Extending G-Nets to Support Inheritance Modeling in Concurrent Object-Oriented Design

Haiping Xu and Sol M. Shatz
Concurrent Software System Lab
Electrical Engineering and Computer Science Department
The University of Illinois at Chicago
{hxu1,shatz}@eecs.uic.edu
Outline

- Why formal methods?
- Extending G-Nets to support class modeling.
- Extending G-Nets to support inheritance modeling.
- Analyzing inheritance anomaly problem
- Our current work: modeling agent-oriented design.
- Concluding comments and future work.
Why Formal Methods?

• To write formal requirement specification, which serve as a contract between the user and the designer.
• To be used in software design. Design errors may be caught in an early design stage.
• To support system verification.
  – model checking
  – theorem proving
G-Nets: A High Level Petri Net

• Defined to support modeling of systems as a set of independent and loosely-coupled modules.
• Provide support for incremental design and successive modification.
• Are not fully object-oriented due to a lack of support for inheritance.
An Example

Figure 1. G-Net Models of Buyer and Seller Objects
Extending G-Nets to Support Class Modeling

• Motivation: to support inheritance.
• Interpret a G-Net as a model of class.
• Instantiate a G-Net $G$:
  – generates a unique object identifier $G.Oid$
  – initializes the state variables defined in $G$
  – *ISP* method invocation becomes 2-tuple $(G’.Oid, mtd)$
Different Forms of Inheritance

- **Augment Inheritance**: new protocols are added to a subclass model.
- **Restrictive Inheritance**: some superclass methods are absent from the protocol of the subclass.
- **Refinement Inheritance**: the subclass contains a method that includes the behavior of its superclass, but extends it in some way.
Extending G-Net to Support Inheritance

• *Default Place*: a default entry place defined in the internal structure of a subclass model.
• The default place is marked only if the method is not defined in the subclass model.
• *Superclass Switch Place (SSP)*: is used to forward a method call to a subobject of the object itself.
A G-Net Model of Unbounded Buffer UB

Figure 2 G-Net Model of Unbounded Buffer UB
A G-Net Model of Bounded Buffer BB

Figure 3 G-Net Model of Bounded Buffer BB
Analyzing Inheritance Anomaly Problem

• Inheritance anomaly refers to the phenomenon that synchronization code can not be effectively inherited without non-trivial re-definition of some inherited methods.

• The inheritance anomaly problem has usually been approached in terms of analyzing the causes, such as partitioning of acceptable states, history-only sensitiveness of acceptance states etc.

• We analyze the inheritance anomaly problem based on clarifying the terminology of “synchronization constraints”.
Synchronization constraints among methods can be specified explicitly or implicitly.

An explicit synchronization constraint refers to the concurrent/mutual-exclusive execution between two methods in an object.

An implicit synchronization constraint refers to cases where acceptance of a method in an object is based on that object’s state.

In either case, the inheritance anomaly problem may be attacked by using refinement inheritance.
A G-Net Model of Bounded Buffer BB1

Figure 4 G-Net Model of Bounded Buffer BB1
Our Current Work: Agent-Oriented Design

• A multi-agent system (MAS) is a concurrent system with autonomous, reactive, internally-motivated agents in a decentralized environment.
• We extend G-Net to support agent modeling based on the BDI agent model.
• To progress from an agent-based design model to an agent-oriented model, we also introduce new mechanisms to support inheritance modeling.
A Framework of Agent-based Model

Figure 5 A Generic Agent-based G-Net Model

Notes: G'.aid = mTk.n.body.msg.receiver as defined later in this section
A Template of Planner Module

Figure 6 A Template of Planner Module
Concluding Comments

• There is an increasing need to ensure that complex software systems being developed are robust, reliable and fit for purpose.

• Petri nets are an excellent formalism for formal specification because they tend to provide a visual, and thus easy to understand, model.

• Extending G-Nets to support inheritance in object-oriented design and agent-oriented design provides an effective way for modeling complex software systems.
Future Work

• Transform the object model and agent model into colored Petri nets, and verify our net models using existing Petri net tools, such as Design/CPN.
• Incrementally design our distributed object system or multi-agent system, and capture early design errors.
• Implement tools to help designer to write formal design specification with our formalism, and automatically verify the behavior properties of the system.