Building Maintainable Scheduling & Resource Allocation Engines: LILCO Experience

Jacob Feldman, IntelEngine, CTO
Phone: (732) 287-1531
E-Mail: feldman@ilog.com

INFORMS Montreal 1998 Meeting

Workforce Scheduling

- **IntelEngine**, Inc. uses a constraint-based OO-technology (**ILOG**) to build customized intelligent engines for workforce scheduling systems
- Two major workforce scheduling projects:
  - Public Utility Scheduling Engines (**LILCO**)
  - Resource Allocation Engine for Strategic Compliance Planning System (**IRS**)

Workforce Scheduling Engines

- A workforce scheduling system usually allocates workforce to workloads satisfying real-world constraints and optimization objectives
- At the heart of such systems are specialized constraint-based engines that should be:
  - maintainable
  - extendable
  - customizable

Utility Scheduler™

- Provides required level of service and skills while minimizing workforce expenses
- Schedules jobs based on their priorities, resource availability and different optimization goals
- Allocates human, equipment and other resources to jobs satisfying user-defined constraints and preferences

LILCO Corporate Resource Management System

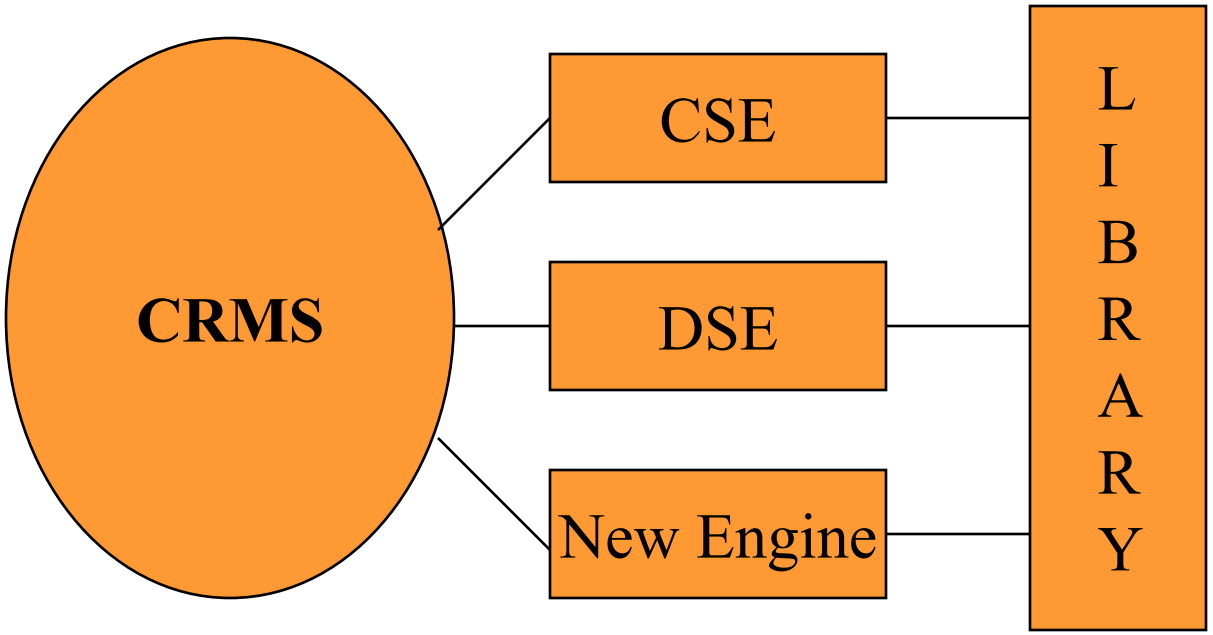
- More than 1 million customers in Long Island, NY
- More than 5000 employees
- Service territory 1,230 square miles
- Hundreds jobs per day
- Job requires a mix of people skills, vehicles and equipment

Multi-objective Work Planning and Scheduling

- Travel time minimization
- Resource load levelization
- Skill utilization (use the least costly skills/equipment)
- Schedule jobs ASAP
- Honor user-defined preferences

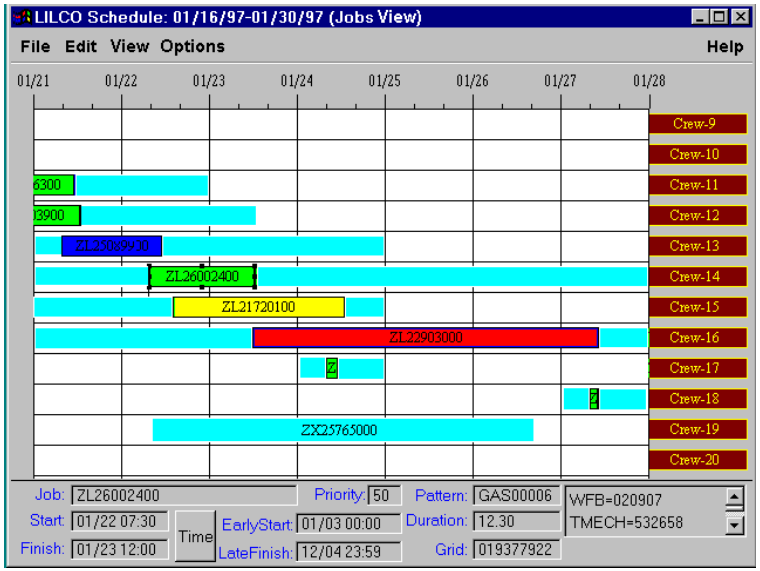
Family of Scheduling Engines

- Construction Scheduling Engine (CSE)
- Designer Scheduling Engine (DSE)

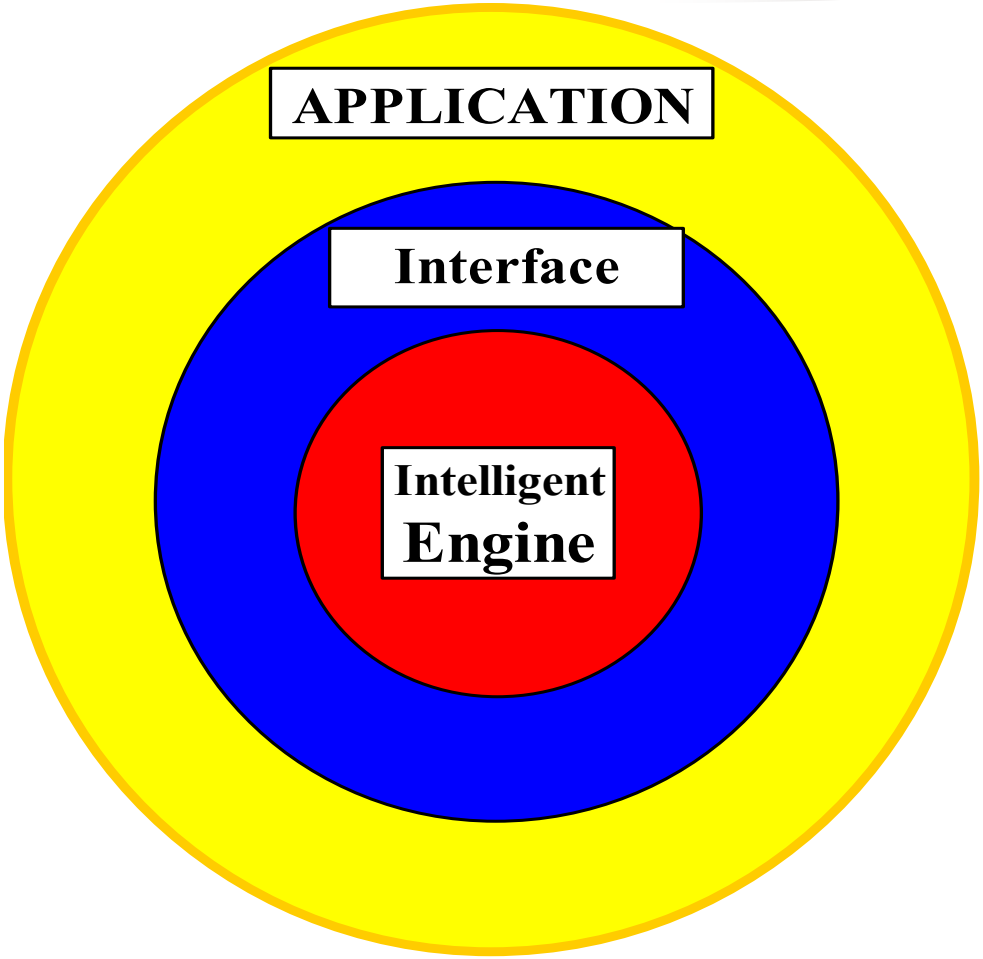


Library of C++ classes for Public Utility Scheduling

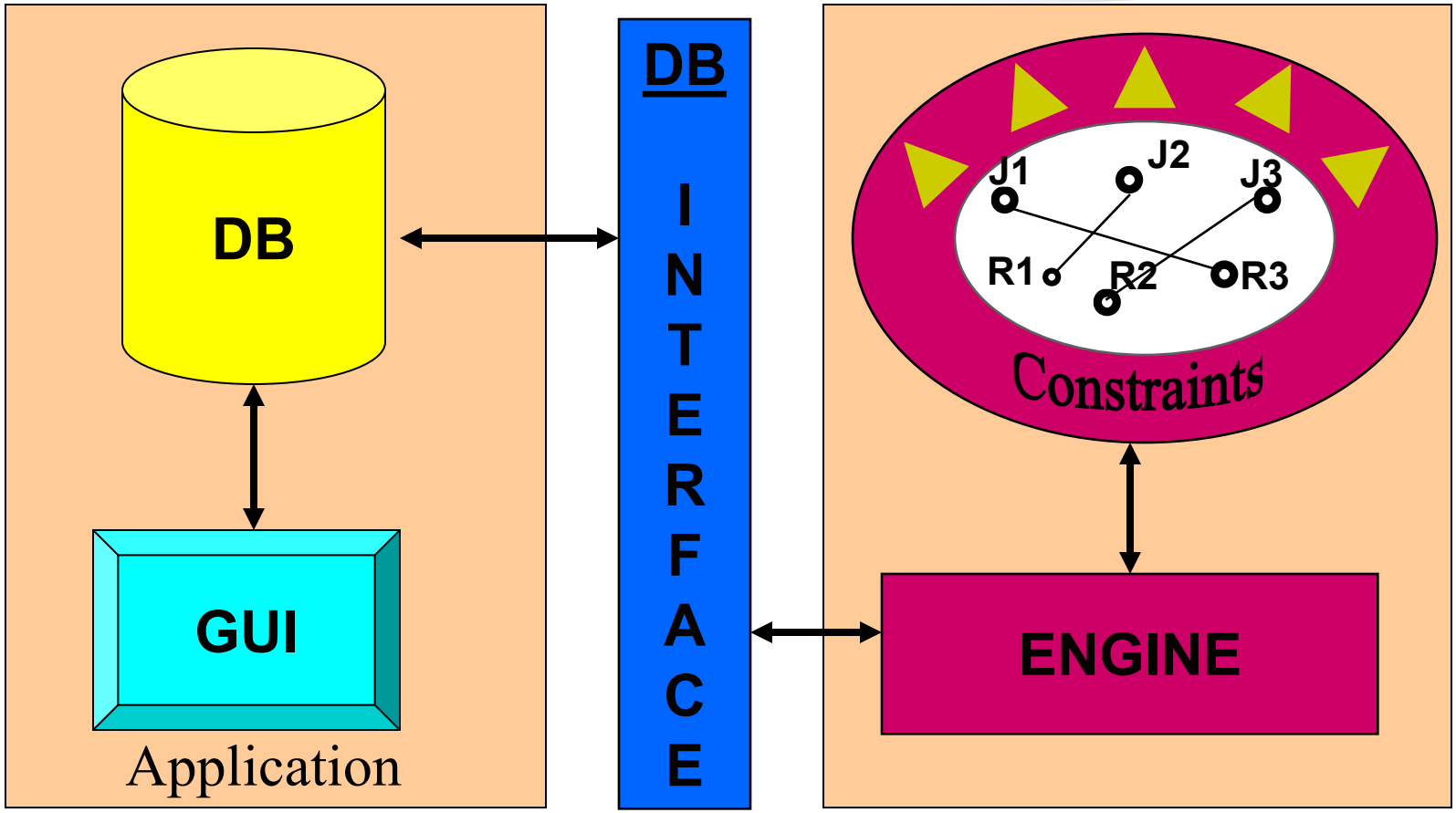
- Utility Scheduler™ as a Library for Public Utility Scheduling Engines
- Architecture Components
- Logical Components
- Graphical Components
- Design Patterns



Application, Interface, Engine



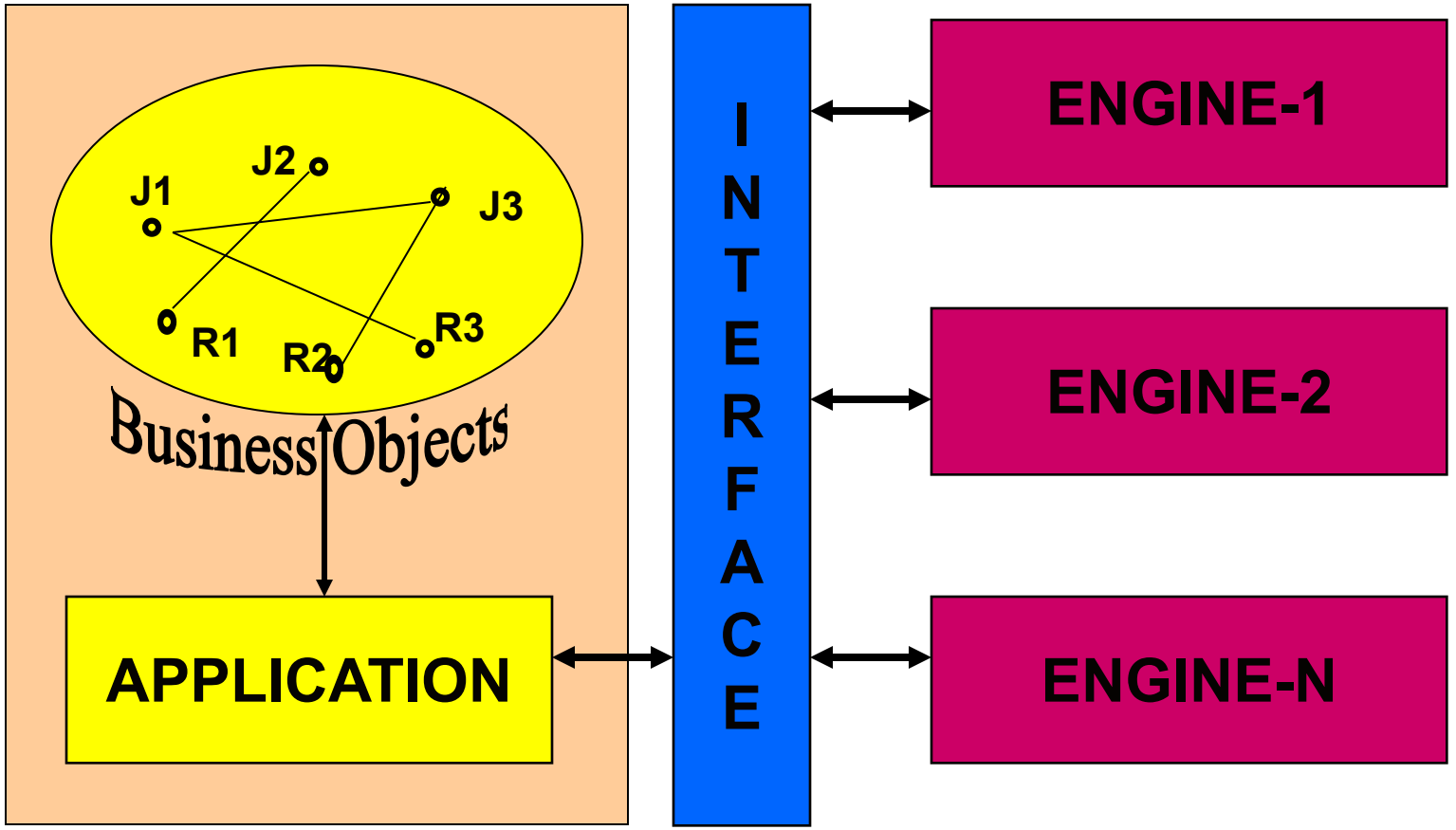
(Anti)Pattern "Batch Engine": Typical Architecture



Batch Constraint Satisfaction: Pros and Cons

- Pros:
  - Simplified development
  - Clear demarcation between the engine's developer and actual customer problems (pros?)
- Cons:
  - Inconsistency (uncontrolled manual overrides)
  - Inefficiency (schedule "all")
  - Redundant Functionality (for GUI and Engine)
  - Difficulties to interpret scheduling results

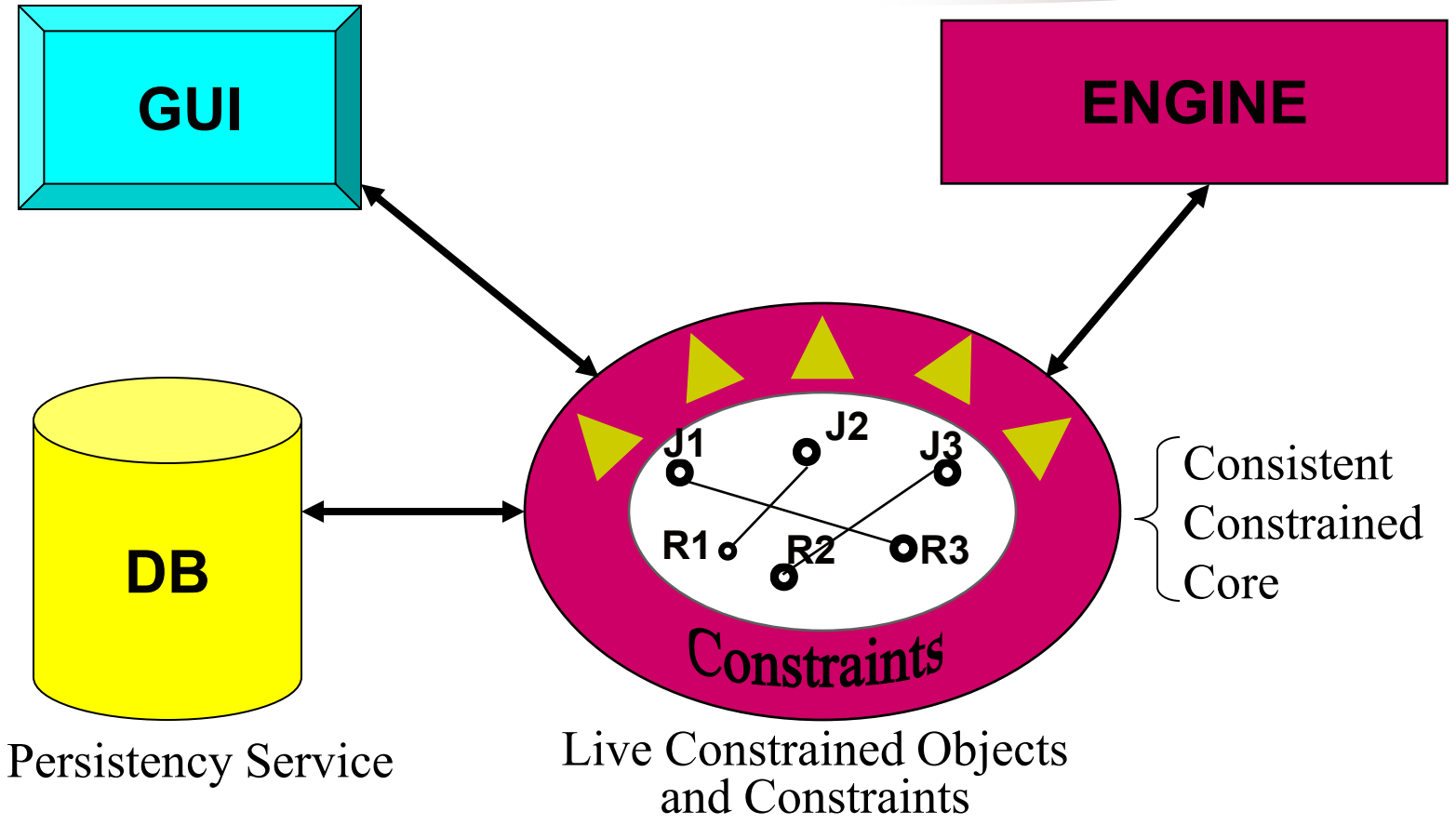
Architecture with Multiple Engines



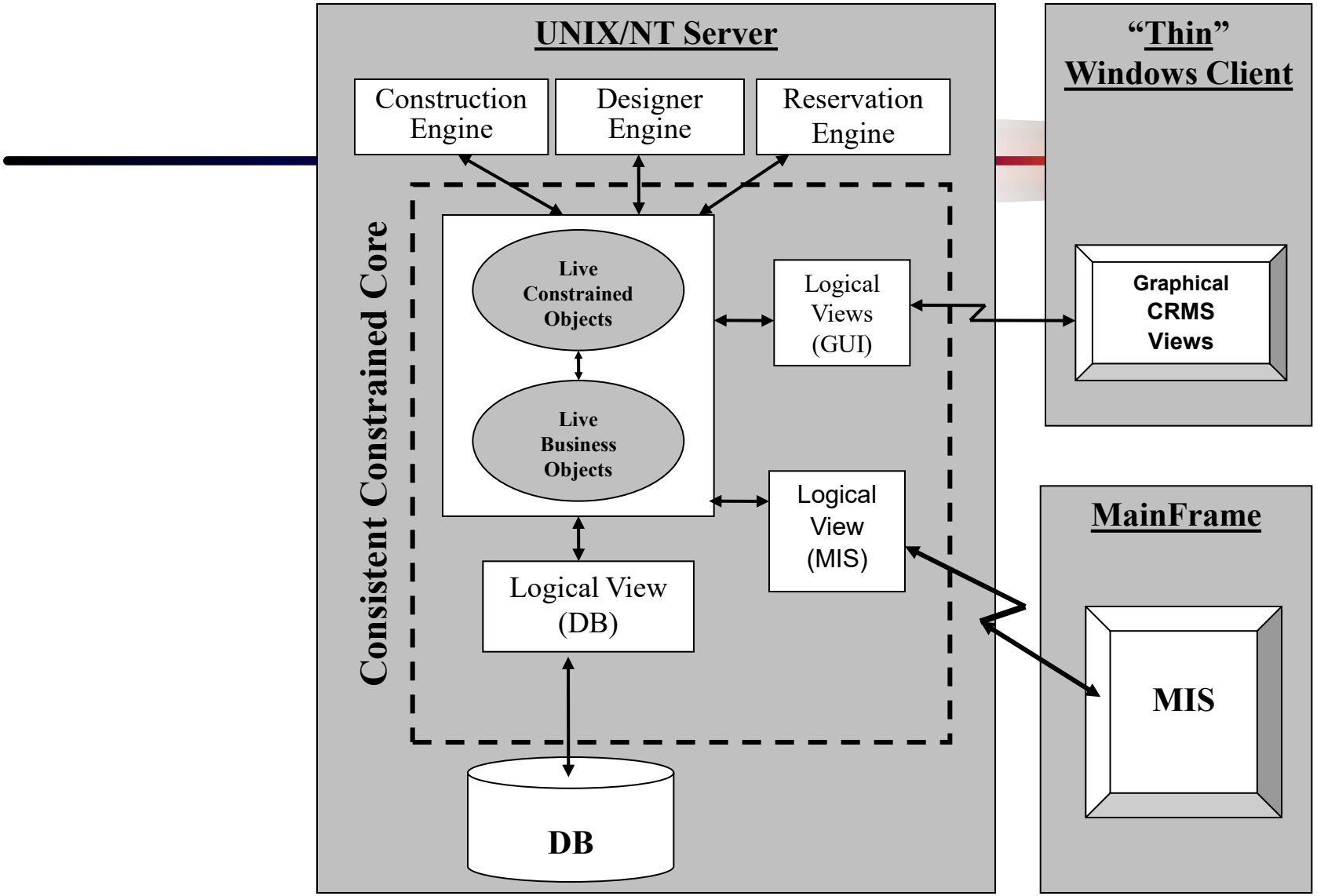
Scheduling Reality means Instant Changes

- When it comes to managing jobs and resources, *change* is the name of the game
- Users want to:
  - make changes quickly and easily
  - update and fine-tune schedule in a flash, whether they're altering jobs' start time/duration or adjusting resources.
- Solution: from "batch" to "interactive"

Pattern "Interactive Engine": Typical Architecture



Interactive Scheduling in LILCO



Interactive Constraint Satisfaction: Pros and Cons

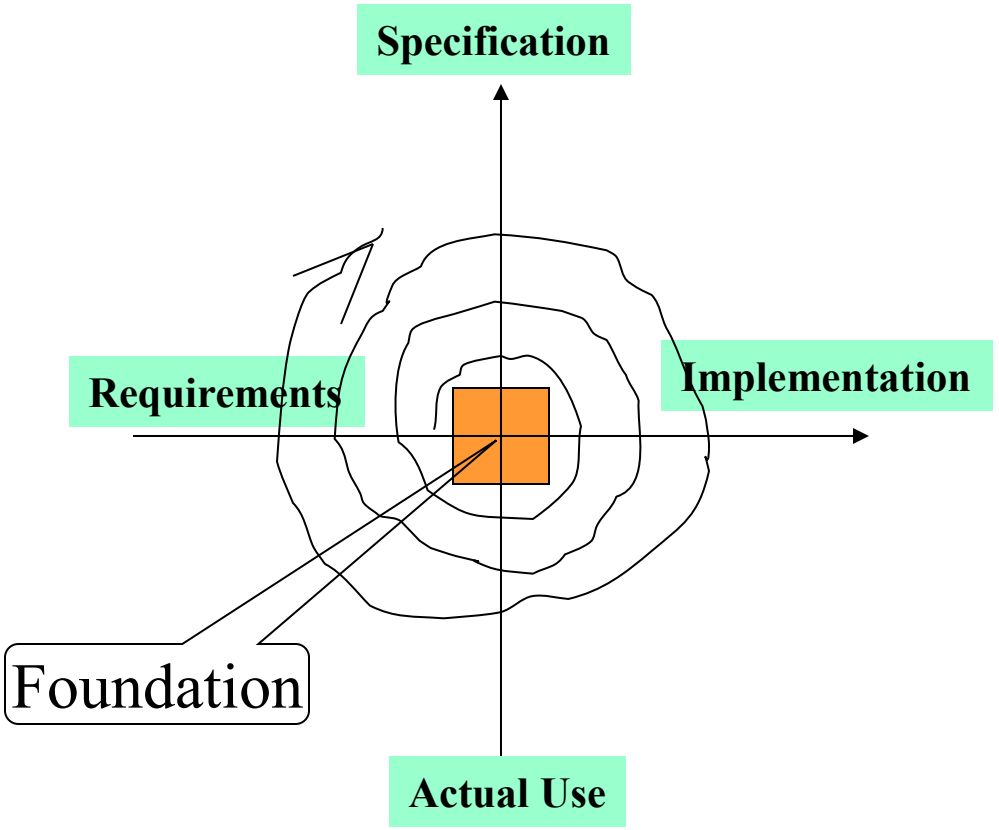
- Pros:
  - Tight integration between GUI and Engines
  - Efficiency
  - What-if analysis support
  - Ability of manual scheduling with controlled constraint propagation
  - Simplified interpretation of scheduling results
- Cons:
  - Complex development

User Involvement

- Development-time involvement

  An end user should be involved during the entire life-cycle of the constraint-based system

- Run-time involvement



Maintainability Patterns

- Several Popular Patterns
  - Pattern “Configurator”
  - Pattern “Strategy”
  - Pattern “Time versus Quality”
  - Pattern “Multiple Objectives”
  - Pattern “What if”

Pattern "Configurator"

- Intent
  - Handling data not specific to the problem but to the way it is going to be solved or optimized
- Also Known As
  - Tweaker, Fine-Tuner
- Motivation
  - Need to take into account user specific, and data specific information. As well, the end-user may help choosing the strategy.

Pattern "Configurator": Applicability

- Iterative solution improvements
- Options of specific constraints
- Alternative search or improvement strategies
- Alternative objective
- Fine tuning
- What-if analysis

Pattern "Configurator"

- Configuration parameters definition via:
  - User profile
  - Environment variables
  - Configuration file (ini-file)
  - Run-time parameters
  - GUI

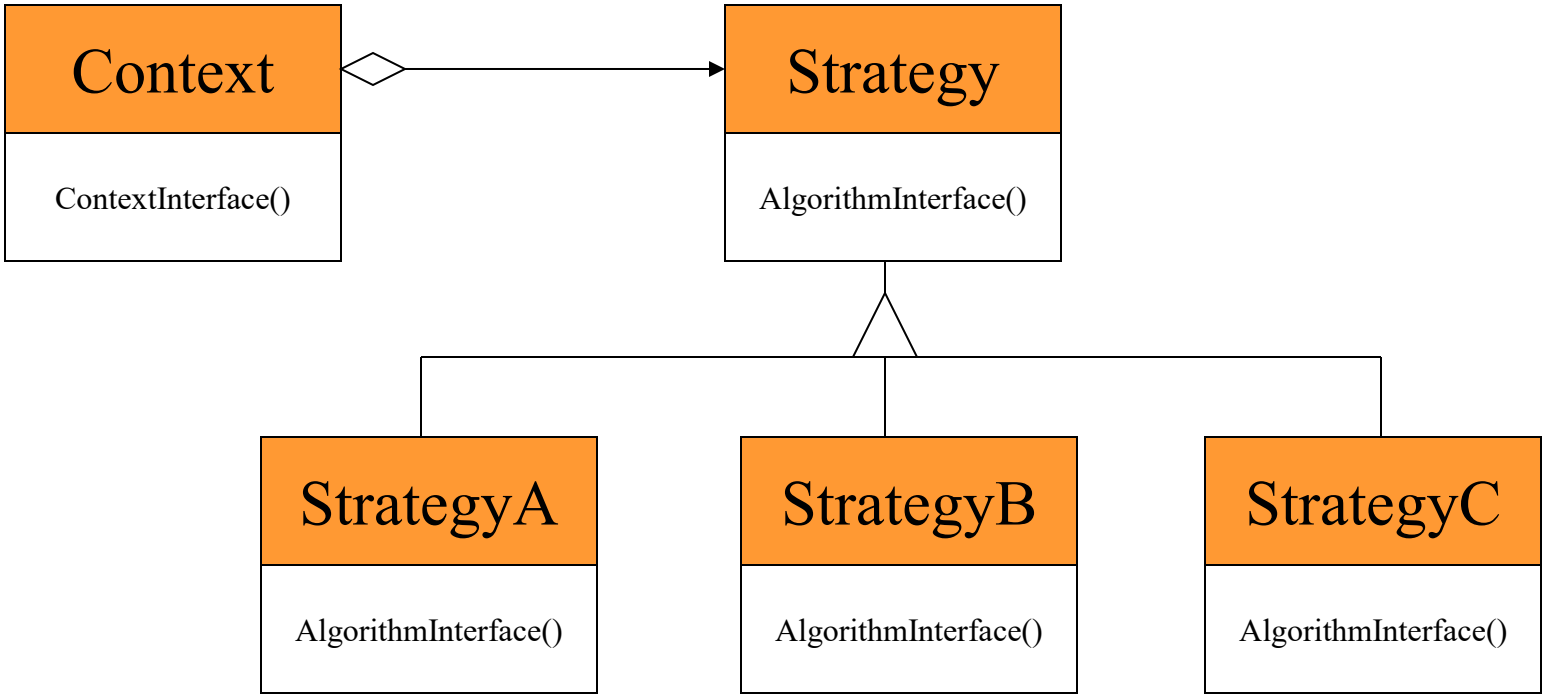
Pattern "Configurator": Sample

```
WEIGHT_OF_LSD=4
WEIGHT_OF_PRIORITY=20
WEIGHT_OF_TRAVEL=75
TRAVEL_MAX=200
WEIGHT_OF_HUMAN_EXCESS=30
WEIGHT_OF_VEHICLE_EQUIP_EXCESS=10
WEIGHT_QUALIFICATION_DISTANCE=10
```

Pattern "Strategy"

- Intent
  - definition of family of algorithms
- Also Known As
  - Policy
- Motivation
  - Common encapsulation tackling different situations
- Confer Gamma's Strategy

Pattern "Strategy": Structure

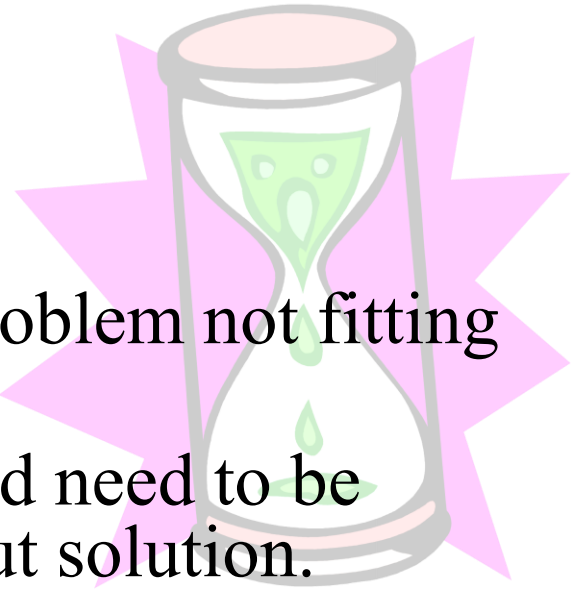


Pattern "Strategy" : Sample

- Two strategies from LILCO engines:
  - Assign Resources to Jobs
    - Select the next most important job
    - Choose the cheapest resource for this job
  - Assign Jobs to Resources
    - For the latest scheduled crew of resources find the most important job this crew can do next

Pattern "Time versus Quality" (1)

- Intent
  - Setting an artificial limit to stop a search
- Also Known As
  - Watchdog
- Motivation
  - Need to be protected against problem not fitting the tried heuristic,
  - Often, a problem is too hard and need to be considered as a problem without solution.



Pattern "Time versus Quality" (2)

- Applicability
  - Iterative improvements
  - Optimization by iterative relaxation
- Implementation
  - Based on the CPU time
  - Based on the backtrack count
  - Based on the "tried" percentage
  - Prorated enumeration
  - User interrupt

Pattern "Multiple Objectives"(1)

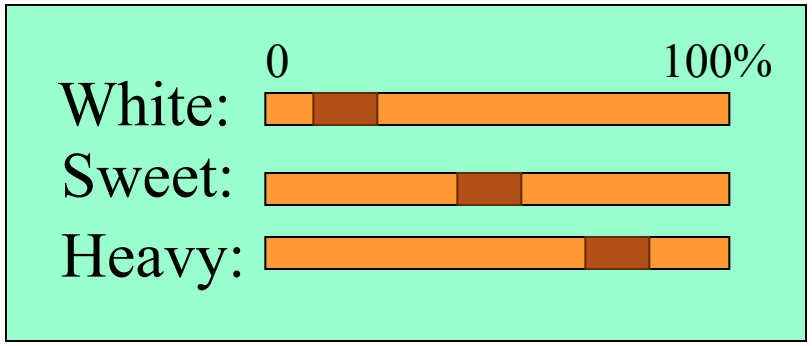
- Intent
  - Giving the respective importance to different optimization objectives
- Also Known As
  - Weighted Cost
- Motivation
  - When your objective is clearly decomposed into comparable/incomparable sub-objective, you want to give the user an opportunity of tuning what his/her real objective is.

Pattern "Multiple Objectives"(2)

- Applicability
  - Aggregation of the quality of the solution
  - Several objectives without clear prioritization
- Sample Implementations
  - GUI Sliders (input)
  - Pie Chart, Bar Chart (output)

Incomparable Objectives

- Applicability
  - How to compare “White” and “Sweet”
- Objective Weights
  - Cost = $\sum_{i}$ w[I] * cost[I]



Pattern "What if"

- Intent
  - Allow the user helping in the solution search
- Also Known As
  - Driver
- Motivation
  - Combine the end-user expertise with the computation power
  - Limit exhaustive search, add "determinism"
  - Take into account preferences not expressed in the model

Pattern "What if"

- Implementation
  - change weights and re-run the Engine
  - set frozen assignments and re-run the Engine
  - request a "different" solution

Over-Constrained Problems

- "...rather than searching for a solution to a problem, we must, in a sense, search for a problem we can solve" (Eugene Freuder)
- Patterns for Partial Constraint Satisfaction could be found in "Over-Constrained Systems", 1996, ISBN 3-540-61479-6
- Check also www.ilog.com: see PACT'98 tutorial "Design Patterns for Constraint Programming"

Conclusion

- Consider a Family of customizable multi-objective intelligent engines
- Be prepared for the migration from Batch to Interactive Scheduling
- Keep users involved during the entire system life-cycle
- Use common design patterns