Applying Meyer's Taxonomy to Object-Oriented Software Systems

Michael English, Jim Buckley and Tony Cahill

University of Limerick, Ireland

Overview

- **Use of Inheritance in existing OO Software Systems**
  - **Meyer's Taxonomy**
  - **Overview of Software System**
  - **Application of Meyer's Taxonomy to this existing system**
  - **Results and Conclusions**

Meyer's Taxonomy

- **Comprehensive Categorisation of Inheritance Relationships**
- **Model/Software/Variation are three main categories**
- **The abstractness/concreteness of parent/child classes facilitate classification by narrowing down the classification space**
- **Examples: (1) Model Inheritance: Subtype Inheritance**

  $shape \rightarrow rectangle$

  (2) Software Inheritance: Implementation Inheritance

  $array \rightarrow stack\_array$

Meyer's Taxonomy

- **Categories of Meyer's Taxonomy**

Model	Software	Variation
Subtype	Reification	Functional and
View	Structure	Type Variation
Extension	Implementation	Uneffecting
Restriction	Facility	

Table 1: Meyer's Taxonomy of Inheritance

Why use Meyer's Taxonomy?

- **Study the use of Inheritance in OO**
- **A basis for system description**
- **Assess/refine/evaluate Meyer's Taxonomy in practice**

Software System

- **A Library of Efficient Data Types and Algorithms (LEDA)**
- **Open Source - source code easily accessible**

Characteristic	Number
Classes	596
Inheritance Relationships	315
Multiple Inheritance Relationships	18
Min. Inheritance Depth	0
Max. Inheritance Depth	4
Inheritance Trees	52
Abstract Classes	15
Stand-alone classes	255
Friend	961
Friend (classes)	218
Friend (functions)	750
Friend (operator functions)	358

Meyer's Taxonomy & LEDA

- **Decision Trees**
- **4 Combinations of concrete/abstract classes**
- **The decisions for concrete parent and concrete child classes.**

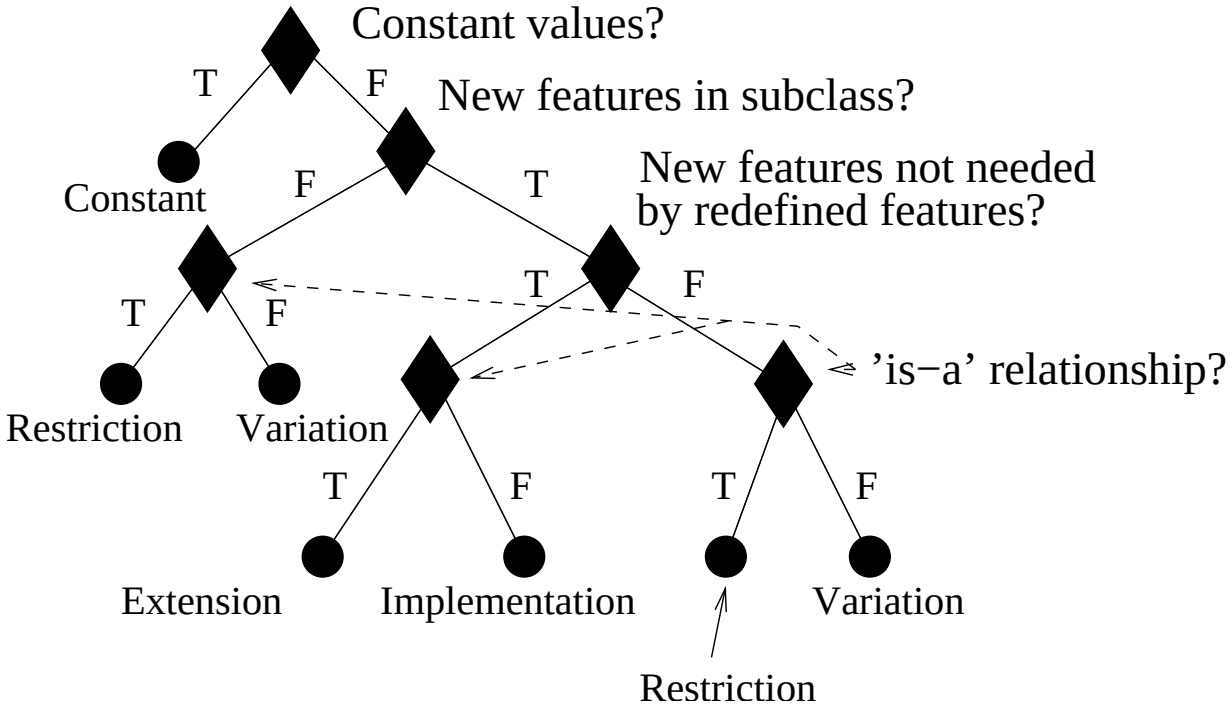

Figure 1: The decision tree when both the parent and child classes are concrete

Results

- **155/315 inheritance relationships examined**
- **140/155 relationships are where both parent and child classes are concrete**
- **78/140 belong to bridge design pattern e.g.** $handle\_base \rightarrow triangle$ **and** $handle\_rep \rightarrow triangle\_rep$
- **Extension, Restriction and Subtype have been clearly identified**
  - **Extension: e.g.** $graph \rightarrow GRAPH < nodeType, edgeType >$
  - **Restriction: e.g.** $graph \rightarrow ugraph(undirectedgraph)$
  - **Subtype:** $Base\_Receiver\_Event0 \rightarrow Receiver\_Event0$
- **Extension/Implementation and Restriction/Variation are prevalent**

Problems and Related Issues

- Determining 'is-a' relationships / Semantic Analysis
- Multiple Inheritance
- Friend Relationships
  - Previous Work
  - Large no. of stand-alone classes
  - Friends which facilitate operator overloading
- Valid uses of Inheritance