Technische Universität München
Department of Mechanical Engineering
Institute of Automation and Information Systems
Prof. Dr.-Ing. Birgit Vogel-Heuser

Bachelorarbeit

Dynamic Generation of Modular Industrial Plant Visualizations on a Manufacturing Execution System (MES) Interface

Author:	Miguel Romero Karam
Advisor:	Dr.-Ing. Daniel Schütz
Supervisor:	Emanuel Trunzer, M.Sc.
Begin Date:	15 April 2018
Submission Date:	15 Oktober 2018

Statutory Declaration

I hereby confirm to have written the present dissertation independently and only with the use of the sources and resources I have indicated. Both content and literal content were identified as such. The work has not been available in this or similar form to any other panel of examiners.

Date: _____ Signature: 

Abstract

...

Kurzzusammenfassung

...

Index

1 Introduction

1.1 Overview and Motivation

The ProcAppCom (Process Application Composer) research project behind this bachelor thesis represents a cooperation between multiple industrial partners, namely 3S-Smart Software Solutions GmbH, Gefasoft GmbH, Johann Albrecht Brautechnik GmbH and APE Engineering GmbH with the Technical University of Munich. The main objective of the ProcAppCom research project is the automated configuration and generation of control code and visualizations for production plants in the field of process engineering.

Gefasoft GmbH is a leading and innovative provider of production-related software solutions. With the product Legato Sapient® Gefasoft offers a web-based Manufacturing Execution System (MES) for controlling and monitoring the production process. Main functions of a MES are production management, supervisory control, maintenance management and the real-time data acquisition, storage and integration to other information systems. These include Enterprise Resource Planning (ERP) and Supervisory control and data acquisition (SCADA) systems as well as programmable logic controllers (PLC). A MES thus spans from the operational management level through the process management level (SCADA), the control level (PLC) and to the field layer or shop floor. Key functionalities of a MES for the process engineering industry include the cross-plant evaluation of messages, alarms, process variables and key figures.

Today, the development of control software and visualization interfaces for the operation of smaller process engineering systems is a major cost driver in process engineering automation projects. Additional to the high up-front implementation cost for the connection and configuration of a MES, operational costs rise rapidly during the MES product life cycle; Creating and later modifying plant-specific visualization interfaces represents a significant technical effort, which translates to these continued increments in operational expenditure. The MES software architecture is often deeply intertwined; Adjustments in any area usually have consequences in others, even rather simple modifications can propagate and lead to important sources of errors, imposing constant software adjustments. A slight shop floor modification, be it a physical change in the plant like the disabling of a temperature sensor or a change to the order of production, might result in significant number of adjustments for the MES. Process visualizations in the Graphical User Interface (GUI) are similarly influenced; being virtual

representations of the physical process facility, they demand frequent adjustments which result in significant overhead for its implementation.

Motivation of this bachelor thesis is the development of a system for the automated generation of dynamic Piping and Instrumentation Diagram (P&ID) visualizations for industrial plants with the goal of reducing implementation and operation expenditure for a MES, so that any enterprises can profit from these software solutions.

1.2 Problem Definition

The present trends in automation technology lead to a permanent increase in the complexity of industrial process facilities and to permanent technical changes. These changes propagate through the documentation, maintenance and operation of mentioned facilities, which represents a major engineering challenge. This leads to the need for the frequent and often manual reconfiguration and adjustment of such systems during its life cycle. Plant-specific visualizations in the graphical user interface (GUI) demand significant efforts for their creation and modification upon any technical change. As virtual representations of the process facility and with the imminent increase in changes, they are subject to constant manual updating. Moreover, this constant change leads to deviation from the industry standards for visualizations, like those for P&ID visualizations. As a result, different companies from different sectors end up each with different standards, which further increases the engineering complexity and results in counterproductive GUIs. For these reasons, production software requires adapting to these demanding trends in the process engineering industry. With respect to process visualizations in the process engineering industry, a system must be developed for the automated generation of modular and dynamic plant P&ID visualizations with minimal user configuration and integration to the MES software at hand.

1.3 Initial Situation

The foundations of this project were already set by previous projects in the context of the ProcAppCom research project at Gefasoft. A general description model for process engineering plants was initially developed. Before the start of this project, it was also possible for plant models to be read and directly transcribed to database tables of the MES Legato Sapient®. This enables the automated connection of the MES to the plant’s control and field levels via the factory edge gateways. A system for the automated generation of P&ID visualizations for the user selected site, area, production unit, process cell or equipment module of the modelled process engineering plant is the culmination of this research project. The Legato Graphic Designer boardlet of Legato Sapient® was developed for the dynamic rendering of visualizations in the form of a single, static xml file. In favor of preventing repetition and to seamlessly integrate to the Legato Sapient Environment, existing code should be leveraged as

much as possible. The final product is dashboard for the creation and rendering of the user selected process engineering plant instance.

<INSERT GLOBAL PROCAPPCOM PROJECT OVERVIEW, ASK DANIEL FOR PHOTO>

1.4 Goals of the Bachelor Thesis

The goals of the bachelor thesis give an overview of what is ultimately intended and enabled a plan to be defined initially in terms of the desired outcomes. The specifics for the fulfillment of the goals came later with the definition of technical and conceptual requirements. The following goals were set to define what was to be ultimately achieved by means of the developed solution:

- Reduce technical effort and accelerate development and modification of Piping and Instrumentation Diagrams (P&IDs) visualization generation of industrial processes.
- Design of visualization components according to the industrial standards for generation of consistent P&ID visualizations.
- Prototypal implementation of the software solution in the web-based MES Legato Sapient® in the form of a user-friendly GUI dashboard for the generation and viewing of P&ID visualizations with abstraction of configurations for the user.

1.5 Project Requirements

Specific requirements where set apart from the project goals for the technical and conceptual aspects of the project. It was intended for all requirements to be met by the end of the project, but the development cycle brought slight variations to some during the course of the project.

1.5.1 Technical Requirements

The technical requirements define the intended outcome of the practical implementation of the project at Gefasoft® Gmbh.

TR1: Library of modular P&ID Visualization Components According to Industry Standards

- Object oriented abstraction of industrial process engineering elements into P&ID classes and categories and corresponding data model as a UML 2.0 class diagram.
- Conception of a library of modular and composable P&ID visualization components in SVG format according to the current industry standards.
- Static geometrical definition of visualization components in a P&ID shapes library implemented as a JSON file for ease of personalization and maintainability.
- Automatic propagation of inherited and composed styles throughout library for ease of user personalization.
- Shapes should display real-time plant process values.

TR2: User Friendly Graphical User Interface (GUI) Boardlet for Creation of P&ID Visualizations

- User friendly boardlet with a simple interface for generating new or updating previously generated P&ID visualizations.
- No configurations needed and abstraction of inner workings for the user.
- File input for the selection of the desired version of the P&ID shapes library.
- Buttons for generating the XML of the visualization, for downloading the visualization in both JSON and XML format, and to upload the XML visualization file to Sapient Engine® file system for the Legato Graphic Designer® boardlet to import and render.
- Visual feedback: progress bar for script run and viewer for the generated XML as text.
- Dashboard with boardlets: P&ID Creator, Node Tree Selector, P&ID Viewer.

TR3: Client-Side Script for the Automated P&ID Visualization Generation as an XML File

- Required from user is only the selection of the desired P&ID shapes library version for the visualization and no additional configurations.
- Script encapsulates all required business logic in a single modular and composable, well-documented JavaScript file.
- Separation of primary concerns: presentation logic, database queries, data mappings, graphing algorithm and xml generation are all separate and inter-independent from each other.

TR4: Prototypal Implementation in the Infrastructure of a MES (Legato Sapient®) and Documentation

- Component-driven, modular design of boardlet implemented with the Ember.js framework used in Legato Sapient®.
- Evaluation of the system: functionality, performance and scalability.
- Clear documentation of code.
- Document with next steps in case of interest on continuing development.

1.5.2 Conceptual Requirements

The conceptual requirements define the core concepts which the proposed technical solution addresses and on which the functionality is based. Furthermore, the subject of these concepts represents the research part of the project and the development of new methodologies to achieve the project goals outlined in section 1.4.

CR1: Object-oriented library of statically defined visualization components

- Geometrical definition of shapes as JSON objects with a defined set of properties to take in a predefined value, or a default value if left blank.
- Shape instances inherit object properties from their parent P&ID class and category based on the data model defined as a UML 2.0 class diagram.
- Semi-colon separated string of styles destructured into styles object for targeted configuration of individual styles. Styles object then concatenated back to string on XML generation.

- Library implemented as a single JSON File for efficiency, portability and ease of translation to other formats (for editing of predefined values for example).

CR2: Mapping of Physical Plant Instances to Corresponding Visualization Component

- Mapping to work with minimal changes to the original data model for an unobtrusive implementation of the automated P&ID visualization generator and to avoid the need for new tables and fields in database.
- Minimum database requirement is a *shapeName* attribute to be property set in the model and thus in the database.
- Connections don't require changes in model. The shapeName attribute for each line shape is determined via logic.

CR3: Automatic Type Detection and Simplification of Connections

- Logic for setting the corresponding *shapeName* property to all connections: differentiate between data, process, connection and signal lines. Because of this, no need to specify a *shapeName* for connections in the data model.
- Connections defined in plant model, and thus also in database, in a logical instance to instance way, but suboptimal manner for the application of P&ID line shapes.
- Connections with multiple waypoints simplified by skipping intermediate ports, until a shape to shape connection (from start source to end target) is reached.
- Orthogonal line shapes optimized for minimal crossings and shortest routing between source and target.

CR4: Declarative specification of Graphing Constraints in Form of Tags

- Declarative approach of tags which allow targeting specific shapes to be positioned according to specific set of positioning rules.
- Tags are loosely coupled so they don't intervene with the algorithm, rather define the algorithm to be run.
- Separates vertex placement logic for shapes to be positioned with distinct positioning rules.
- Vertex placement algorithm can be easily progressively enhanced through the addition of more and more tags.

CR5: P&ID Graphing Algorithm

- Research and analysis of state of the art graphing algorithms.
- Simplicity over efficiency of the algorithm as to allow later improvements and since the creation of P&ID visualizations is not time critical.
- Depth-first post-order instance hierarchy traversal to get nodes in graphing order.
- Block-packing algorithm for the positioning of groups in groups to minimize the area.
- Algorithm concept for P&ID visualizations works no matter the complexity of the modelled process engineering plant.
- Ability of progressively enhancing the algorithm for creation of better and more complex visualizations without change in concept.

- Implementation of the algorithm for the example Aida Brewery plant.

CR6: Dynamic Real-Time Display of Process Variables in the P&ID Visualization

- P&ID visualization with real-time updating shapes and shape labels.
- Different types of display of process values depending on data type of process variable and on shape category (for example: Boolean values set fill color for valves, but not for tanks).
- Components encapsulate a uniform set of data bindings to the actual process values and display values in real time.
- Default settings to override labels for shapes with data bindings to empty values.
- One-way data bindings that update automatically on the client-side instead of on the server-side for optimizing performance.
- Data bindings implemented with the sapient-bind property of the shape's XML user-defined object which uses the mxGraph API already (placeholders).

1.6 Composition of the Bachelor Thesis

1.6.1 Project Management

In favor of lightweight and flexible project management, the GIST methodology was preferred over more traditional agile methods. GIST is called after its main building blocks: Goals, Ideas, Sprints and Tasks, each with distinct planning perspective and frequency of change [1]. Instead of initially declaring tasks, goals where set, which enabled a plan to be defined in terms of the desired project outcomes. The goals stated in section 1.4 lead the decisions from beginning to end of the project and where maintained for the most part. Ideas where tracked during the entire project's life cycled and reconsidered for implementation or discarded at the beginning of each sprint. Sprints where executed until all tasks where completed, though tasks of previous and future sprints where sometimes worked upon outside of the corresponding sprint. Tasks themselves where reconsidered weekly for the current sprint and tracked with a Kanban board.

1.6.2 Project Sprints

S1: Research and Choice of Tools and Technologies

Before any work was done, the tools and technologies with which the goals would be achieved had to be at least preliminary decided. Though many diagramming frameworks and libraries exist, not all where optimal for the task at hand, therefore comparisons where done between the available technologies after meticulous analysis of the projects technical and conceptual requirements. The open-source mxGraph JavaScript diagramming library was chosen due to its lightweightedness, robustness between distinct web browsers and compatibility with the

diagramming tool draw.io, built with mxGraph. Furthermore, the Legato Graphic Designer Boardlet in which the visualization is to be view is implemented with the mxGraph library.

S2: Creation of a P&ID Shapes Library

The second project sprint was the conception of an object-oriented library of modular shapes conforming to the industry standards for P&ID visualizations. This task was further divided into subtasks. first of which was the analysis of the mxGraph application programming interface (API), with which the visualizations were to be implemented in the browser. These consisted of the manual creation and analysis of example visualizations as well as a thorough reading to the API's documentation. mxGraph is a fully client-side JavaScript diagramming library that uses SVG and HTML for rendering. The predefined process engineering shape library was used as a base for the next step: the creation of a statically defined, modular and composable object-oriented shapes library for P&ID elements. It was decided that this library was to be defined in JSON format, to facilitate user modification and tuning of the geometries, rather than in XML format like the visualization file itself.

S3: Requirements and Design of Software Architecture

After the creation of the statically defined P&ID shapes library file in JSON format came the conceptual elaboration of a preliminary software architecture for the project's technical implementation in the MES Legato Sapient®. This task preceded the commencement of the agile development life cycle.

S4: Boardlet Design

Aligning to the Legato Sapient® design and coding principles implemented in the component based Ember.js framework, the start of the development phase consisted in setting up boilerplate code for the P&ID Creator Boardlet. After the creation of both a JavaScript (.js) and handlebars (.hbs) templating file, the preliminary wireframe design for the boardlet was made and coded. Attaining to principles of component- based design, the handlebars template was designed and developed modularly with both new and reused ember components. After having the boardlet up and running on the Sapient Engine® it became possible to start the progressive development of the business logic for the automated P&ID visualization generation.

S5: Generation of the XML file of the P&ID Visualization

The first development sprint for the generation of the XML file of the visualization where made before establishing a database connection for fetching of the plant instances on a separate testing boardlet. This testing boardlet allowed for constant modification and experimentation of the algorithms with pure JavaScript, HTML5 and CSS. These allowed for rapid coding without needing to be connected to the full sapient architecture. The plant hierarchy was first modelled statically in form of JSON files in place of the database queries which return equivalent JSON responses. The file input component for the uploading of the P&ID shapes library was recycled to directly load the needed files in the client, thus enabling faster trials and testing of the script in development until XML of unplaced, overlapping vertices was correctly generated.

S6: Connection and Fetching of Plant Instances from Database

After the script successfully and automatically generated an XML file of the P&ID from static JSON files of plant instances, the connection to the database and registering of database tables in the Legato Configuration Center® (LC2) followed. This allowed to test the XML generation script now with actual plant data queried from the database. This required a global data map of all required tables and fields. With the data map, name mappings where done and a function to fetch the data with custom filters implemented via the available Legato getRecords() function. Modification of the database queries could now be done only by modification of the passed parameters for the query. The result of the XML generation algorithm with the actual plant data corresponded now with what was originally modelled for the plant. By now, all vertices and edges where correctly instantiated in the diagram, but vertices where placed one on top of the other. This lead to the start of the graphing algorithm for the placement of these vertices.

S7: P&ID Graphing Algorithm

The main task during this sprint was the development of a vertex placement algorithm to set the x and y properties of each vertex according to a defined set of positioning rules. Both a declarative, rule-based approach and an algorithmic approach for the optimization of area where used. First part of the algorithm consisted in the declaration of constraints in the form of tags based on shape attributes. This way, the positioning logic could be later better targeted at the distinct tags individually, since distinct shapes are to be positioned based on a distinct set of rules. The loosely-coupled tags where specified first and apart from the positioning logic, as this part was based on algorithmic optimization rather than classification. Afterwards, distinct sets of positioning rules where defined for each of the tags. The shapes would thus be iteratively positioned by the algorithm in a distinct and independent way. Furthermore, a block-packing algorithm was to be implemented for the positioning of groups in groups to minimize the area. Though much progress was made in a short time, a standpoint was later reached. Though the algorithm could still be made better, the time invested was too much compared to the progress, and because of the lack of time, the algorithm was left as is in order to continue with the last sprint.

S8: Dynamic Real-Time Display of Process Variables in the P&ID Visualization

Although the output of the P&ID Creator boardlet and the input for the P&ID Viewer is a static XML file of the P&ID Visualization, it must contain the data bindings for the dynamic display of the process variables. The data binding should be independently set based on the data type of the process variable via the sapient-bind attribute in each XML object. This logic had to be set before the string generation so that the distinct data types of the process values result in distinct labels or colors for each shape instance. The Legato Graphic Designer requires the ID (primary key) of the value in the database and automatically fetches the value in the background whenever it changes. This functionality is implemented as a mxGraph placeholders and allows for the data bindings to be also included in the static XML file of the P&ID visualization.

S9: Prototypal Implementation and Evaluation in the Infrastructure of MES Legato Sapient®

Although plant instances where already modelled and available from the database during the project, no connection to the gateway of the example Aida Brewery plant existed. This