# Online Application of Modelica Models in the Industrial IT Extended Automation System 800xA

Rüdiger Franke
ABB AG, Power Technology Systems
Kallstadter Str. 1
68309 Mannheim, Germany

Jens Doppelhamer
ABB Corporate Research
Wallstadter Str. 59
68526 Ladenburg, Germany

## Abstract

The Modelica technology and the increasing availability of model libraries allow an efficient modeling of complex dynamic processes. Having a good process model at hand one might want to apply the model online to improve the operation of the real process. These online applications range from the generation of high-level information like performance indices from process measurements over the estimation of unmeasured quantities in a so called soft sensor up to model based control and online optimization.

This paper discusses the online application of Modelica models in an industrial control system. The models are developed and tested using a standard Modelica tool. Afterwards they are imported into the control system. Here the model variables can be associated with process signals. This way a model can be initialized with current process values. A numerical solver performs simulation, estimation or optimization activities. Solution results can either be used for diagnostics or they can be fed back to the process as manipulated variables.

The Dynamic Optimization system extension has been developed for the Industrial IT System 800xA. Exploiting Aspect Object technology, the required functionality for model-based applications can be integrated seamlessly with the control system. Model based applications can be set up in a modularly structured way.

The Dynamic Optimization system extension has been used to deploy different model-based applications. A Nonlinear Model-based Predictive Controller (NMPC) for the start-up of steam power plants is discussed as an example. The overall NMPC application consists of several model-based activities, including preprocessing of process values, estimation of model states, prediction of optimal operations, and post-processing of optimization results. A scheduler periodically triggers these activities online.

*Keywords: Modelica, 800xA, Industrial IT, control system, online optimization, NMPC*

## 1 Introduction

The Modelica technology clearly separates between model specification and model solution. This way not only existing models can be used with different tools, but also different kinds model based activities can be performed for one and the same model. Such activities include, besides the solution of initial-value simulation problems, also the estimation of model parameters, the optimization of the design of a modeled process and model-based control.

Also the increasing availability of libraries for fluid processes is making Modelica more and more suitable for process applications, see also [5, 8, 4].

Additional things that have to be treated in a real model based application include the signal exchange with the process, concepts for security and redundancy as well as real-time scheduling of model based activities and the operator user interface.

This paper discusses the integration of dynamic optimization with the Industrial IT System 800xA by ABB, allowing a rapid online deployment of model based applications once appropriate models and model based activities have been set up offline.

## 2 System 800xA overview

The architectural framework for the Industrial IT System 800xA is built upon ABB's Aspect Object technology [1]. Aspect Objects relate plant data and functions – the aspects, to specific plant assets – the objects. Aspect objects represent real objects, such as process units, devices and controllers. Aspects are informational items, such as I/O definitions, engineering

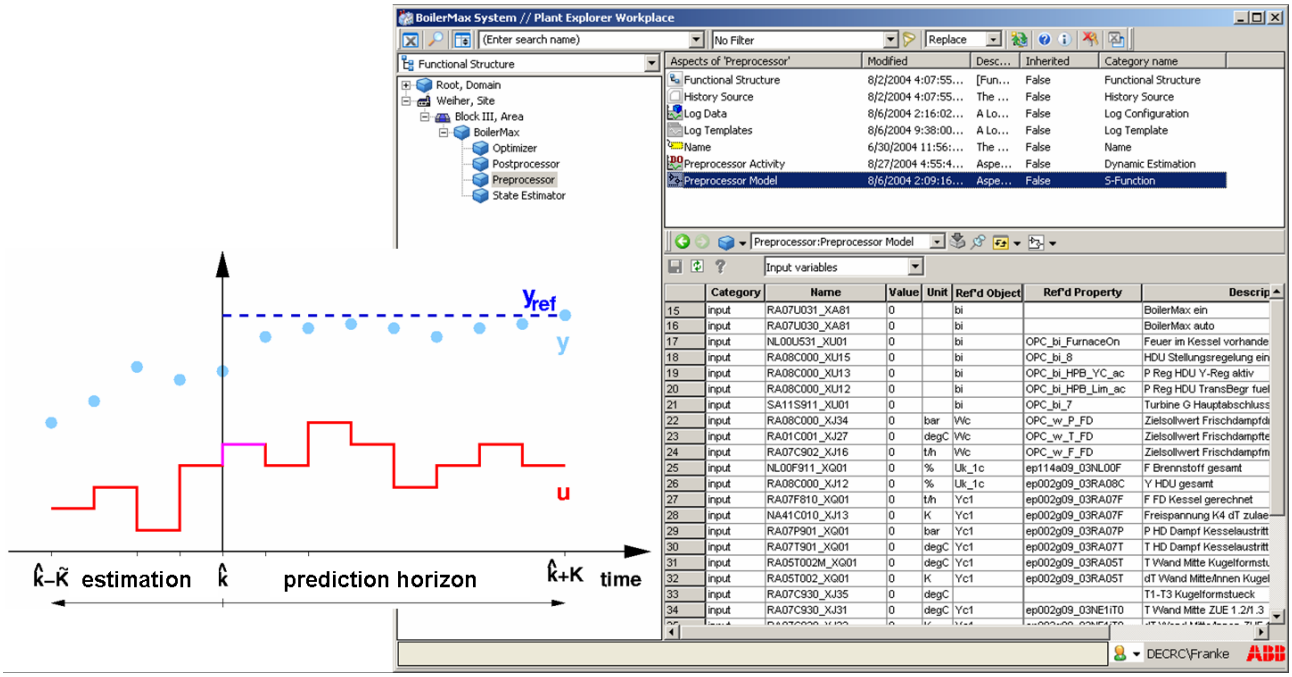


Figure 1: Plant Explorer Workplace showing the Functional Structure of an NMPC.

drawings, process graphics, reports and trends that are assigned to the objects in the system.
Aspect Objects are organized in hierarchical structures that represent different views of the plant. One object may be placed multiple times in different structures. Examples for different types of structures are:

**Functional Structure:** Shows the plant from the process point of view.

**Location Structure:** Shows the physical layout of what equipment is located where in the plant.

**Control Structure:** Shows the control network in terms of networks, nodes, fieldbuses, and stations.

The idea of placing the same object in multiple structures is based on the IEC standard 1346 [9, 2]. A controller is a typical example: the real controller is represented by an Aspect Object. This object is placed in the control structure showing the logical arrangement of the controller in the control system, in the location structure showing the actual location, and in the functional structure showing the function of the controller for the operation of the process.
The Plant Explorer Workplace is the main tool used to create, delete, and organize Aspect Objects and aspects. It is based on a structural hierarchy, similar to Windows Explorer, as demonstrated in Figure 1. The object hierarchy is visible on the left hand side of the window. The upper right pane shows the aspects of an object and the lower right pane views a selected aspect.

# 3 Integration of model based control

## 3.1 Example for a complex model-based control application

Figure 1 shows how the functional structure is set up for an Nonlinear Model-based Predictive Controller (NMPC) using Aspect Object technology and the Dynamic Optimization system extension discussed in this section. Different Aspect Objects represent the major processing steps of the NMPC algorithm.

1. The Preprocessor reads current measurements, validates the data, and generates a guess for the model state. Furthermore a short term history is assembled.

2. The State Estimator uses the short term history and estimates the initial model state.

3. The Optimizer predicts the optimal control into the future, starting from the estimated initial state.

4. The Postprocessor checks optimization results and communicates set points to the underlying control system.

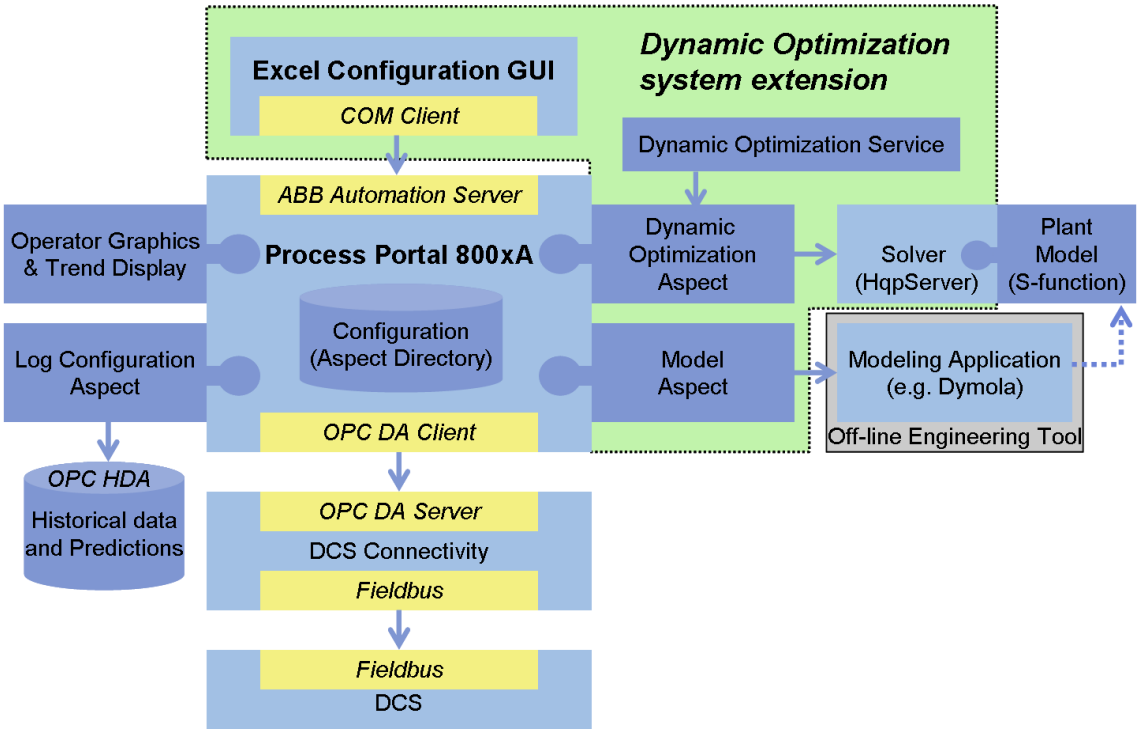


Figure 2: Software architecture of the Dynamic Optimization system extension.

An additional scheduler activity periodically triggers the other activities and supervises their successful completion.

The state estimator and the optimizer are based on the same plant model. This model is built efficiently on available model libraries [5]. Moreover, specific preprocessor and the postprocessor models are formulated as computational algorithms in Modelica. The scheduler model is formulated as a state graph [12].

Based on the models, the activities are formulated as estimation (State Estimator), optimization (Optimizer) or initial-value simulation (Preprocessor, Postprocessor, Scheduler). The Dynamic Optimization system extension provides the required aspects.

## 3.2 Dynamic Optimization system extension

Figure 2 shows the Dynamic Optimization system extension in the context of System 800xA. The framework underlying the Industrial IT System 800xA contains a scalable client-server object oriented database as one of its key components, seen as Aspect Directory in Figure 2. This database is generally used to store configuration data.

The System 800xA provides multiple predefined aspects that cover the basic functionality of a control system, such as control connection, process graphics and history logs of process values. Additional functionality is added through system extensions.

The Dynamic Optimization system extension provides two new aspects: the Model aspect and the Dynamic Optimization solver aspect. The new aspects allow the seamless integration of model-based applications. Moreover a configuration GUI is provided as an add-in for Microsoft Excel, allowing the efficient engineering of model based activities. Last but not least the Dynamic Optimization service manages the instantiation of solver activities in online applications.

The model integration exploits the principle of Modelica to clearly separate model specification and model analysis [10]. For the applications conducted so far, the tool Dymola [3] is used for model editing and translation. The implementation of the executable model is treated as a Simulink S-function [13]. The used solver HQP [6] allows the treatment of different model based activities, including initial value simulation, estimation of model parameters and initial states as well as nonlinear optimal control with constraints on model inputs and outputs.

## 3.3 The Model Aspect

The model aspect of an aspect object provides applications with the necessary information to apply a model

based activity in the context of the aspects object. A model aspect thus augments the aspect object representation of a real-world object with model meta-data. The responsibilities of the model aspect are:

1. Persistent Storage of the model meta-data
2. Exposing a convenient API for the programmatic retrieval and manipulation of the model meta-data and
3. Providing a user interface to allow viewing and manipulating the model meta-data.

The Model aspect does not provide any functionality nor does it deal with implementation details. Instead it references an external implementation. In this way available modeling tools can be applied, such as Dymola, and expensive re-implementation is avoided.

#### 3.3.1 Persistent Storage of the Model Meta-Data

The responsibility of actually storing the model meta data delegated by the model aspect to the Aspect Directory, ensuring qualities like security, redundancy and scalability and providing functionality like backup/restore and import/export of data. The stored data includes:

- Declaration of model variables in categories (Parameter, Input, Output, State, Generic),
- Values for model variables, e.g. for parameters,
- References to process signals, e.g. for inputs and outputs,
- Structural information for hierarchical sub-model structure,
- Reference to the implementation of the model.

#### 3.3.2 APIs for Retrieval and Manipulation of Model Meta-Data

The 800xA framework also defines a generic means of exposing the data of aspect objects: Aspects can make their data available as a set of named properties with values of simple types (String, Real, Boolean, etc). Through these framework defined, generic interfaces to aspect properties, the model data can be made available to generic applications, i.e. non model-based ones. These generic interfaces were specially designed to ease access from many programming environments and languages. As an example, a tool providing import and export of aspect data to and from Excel could use these generic interfaces. Another key component of the Industrial IT System 800xA can make data exposed as aspect properties available for clients using the widely recognized OPC standard for data access.
For the convenience of model based applications, the model meta-data is also made available in a more structured way, using collections of complex types for model variables, their connection to process signals and other model meta-data. These API can be seen as a facade of the underlying, lower-level data structure exposed via the generic interfaces described above. Support for resolving references to process variables especially suited for modeling applications is added on this layer.

#### 3.3.3 User Interface Integration

The framework underlying the Industrial IT System 800xA strongly supports user interface integration of the constituent applications. A consistent look and feel, support for services like drag-and-drop or copy-and-paste and the seamless integration of an applications user interface into workplaces like the Plant Explorer Tool can be easily achieved based on that support as well as role based customization and security of a workplace, i.e. the ability to adapt and restrict the applications user interfaces depending on the role of the current user of the system. Based on this framework features, and analog to the two levels model meta-data API described above, the model aspect provides three views of the model meta-data:
The first one reflects the lower-level data structure as a set of aspect object properties that can be viewed and individually manipulated (if sufficient permission is granted). This generic UI component is not specific to modeling application; it actually ships with the Industrial IT System 800xA Core and is reused by the model aspect to provide users with a well known view of the underlying data.
The second view specially presents information about the model variables in an Excel-like grid. Model variables can be sorted and filtered by their category (input, output, parameter, state or generic) and associated with process variables by drag-and-drop of aspect objects, e.g. from a tree view in a Plant Explorer, and selection of control connection aspects and properties from combo boxes in the grid. Features like undo functionality, sorting and Excel-like auto fill of this variable table is provided by the underlying, 3rd party, grid implementation.
Last but not least the third view embeds an Internet Explorer control that can be configured with an URL.

This view can e.g. be used to launch the modeling application Dymola to view the Modelica model graphically.

#### 3.3.4 Mathematical view on a model

Mathematically, a model has the form of a hybrid differential algebraic equation system (hybrid DAE)

$$\begin{aligned} \mathbf{0} &= \mathbf{F}[\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{m}(t), \mathbf{u}(t), \mathbf{z}(t), \mathbf{y}(t), \mathbf{p}, t], \qquad (1)\\ \mathbf{F} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_m} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_y} \\ &\quad \times \mathbb{R}^{n_p} \times \mathbb{R}^1 \mapsto \mathbb{R}^{n_x}, \\ \mathbf{m}(t) &:= \mathbf{G}[\mathbf{x}(t), \mathbf{m}(t), \mathbf{u}(t), \mathbf{z}(t), \mathbf{y}(t), \mathbf{p}, t], \qquad (2)\\ \mathbf{G} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_m} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_y} \\ &\quad \times \mathbb{R}^{n_p} \times \mathbb{R}^1 \mapsto \mathbb{R}^{n_m}. \end{aligned}$$

Here $\mathbf{x}$ denote continuous-time states, $\mathbf{m}$ are discrete modes, $\mathbf{u}$ and $\mathbf{z}$ are controlled and not-controlled inputs, respectively, $\mathbf{y}$ are outputs and $\mathbf{p}$ are model parameters. Discrete modes are variables that change their values only at discrete time instants, so called event instants $t_e$, see [10].

### 3.4 The Dynamic Optimization solver Aspect

A model can be applied to perform one or more model-based activities. A second aspect, the Dynamic Optimization aspect has been developed to interface a numerical solver, hold the solver configuration, and to exchange data between the solver and the control system. The exchanged data includes: configuration data, current process values (like sensor values and controller set-points), and history logs. Predictions are written back to the control system as history logs with future time stamps. Each aspect is working with its own instance of the numerical solver, allowing multiple model-based activities to run at the same time.
The integrated solver HQP is primarily intended for structured, large-scale nonlinear optimization [6]. It implements a Sequential Quadratic Programming algorithm that treats nonlinear optimization problems with a sequence of linear-quadratic sub-problems. The sub-problems are formed internally by simulating the model and by analyzing sensitivities. They are solved with an interior point method that is especially suited for a high number of inequality constraints, e.g. resulting from the discretization of path constraints. See [7] and [6] for more details about the solver.
Based on the system model (1),(2), several model-based activities can be formulated and solved numerically over a time horizon $[t_0, t_f]$. The treated model based activities include

- Initial value simulation for specified initial states $\mathbf{x}(t_0)$ and model inputs,
- Estimation of model parameters and initial states,
- Nonlinear optimal control with constraints on model inputs and outputs,
- Steady-state simulation, estimation and optimization at one time instant.

An initial-value simulation covers hybrid DAEs (1),(2). However, optimization and estimation problems can currently only be solved for a simplified hybrid DAE $\mathbf{F}$, $\mathbf{G}'$ of the form:

$$\begin{aligned} \mathbf{m}(t) &:= \mathbf{G}'[\mathbf{m}(t), \mathbf{z}(t), t], \qquad (3)\\ \mathbf{G}' &: \mathbb{R}^{n_m} \times \mathbb{R}^{n_z} \times \mathbb{R}^1 \mapsto \mathbb{R}^{n_m}, \end{aligned}$$

where discrete modes do not depend on states or optimized variables.

#### 3.4.1 Simulation Problem

The model behavior is completely determined by the system equations $\mathbf{F}$ and $\mathbf{G}$, if initial states $\mathbf{x}_0 = \mathbf{x}(t_0)$, external inputs $\mathbf{u}(t), \mathbf{z}(t), t \in [t_0, t_f]$, and parameters $\mathbf{p}$ are given. The outputs $\mathbf{y}(t), t \in [t_0, t_f]$ can then be obtained by solving the system of differential equations using initial-value simulation.
However, often some of the required information is not explicitly known, but can be obtained by minimizing a cost function. In many of those cases, a feasible solution can be further specified by constraining model variables. Optimization is a universal tool for treating those inverse problems.

#### 3.4.2 Estimation Problem

An example for an inverse problem is the estimation of unknown parameters $\mathbf{p}$ and/or initial states $\mathbf{x}_0$ based on measured inputs and outputs. The estimation problem can be solved by minimizing a least squares criterion

$$\sum_{i=1}^{n_{\bar{\mathbf{y}}}} \|\mathbf{y}(t_i) - \bar{\mathbf{y}}(t_i)\|^2 \rightarrow \min_{\mathbf{x}_0, \mathbf{p}} \qquad (4)$$

for the set of measurement data $\{\bar{\mathbf{y}}(t_i), t_i \in [t_0, t_f], i = 1, \ldots, n_{\bar{\mathbf{y}}}\}$.

#### 3.4.3 Optimization Problem

The control inputs $\mathbf{u}(t), t \in [t_0, t_f]$ or the initial states $\mathbf{x}_0$ might be free to be chosen so that a criterion

$$F_0[t_f, \mathbf{x}(t_f)] + \int_{t_0}^{t_f} f_0[t, \mathbf{x}(t), \mathbf{u}(t)]\, dt \rightarrow \min_{\mathbf{x}_0, \mathbf{u}(t)}, \qquad (5)$$

$$F_0 : \mathbb{R} \times \mathbb{R}^{n_x} \mapsto \mathbb{R},$$
$$f_0 : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}.$$

is minimized subject to constraints on model inputs $\mathbf{u}_{\min}(t) \leq \mathbf{u}(t) \leq \mathbf{u}_{\max}(t)$ and outputs $\mathbf{y}_{\min}(t) \leq \mathbf{y}(t) \leq \mathbf{y}_{\max}(t),\ t \in [t_0, t_f]$.
Generally it cannot be guaranteed that a solution exists for an optimization problem with output constraints as the model outputs are determined by model states and model inputs. This is why output constraints should be relaxed to soft constraints, augmenting the optimization criterion (5) with penalties for violations. The HQP solver provides support for soft constraints.

#### 3.4.4 Steady-state problem

The dynamic estimation and optimization problems discussed above can also be formulated as steady-state problems at one time instant $t = t_0 = t_f$. The steady-state condition

$$\dot{\mathbf{x}}(t) = \mathbf{0} \tag{6}$$

is formulated as constraint for the HQP optimization solver.

### 3.5 Discrete-Time Optimal Control Problem

Dynamic Optimization and Estimation problems are treated internally as discrete-time optimal control problems, applying multi-stage control vector parameterization. The time horizon $[t_0, t_f]$ is divided into $K$ stages with $t_0 = t^0 < t^1 < \ldots < t^K = t_f$. The controls $\mathbf{u}(t)$ are described in each interval $[t^k, t^{k+1}], k = 0, \ldots, K-1$ as function of the discrete-time input variables $\mathbf{u}^k \in \mathbb{R}^m$. The unknown parameters $\mathbf{p}$ are converted to state variables with the state equation $\dot{\mathbf{p}} = \mathbf{0}$ and with unknown initial values $\mathbf{p}_0 = \mathbf{p}(t_0)$. They are described together with the continuous-time model states $\mathbf{x}(t)$ with the discrete-time state variables $\mathbf{x}^k \in \mathbb{R}^n, n = n_x + n_p$. The state equation (1) is solved for the stage $k$ with the initial values $\mathbf{x}^k$ and the controls $\mathbf{u}^k$ using a numerical integration formula.
This results in the multistage optimization problem:

$$F^K(\mathbf{x}^K) + \sum_k f_0^k(\mathbf{x}^k, \mathbf{u}^k) \rightarrow \min_{\mathbf{u}^k, \mathbf{x}_0}, \tag{7}$$
$$F^K : \mathbb{R}^n \mapsto \mathbb{R}^1,\ f_0^k : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^1$$

with respect to the discrete-time system equations

$$\mathbf{x}^{k+1} = \mathbf{f}^k(\mathbf{x}^k, \mathbf{u}^k), \tag{8}$$
$$\mathbf{f}^k : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$$

and the additional constraints

$$\begin{aligned} \mathbf{c}^k_{\min} &\leq \mathbf{c}^k(\mathbf{x}^k, \mathbf{u}^k) \leq \mathbf{c}^k_{\max}, \\ \mathbf{c}^K_{\min} &\leq \mathbf{c}^K(\mathbf{x}^K) \leq \mathbf{c}^K_{\max}, \\ \mathbf{c}^k &: \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^{m_k}, \mathbf{c}^K : \mathbb{R}^n \mapsto \mathbb{R}^{m_K}. \end{aligned} \tag{9}$$

Note that initial conditions of the system model are formulated as general constraints (9) as well. Discretization formulae, known parameter values, and predetermined disturbances are included into the discrete-time functions $F^K$, $f_0^k$, $\mathbf{f}^k$, $\mathbf{c}^k$, and $\mathbf{c}^K$. The discrete-time functions are assumed to be two times continuously differentiable with respect to their variables.

### 3.6 Large-Scale Nonlinear Programming Problem

Discrete-time optimal control problems can be solved as structured large-scale nonlinear optimization problems. This has the main advantage that powerful methods for large-scale nonlinear optimization can be applied to their efficient solution [11].
The discrete-time control and state variables for all stages $k$ are collected to one large vector of optimization variables

$$\mathbf{v} = \begin{pmatrix} \mathbf{x}^0 \\ \mathbf{u}^0 \\ \mathbf{x}^1 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{x}^{K-1} \\ \mathbf{u}^{K-1} \\ \mathbf{x}^K \end{pmatrix}. \tag{10}$$

One specific feature of the optimization approach discussed here is that the discrete-time state variables at all stages are treated as optimization variables as well, even though they are determined by initial conditions and the control parameters. This leads to a significant increase of the size of the optimization problem. However, the consideration of states as constrained optimization variables generally improves robustness and efficiency of the solution. For instance trajectory constraints can be formulated directly on the discrete-time state variables. Furthermore the separation of the overall problem into multiple stages often leads to a reduction of the required number of nonlinear iterations. The computational overhead is relatively low if the number of state variables $n_x$ is not too high, compared to the number of control variables $n_u$ and if the sparse multistage structure of the large-scale nonlinear optimization problem is exploited appropriately.

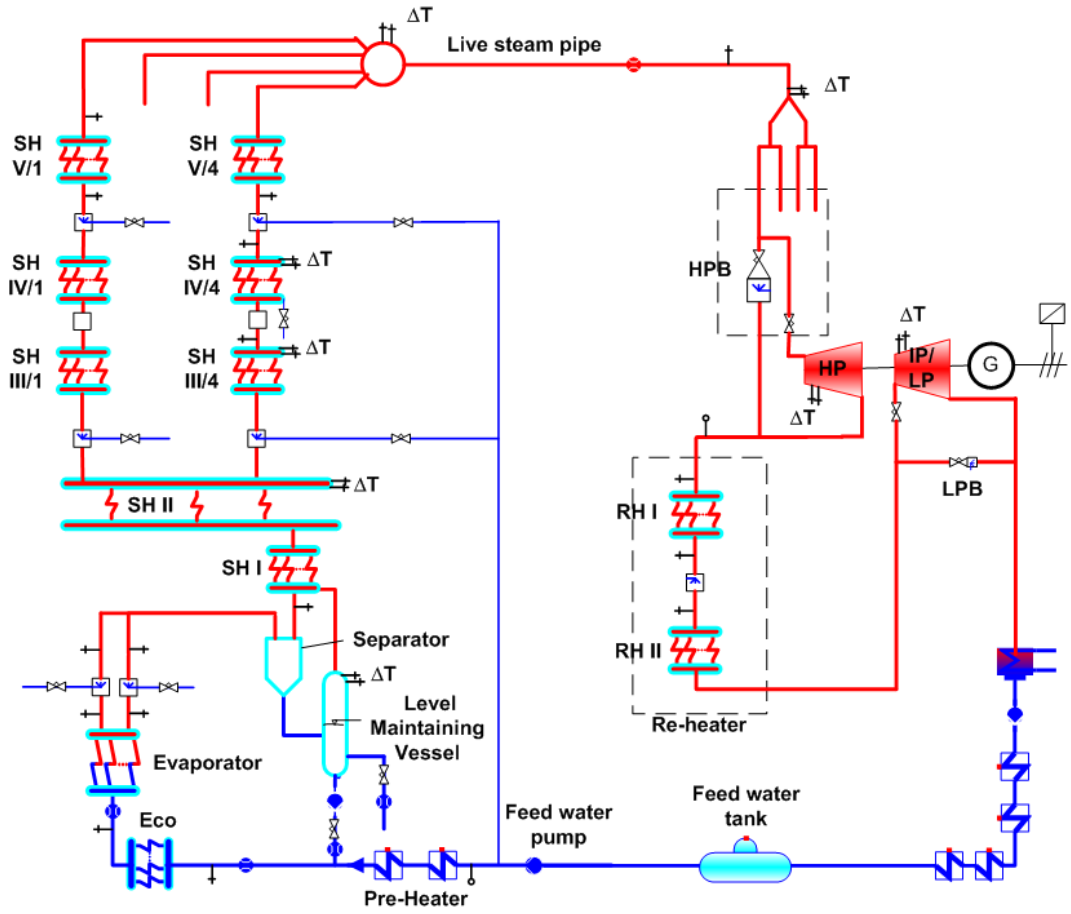


Figure 3: Simplified process diagram of a power plant.

## 4 Application example

A Nonlinear Model-based Predictive Controller (NMPC) for power plant start-up serves as example. The start-up problem is challenging as it is highly nonlinear in the covered large range of operation. Thermal stress occurring in thick walled components needs to be kept in given limits. Multiple manipulated variables have to be coordinated. A long prediction horizon is required to fulfill the constraints during a start-up.

Figure 3 shows a process diagram of a power plant. Feed water enters through pre-heaters and the economizer into the evaporator (lower left side). Saturated steam leaving the evaporator gets super-heated within several super-heater stages (the example diagram shows five super-heater stages and 4 parallel streams in the upper left part). The live steam leaving the boiler goes to the turbine (the example shows 2 turbine sections). There the thermal energy gets transformed to mechanical energy, driving the generator. Afterwards the steam gets condensed and water flows back to the feed water tank (lower right side of the diagram).

During start-up, the boiler first has to produce steam as required for starting the turbine. Within this phase, the steam bypasses the turbine through the high-pressure (HP) and low pressure (LP) bypass valves. The boiler gets heated up by several hundred degrees centigrade. This causes spatial temperature differences in thick walled parts, in particular headers behind the superheaters and spherical fittings in the live steam pipe. Depending on the material properties, the spatial temperature differences cause thermal stress, which again causes fatigue up to destruction. This is why the thermal stress needs to be carefully observed and kept in prescribed limits.

A boiler model was built using the Modelica technology [5]. The model needs to be carefully designed so that is expresses the relationship between optimized control actions (fuel flow rate and valve positions) and constrained process values (pressures, temperatures and thermal stresses). In the example described here, a system of differential-algebraic equations (DAE) with about 1000 variables was built. The Dynamic Opti-

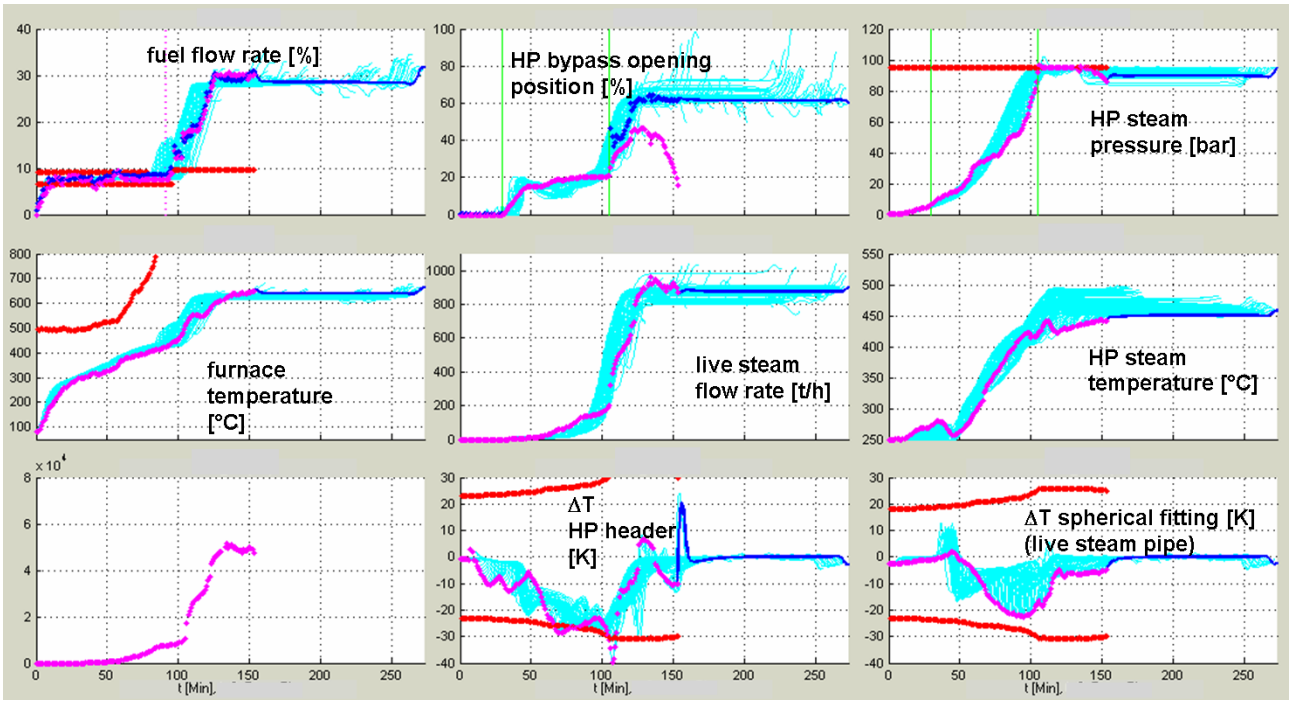


Figure 4: Traditional start-up. The dots show actual process values and limits, the light lines show predictions of process values over 90 minutes that are recalculated every minute. The dark lines show the most recent prediction.

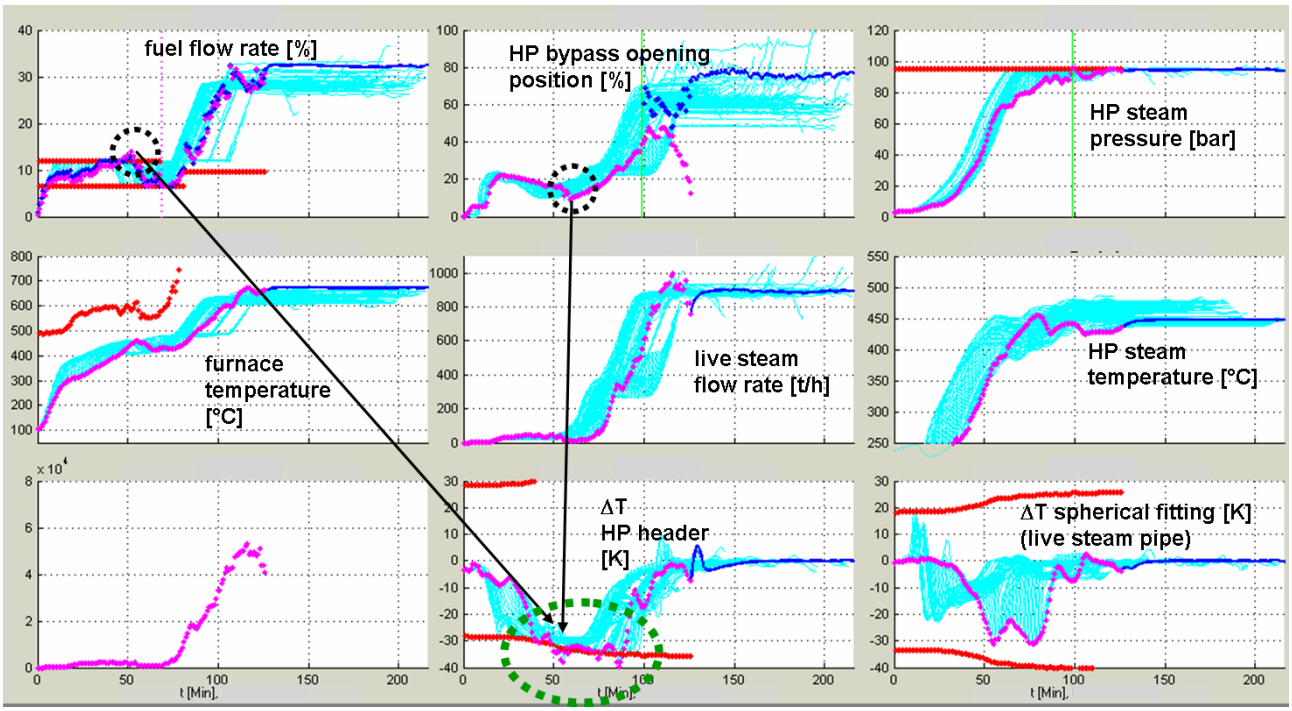


Figure 5: Optimized start-up performed with the NMPC online in closed loop.

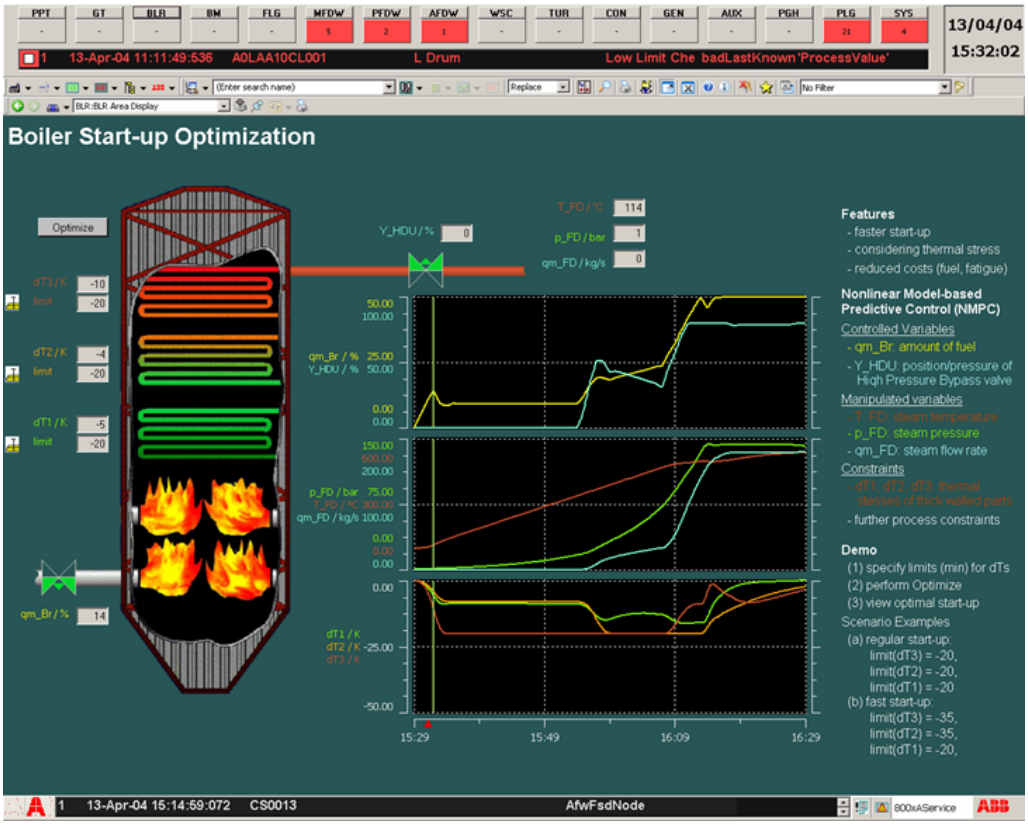


Figure 6: Operator display showing the optimal start-up predicted by the NMPC, in addition to current process values and history logs.

mization aspect system was used offline to identify model parameters based on data logs available for historical start-ups.

Figure 4 shows major process values for a start-up that was performed using well tuned standard control. The plant model was used online, open loop to check its ability to predict the future behavior of the process. The fuel flow rate and the HP bypass position are manipulated variables. The most important process variables are the furnace temperature, live steam pressure, temperature and flow rate, as well as thermal stresses. It can be seen that the allowed limits for thermal stress are not exploited during long time periods on the one hand side and that they exceed the allowed limits at other times (in particular $\Delta T$ HP header). The model was able to predict the behavior of the plant sufficiently well.

During a run of the NMPC, an optimization problem is solved online every minute. The time horizon (prediction and control) is 90 minutes in the example. It gets divided into 90 sample periods. The optimized manipulated variables are parameterized piecewise linear. All other model variables are evaluated at the sample time points. This means that overall about 91000 variables are present in the online optimization problem. The solution time is about five minutes for a cold start of the solver and about 40 seconds for a subsequent solver run.

Figure 5 shows the results of a start-up performed with the NMPC. Due to optimized use of the manipulated variables fuel flow rate and HP bypass position, the constraint on thermal stress of the HP header stays active during almost one hour. After about 50 minutes the fuel flow rate accidentally shot over, resulting in a violation of the thermal stress constraint. It can be seen how the NMPC reacted by immediately throttling the HP bypass valve and by reducing the fuel flow rate. Overall the start-up time could be reduced with the NMPC by about 20 minutes and the start-up costs by about 10% in a 700 MW coal fired power plant.

Figure 6 shows an operator display for boiler start-up optimization. Traditionally an operator display shows current process values and history logs. As a by-product of model predictive control, the operator can additionally see the prediction of the future behavior of the plant. As the NMPC runs integrated with the control system, this display can easily be configured.

# 5 Conclusions

The Modelica technology and the available model libraries allow an efficient modeling of many processes. Nevertheless nowadays the application of the models normally remains restricted to simulation studies conducted offline. A considerable additional effort is required to bring a model online and to deploy a mature model-based application.

The Dynamic Optimization system extension has been developed for the Industrial IT System 800xA by ABB to integrate model-based applications. Exploiting the powerful framework of the System 800xA, the effort for the development of the Dynamic Optimization system extension could be restricted to few additional software components. The new Model aspect exposes model data to the System 800xA. An additional modeling application like Dymola is used to build a Modelica model and to export C-code. The C-code is compiled to a stand-alone executable Dll and loaded by the HQP optimization solver at runtime. The new Dynamic Optimization aspect configures the HQP solver for a specific model based activity and it exchanges data like model parameters, process values and history logs between System 800xA and the HQP solver. The new aspects can be combined with other existing aspects in Aspect Objects. This allows the flexible structuring of complex model-based applications, consisting of multiple models and model-based activities. A configuration GUI has been developed as Excel add-in, which turned out to be a good compromise between development effort and achieved productivity. The Dynamic Optimization service manages the instantiation of solver activities in online applications.

The Dynamic Optimization system extension has been applied so far in a number of different model-based applications. Nonlinear Model-based Predictive Control (NMPC) for the start-up of power plants is discussed in this paper as an example. The overall controller consists of four different model-based activities, including the pre-processing of process signals, the estimation of the model state, the prediction of the optimal start-up, and the post-processing of optimization results. The process models are based on the Modelica.Media and Modelica.Fluid libraries. The scheduling and supervision of the four activities has been implemented in the same framework as additional model-based activity, based on the Modelica.StateGraph library.

After the successful application of the NMPC to the start-up of a 700 MW coal fired power plant, several more start-up optimizations are currently being deployed in gas, oil and coal fired power plants.

# References


[1] ABB Automation Technologies. Industrial IT System 800xA – System Architecture Overview. http://www.abb.com, Document Id: 3BUS092080R0101, 2005.

[2] L.G. Bratthall, R. van der Geest, H. Hoffmann, E. Jellum, Z. Korendo, R. Martinez, M. Orkisz, C. Zeidler, and J. S Andersson. Integrating hundred's of products through one architecture – the Industrial IT architecture. In *International Converence on Software Engineering*. Orlando, Florida, USA, 2002.

[3] Dynasim AB. Dymola: Dynamic Modeling Laboratory. http://www.dynasim.se.

[4] J. Eborn, H. Tummescheit, and K. Prölß. Airconditioning – a Modelica library for dynamic simulation of AC systems. In *Proceedings of the 4th International Modelica Conference*. Modelica Association, Hamburg-Harburg, Germany, March 2005.

[5] H. Elmqvist, H. Tummescheit, and M. Otter. Modeling of thermo-fluid systems – Modelica.Media and Modelica.Fluid. In *Proceedings of the 3rd International Modelica Conference*. Modelica Association, Linköping, Sweden, November 2003.

[6] R. Franke, E. Arnold, and H. Linke. HQP: a solver for nonlinearly constrained large-scale optimization. http://hqp.sourceforge.net.

[7] R. Franke, K. Krüger, and M. Rode. Nonlinear model predictive control for optimized startup of steam boilers. In *GMA-Kongress 2003*. VDI-Verlag, Düsseldorf, 2003. VDI-Berichte Nr. 1756, ISBN 3-18-091756-3.

[8] R. Franke, K. Krüger, and M. Rode. On-line optimization of drum boiler startup. In *Proceedings of the 3rd International Modelica Conference*. Modelica Association, Linköping, Sweden, November 2003.

[9] International Electrotechnical Commission. Industrial systems, installations and equipment and industrial products – structuring principles and reference designations. IEC Standard 61346, 1996.

[10] Modelica Association. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, Version 2.2. http://www.modelica.org, 2005.

[11] Walter Murray. Sequential quadratic programming methods for large-scale problems. *Computational Optimization and Applications*, 7(1):127–142, 1997.

[12] M. Otter, J. Årzén, and A. Schneider. StateGraph – a Modelica library for hierarchical state machines. In *Proceedings of the 4th International Modelica Conference*. Modelica Association, Hamburg-Harburg, Germany, March 2005.

[13] The MathWorks, Inc. Simulink: for model-based and system level design. http://www.mathworks.com.