CLOUD COMPUTING SERVICES FOR THE DESIGN AND OPTIMAL MANAGEMENT OF BUILDINGS

B. Delinchant[1], P.Y. Gibello[2], F. Verdière[3], F. Wurtz[1]

[1]G2Elab, Electrical Engineering Lab, Grenoble University, France
[2]Linagora, 8 place R. Schuman, Grenoble, France
[3]Vesta-System: 22, av. Doyen L. Weil, 38000 Grenoble, France

ABSTRACT

The optimal building management models require predicting the behaviour of building systems. To achieve this goal, we propose to make available on the web remote computing services for the composition, simulation and optimization of equipments. A paradigm shift is necessary in order to fulfil this goal. We show in this study that models can be embedded in software components and made available on the web without any special programming skills. A proof-of-concept prototype is provided as demonstration, and a remote optimization is done without ever holding the model as a possible use of such a concept.

PERSPECTIVES OPPENDED BY CLOUD COMPUTING IN BUILDING MANAGMENT

Building management

The building, with 43% of primary energy consumption and 66% of the electrical energy consumption in France, is a major challenge for the coming years in most of countries. The problem described here is to imagine new approaches to help improving both the design (simulation, optimization, ...) and monitoring (proactive and reactive optimal management, ...) of buildings. It may take advantage of the accessibility of large information networks in buildings and a major phenomenon: the "cloud" or “cloud computing”.

Energetic network and information network

The following figure illustrates a possible link between two ubiquitous networks, the information and the energy ones. Indeed, we can imagine that all components related to energy building are interconnected via two channels: energy and information technology. This is the architecture that we can call “smart building”. This link can be real or virtual. Indeed, we can consider that in the near future it is utopian to make "intelligent" all our equipments, in homes or offices. For cons, the same way that the measurements made by the sensors on the equipment may be available on standard data servers (OPC[1], ...), it is quite conceivable that simulations are available on the servers, "virtualizing" operation of individual systems or the overall building. These models can be supplied by equipment manufacturers or by integrators, and can be available on computational servers to provide solutions to design and / or management buildings “smarter”.

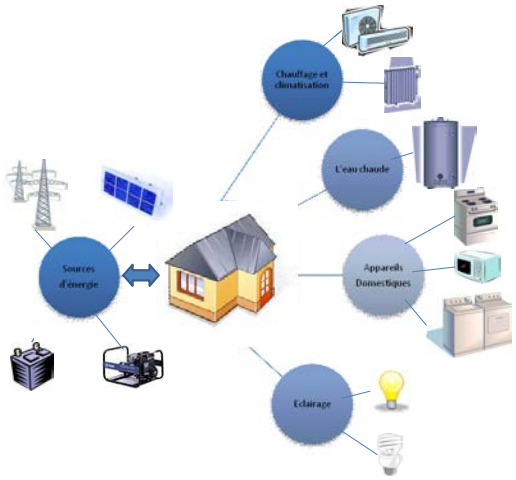

Figure 1 Information and Energy Nets convergence

This new vision of building simulation offer a new way of business and many company are going in this way. For instance, data monitoring and associated services are just emerging like *Panoptix™* platform from *Johnson Control* (Panoptix, 2012).

From this kind of cloud architecture, we can imagine many services such as:

- Pre-diagnosis / Audit: Estimated potential energy savings.
- Maintenance: locating malfunctions on installation
- Energy monitoring: monitoring and control to quickly identify and correct the sources of drift energy and discomfort.

For instance, *BuildingIQ* company is offering predictive energy optimization software that have

[1] OPC: Object linking and embedding for Process Control

been planned to be connected to such a cloud platform (PikeResearch, 2012).

These new businesses are opportunities for building simulation to go further than building design. In this paper, we are giving guidelines to bring such opportunities to scientific simulation codes.

Cloud computing

We present quickly the potential benefits of services computing, simulation and optimization available in remote web servers:

- We would like to use remote computation resources :
  - High-Memory (up to 68.4 GB),
  - High-CPU (up to 20 computation units)
  - Wide band CPU Clusters (up to 88 comp. units, 10Gbps network),
  - GPU (2x515 Gflops double, 2x448 cores) …
- Reducing complexity :
  - No need to buy it and making it work (skills, maintenance, …).
- Reducing cost: pay-as-you-go pricing
  - On demand, reserved or spot Instances
  - Adjusting resources depending on the problem size and solution delay requirements.
- Executing HPC
  - anywhere using simple remote requests
  - at any time with no queues due to scale factor

Beside these classical advantages, cloud computing can be used when models cannot be available easily:

  - Intellectual property
  - Software licenses for numerical modeling software, optimization algorithm, ...

By cons, there are legal issues arising out of cloud computing. They can be categorized as operational, legislative or regulatory, security, third party contractual limitations, risk allocation or mitigation, and those relating to jurisdiction.

IPR (Intellectual Property Rights) are also in question for building simulation models used for cloud computing. It is not our goal to answer such legislative question but we are thinking that several ways are available like it is for model libraries (open source, binaries, …).

A DEDICATED ARCHITECTURE: FROM COMPONENT TO WEB-SERVICES

Needs for a paradigm shift

Perspectives that we discuss can not be achieved without the cost of a paradigm shift in the structure of simulation codes currently used. This requires going beyond the paradigm of standalone tools that are used today which rely on non-distributed computing environments and non-interoperable. For example, models are typically developed in a specific software tool which do not offering interoperability solution. This is not suited to the increasing complexity of building simulation which requires taking into account the overall system.

This is why we are proposing to rely on a paradigm of open framework, which is based on the concept of software component, which are autonomous, capitalizable, and conceptually dedicated to interoperability. The component approach follows the object-oriented paradigms in the history of structuring software in computer science (Szypersky C. 98). It is within this framework that was developed ICAR-MUSE components studied in french national projects SIMINTHEC and PLUME (Gaaloul S. et al, 2011).

Benefits of the component approach

Software components can automatically handle issues such as communication between software tools, the heterogeneity of the programming language and operating system. It also allows the capitalization and making available "off the shelf" models, in the case of compound models. It can be noticed that our software component norm (ICAr: Interface for Component Architecture) has been developed a decade ago (Delinchant et al, 2004) in the field of optimal design of electrical engineering systems. It offers now solutions for interoperability between tools or languages such as Modelica, TRNSYS, Pleiade/Comfie, etc.

One of the main characteristic of software component is their capacity for autonomy. This is an important feature; they can be exchanged between partners, or simply made available for download on sites dedicated to the capitalization/reuse of models such as DIMOCODE (http://www.dimocode.org). In this work, we are not addressing meta-information associated with computable model. This meta-information, like modelling hypothesis, variables range, and all information required for reuse purpose, must be founded beside computable models as it is in DIMOCODE web site.

The next step that we propose here is to make the computing capabilities of a model, encapsulated in a component, directly available remotely on a web-server.

Making services available online

The solution we have implemented in the project SIMINTHEC respects the following techniques recommendations:

- Technology based on a common protocol for the Internet: HTTP, REST (Representational State Transfer), (Fielding, 2000)
- Deployable on a framework for common use: Java Web Application (Apache Tomcat application server)

- Neutrality regarding protocols, technologies and languages (low normative approach): JSON encoding (JavaScript Object Notation).

We are claiming that, due to their basic protocols, such specified components are easy to develop, implement and assemble, and do not induce limitations to future innovation.

If we take the example of a simple model of heat conduction in a wall (Figure 2). Fourier's law applied to equation (1) gives us the expression of the heat flux density (2) and therefore the total flow (3). We simply retain the final equation (4) providing the difference in temperature depending on the flow and the thermal resistance of the wall.

$$T = T_1 + \frac{x - x_1}{e}(T_2 - T_1) \quad (1)$$

$$\varphi = -\lambda\frac{dT}{dx} = \frac{\lambda}{e}(T_1 - T_2) \quad (2)$$

$$\Phi = \frac{\lambda S}{e}(T_1 - T_2) \quad (3)$$

$$\Delta T = R \cdot \Phi \quad (4)$$

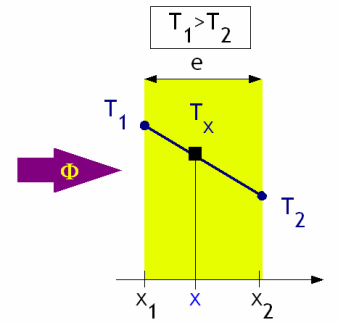

Figure 2 A simple heat conduction example

Of course, many other "conduction models" can be defined. We are not trying to reach an holistic modelling framework, but a framework were many models can be computed with their own goals and associated hypothesis.

This model can be described for instance in the tool CADES[2] (www.cades-solutions.com) [DEL 07] which generates a software component (ICAr[3]). This component is also WAR[4] compatible, so it can be uploaded to an application server (like Tomcat apache.org) and is available for acting as a computation web-service.

We have also developed a web interface to upload such components on a server, accessible via login, which allows managing its services (Figure 3).

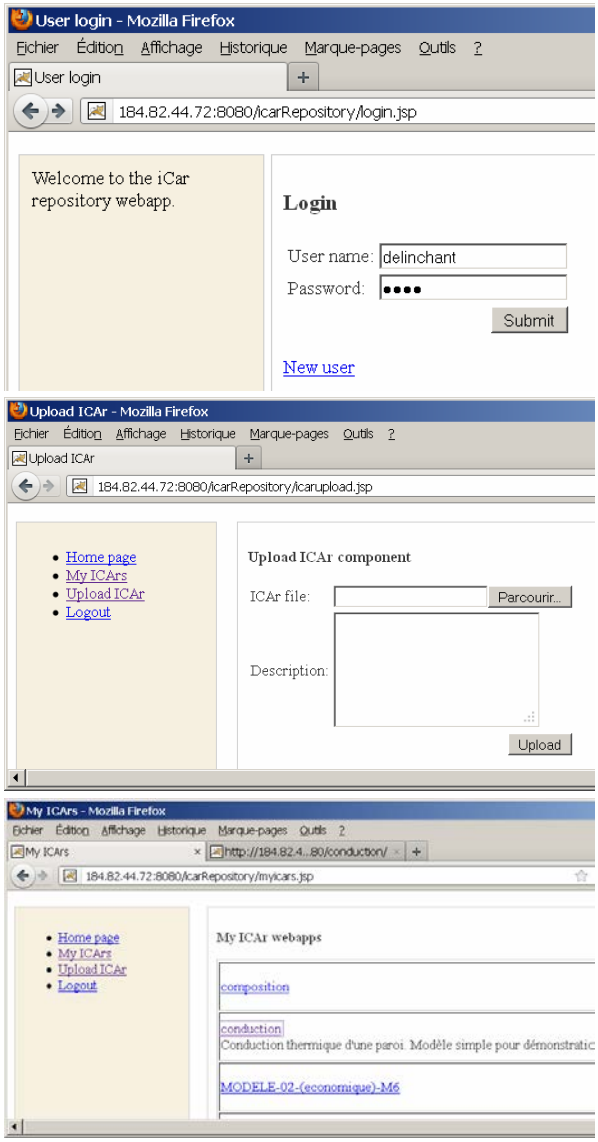

Figure 3 Web-service management platform

Once the component is loaded, the list of available services and details of ports in different services may be displayed by a simple HTTP request (Figure 4).

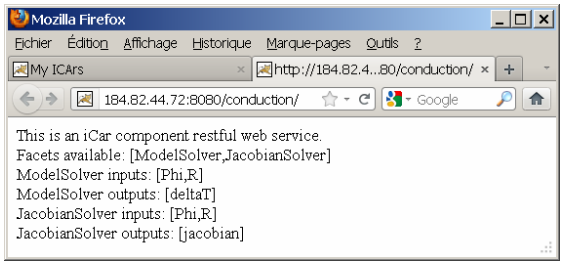

Figure 4 List of services and ports of "conduction" web-service

The request is made over HTTP on an URL[5] that represents a resource. The application consists of parameters (QUERY_STRING part) that the

[2] CADES: Component Architecture for Design of Engineering Systems
[3] ICAr: Interface for Component Architecture
[4] WAR: Web Application aRchive
[5] URL: Uniform Resource Locator

resource will then interpret. This can be done simply in the web browser, but also in any language (Python, Java, dot Net...) and very easily. For example, in the case of the previous uploaded service, you simply have to invoke the following query:

```
Request:
http://184.82.44.72:8080/conduction/facets

Answer: [ModelSolver,JacobianSolver]
```

The component may provide metadata descriptions such as a list of its input and output ports. Here, the input ports:

```
Request: http://184.82.44.72:8080/conduction
                    /facets/ModelSolver/inputnames

Answer: [Phi,R]
```

It is then possible to invoke this service to perform a model simulation, for example (calculation of u for $i = 3$ and $r = 2$):

```
Request: http://184.82.44.72:8080/conduction
      /facets/ModelSolver/resolve?{R:3.0,Phi:2.0}

Answer: {deltaT:6}
```

So, in this simple example, we can see that :

- model is made available automatically as a software component using appropriate modelling tools (e.g. CADES),
- model is made available automatically as a web service using our management platform web site,
- model can be introspected (what can it do ?), and used (simulation, or optimization services) with a simple protocol (HTTP commands).

From this first automated chain, we are thinking that models reuse capacity is improved.

In the next part, we are trying to see futher than simple model computation using HTTP commands, which was done for demonstration purpose.

USE CASE: OPTIMIZATION OF COMPOUND MODEL COMPUTED REMOTLY

Design problem description

In order to illustrate our approach; we would like to determine the optimal control of a heating system as a function of the user temperature set point.

For this, two models have been used:

- *Envelope model*, in order to compute the internal temperature as a function of external temperature, and sources. This simple model is based on the electrical analogy and is solved symbolically to explicit the temperature depending on time and sources.
- *Controlled heating system*, injecting a heat output based on a set point and the current temperature value.

Web-service creation

Technology we developed for creating a web service is based on the concept of automatic code generation from a model description (Modelica) or existing simulator (FMI, TRNSys, Pleiade/Comfie...). In Gaaloul S. et al, 2011, we have shown the complementary approaches of “white box” modelling based for example on Modelica language, and the ”black box” modelling based on software components. These two approaches are mixed using “plug’ out” and “plug’ in” allowing the extraction of simulation models from building simulation tools and their use in composition framework such as Matlab[6] or Dymola[7] thanks to component approach.

For instance, we are using Java language, which is directly compatible with application server technologies (Apache / Tomcat). In Verdière F. et al, 2012, we introduce a software component generator of Modelica model in full Java, including jacobian computation which can be used for DAE[8] solving or for optimization.

In this paper, we are using such a generator, including Jacobian automatic programming in order to make efficient optimization. CADES software is then used to describe the compound model, based on algebraic equations and algorithms:

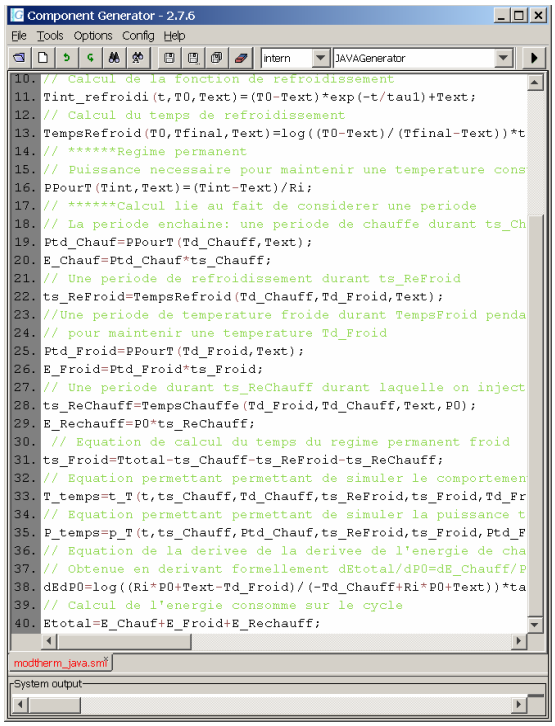

Figure 5 Algebraic equation modelling in CADES

[6] Matlab™ : *www.mathworks.fr/products/matlab*
[7] Dymola™ : *www.dymola.com*
[8] DAE : Differential Algebraic Equation

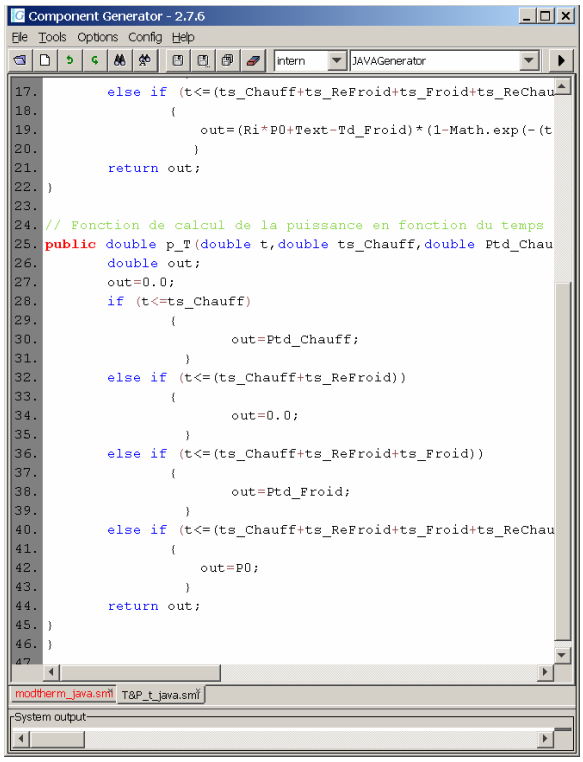

Figure 6 Algorithmic modelling in CADES

The software component is automatically generated (model analysis, definition of incidence graph computation, Java code generation, compilation, packaging). The file is then uploaded on an application server through a web browser. It is automatically available, ready to be used as a web service.

Web-service connected to an optimizer

We can now imagine a different actor, allowed to use this web-service, which wants to make an optimization. He can connect the web-service to its optimization tool (Delinchant B. et al. 2007) using a plug' in. Only the URL (http://184.82.44.72:8080/...) is requiered to connect the computation service to the optimizer.

In our example, this optimization allows to determine, for a set of weather data and characteristics of the envelope, the optimal control satisfying the user comfort and minimizing the energy consumption. This optimal solution was founded in a transparent way without owning any modelling software in local.

Figure 7 illustrates this optimization, in which the system that is optimized, is virtually present locally beside the optimization algorithm, but is executing building simulation program remotely in reality.

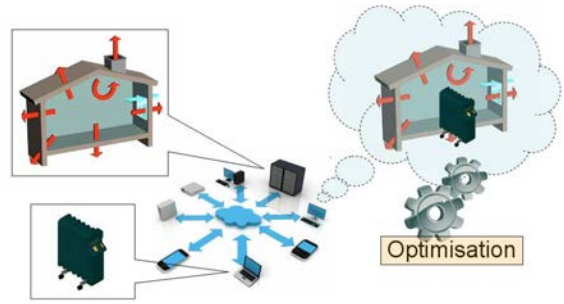

Figure 7 An example of optimization with distributed components compounded in the "cloud"

Figure 8 illustrates technologies that are behind this automation of remote access. Major parts are :

- Model as software component
- Application server (Apache)
- HTTP / REST protocols over the internet
- Plug'in for each tools in which web-service wants to be used.

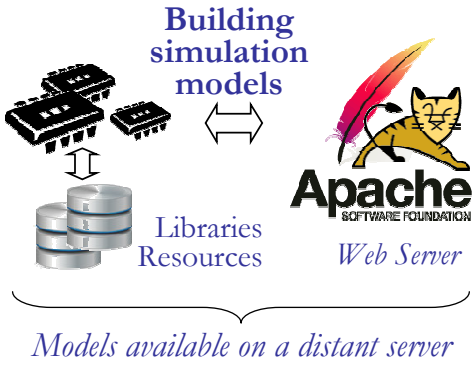

Figure 8 An example of optimization with distributed components compounded in the "cloud"

Figure 9 is showing the optimization that is performed in 10 iterations (fast thanks to the availability of the model Jacobian) and 3.7 seconds. Optimization results are used to determine the heating power (P_0 = 100W), the optimal temperature stimulus (Td_{froid} = 2 °C) to minimize energy for a day (E_{total} = 647 J).

CONCLUSION AND PERSPECTIVES

We have developed an automated chain, from modelling to uploading web service in the "cloud". This chain is based on software component and plug' out available for community's modelling software. A simple web service protocol has been defined in order to improve models reuse capacity. Then, this chain allows the availability of building model during it life cycle, from design to energy management. It is a key feature to achieve smart building, in which information network will merge with energy network. This smart building will require data measurements, but also models and optimization algorithm. It has been shown that several models can be composed and optimized, using our architecture, to design systems or to predict optimal management scheme. Optimization algorithms can also be used to fit model using data, and we can imagine many other interesting services based on building models. For reasons discussed in this paper, we thought that a part of the building intelligence will be delocalized in the cloud using this kind of paradigm and associated technologies.

ACKNOWLEDGEMENT

The work presented in this paper was performed in the scope of the SIMINTHEC project, funded by the French National Research Agency.

REFERENCES

Panoptix™ : www.johnsoncontrols.com/panoptix

Pikes Research, 2012 : www.pikeresearch.com/tag/smart-buildings-practice

Szypersky C., Component Software – Beyond Object-Oriented Programming, Addison-Wesley, 1998

Gaaloul S. et al 2011. Software Components For Dynamic Building Simulation, Proceedings of Building Simulation 2011, Sydney.

Fielding R., Representational State Transfer (REST), Chapter 5, PhD, 2000

Verdière F. et al 2012, Modelica models translation into Java components for optimization and DAE solving using automatic differentiation, IEEE UKSim2012

Delinchant B. et al. 2004, "A component-based framework for the composition of simulation software modeling electrical systems", Journal of Simulation, Special Issue: Component-Based Modeling and Simulation. vol. 80: pp 347 - 356.

Delinchant B. et al. 2007, An Optimizer using the Software Component Paradigm for the Optimization of Engineering Systems, COMPEL, Vol. 26 No. 2, 2007, pp. 368-379

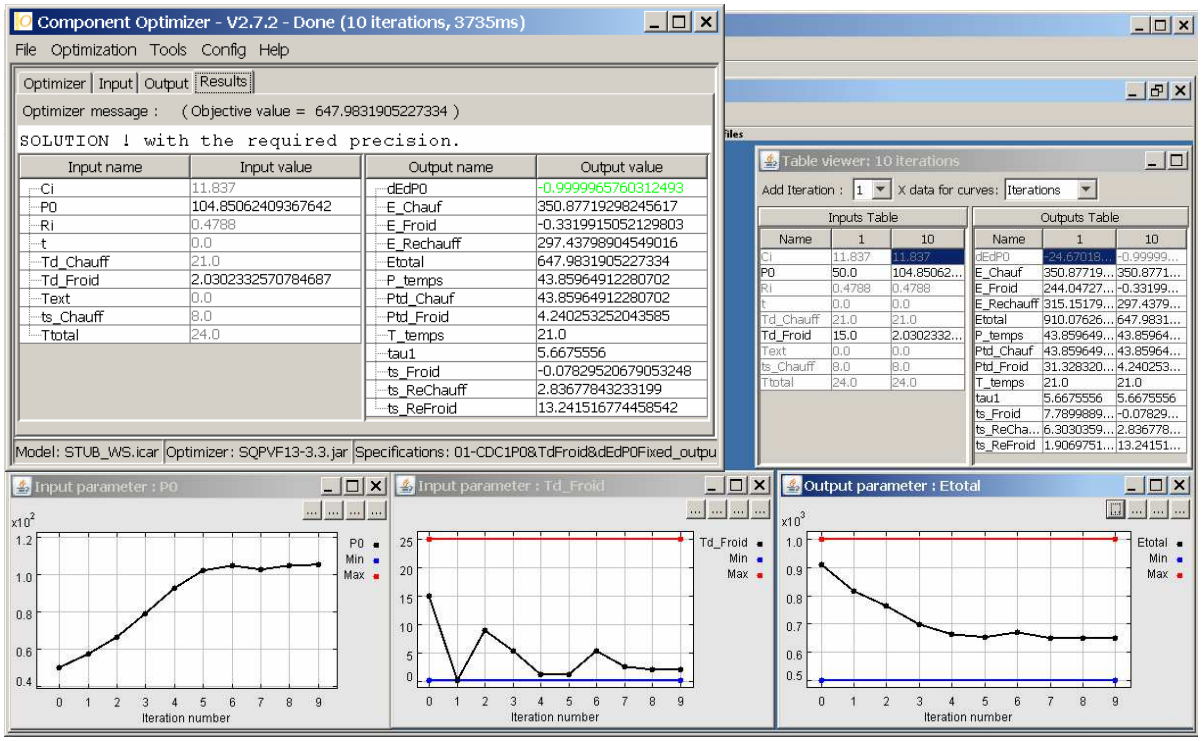

Figure 9 Optimization results based on remote web-service simulations