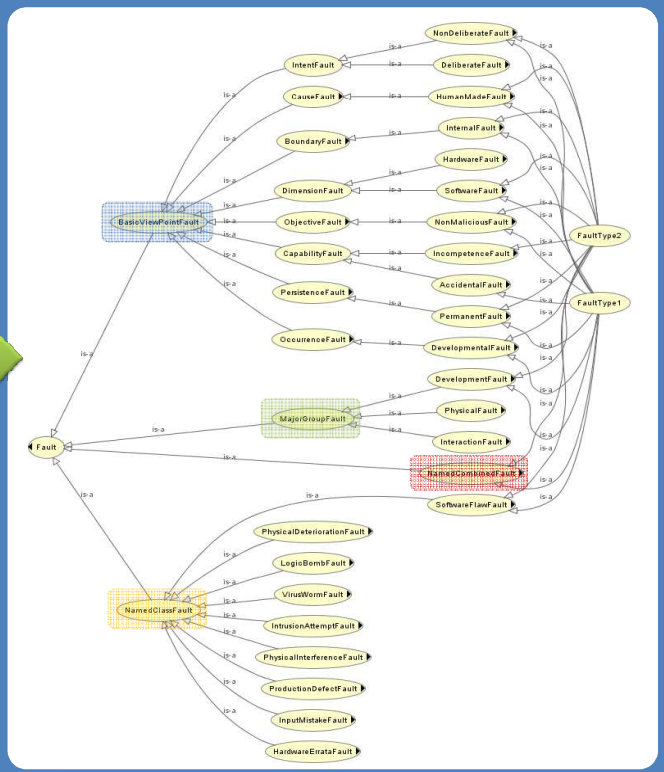


Figure 3

Structure of the View Inheritance ODP for the representation of Fault

For simplicity, only 2 types of faults are shown out of the 31 types defined



This work is supported under the ReSIST Network of Excellence (NoE) which is sponsored by the Information Society Technology (IST) priority of the EU Sixth Framework programme (FP6) under contract number IST-4-026764-NOE