CIS 525 Software Development of Parallel and Distributed systems

RAPID PROTOTYPING

- **Rapid Prototyping tool**
- **Behavioral validation**
- **Environmental integration**

1. What it does?

- To express specification
- To satisfy corresponding properties
- To generate programs adapted to target architectures

2. Issues of automatization of Rapid Prototyping:

- How to get parallel activities from specification?
- How to allow interfaces between the created prototype and its environment?
- How to optimize generated code?

3. Formal approach allows major advances in:

- Reliability
- Robustness
- Modification
- Reusability

4. Verification of systems before installation:

- Large cost reduction in development and maintenance
- Petri nets allow validation of behavioral properties (locks, etc.)

- Petri nets allow abstractions of subsets of the outside environment the prototype has to deal with

Main ideas:

- management of external environment interfaces
- description of the complete process

Specification:

- modeling of the system behavior
- modeling of the prototype environment behavior (needed to validate the system behavior in its environment) – calls of interface primitives are required

EXAMPLE: Data processing of characters between 3 processes

- Petri net model has to be compilable (i.e. it is well structured and respects some properties)
- Compilable model may be decomposed into sets of objects
- The decomposition is performed before code generation

Types of software objects involved

1. Process – subsets satisfying invariant properties and characterize a possible concurrent program unit

2. state_process – is a place belonging to a process (simple, alternative, terminate)

3. Resource – private, shared

4. Action (from transitions) can be:

- simple (one process)
- synchronized (>=2)
- guarded (resources)

Definition: A prototyping process is a mapping from a Petri net to a prototype.

Steps of Prototyping Process

1. **Identification:** (neither language nor architecture dependent)

- Identifies and checks the use of external components
- Behavioral validation of external component must happen before validation

2. **Analysis:** (neither language nor architecture dependent)

- Decomposing the model into sets of objects (using Petri net invariants called also semi-flows)
- Several decompositions are possible and all of them can be explored

3. **Location Step:** (architecture dependent)

- Distributes instances of the process decomposition upon the target architecture (if several processors) – external component location constraints have to be taken into account

4. **Code Generation:** (target language dependent)

- Prototype manages control of all actions described under specification

- External component must be program in the target language; the code must be linked to prototype
- Code is generated using object decomposition from analysis
- Prototype code is optimized using:

  - attributes associated to each object (simple, alternative, terminate)
  - relation between objects.

Prototyping Tool

1. **Identification Step:** interaction of external component with a system
2. **Analysis Step:**

   - **phase 1**: semi-flow computation
   - **phase 2:** compute all possible process decompositions using semiflows

3. **Processes:**

Process#1: ProdReady +ProdWait + ProdInit + ProdEnd

Process#2: Cust1_1 + Cust1_2 + Cust1_End

Process#3: Cust2_1 + Cust2_2 + Cust2_End

(tool may ask human for a choice of decomposition; processes have to be named by system designer)

- **phase 3:** all objects and attribute objects are deduced from the model and its process decomposition

4. Location and Generation:

- Each process is implemented by a task type (Prod, Cust1, Cust2 – are deduced from specification)
- Other tasks manage synchronized Actions also Resources (Resource Manager) and control (Application Manager) – management tasks