

# Process Modeling and Programming

- Started as conceptual schema for project management (e.g. Gantt charts)
- Process programming languages
- Not yet had big impact
- (But configuration management systems are becoming the *de facto* process police)

# Software Life Cycle Model

- Descriptive/prescriptive characterization of software development process
- Macro level; break down into major stages
- Purposes
  - Management guidelines, documentation outline, methodology guidance, measurement framework, basis for empirical studies

# Software Process Model

- More detailed than life cycle model
- Networked sequence of activities, objects, events, and transformations
  - Task chain
  - (Work flow)

# Traditional Life Cycle Models

- Waterfall
- Stepwise refinement (programming)
- Incremental development/release
  - Possibly with prototyping
- Industrial/military standards
- Capability models (CMM)

# Non-Traditional Models

- Rapid prototyping
  - Throw-away, mock-up, demos, quick-and-dirty, evolutionary
- Joint Application Development (JAD)
  - Involve users (3-6 month iterations)
- Reuse
  - Acquisition, analysis, indexing, composition, library management, (reward system)

# Non-Traditional - 2

- Application generation
  - Domain-specific languages, product lines
- Software documentation support environments
- Rapid iteration, incremental evolution, evolutionary delivery
- Program evolution models
  - Belady and Lehman
  - Continuing change, increasing complexity, statistically self-regulating, invariant work rate, incremental growth rate

# Production Process Models

- Spiral (Boehm)
  - Inner: analysis and prototyping
  - Outer: classical with built-in risk analysis; equal angular displacement
- Operational models
  - Formal models with operational interpretation
- Software automation and programming
  - Backward and forward chaining, behavior, object type structure, process dynamism, constraints, goals, policies, user interactions, plans, environments, knowledge-based

# Process Programming Languages

- Knowledge-based
- Rule-based
- Petri Nets
- Client-server

# Goals / Uses

- Communication
- Estimation and planning
- Management and replanning
- Measuring
- Configuration
- Reuse
- Enaction
- Verifying

# Features

- Abstraction, non-determinism, modularity, genericity
- Concurrency, synchronization, external events, scheduling, logical precedence, deadlines
- Persistence, versioning, long transactions
- Incompleteness, ambiguity, mixed agent (machine and human), tool invocation, measurement

# Open Issues

- Representing and interacting with humans
- Failure management with unforeseen events
- Dynamic process modification
- Integrated support for modeling and management
- Integrated support for process and data modeling

# Example Language: JIL

- Process is a *composition* of steps
- Step is a set of elements
- Element
  - Object, declaration (parameters and local data), resource (people, software, hardware), substep, activity (imperative definition of step actions), reaction to triggered event (artifact update, process control, exception), reaction (substep invocation), step execution constraint, precondition, postcondition, consistency condition (on product, process, or resource state), exception handler

# Step Composition Operators

- ORDERED
- UNORDERED
- PARALLEL
- TRY (ordered until one succeeds)

# Other Features

- REACT for exceptions
- Resource classification hierarchies
- External execution agents (humans or machines)

# Little-JIL Visual PPL

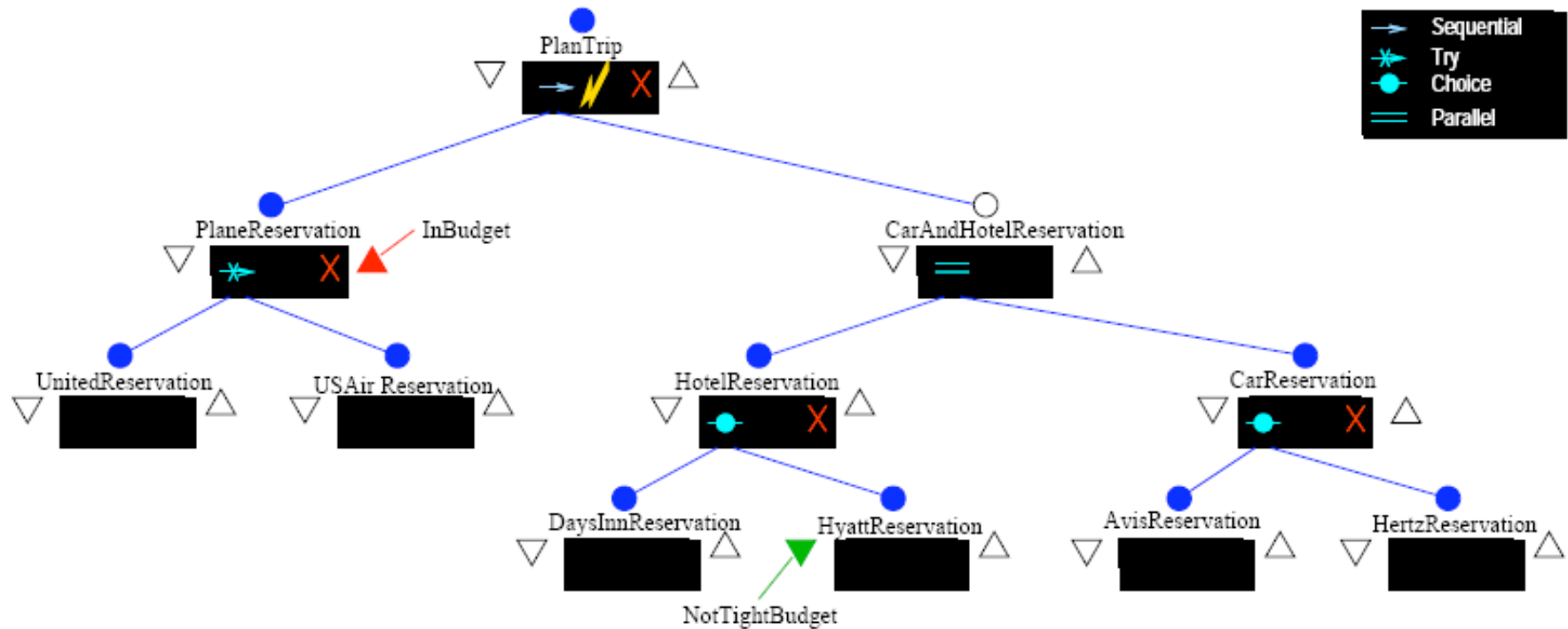


Figure 2. Reservation process showing proactive control: step kinds, requisites.

# Future Directions for Modeling

- Process simulation
- Web-based models and environments
- Process re-engineering
- Global, distributed, open-source, extreme programming, virtual