Rule Engine

A Rule Engine is a Component which executes Business Rules which can be directly created or updated without going through the heavy classical development process.
The main advantage is “time to deploy Rules”.
So the Rule engine is well adapted to rules which **often change** like Pricing Rules.
Other advantages are **Visibility** on existing rules, and **Simulations** of new rules.

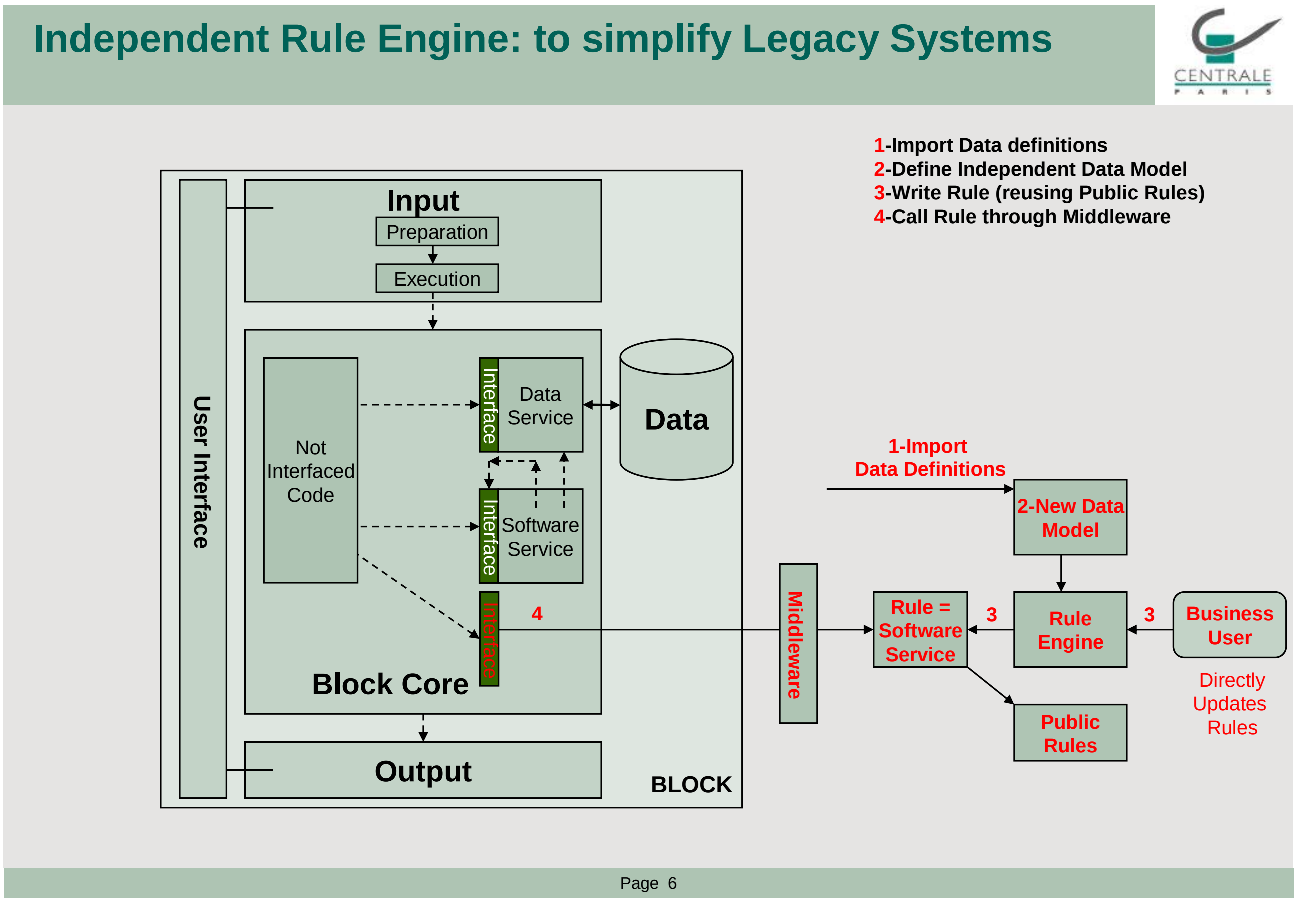
But isolating Rules supposes to answer some questions :

- Target questions
  - How to **interface** the rule engine to the Legacy System?
  - How to **find** rules (for example when updating): the rules are ordered in a set which is independent from applications?
  - How to **reduce the scope of data** proposed for each rule
  - How to offer an easy **user interface**?
  - How to **test a rule**?
  - How to provide good **performances**?
  - How to **reuse** existing rules?
  - How to **test the application** which calls the rule?
  - How to offer **simulation** tool?
- Migration questions
  - How to **find** rules inside existing code?
  - How to **replace** existing rules by a call to the Rule Engine?

As Time to Market becomes a priority and Rule Engines becomes more popular, a new question arises: how to **integrate** the Rule Engine inside a more global application?
Example of “Product Factory” which allows marketing teams to directly create or modify Products by defining Product Structure, assembling Services and managing Rules like pricing rules, commission rules or eligibility rules, without the help of IT teams.

1 Independent or Integrated Rule engine?

1.1 Independent Rule engine



An Independent Rule Engine allows to build rules in its specific environment. These rules may then been callable by external Blocks. The main advantage is that existing Blocks built on classical technologies can benefit from a Rule Engine.
And new Blocks, built on a technology which does not include a Rule Engine in the Development Environment, can also have access to Rule Engine power.
The difficulty is then to:

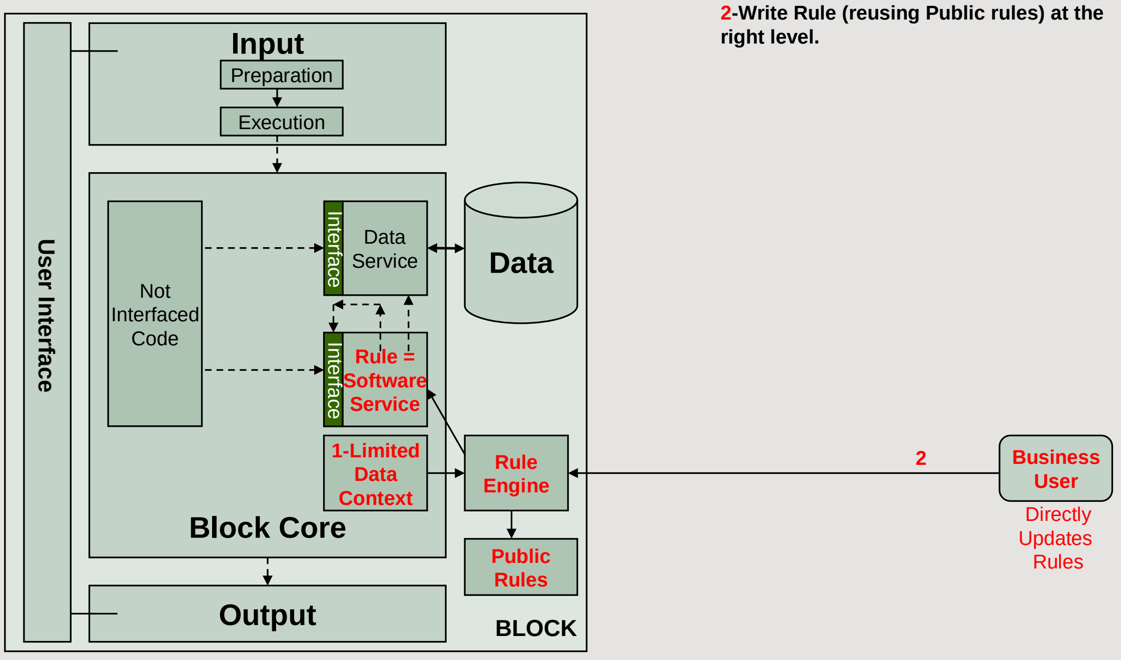
- Allow Blocks to **call** the Rule Engine.
- Manage all rules in an **independent** set which is not structured as the Blcoks: how to find its way in the Rule set?
- Provide to Rule designers, access to a **limited number of data** and not all possible data**.**

One solution is to **redefine** a limited data environment in the Rule Engine context: class model, data description of each class, and interface with Legacy system to import and export data. This is a good way to limit the data model to what is sufficient to Rule Designers. But it means that there is a new model to maintain and keep consistent with the Legacy System.

1.2 Integrated Rule Engine

Integrated Rule Engine: for new applications

1-Define limited data Context by Rule type
2-Write Rule (reusing Public rules) at the right level.



Page 5

Another way is to **integrate** the Rule Engine inside the Development Environment to allow the Application Developer to install the Rule Engine inside the application. This is more a solution for **new applications** which can reuse embedded technologies. The Rule Engine allows to position the Rule where it is necessary in the Application: the Rule Engine is more a common tool reused by each application. It allows to simplify the work of people who adapt Business Applications because they can create, modify, inquire rules inside the application.

The **best solution** would be to merge both approaches:

- Use an **integrated** Rule Engine for **new** applications
- Use an **independent** Rule Engine for **existing** applications
- Obtain to do it with the **same tool**

2 Data Context for each Rule type

To simplify activity of the Product Designer, you must offer not only a Rule Engine, but also **limit the data** to concentrate on what is really useful for him.
Rules can be **classified** by Rule Type: pricing, eligibility, commissioning, selection for a marketing campaign. This Rule Type can be more precisely defined by other criterias like product family.
Instead of giving access to all data of the Information System, we must define for each Rule type, the data subset he could use.
Definition of this subset is done by **Business** people, but **implementation** must be done by **IT** teams who know **where** to catch the data in the Information System (for example, if some data are duplicated, only IT teams know where is the right data), **transform** types if necessary, **optimize** access for Rule Engine performance, …

- For dissociated business rule engine, it is done by import and export functions
- For integrated business rule engine, it is done by defining subset of data in each Rule Type context.

Then the Rule Designer may work on this limited context to build as many Rules as he wishes.

3 Private Rules and Public Rules

Most Business Rules are **Private Rules** and not reusable by other private rules. They can be copied to create new Private Rules, but these rules are not maintained together.
Ex: the rating rule for Service “window break” of Product “Car Insurance for young drivers” is a Private Rule.
But some rules like Actuarial Rules or Mortality Rules are **Public Rules** reused by the Private Rules. They come with a precise interface.
Public rules may call other Public Rules: they are classified in **catalogs** to ease research.

4 How to create a Rule?

A Rule (private or public) must be expressed as simply as possible, which means that several levels of complexity should be allowed

- As a constant, or proportional value
- As a Table with 1, 2 ou more dimensions
- As a formula
- As a table of formulas

A Formula must be checked with a **syntactic control** offered by the Rule engine, to warn the Rule Designer of possible syntax errors.
The Rule Engine must manage **numeric** or **logical** data.
The Rule Engine must provide a **test tool** to immediately test the Rule.
The Rule Engine must provide an integrated **debugger**.
The Rule should have a **status** which reflects its life cycle: in construction, waiting for validation, active, abandoned, cancelled, …
The constants used by a Rule (like VAT %, mortality table, …) must be directly imported inside the Rule Engine.

5 How to find Rules ?

Application developers may call Private and Public Rules.
A **Rule Repository** offers tools to find Rules.
Rules should be accessible by:

- Name and version (or date)
- Or any Rule family or classification which helps to find them
- Or by a multi Criteria query

Private Rule designers may reuse Public Rules.
To help them, a **Public Rule Repository** should be offered which include: Rule name, version, interface and description of the service it offers.

6 How to call a Rule?

Rules must be accessed by any **middleware**.
Rules must be **versioned** and validity is defined by **beginning** and **end dates**.
Rules must be reusable by **batch** or **transactional** applications.
To reach excellence performances with batch applications, Rules must be launched and executed by **set of Rules**.
Rules should be used to **simulate** Products.