Changing the rules of business™

June 16, 2005

Integration of Rules and Optimization in Plant PowerOps

Rules and Optimization in PPO

Changing the rules of business™

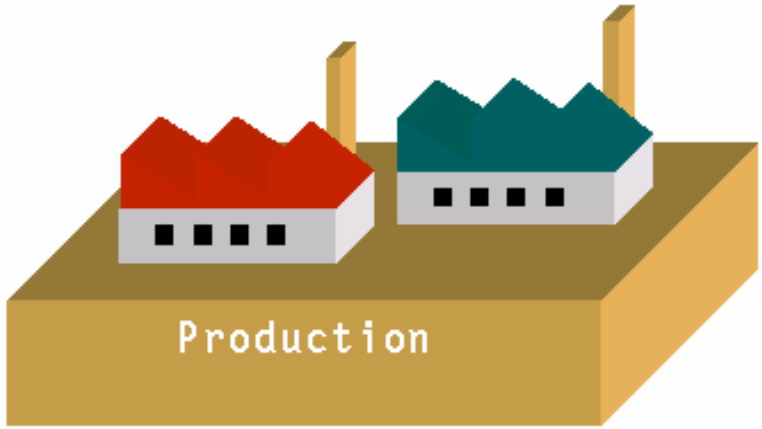
Agenda

- Plant PowerOps overview (1)
- Short demo
- Motivation
  - Between package and custom development
  - Learn from the industrial success of Business Rules
  - The challenge
- Plant PowerOps overview (2)
- Business Rules interface
- Business Rules use cases
  - Rules for Decision Support
  - Rules for Operations
- Business Rule integration
- Discussion and open questions
  - Loose vs. tight integration
  - Rules and search
  - Rules and GUI
  - Developer and End user
- Conclusion

Plant PowerOps overview (1)

Changing the rules of business™

Plant (and Transport) PowerOps



Plant PowerOps
(PPO)

Transport PowerOps
(TPO)

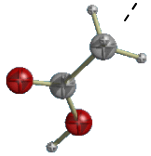
Tomato factory

Changing the rules of business™

Problem description

crusher

cooker



filler

Bottled Tomato Ketchup

Canned Tomato Sauce

Aseptic Bags filled with tomato sauce or ketchup

washer

sterilizer

Motivation

Changing the rules of business™

Between package and custom development

The customer's perspective

- **Two possible ways to cover a software need:**
  - **Buy a package (e.g. SAP APO for Supply Chain Optimization)**
    - Force the requirements into a rigid pre-defined generic solution (time and money)
    - Incompatibilities are found at last stages of the implementation, i.e. too late (risk)
  - **Launch a custom development**
    - Everything can be done: where to stop the development? (time, risk)
    - Only few companies can afford it (money)
- **In both cases evolutions are difficult**
- **Many companies bought complex supply chain packages and ended up using Excel**

Motivation

Between package and custom development

- The “package vs. custom development” dilemma is not proper to optimization.

BUT

- In optimization is particularly important.
- In fact, optimization only makes sense if you optimize the right problem using correct data!

Motivation

Between package and custom development

ILOG's perspective: common problems optimization experts face

- **Without application prototypes, it's**
  - **Hard to get buy in and funding for optimization projects**
  - **Hard to refine the optimization models with business/operations personnel**
- **Users don't understand the reasons for a recommended plan**
  - **It comes out of a black box, making it difficult to trust**
- **Users can't interact with the recommended plan or schedule**
  - **Need to do "what-if" analysis & scenario comparison to be comfortable**
  - **Need sensitivity analysis and explanations**
  - **Need to modify a model-based plan, using it as a starting point**

Motivation

Between package and custom development

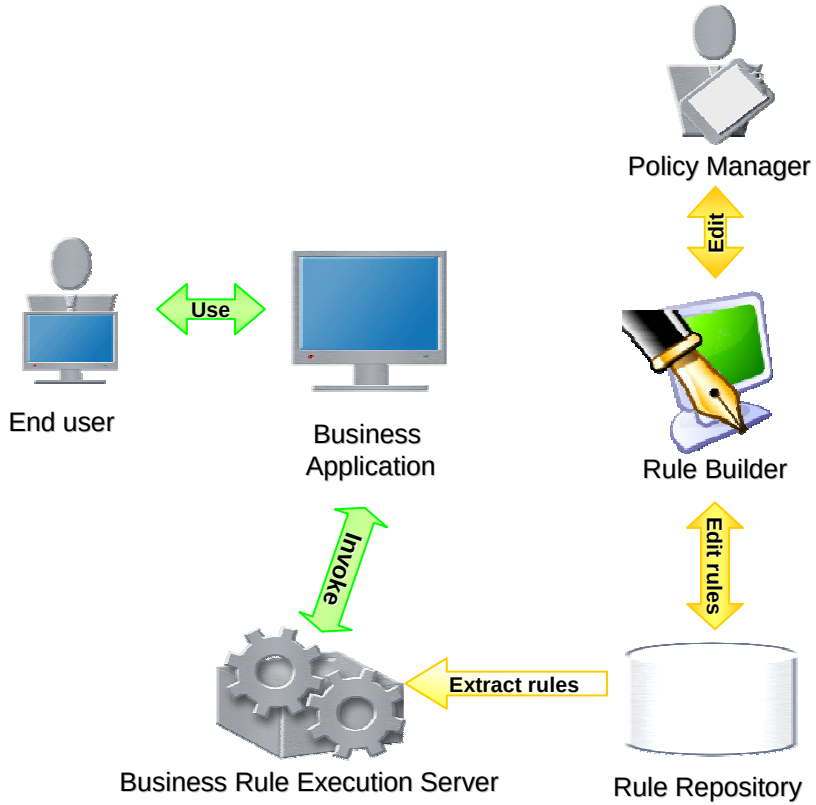
A possible solution

- **Go more “vertical”, yet without building a package:**
  - **Define a class of problem (e.g. manufacturing scheduling)**
  - **Build pre-defined models and search methods**
  - **Build pre-defined connectors to legacy systems**
  - **Allow consultants and end users to adapt it to the specific business**

Motivation

Changing the rules of business™

Learn from the success of Business Rules



Motivation

Learn from the success of Business Rules

- Separation of business logic from application
  - Enables business users to maintain and modify the business logic
- Reduce the development time and the maintenance cost (easy to modify)
- Increase visibility and understanding of business policies

- Shouldn’t all this apply to optimization applications?
- Doesn’t all this answer most of the criticism against optimization packages?

Motivation

The challenge

- **Build a "semi-finished" product such that:**
  - **fit / no-fit assessment is easy (verticalization)**
  - **it can be quickly transformed into a solution by optimization experts and software engineers (rules as scripting language)**
  - **it can be maintained by IT departments (rely on business rules management system)**
  - **it can be configured by business users (business rules language)**

- **We hope to improve the acceptance and usability of optimization applications**
- **We claim that a business rule interface on top of an object model provides the flexibility and adaptability needed to solve the real needs of the customers.**

Plant PowerOps overview (2)

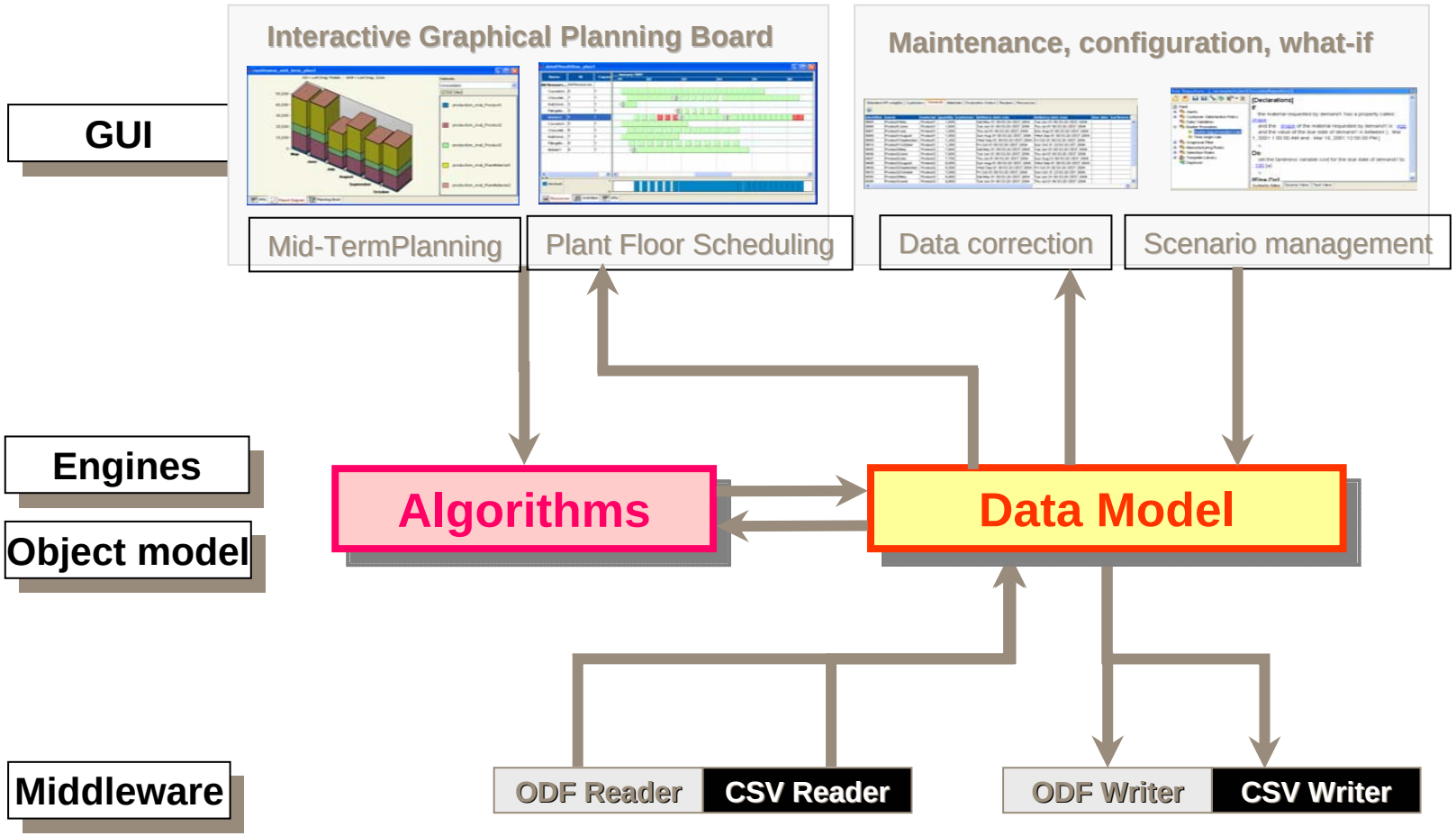
Changing the rules of business™

- Is an Advanced Planning and Scheduling system
  - Specialized for the process industry
- Targeting companies which need
  - optimization
  - agility and flexibility in the production system
- Providing
  - a pre-defined data model
  - effective optimization models and algorithms (based on CPLEX and Scheduler)
  - a graphical planning board
  - a pre-defined connector to the legacy system
  - a rule-based interface for customization and scenario management

Plant PowerOps overview (2)

Changing the rules of business™

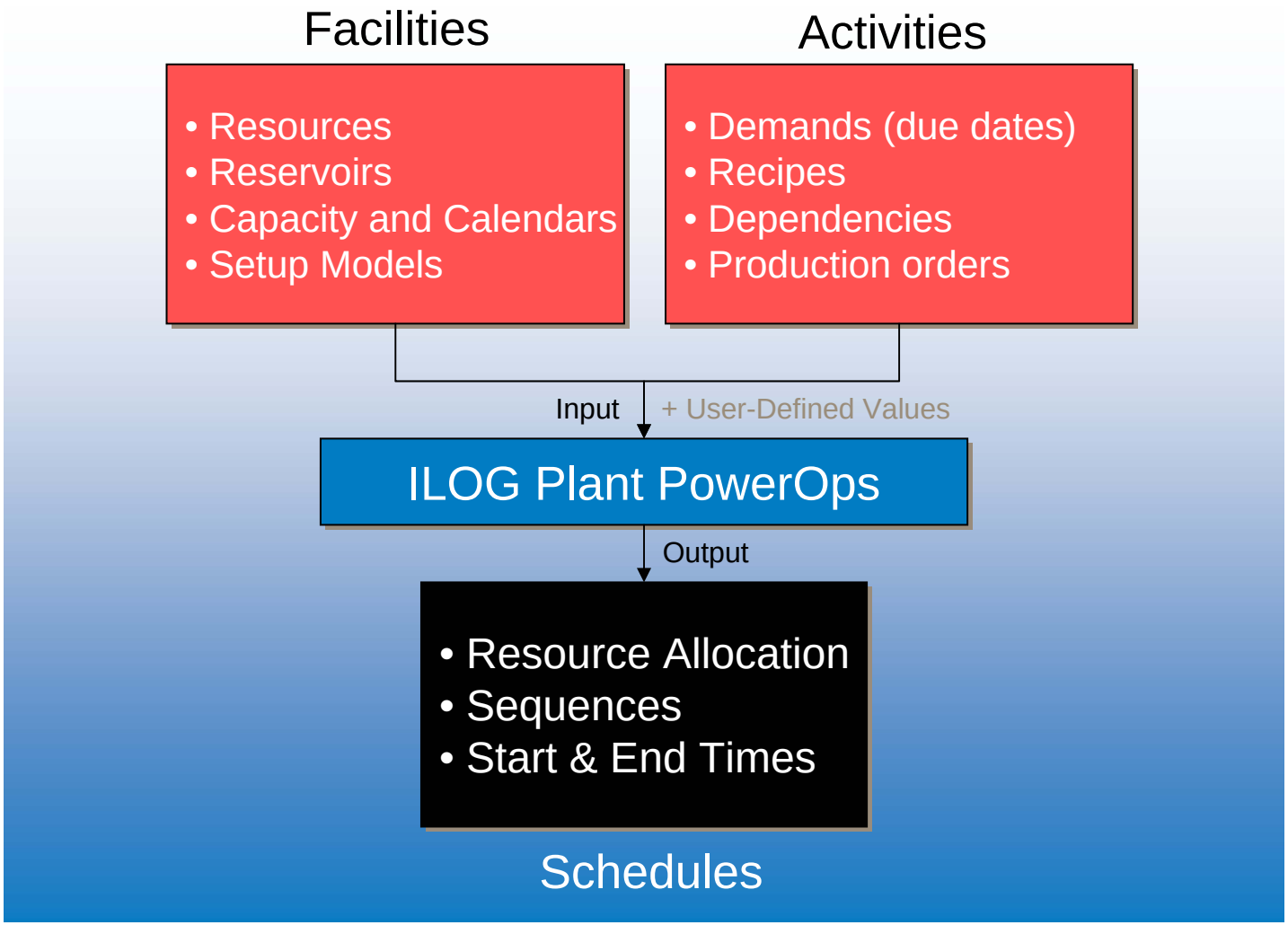
Architecture



Plant PowerOps overview (2)

Changing the rules of business™

Manufacturing scheduling



Business Rules interface

Changing the rules of business™

Rule language

- Production rules executed in forward-chaining (using RETE)
- Technical rule language (Java-like)
- Natural language syntax

When
    conditions
Then
    actions

- *Conditions*
  - methods returning booleans on objects of the PPO data model
- Actions
  - Any method of the PPO data model
  - Insertion in the working memory (new)
  - Retraction from the working memory (delete)

Business Rules interface

Changing the rules of business™

Example

"for every activity that is due on Jan 1st 2005, set its due date on Jan 2nd 2005"

IloMSActivity is a class of the PPO object model

Providing the methods "getDueDate()" and "setDueDate(Date d)"

Technical rule language

When
    conditions
Then
    actions

- ***Conditions***
  - **Evaluate(theActivity.getDueDate() equals "Jan 1, 2005")**
- **Actions**
  - **Modify(theActivity.setDueDate("Jan 2, 2005"))**

Natural language

- ***Conditions***
  - **If the activity is due on "Jan 1, 2005"**
- **Actions**
  - **Set the due date of the activity to "Jan 2, 2005"**

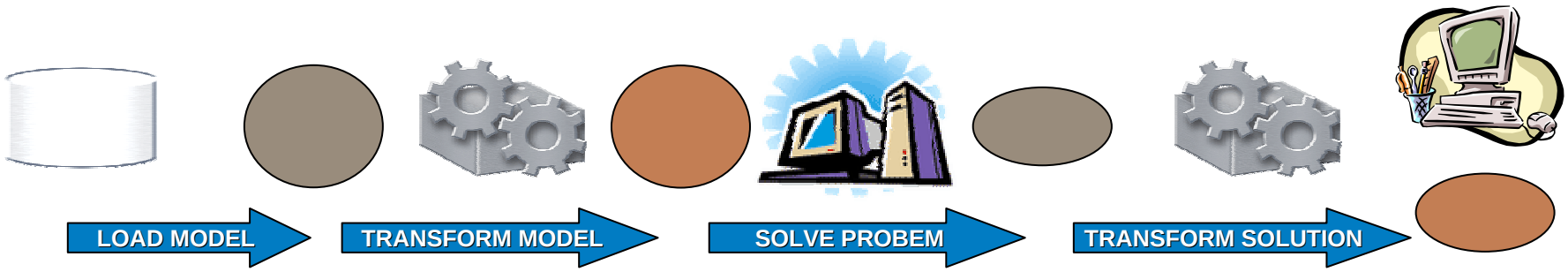
Business Rules interface

Changing the rules of business™

Rules and Object model

- **Rules are applied either in a pre-processing or in a post-processing step**
  - **Pre-processing rules (before optimization)**
    - Transform the model before optimization (create the optimization model)
  - **Post-processing rules (after optimization)**
    - Transform the solution after optimization

- **Rules do not solve the problem. Rules state what the problem is about.**



Business rules use cases

Changing the rules of business™

- **Decision Support: management**
  - What-if analysis
  - Business policies
  - Model preprocessing
  - Tune the engine

- **Operations: planner**
  - Data validation
  - Solution checking
  - Graphical rules

What-if analysis

Changing the rules of business™

- **Perform massive and complex data modifications to simulate an event**
- **Example: expected increase of sales generated by forthcoming agreement with a customer**

Declarations
for *the customer order*, instance of demand
where *this demand* is a customer order
If
the name of the customer for *the customer order* is “Hane”
Then
set the quantity of material requested by *the customer order* to
the quantity of material requested by *the customer order* × 2

Business policies

Changing the rules of business™

- **Define those business policies that enable to best deal with events**
- **Example: in order to obtain a selling agreement, the CEO of the company has promised to never deliver large orders for "Hane" late**

Declarations

for *the customer order*, instance of demand
where *this demand* is a customer order

If

the name of the customer for *the customer order* is "Hane"

and the quantity of material requested by *the customer order*

is greater than 200

Then

set the tardiness variable cost for the due date of
the customer order to 100

Model preprocessing

Changing the rules of business™

- **Incorporate implicit constraints in business rules instead of polluting the legacy system**
- **Example: Product for the same customer order are to be produced on the same production line**

Declarations

for *the demand*, instance of demand,
for *production order A*, instance of production order
where *the demand* is satisfied by *production order A*,
for *production order B*, instance of production order
where *the demand* is satisfied by *production order B*

If

Then

insert in the working memory a new compatibility constraint so that
production order A and production order B
are processed on the same line

Tune the engine

- **The priorities can be driven by the current data**
- **Example: emphasize customer satisfaction**

If
the percentage of gold customer is less than 20
Then
set the total setup cost weight to “high”
and set the total tardiness weight to “low”
Else
set the total setup cost weight to “low”
and set the total tardiness weight to “high”

Data validation

- **Clean the data before entering the optimization**
- **Example: minimal order quantity**

Declarations
for *the demand*, instance of demand
where *this demand* is a customer order
If
the quantity of material requested by *the demand* is less than 3
Then
display the name of *the demand*
and display “request less than 3 product units”

Solution checking

Changing the rules of business™

- **Check the solution against user-defined rules**
  - **This is fully integrated in the solution checker**
- **Example: temporal dispersion of related activities**

Declarations

for *production order A*, instance of production order,
for *activity 1*, instance of activities generated from *production order A*
for *activity 2*, instance of activities generated from *production order A*
for *the solution* is the current scheduling solution

If

the start time of *activity 2* in *the solution* is greater than
the end time of *activity 1* in *the solution* + 15

Then

add in the checker of *the solution* a violation “Dispersed activities”

Graphical rules

Changing the rules of business™

- **Use business rules to filter, colorize or select**
- **Example: select late activities that belong to a gold customer**

Declarations

for *the customer order*, instance of demand
where *this demand* is a customer order,
for *the production order*, instance of production order
where *the customer order* is satisfied by *this production order*,
for *activity 1*, instance of activities generated from *the production order*

If

the category of the customer for *the customer order* is gold
and the tardiness cost of *activity 1* in the current scheduling solution
is greater than 0

Then

add *activity 1* to selection

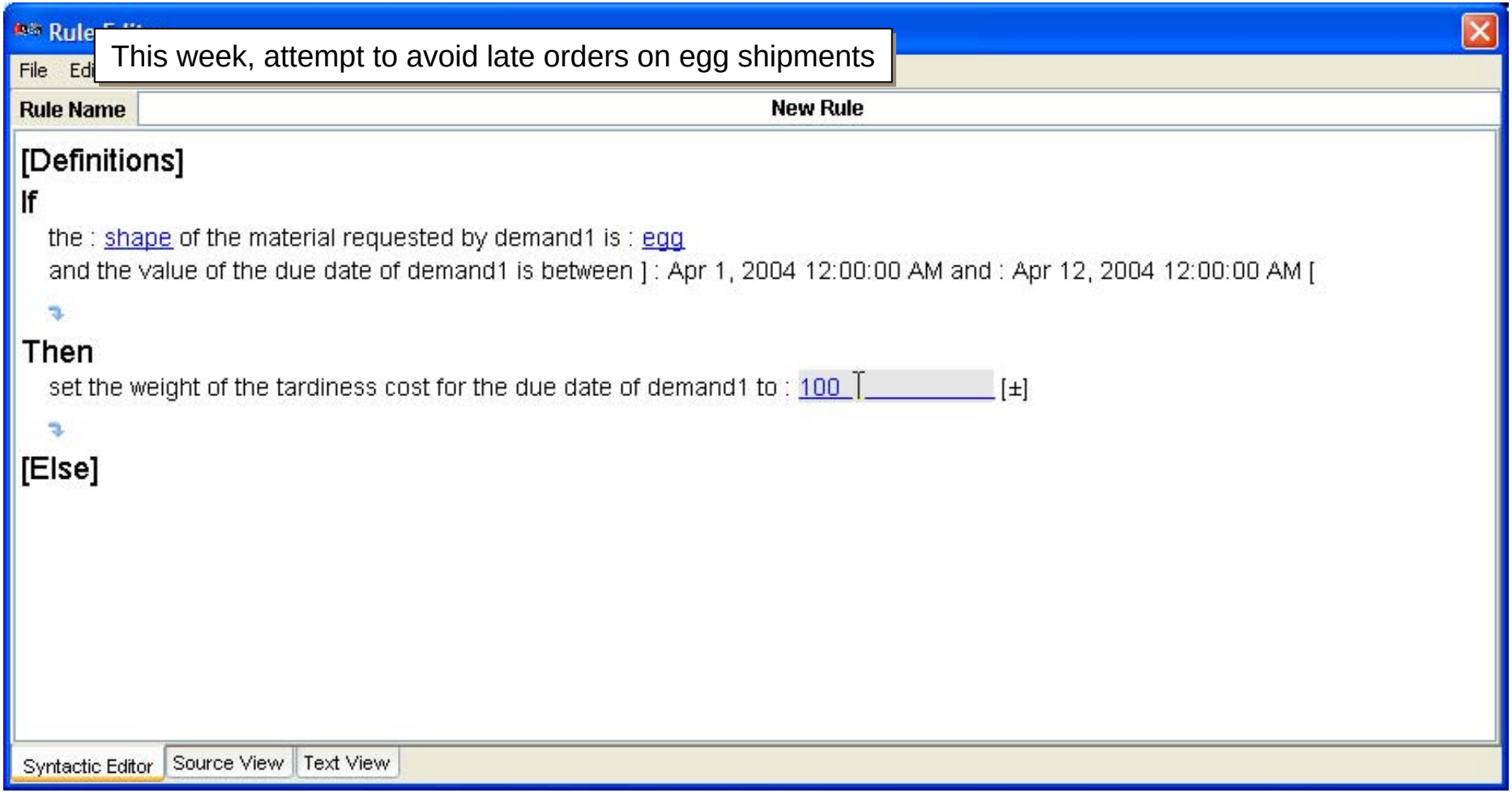
Then

remove *activity 1* to selection

Business rules integration

Changing the rules of business™

Rules can be defined using the syntactic editor

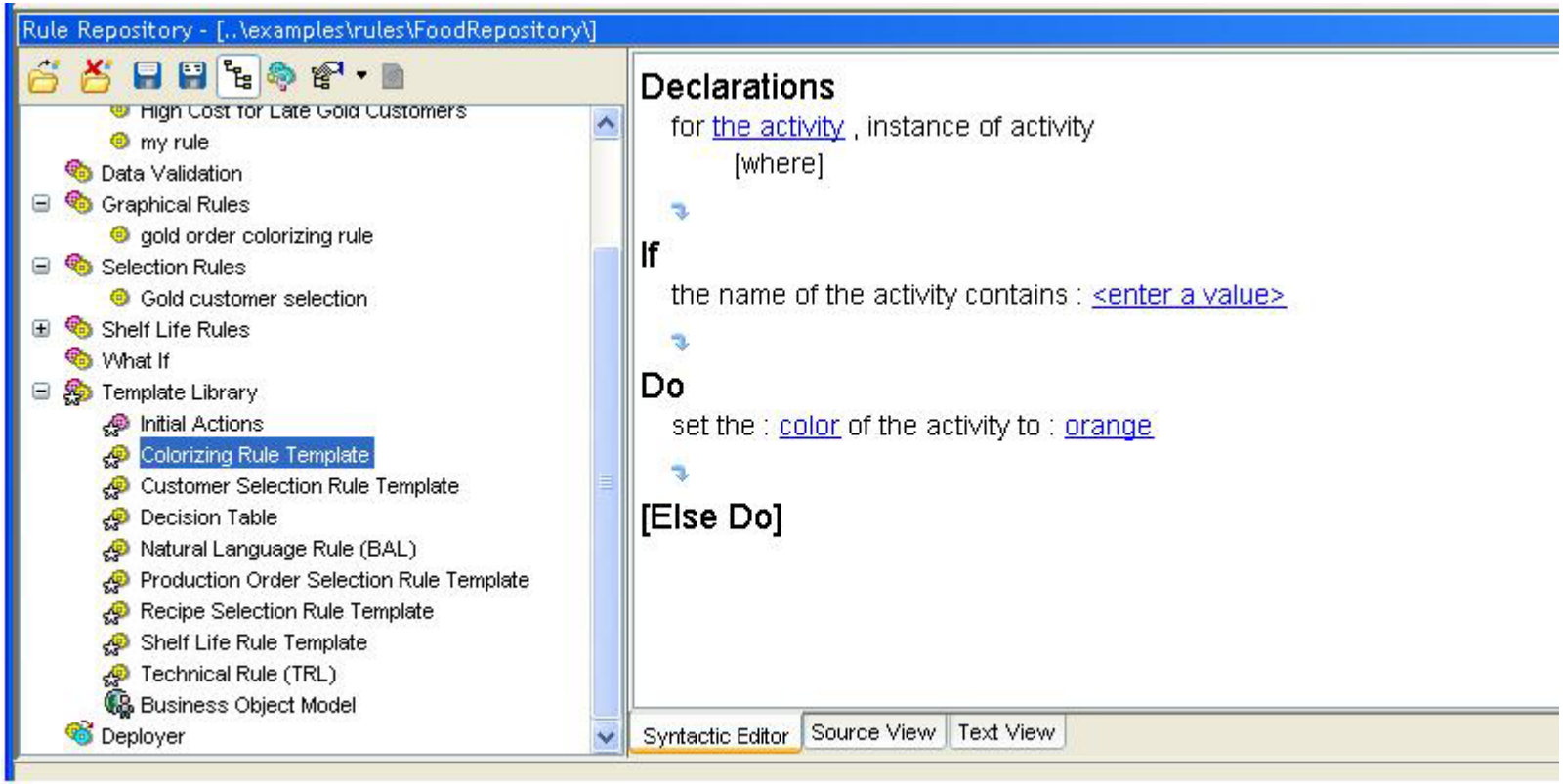


Business rules integration

Changing the rules of business™

Rule can be defined using rule templates

- **Create a new rules by filling in placeholders of existing templates**

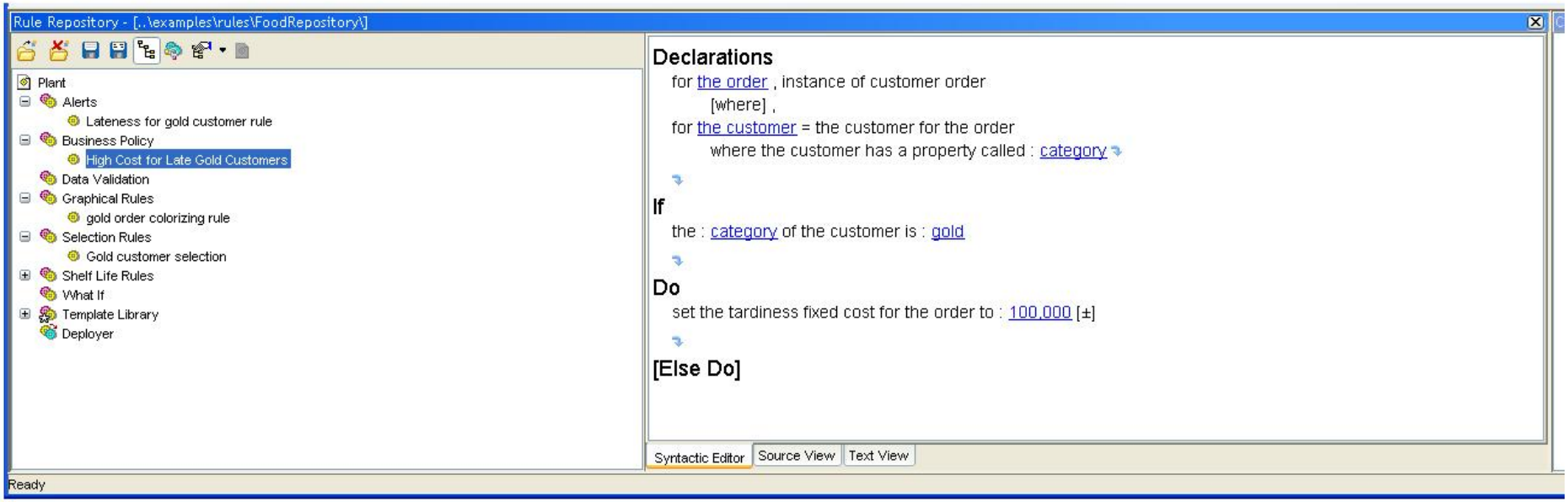


Business rules integration

Changing the rules of business™

Rule repository

- Rules are defined in rule packages
- Each rule package is organized in repositories

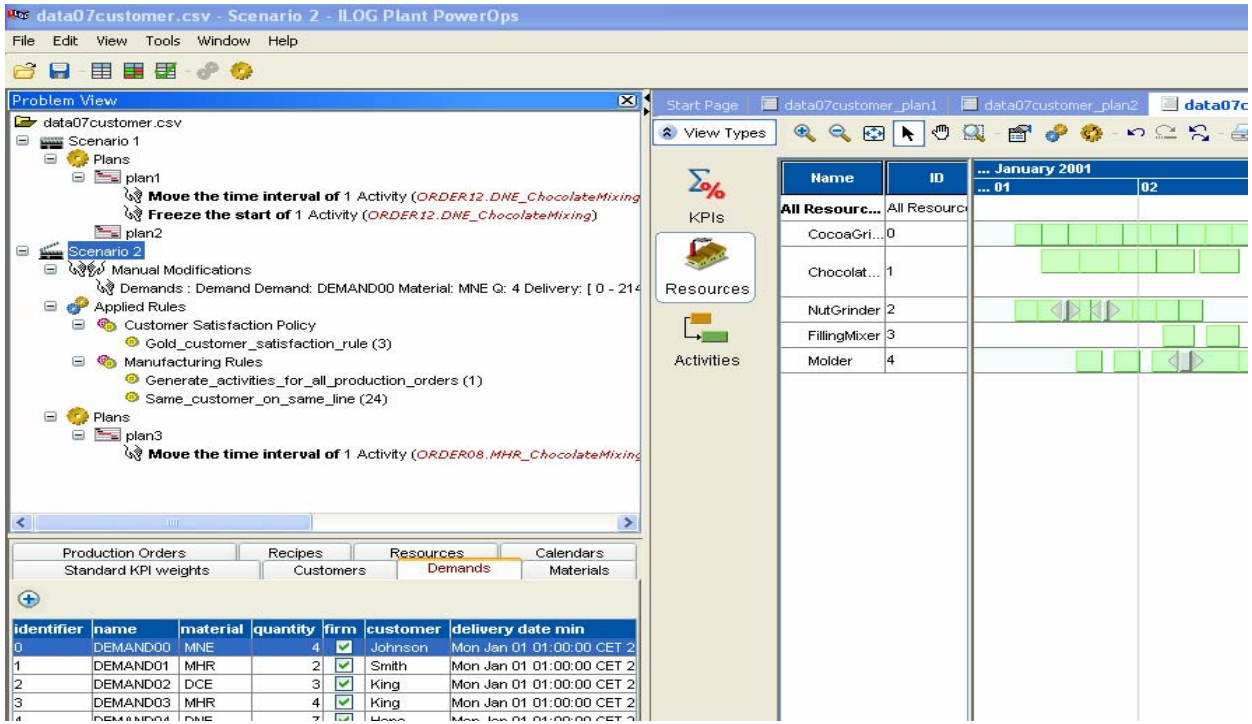


Business rules integration

Changing the rules of business™

Scenario management

- **Model modifications (manual editing or pre-processing rules) apply within a scenario.**

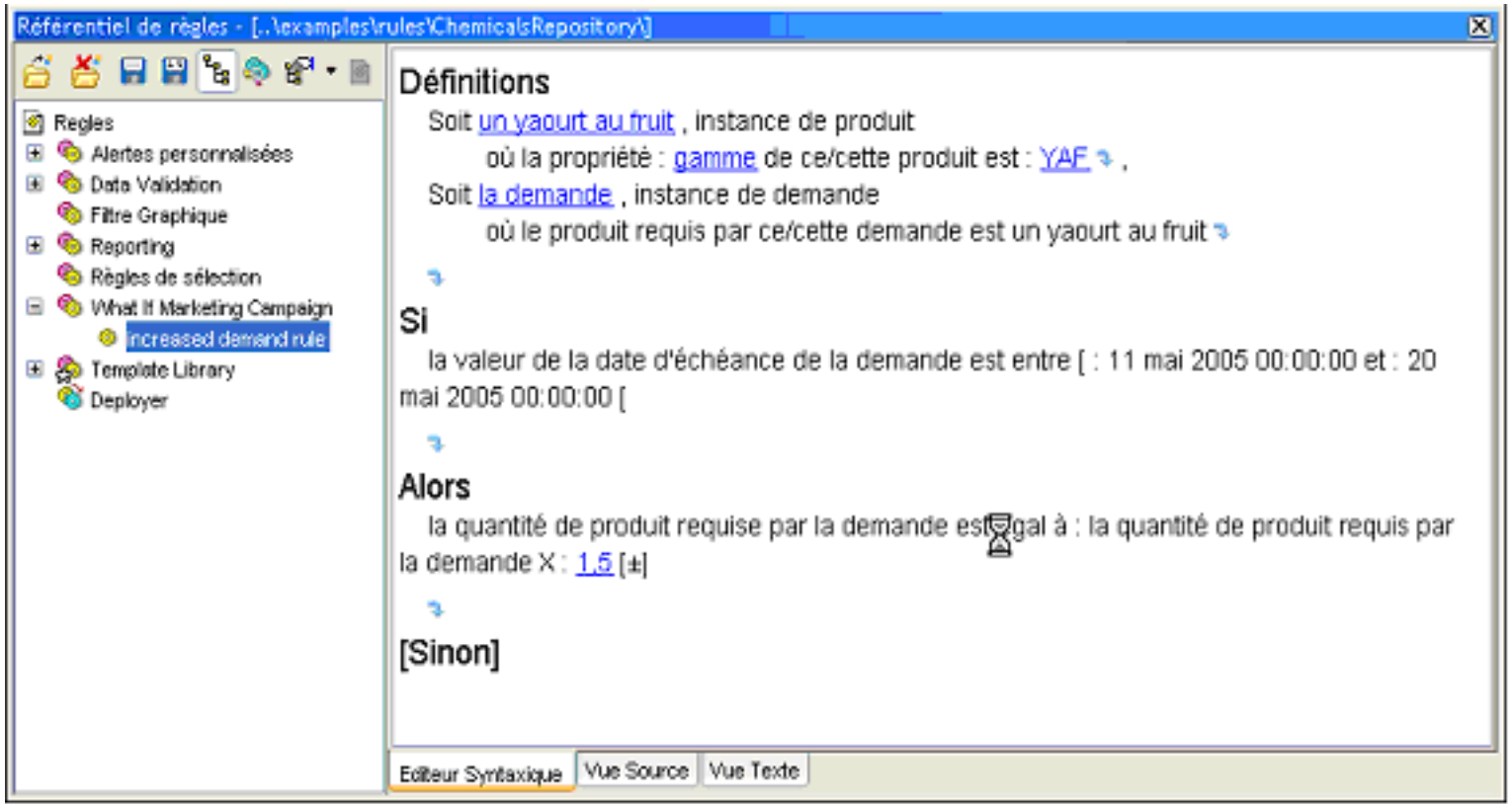


Business rules integration

Changing the rules of business™

Scenario management

- **An example of simulation for one of our French prospects – what if marketing campaign -**

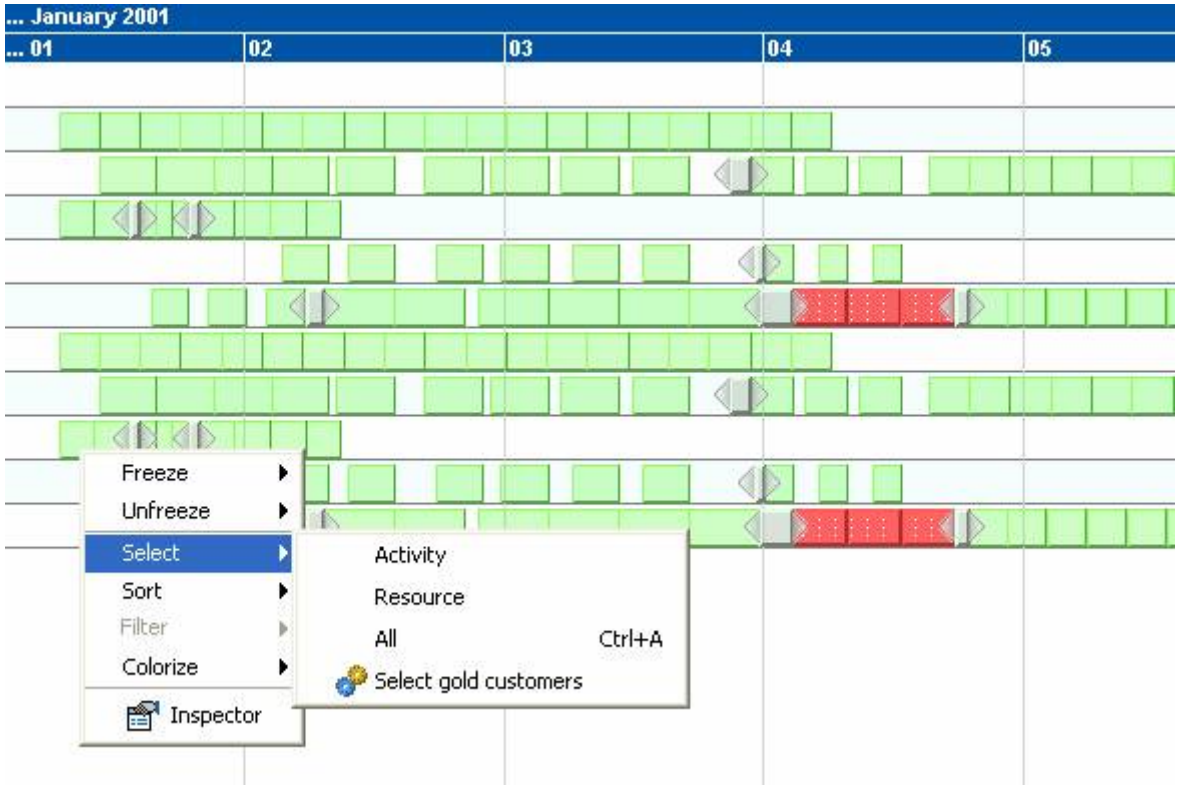


Business rules integration

Changing the rules of business™

Integration in context-sensitive menus

- **Rule packages can be “tagged” to declare its purpose to use them from the Gantt context-sensitive menu**

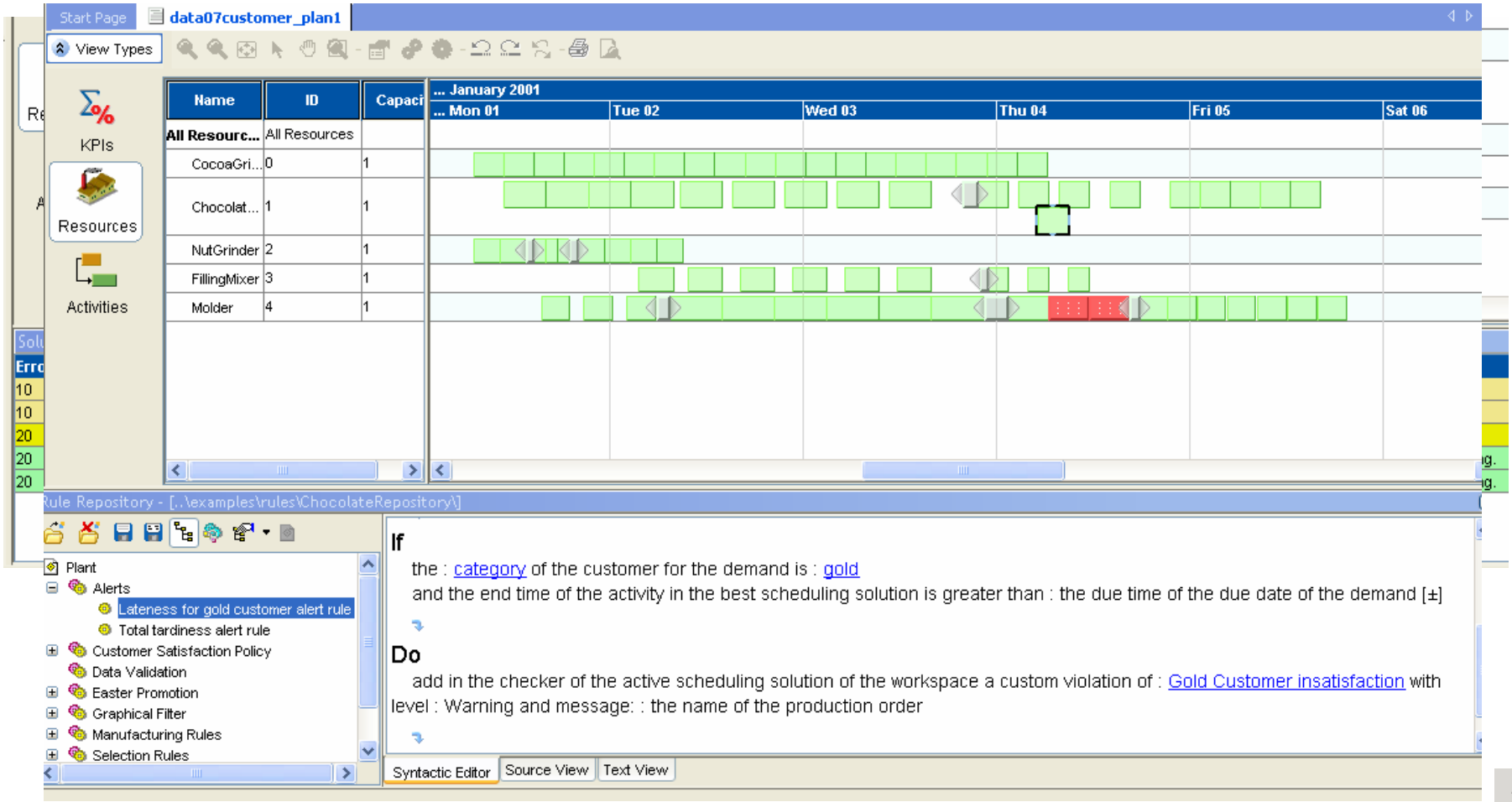


Business rules integration

Changing the rules of business™

Integration in checker

- **Checking rules are integrated in built-in checker**



Discussion and open questions

Loose vs. tight integration

- Rules are applied either before or after solving (not during)
  - Pre-processing rules and post-processing rules
- The rule engine transform a models and/or solutions
  - The optimization engines have no knowledge of the rule engine
- The object models are “closed”. No access to decision variables
  - The rule engine has no knowledge of the optimization engines

- We claim that loose is better than tight (for end user applications)
  - Simple integration
  - Globally robust system
  - Avoid developing a rule-based language

Discussion and open questions

Business Rules vs. scripting language

- Since rules are used to transform models and solutions, why not defining a scripting language instead?

- We can use Java to extend the model
- Rules are “visible”
- Rules can be understood and modified by business users
- Rules can be organized in “rule packages” in a database
- Business rule is becoming a widely accepted and known technology

Discussion and open questions

- **Interaction of rules and search for**
  - **Constructive search methods**
  - **Defining local moves**
  - **Repairing infeasible plans**
  - **Defining soft constraints (wishes)**

Discussion and open questions

- **Rules and GUI**
  - **Rules provide a powerful user interface**
  - **What should be provided as rules and what should be provided as GUI?**
  - **From rule language to rule templates to decision tables (excel-like interface)**

Conclusion

- Rules interface to optimization models of PPO
- Same expressive power as the C++ or Java API
- Flexibility and adaptability needed to easily configure and maintain the application
- Externalize the business logic
- Rule based scenario management and what-if analysis

Conclusion

- **This approach can be generalized for any optimization application**
- **Still a lot of work need to be done**
  - **Interaction between rules and search**
  - **More experience with end-user project is needed**