Stefan Hennig

Design of Sustainable Solutions for Process Visualization in Industrial Automation with Model-Driven Software Development

Beiträge aus der Automatisierungstechnik

Stefan Hennig

Design of Sustainable Solutions for Process Visualization in Industrial Automation with Model-Driven Software Development

VOGT

Dresden 2012

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Bibliographic Information published by the Deutsche Bibliothek
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibiograpic data is available in the internet at http://dnb.ddb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2012

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „Design of Sustainable Solutions for Process Visualization in Industrial Automation with Model-Driven Software Development“ von Stefan Hennig überein.

Gesetzt vom Autor

ISBN 978-3-938860-49-6

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik
Institut für Automatisierungstechnik

Entwurf nachhaltiger Lösungen zur Prozessvisualisierung in der industriellen Automatisierungstechnik mittels modellgetriebener Softwareentwicklung

Design of Sustainable Solutions for Process Visualization
in Industrial Automation with
Model-Driven Software Development

Stefan Hennig

Der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs
(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. Ralf Lehnert
Gutachter:	Prof. Dr. techn. Klaus Janschek Prof. Dr. Kris Luyten
Tag der Einreichung:	02.03.2012
Tag der Verteidigung:	07.06.2012

Acknowledgments

Successful research has at least three preconditions: (1) a reliable and carefree financial situation, (2) passionate scientific advice, and (3) many enthusiastic supporters. To meet the first requirement, this work was funded as "Landesinnovationspromotion" by the *European Social Fund* and the *Free State of Saxony*. This funding gave me the independence to research freely from conflicts of interests.

Prof. Dr. techn. Klaus Janschek, Managing Director of the Institute of Automation, Technische Universität Dresden, ensured the second requirement for successful research. I would like to thank him for his guidance and for the resources which made this work possible. With a balanced relationship between *encouragement* and *challenge*, he left me all the creative freedom while he ensured a successful conclusion.

Prof. Dr. Kris Luyten, Professor at the Expertise Centre for Digital Media, Universiteit Hasselt, Belgium, invited me to a research stay in his team. During this short time, we implemented several exciting ideas and I was able to learn a lot about the real business of research. Particularly, Kris' passion for science and for motivating support inspired me.

This work was realized as part of the research of the *Teleautomation* working group. It is based on collaborative work and a great number of shared ideas. Many thanks go to the fabulous team forming that group, particularly to the team leader *PD Dr.-Ing. Annerose Braune.* She asked countless critical questions and always had an open mind about unconventional approaches. I would like to thank *Evelina Koycheva*, *Matthias Freund*, and *Henning Hager* for the strong company and all the fruitful discussions. Henning had accompanied my project over several years at the narrowest. No one else has had such deep insight into my work.

The third requirement was met, on one side, by many students. Some parts of this work were greatly influenced by them, other parts were collaboratively developed. In particular, many thanks are devoted to *Nicolas Dingeldey*, *Alexander Witkowski*, *Ying Su*, and *Martin Halfter.* On the other hand, I am very grateful to the secretary of the institute, *Petra Möge*, and to the institute's backbone, *Matthias Werner*, for all the things they arranged for me during the past six years. Furthermore, I want to thank all the other doctorate students of the institute for having such a great time. Particular thanks go to *Thomas Kaden* who helped me a lot with preparing the doctoral viva.

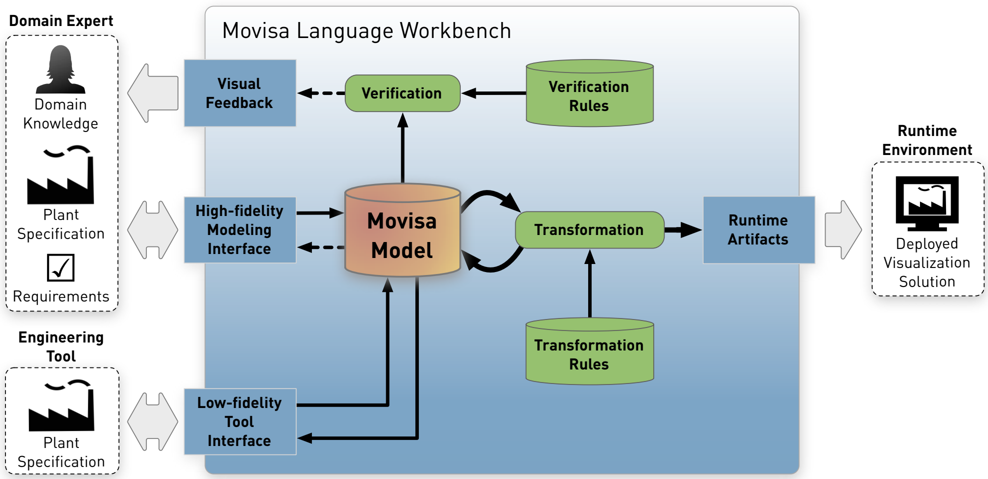
More important than getting technical support, however, is a strong support in “real life”. Hence, I would like to thank my friends for always supporting me during all the years of hard work. Special thanks go to *Maik Biebl* for being so patient with me and for his insatiable interest in my work despite not understanding a single word. Special thanks also go to *Bärbel Wöhlke* for polishing my English.

“What you have started, must also be brought to an end!” As I grew up under this premise, my *parents*, Lorita and Günter, and my *brother* Sebastian have always given me the courage to finish what I had started and the freedom to make my own decisions. I am in great debt to them.

And finally, I thank *Katrin Holinski* for reminding me so often that there is a life beyond technology.

Abstract

Industrial facilities are supervised using dedicated *Supervisory Control and Data Acquisition* (SCADA) applications. These applications, however, suffer from being developed using platform specific terminologies which cause that their operative characteristics are strongly merged with aspects of the technical realization. Platforms executing these applications are characterized by short innovation cycles, thus, decreasing the life time of SCADA applications. Industrial facilities, however, are required to be in operation for decades which possibly requires repeated redevelopment of these applications even if the operative characteristics remain the same. *Model driven* techniques are promising design approaches to foster sustainability of SCADA applications: They separate operative characteristics from their technical realization using *Domain Specific Languages*.

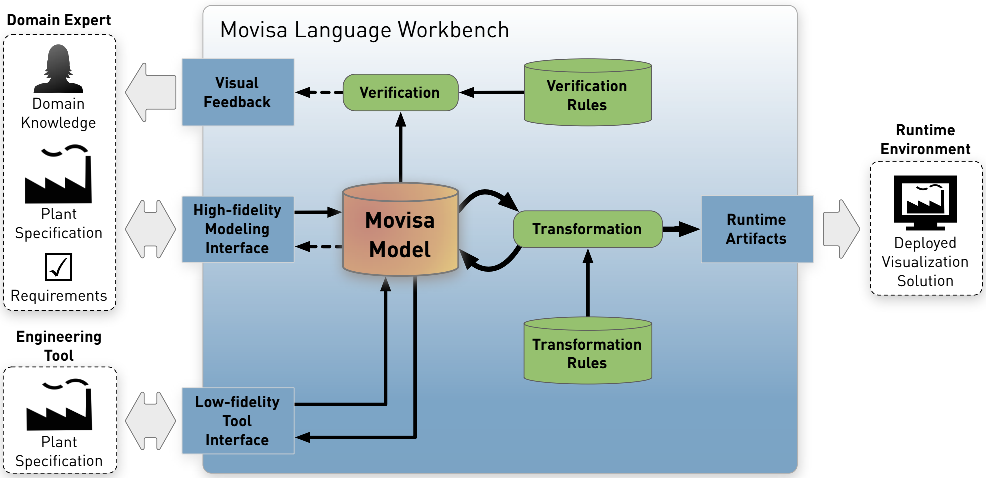


This thesis proposes the domain specific modeling workbench MOVISA. Its core consists of a domain specific modeling language enabling to capture operative characteristics of SCADA applications. For this purpose, it contains building blocks to create user interfaces, process data and communication relationships with automation specific data servers, and it allows to express custom functionality through an Executable UML realization. *Language Constraints* and model-integrity checks allow to identify errors in early design stages and ensure the correctness of models. Transformation rules capture aspects of the technical realization: They allow to process MOVISA models either to modify these models or to automatically create runtime artifacts. In this context, different kinds of transformations are provided in order to support modelers in their assignments and, thus, to reduce the overall development effort. This complexity is encapsulated behind a high-fidelity modeling interface to be exploited by domain experts. It allows to solve problems with a common terminology that is very close to the respective solution space. Furthermore, engineering tools are able to populate MOVISA models via the low-fidelity tool interface.

Case studies from different fields of the domain *production automation* prove the language to be able to describe SCADA applications, thus, meeting related requirements of industrial automation. Sustainability of these applications can be ensured, among others, through automatic transformations, by reusing models and transformations in future projects and through having only one tool to master. The quintessence of this thesis is that even though model driven approaches are challenging with respect to provide effective tool environments, they are very promising means for creating sustainable software designs.

Kurzfassung

Unter *Supervisory Control and Data Acquisition* (SCADA) wird das Überwachen und Bedienen technischer Produktionsprozesse verstanden. Handelsübliche SCADA-Systeme erzeugen Lösungen auf Basis plattformspezifischer Terminologien. Daraus folgt eine enge Verzahnung funktionaler Inhalte mit Aspekten ihrer technischen Realisierung. Plattformen, auf denen SCADA-Lösungen genutzt werden, entstammen zunehmend dem Endverbrauchermarkt und unterliegen damit einer hohen Innovationsrate. Sich ändernde Plattformeigenschaften ziehen selbst bei gleichbleibenden funktionalen Inhalten eine Neuentwicklung dieser Lösungen nach sich. Die Einsatzzeit der SCADA-Lösungen ist somit verglichen mit der der industriellen Anlagen gering. Die modellgetriebene Software-Entwicklung bietet einen vielversprechenden Ansatz zur Erzeugung nachhaltiger SCADA-Lösungen, indem sie funktionale Inhalte von Aspekten ihrer technischen Realisierung auf Basis *Domänenspezifischer Sprachen* erlaubt.



Diese Arbeit schlägt zur Lösung der genannten Problemstellung die domänenspezifische Werkzeugkette Movisa vor. Zentraler Bestandteil ist eine domänenspezifische Modellierungssprache, die funktionale Inhalte von SCADA-Lösungen zu beschreiben vermag. Dazu stellt sie Sprachmittel für die Beschreibung der Benutzungsschnittstelle sowie der Prozessdaten und Kommunikationsbeziehungen zur Verfügung. Aufgrund der Vielfalt technischer Prozesse enthält sie außerdem eine *Executable UML*-Realisierung, um die damit verbundenen Anforderungen zu adressieren. Mittel der Modellverifikation und -integritätsprüfungen ermöglichen die Identifizierung von Fehlern bereits in frühen Entwurfsphasen und garantieren die Korrektheit der Modelle. Transformationsregeln enthalten die Aspekte der technischen Realisierung. Hinsichtlich einer automatischen Erzeugung der Laufzeitartefakte werden diese den MOVISA-Modellen zugeführt. Weitere Transformationen verarbeiten MOVISA-Modelle, um den Modellierer in seinen Aufgaben zu unterstützen und damit den Entwicklungsaufwand zu reduzieren. Die mit diesen Komponenten verbundene Komplexität bleibt dem Modellierer durch einen *high-fidelity* Arbeitsraum im vorgeschlagenen Werkzeug verborgen. Dies ermöglicht das Arbeiten mit einer Terminologie, die sich nah am Lösungsraum des jeweiligen Problems befindet. Über eine *low-fidelity* Schnittstelle erhalten Engineering-Werkzeuge Zugriff auf das MOVISA-Modell.

Fallstudien aus verschiedenen Anwendungsfeldern der Domäne *Produktionsautomatisierung* belegen, dass die vorgeschlagene domänenspezifische Sprache imstande ist, SCADA-Lösungen zu beschreiben. Die Nachhaltigkeit dieser Lösungen ist unter anderem durch automatische Transformationen, durch Wiederverwendung der Modelle und Transformationen in späteren Projekten sowie durch die Pflege nur eines Werkzeugs sichergestellt. Als Quintessenz dieser Arbeit wird festgestellt, dass modellgetriebene Ansätze zur Softwareentwicklung zwar vor dem Hintergrund der Bereitstellung effizienter Werkzeuge herausfordernd sind. Doch zeigen sie sich vielversprechend für den Entwurf nachhaltiger Softwarelösungen.

Contents

List of Figures

List of Tables

List of Listings

Nomenclature

API	Application Programming Interface
ASL	Action Specification Language
ATL	ATLAS Transformation Language
AUI	Abstract User Interface
CAE	Computer Aided Engineering
CAEX	Computer Aided Engineering eXchange
CRF	Cameleon Reference Framework
CUI	Concrete User Interface
DES	Discrete Event System
DSL	Domain Specific Language
EBNF	Extended Backus-Naur Form
EGL	Epsilon Generation Language
EOL	Epsilon Object Language
EVL	Epsilon Validation Language
FCML	Facility Control Markup Language
FDA	U.S. Food and Drug Administration
FLEPR	Flexible Workflow for early User Interface Prototypes
FUI	Final User Interface
FUML	Semantics of a Foundational Subset for Executable UML Models
GUI	Graphical User Interface
HCI	Human Computer Interaction
HMI	Human Machine Interface
HTML	Hypertext Transfer Markup Language
JET	Java Emitter Templates
M2M	Model to Model (Transformation)
M2T	Model to Text (Transformation)
MARIA	Model-based lAnguage foR Interactive Applications
MBUID	Model Based User Interface Development
MDA	Model Driven Architecture
MDSD	Model Driven Software Development
MOF	Meta Object Facility
OAL	Object Action Language

OCL	Object Constraint Language
OMG	Object Management Group
OPC	OLE for Process Control *(Nowadays, OPC is used without referring to an abbreviation, as the importance of the OLE interface decreases.)*
OPC UA	OPC Unified Architecture
PLC	Programmable Logic Controller
PUC	Personal Universal Controller
QVT	Query/Views/Transformations
SCADA	Supervisory Control and Data Acquisition
Scrall	Starr's Concise Relational Action Language
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
T&C	Tasks and Concepts
UCD	User Centered Design
UI	User Interface
UIML	User Interface Markup Language
UML	Unified Modeling Language
UsiXML	User Interface Markup Language
VBA	Visual Basic for Applications
WSDL	Web Services Description Language
XIML	eXtensible Interface Markup Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformation
XUL	XML User Interface Language
XVCML	eXtensible Visualization Components Markup Language

Chapter 1

Introduction

Automation systems consist of technical processes and the required automation equipment [Lunze, 2008]. A technical process is, according to Johannsen (1993), a physical-technical or a chemical-technical procedure with material, energy, and/or information flows at input and output. Johannsen (1993) instanced a procedure on a milling machine with a raw workpiece and electrical energy as input, the processed workpiece, chips, and thermal energy as output. The automation equipment is composed of devices required to transform material and energy. These devices impact on the technical process, e.g. a servo motor moving the milling head.

Human operators are responsible for a safe operation of an automation system [Johannsen, 1993]. Hence, it is monitored and operated by human operators through appropriate *Human Machine Interfaces* (HMI), constituting the connection between the human operator and the automation system: It allows for monitoring the operative states of the automation system by presenting relevant information that have appropriately been prepared. Furthermore, it consists of input devices for entering information. In this way, human operators are enabled to have an impact on the technical process according to its actual state and given goals. Sheridan (1992) makes a distinction between *Human Computer Interaction* (HCI) and *Supervisory Control*: While in HCI one uses computers to operate other computers or databases as end objects, in *Supervisory Control* computers are only mediators between a technical process and human supervision. Sheridan (1992) defines *Supervisory Control* as follows: "[...] one or more human operators are intermittently programming and continually receiving information from a computer that itself closes an autonomous control loop through artificial effectors and sensors to the controlled process or task environment"[1]. Figure 1.1 shows the basic paradigm behind this definition. Appendix A exemplarily presents concrete supervisory control solutions.

[1]Cassandras and Lafortune (1999) introduce *Supervisory Control* in automata theory for a

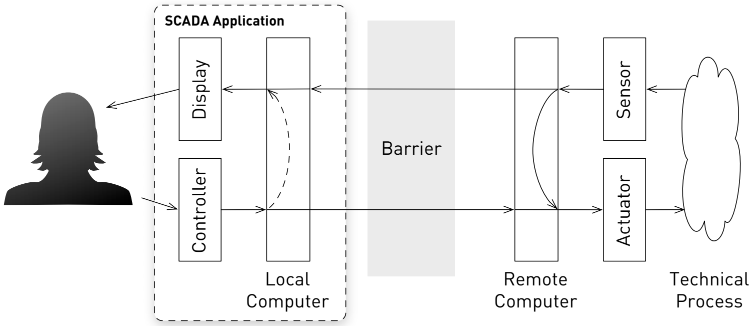


Figure 1.1: Basic Supervisory and Control Paradigm (based on [Sheridan, 1992]).

A remote computer closes a control loop by observing values actually measured by sensors and by setting particular actuators according to these values and given (control) programs. Human operators supervise this process through *graphical displays* representing the technical instrumentation and employ appropriate controllers to modify parameters in the remote programs. Both, the operator and the remote device, might be separated by "a barrier of distance, time, or inconvenience" [Sheridan, 1992]. To implement this general concept, *Programmable Logic Controllers* (PLC) that act independently and close to the process are usually used as remote computers. *Supervisory Control and Data Acquisition* (SCADA) applications constitute the local part depicted in Figure 1.1. Bailey and Wright (2003) refer SCADA "to the combination of telemetry and data acquisition": It encompasses the collection of relevant information, carrying out any necessary analysis and preparing that information to be presented on a number of operator screens or displays. Required control actions are then conveyed back to the process. A SCADA application is, according to Daneels and Salter (1999), a purely software package that is positioned on top of hardware. Thus, SCADA applications form the *Human Machine Interface* for process visualization.

Definition 1.1: *A* Visualization System *is a software tool to develop SCADA applications. A* Visualization Solution *is a particular SCADA application tailored*

given *Discrete Event System* (DES) "whose behavior must be modified by feedback control in order to achieve a given set of specification." This DES is modeled by a graph G with an event set E. If the behavior of G is not satisfactory, it must be controlled by a supervisor S. S observes all events that G executes and then, "S tells G which events in the current active event of G are allowed next."

to supervise a concrete technical process, thus being the User Interface *to this process.*

In the following, Section 1.1 explains the necessity for a new approach to the development of visualization solutions and introduces the solution proposed in this thesis. Section 1.2 provides an overview of structure of this thesis.

1.1 Problem Statement and Aims

Vital requirements for graphical human machine interfaces for process control are among others *efficiency*, *ergonomics*, and that they "shall be so designed as to allow the operator to perform [her] activities in accordance with [her] capabilities, skills and needs, as required to achieve [her] objectives" [VDI/VDE, 2005]. For this purpose, the guideline VDI/VDE (2005) recommends to involve operators into the design phase. Therewith it proposes a *User Centered Design* (UCD) approach, thus entailing a significant amount of the overall system design and development effort. On the other hand, platforms executing SCADA applications in industrial automation are characterized by diversity as illustrated in Figure 1.2.

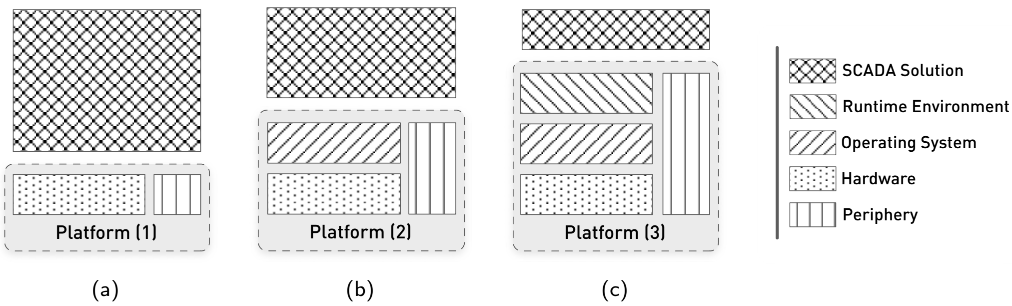


Figure 1.2: Illustration of three SCADA applications operated on different platform configurations: (a) shows an embedded device with a native SCADA realization to be exploited close to the process, (b) presents a mobile version for remote access, and (c) illustrates a classical installation of a SCADA application to be operated on a workstation in control rooms. These platform configurations can also be combined.

Currently, visualization solutions are developed using platform specific tools and terminologies, such as a specific programming language. As a consequence, operative characteristics are strongly merged with aspects of their technical realization. Thus, a human machine interface has to be redeveloped for and tested on each platform using different and probably incompatible tools, even if the operative characteristics remain the same. Moreover, as Menzel et al. (2003) state, the situation is exacerbated by the fact that standard end-user platforms are used almost exclusively. These platforms are characterized by short innovation cycles decreasing the life time of SCADA components to less than five years due to regular releases of new software versions [Menzel et al., 2003]. Given the fact that industrial facilities are required to be in operation for decades and given the complexity of SCADA applications in order to ensure correctness, reliability, and usability, new design approaches are demanded to suit the rapid development of the platforms.

Model driven techniques are promising design approaches, as they express operative characteristics through a platform independent terminology that is based on models. Technical aspects are separated into transformations as carriers of platform specific terminologies. For the given situation, model driven approaches enable to capture the operative characteristics of a SCADA solution by creating a platform independent model. Deploying this solution to a particular platform requires an appropriate transformation that translates the platform independent terminology, provided by the model, into a platform specific terminology. If a compatible platform evolves over time, e.g. through an update of the operating system, causing incompatibility or to equip another or a new platform with this existing SCADA solution, only suitable transformation is required while the respective model remains unchanged. Consequently, operational characteristics need to be expressed and tested for correctness, reliability, and usability only once. Tested and correct transformations are expected to produce always correct platform specific runtime solutions. Additionally, model driven approaches offer the following more general benefits:

(1) Functional aspects can be reused in future projects, even if they aim at another platform.
(2) Technical realizations can be reused in future projects, even if the functionality is a different one.
(3) High-quality and reproducible solutions can be generated through tested transformations.
(4) Various technical solutions can automatically be created from a single functional description, simply by invoking another transformation.
(5) The repertoire of required tools can be slimmed down.

Nichols, Chau, and Myers (2007) have already proven the viability of model driven approaches for the development of user interfaces for office applications. Aquino et al. (2010) additionally proved that model driven development procedures are promising even for automatically deploying user interfaces to different devices. Visualization solutions in industrial environments, however, are characterized by specific requirements and constraints that the aforementioned approaches to model office solutions do not meet, mainly due to the fact that office applications are connected to databases, whereas industrial solutions are connected to technical processes.

> This thesis transfers the aforementioned advantages of model-driven approaches into the domain of industrial *production automation*[2]. It proposes a *Domain Specific Language* which enables the development of sustainable visualization solutions. Other than model-driven approaches for office applications, this modeling languages is tailored to meet the requirements of industrial automation: (1) It defines a sufficient set user interface components with appropriate *representation*, *animation*, and *interaction* properties. (2) It provides a solid abstraction to the variety of industrial automation specific process communication means. (3) It contains an *Executable UML* realization to take into account the diversity of industrial processes and their individual requirements.

This thesis contributes the domain specific modeling workbench Movisa, as depicted in Figure 1.3: Movisa models capture operative characteristics of visualization solutions. Transformation rules capture aspects of the technical realization. The verification tool ensures the correctness of models and the transformation tool processes Movisa models either to modify these models or to automatically create runtime artifacts. This complexity is encapsulated behind a high-fidelity modeling interface to be exploited by domain experts. Engineering tools are able to populate Movisa models through low-fidelity tool interface.

[2] Johannsen (1993) brings the fields of *process industries*, *factory automation*, and *energy supply systems* under the umbrella of production automation together.

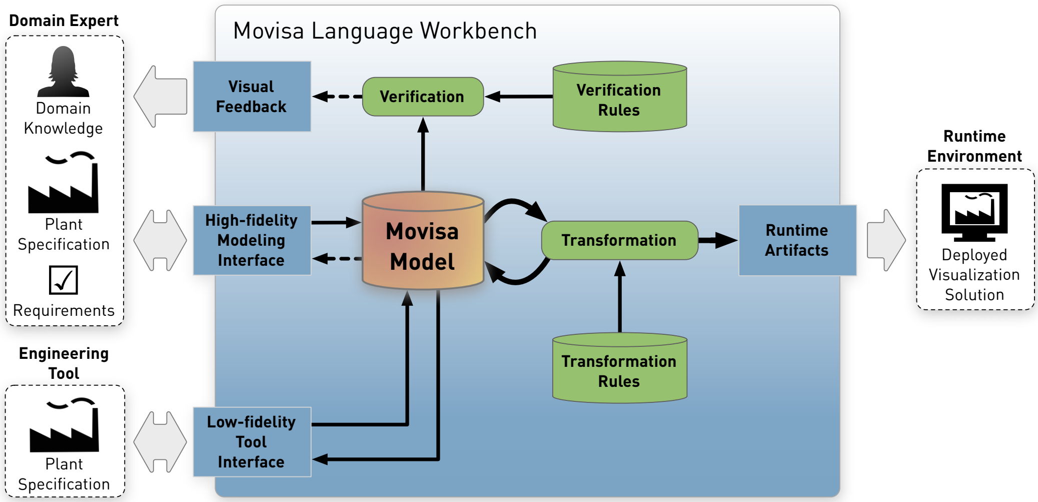


Figure 1.3: Basic concept, architecture, and functionality of the proposed MOVISA modeling workbench.

1.2 Thesis Structure

This thesis is structured as follows:

Chapter 2 gives an overview of the state of the art and its particular background. This chapter concludes with a detailed requirements definition used for pointing out deficiencies of the state of the art approaches and how this thesis contributes to them.

Chapter 3 formalizes the *Language Model* of the *Domain Specific Language* MOVISA by abstracting its *Target Domain* into the *Core Language Model*. Section 3.3 explains *Language Constraints* and Section 3.4 presents the *Language Behavior* definition.

Chapter 4 works out and discusses a concrete syntax notation that enables modelers to work and to think their domain.

Chapter 5 discusses required aspects of a modeling workbench for creating, using, and maintaining models of the language that was created in Chapter 3 which encapsulates the complexity of the model driven approach.

Chapter 6 evaluates the feasibility of the MOVISA modeling workbench by exploiting it on representative case studies.

Chapter 7 elaborates a transformation based framework enabling to incorporate the MOVISA modeling workbench in higher-level engineering procedures.

Chapter 8 draws the conclusions of the findings of the previous chapters.