Design Space Exploration for Building Automation Systems

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Ing. A. Cemal Özlük
geboren am 11. Januar 1977 in Ankara, Türkei

Gutachter:
Prof. Dr.-Ing. habil. Klaus Kabitzsch,
Technische Universität Dresden

Prof. Dr.-Ing. Alexander Fay,
Helmut-Schmidt-Universität der Bundeswehr Hamburg

Tag der Verteidigung: 29.11.2013

Dresden, im Dezember 2013

Abstract

In the building automation domain, there are gaps among various tasks related to design engineering. As a result created system designs must be adapted to the given requirements on system functionality, which is related to increased costs and engineering effort than planned. For this reason standards are prepared to enable a coordination among these tasks by providing guidelines and unified artifacts for the design. Moreover, a huge variety of prefabricated devices offered from different manufacturers on the market for building automation that realize building automation functions by preprogrammed software components. Current methods for design creation do not consider this variety and design solution is limited to product lines of a few manufacturers and expertise of system integrators. Correspondingly, this results in design solutions of a limited quality. Thus, a great optimization potential of the quality of design solutions and coordination of tasks related to design engineering arises. For given design requirements, the existence of a high number of devices that realize required functions leads to a combinatorial explosion of design alternatives at different price and quality levels. Finding optimal design alternatives is a hard problem to which a new solution method is proposed based on heuristical approaches. By integrating problem specific knowledge into algorithms based on heuristics, a promisingly high optimization performance is achieved. Further, optimization algorithms are conceived to consider a set of flexibly defined quality criteria specified by users and achieve system design solutions of high quality. In order to realize this idea, optimization algorithms are proposed in this thesis based on goal-oriented operations that achieve a balanced convergence and exploration behavior for a search in the design space applied in different strategies. Further, a component model is proposed that enables a seamless integration of design engineering tasks according to the related standards and application of optimization algorithms.

Acknowledgments

I am grateful to many people who made this thesis possible. I would like to thank Professor Kabitzsch for his disposition and collaboration, and Professor Fay for motivation. I would like to thank further my parents Elif and Haydar, my sister Dilek and my nephew Ekim for their support. My grandmother Nazik has been at my disposal with her good wishes. My colleagues Bastian, Henrik, Linh, Matthias and Uwe have been there for support, discussions and collaboration. I enjoyed working with them. I would like to thank people of Dresden for being very kind and quite which allowed me to make observations and to concentrate on research ideas without interruption. I would like to thank Baris and Jens for making music with me in different times. I would like to thank International Office of TU Dresden for organizing cultural activities and excursions for PhD students. Thanks to the city of Dresden and its environment for wonderful culture and nature, and for hosting the beautiful river Elbe.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ACO	Ant Colony Optimization
ADT	Abstract Design Template
AUDRAGA	Automated Design of Wireless Sensor Networks for Building Automation Systems
AUTEG	Automated Design of Building Automation Systems
BA	Building Automation
BACnet	Building Automation and Control Networks
BAS	Building Automation System
BPC	Binding Pair Candidate
C	Coverage
CAD	Computer-Aided Design
COM	Component Object Model
COP	Combinatorial Optimization Problem
COTS	Components Off-The-Shelf
CSMA	Carrier Sense Multiple Access
CSV	Component Space Variation
D	Directed Variation
DCOM	Distributed Component Object Model
DDT	Detailed Design Template
EC	Evolutionary Computation
ECJ	A Java Based Evolutionary Computation Research System
ETS	Engineering Tool Software
EVA2	A Java Based Framework for Evolutionary Algorithms
FB	Function Block
FBC	Function Block Combination

GD	Generational Distance
GRASP	Greedy Randomized Adaptive Search
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
IFC	Industry Foundation Classes
ILS	Iterated Local Search
ISO	International Organization for Standardization
JCLEC	A Software System for Evolutionary Computation Research
KNX	Konnex
LNS	LonWorks Control Networking Software
LON	Local Operating Network
MOCell	Cellular Genetic Algorithm for Multi-Objective Optimization
MOCO	Multi-Objective Combinatorial Optimization
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
NSGA2	Non-Dominated Sorting Genetic Algorithm 2
OSI	Open Systems Interconnection
PC	Personal Computer
PLC	Programmable Logic Controller
PP	Problem Property
PSO	Particle Swarm Optimization
R	Random Variation
RA	Room Automation
RF	Radio Frequency
S	Spread
SA	Simulated Annealing
SD	Semi-Directed Variation
SPEA2	Strength Pareto Evolutionary Algorithm 2

SSV	Solution Space Variation
TS	Tabu Search
TSP	Traveling Salesman Problem
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
VLSI	Very-Large-Scale Integration

Chapter 1

Introduction

Modern building automation systems (BASs) are large intelligent networks installed in functional buildings such as office buildings, schools, and hospitals. The principle motivations are reduction of energy consumption and achievement of a high level of comfort. As for being energy efficient BASs contribute to the protection of environment and are employed to control industries such as lighting, shading, heating, ventilation and air conditioning in a building. A BAS installation is related to high costs which are compensated after setting the system in operation after a few years by the amount of saved energy costs.

The installation of building automation systems comprise a chain of individual tasks starting from planning, expanding to system integration and installation followed by commissioning and setting in operation. One of the most important tasks among these is the system integration for its close relation to the installation and operation costs. Quality of a system that will be installed depends on the quality of the components that are planned for the system integration and on the quality of the automation network that comprises these components. Further, system integration does not only have an important impact on the installation and operation costs, but also on the efforts for maintenance of the system in its life-cycle.

In system integration, for given requirements, a system configuration emerges as a software application oriented design of the system with often prefabricated devices. These devices are set in communication relations to form the functional topology of the building automation system. In the market for building automation domain, there exists a large variety of devices from different manufacturers. For the achievement of high quality designs in system integration, this variety must be taken in account which allows a creation of various design alternatives. These alternatives can be compared to each other to identify a design with the best quality, if criteria corresponding to the quality requirements for such a comparison are provided. Hereby, design quality requirements must be fulfilled at a degree as high as possible. By consideration of increasing number of devices available on the market, an exponentially growing number of design alternatives emerge among which optimal designs would like to be obtained. This is not trivial and a problem of a high combinatorial complexity arises. The problem is a multi-objective combinatorial optimization problem with the task for creation of optimized functional topology designs for building automation systems.

The objective of this thesis is to develop a method for the generation of optimized functional topology designs for building automation systems mapping specified requirements onto optimized design alternatives. The developed method takes the design specific characteristics into consideration.

1.1 Background

Automation technologies gain importance increasingly inter alia for saving resources for humans such as energy, time, capital and effort. As the requirements grow, automation systems become always larger and more complex. In order to attain the goal of optimal use of resources, especially in large and complex systems, installation and configuration of such systems, considered as a chain of tasks, must be performed by effective methods. Further, tasks belonging to this chain must also cooperate optimally in accordance with each other.

Modern BASs are composed of distributed intelligent components that do not need central coordinating nodes or computers for function and communication. There exist wireless and wired BASs as well as hybrid system that consist of components from both wireless and wired BA technology. Communication model in wired BAS networks is generally peer to peer and realized over a field bus.

Components are BAS network nodes that are prefabricated automation devices, with embedded software modules of varying functional types. The building automation functions are realized by the employed devices that communicate via network variables implemented on their software modules. The communication relationships among the device software modules are typically specified in a software-based design which is the functional topology of a BAS.

The specification of a BAS design for a building is subdivided into designs for different building structure elements such as room, area, storey. BAS designs are created based on the requirements defined for individual building structure elements. For the elements of the same type (e.g. in an office building, 100 office rooms of an identical floor plan), generally identical requirements are defined.

The principal characteristics of BAS functional topology designs are:

- **component-based design architecture with prefabricated devices**: Designs are application oriented and consist of

  - physical components such as devices and
  - functional components such as software modules.

  The intelligence of devices are provided in the logic embedded in the applications that is represented by algorithms for the realization of BA functions. Applications are provided by the preprogrammed software modules embedded in the devices. A design represents the binding-schema of device-specific software modules which is a functional communication schema of these devices;

- **diversity in function coverage of devices**: The implementations of applications on software modules vary by device portfolios of different manufacturers. Consequently, the distribution of automation functions onto software modules, thus onto devices, can also be different due to the manufacturer specific implementation. Hence, the devices of a manufacturer may provide a different set of functions when compared to devices another manufacturer for serving different user groups to increase the chances of the manufacturer's share in the market;

- **device interoperability**: For valid designs a necessary condition is the interoperability of devices that are communication partners. This requires that the software modules of such devices are bound via compatible data points. These data points allow message passing from the sender to the receiver ensuring that the messages are semantically identical to both communication partners;
- **similar requirement patterns for same type of building structure elements**: Since the requirements exhibit similar patterns for the building structure elements of the same type, also similar design patterns emerge;
- **intermeshed functional connections**: The connections among software modules are highly intermeshed. Since the designs exhibit scalable component architecture, one-to-many and many-to-one bindings emerge frequently among the software modules for the realization of communication of concerned BA functions.

1.2 Motivation

Current methods for the creation of functional topology designs for BAS are not capable of creating optimized designs due to two major shortcomings of these design methods. These are:

- existing device variety of open systems available on the market is not considered, since a system integrator is used to choosing components of a few certain manufacturers and evaluation of device interoperability without tool aid is a complex and time consuming task, and
- criteria for creation of designs are not conceived, that can flexibly be customized.

As a result, suboptimal solutions emerge with the devices of a few manufacturers from a poor variety, instead of a set of competitive alternative solutions. Further, results of different tasks related to design engineering such as requirement elicitation, planning and design creation are often not in accordance with each other. A created design may not fulfill the customer's requirements as specified in the elicitation phase which requires some iterations of these tasks and that is related to much additional effort and extra engineering costs.

This explained situation implies that there is a high optimization potential in the design creation task and for quality of created designs.

1.3 Goals and Use of the Thesis

If the high optimization potential in the design creation task and for quality of created designs (as explained in Section 1.2) is considered, following research question arises:

> **How can optimized functional topology designs be created for building automation systems**?

In the literature, there exist methods that create optimized system designs for various domains of engineering applications. These methods must first be investigated for their suitability for solving the design creation problem. In this regard, these methods must be capable of automatically creating optimized designs considering the presented principal problem characteristics and finding optimized design suggestions for the given design optimality criteria. The hypothesis of this dissertation is:

> **The developed method is suitable for solving the design creation problem and can automatically create optimized functional topology design suggestions for building automation systems in reasonable time.**

The goal of this thesis is to improve the design creation task for BAS and to improve design solutions by developing a method for the automated creation of optimized building automation functional topology designs for given requirement specifications from a large number of design suggestions resulting upon a large variety of design components. This method conceives the creation of detailed system designs based on functional specification of the system in form of an abstract design as illustrated in Figure 1.1. Furthermore, this method must be evaluated using appropriate performance metrics to make a judgment on its performance in relation with consumed resources such as computation time, memory usage, processor speed.

The proposed method addresses on the one hand the problem creation of optimized design suggestions in the presence of potentially large variety of design components, and on the other hand the modeling of the problem as well as the evaluation of created designs. For this scope, the structural characteristics of functional topology of system designs are determined in connection with the competing quality criteria which can guide the design creation method to optimized design suggestions. Further, this method is conceived based on a component model that enables a productive coordination of design engineering tasks according to guidelines provided in standards.

The relevance of the design creation problem is related to the increasing interest and demand of building automation installations. The quality of building automation installations depend on the quality of building automation system designs. Optimized building automation system designs can be achieved by using devices of different manufacturers available in the market. A method for the automated creation of optimized designs can be useful to decrease the installation and maintenance costs. Hence, the automated design creation can be a useful tool for the planers and system integrators.

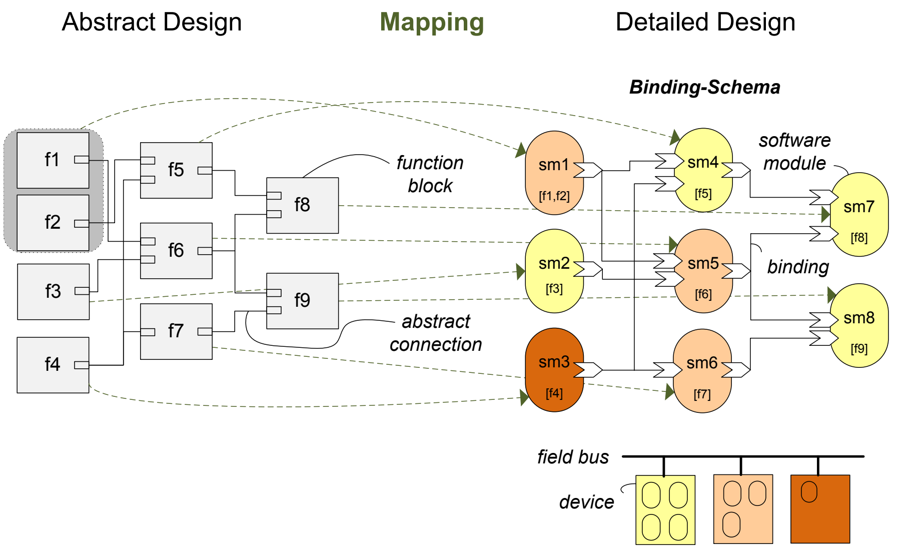


Figure 1.1: Creation of a Detailed Design by Mapping

1.4 Solution Concepts

The proposed method for the creation of optimized designs consists of:

- *representation* of the designs containing design components such as function blocks that depict functions, software modules, devices; relevant properties and attributes of these components for the design creation, and communication relationships among components. This representation is subdivided into two abstraction levels:
  - *abstract design*: a functional model specification of the required BAS created for a given set of requirements which is defined independent from platform or manufacturer details, and
  - *detailed design*: a platform and manufacturer dependent specification of the required BAS, the major outcome of the design creation method;
- *design optimality criteria* which are essential for the quantification for the suitability of the created detailed designs according to different criteria as problem solutions;

- *algorithms* that map the abstract designs onto detailed designs by composing detailed designs using the platform and manufacturer specific components and that search for optimized detailed designs according to the optimality criteria.

Moreover, for the mapping of abstract design onto detailed design alternatives, certain knowledge on the components is necessary. The knowledge such as the identification of components that realize the requirements and various component properties which are necessary for the interconnection of components and the evaluation of created detailed designs are provided by a component repository.

In this thesis, a design creation method is proposed which considers the characteristics of the BAS functional topology designs as introduced in Section 1.1. The abstract designs are specified according to the German standard VDI 3813 [The11a, The11b] by the used abstract design generator. Since the chosen component repository consists of component entries that are conform to this standard, mapping of the abstract design elements (function blocks and abstract connections) onto detailed design elements (software modules, bindings and devices) is possible. In addition, it is also possible to evaluate the mapped detailed design alternatives for various objectives such as validity of the design (mapping of abstract connections onto bindings which requires the interoperability of the devices), device costs, and additional quality criteria.

The design creation problem is a combinatorial optimization problem with multiple objectives. Different design candidates can be put in comparison for these objectives and competitive design solutions are determined using a method introduced by Vilfredo Pareto. Using this method different trade-off solutions for different criteria can be achieved. Thus, by handling the design creation problem with Pareto approach, the problem is identified as a multi-objective combinatorial optimization problem. Since, there exist a very large number of design alternatives; the search for optimized designs is an iterative process. In each iteration, using a population-based approach, new design alternatives are planned to generate by using problem-specific operations, which are better than the design alternatives obtained by algorithms that apply a blind search. Best design alternatives with some diverse solutions are entitled to undergo next iterations and after a number of iterations, optimized trade-off design solutions emerge.

The developed design creation method can be employed by planers or system integrators who can specify standard conform abstract designs. For a given abstract design, a set of detailed design trade-off solutions as detailed design suggestions are created. The resulting set of trade-off solutions may further be limited by the user's definition of priorities for the design creation objectives. The user can consider the best solutions and chooses a solution of his favor. The chosen design solution may be downloaded in the design database of the system as a system configuration.

1.5 Organization of the Thesis

This thesis consists of three parts:

1. an introduction to the design engineering in the BA domain and the design creation problem, a theoretical consideration of the problem and a presentation of candidate algorithms to solve the problem as well as a state-of-the-art analysis for candidate solution methods, and choice of an algorithm;

2. presentation of the proposed solution method for the design creation problem with problem-specific adaptation of the chosen algorithm accompanied with highly BA domain relevant practical considerations; and

3. validation of the proposed solution method by performing an empirical analysis on practical representative problem instances, presentation of results obtained with different algorithms and parameter settings, and presentation of the optimization framework that is conceived and implemented to perform the validation followed by conclusions.

In Chapter 2, 3 and 4 the first part is presented. In Chapter 5 the second part and in Chapter 6, and 7 the third part are presented respectively.

Chapter 2

Design Creation for Building Automation Systems

This chapter provides the background information for the design creation of BAS as the main focus handled in this thesis. A basic introduction of the main frame of the problem in its specific domain is presented including crucial properties of automation systems for the problem.

2.1 Background

Building Automation Systems, also called Building Automation and Control Systems (BACS or BAS) attain goals such as reduction of energy production and consumption, providing maximal comfort, security and a flexible change of use in functional buildings, e.g. office buildings, schools, hospitals, etc. These goals are attained by BAS components that realize automation functions from various industries such as HVAC, lighting, safety alarms, access control, etc. as illustrated in Figure 2.1 coupled to overall system functions for operation, maintenance and management. Buildings equipped with BA technologies are also called *intelligent buildings*. In such buildings systems with focus on user requirements are provided to increase the productivity in the working environment. In the past industries of the building automation were designed and operated as individual tasks performed isolated from each other. Currently, building automation industries can be used integrally due to the development of decentralized communication systems.

Figure 2.2 illustrates the general BAS communication model according to the international standard ISO 16484-2 [Int04] given by the amount of system components present in each level in the automation pyramid for BAS [The99, KHT00]. This model consists of three levels of automation hierarchy distinguished by the components assigned to each level for the type of realized system functions. These functions realize the requirements on various industries[1] such as heating, ventilation, air-conditioning (HVAC) , lighting, blinding, access control, etc. Moreover, the functions are realized using software modules provided on devices such as sensors, automation stations, management computers or elements with interfaces connected to the automation system. A software module or a functional profile [RFK00, BDR00, Int12b] is a software program implementing one or more functions as an element (component) of a BAS and communicates with interconnected software modules via interfaces.

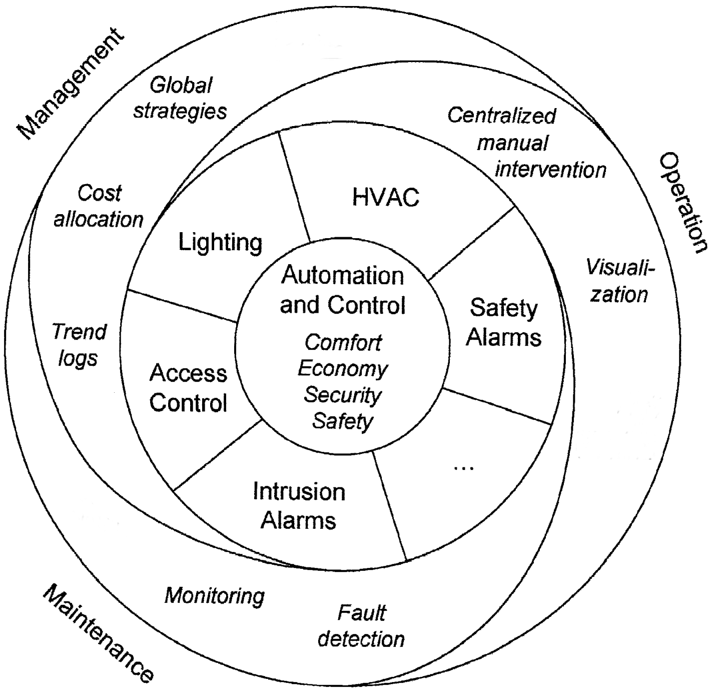


Figure 2.1: Functions in BAS [KNSN05]

In the automation pyramid the field level accommodates components that collect information from the physical environment of the installation location. This information is transformed into a representation according to a specific communication protocol and made available for the system to be transmitted and processed [KNSN05].

The automation level consists of components that autonomously execute the embedded algorithms to locally control the environments in various building structure elements e.g. the whole building, a storey or a room. The necessary information here is provided by communication with the components of the field level via logical connections. Hereby, the transmitted information can also be shared globally such as the measured value for the outdoor temperature.

[1]In the literature industry is also called “aggregate” [DS99] and in the standard VDI 3813-2 [The11b] “trade”.

Information exchange within the same level is called *horizontal communication*. Information from the automation level may be transmitted to the management level e.g. for the storing some process values to perform analysis for e.g. energy management. The information exchange between different levels is called *vertical communication*. The amount of components in the automation pyramid grows from up to bottom, whereas the amount of accumulated information grows from bottom to up, such that at the management level data from the whole system are available. A vertical communication access is performed generally on demand and is related to the information situated in a single node.

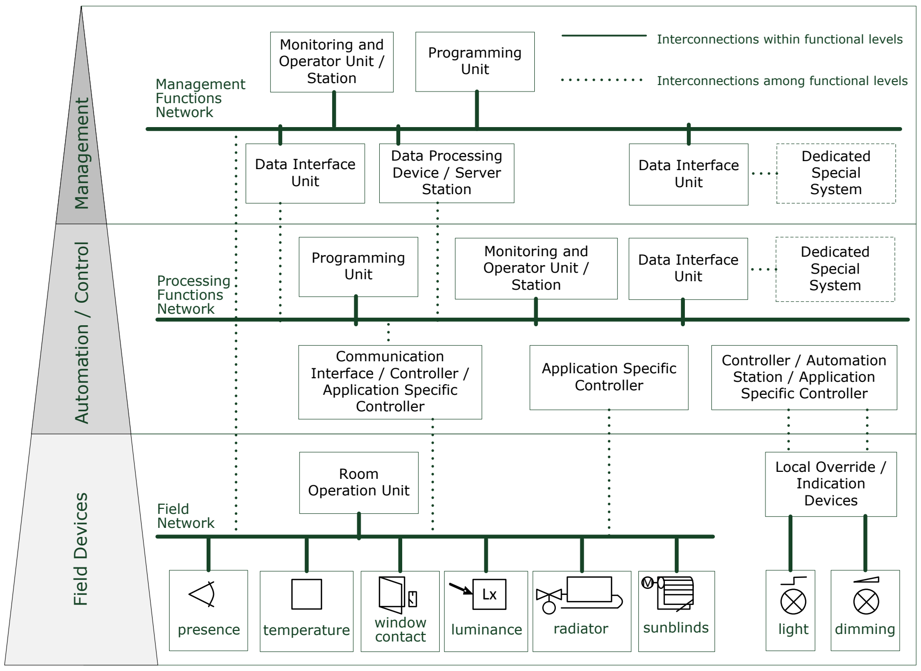

Management
Automation / Control
Field Devices
Interconnections within functional levels
Interconnections among functional levels
Management Functions Network
Monitoring and Operator Unit / Station
Programming Unit
Data Interface Unit
Data Processing Device / Server Station
Data Interface Unit
Dedicated Special System
Processing Functions Network
Programming Unit
Monitoring and Operator Unit / Station
Data Interface Unit
Dedicated Special System
Communication Interface / Controller / Application Specific Controller
Application Specific Controller
Controller / Automation Station / Application Specific Controller
Room Operation Unit
Local Override / Indication Devices
Field Network
presence
temperature
window contact
Lx
luminance
radiator
sunblinds
light
dimming

Figure 2.2: Automation Hierarchy According to ISO 16484-2 [Int04]

At the management level the available data can be used to invoke alarms for critical states in case of system faults. Since the processes at the field level are distributed on many different components with own processors, a simultaneous processing is carried out with advantages of great importance such as an uninterrupted operation of the rest of the system in case, if a failure in a component or a subsystem occurs and maintenance operations must be performed; reduction of the risk of system overloading, and reduction of latencies [KNSN05].

In spite of the introduced three level system model in Figure 2.2, as a consequence of growing complexity of requirements on functionality of devices, many systems in practice show an increasing trend in carrying the functionality and intelligence of the automation and even management level to the field level, and employ field devices with controller functions. Currently, most of the buildings with building automation installations comprise control networks for each floor individually connecting sensors and actuators installed for the rooms.

The application model of a BAS is a logical network of basic data representations called data points. The network of data points reflects the intercommunication of physical processes performed by the devices. Data points are encapsulated in a node, and depending on the realized functionality, several data points of a node are aggregated to a software module to provide high-level system integration. Moreover, this enhances the functional modularity of the system and even very complex applications can be realized as a composition of the software modules. The communication of the data points are realized over the encapsulating nodes that are physically connected to the field bus (cf. Figure 2.3) [KNSN05]. Each data point is related to the realization of a certain application e.g. occupancy detection. Data points are in two special forms: output and input data points. Output data points provide useful data to the input data points via interconnecting *bindings*. By bindings among the data points, thus the system components, the logical communication in the network is performed for the realization of the corresponding applications.

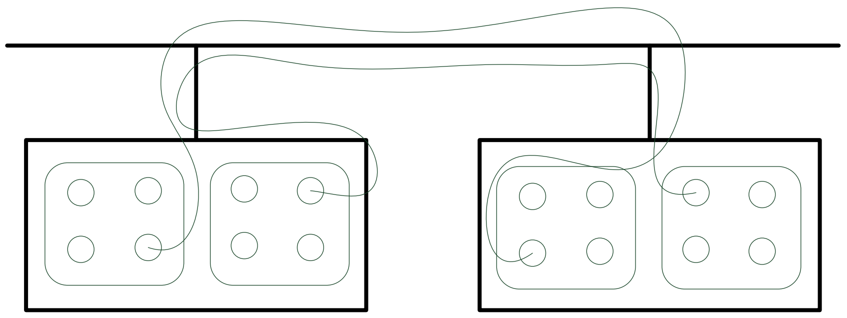


Figure 2.3: Communication of Data Points

Table 2.1 introduces a comparison among different branches of automation systems for aspects such as complexity of system design and efficiency of the design creation. The complexity of an automation system design increases by growing number of employed devices, and requirements on real-time capability and operation safety. The existence of design patterns in an automation domain is particularly of great importance for an effective design creation with focus on design costs per device.

One of the most important characteristics of BAS is that installations are very often for large volumes e.g. high-rise buildings or building complexes, etc. Designs consist of ten thousands of devices [BDR00], which is possible, since a whole system is divided in subsystems installed e.g. for rooms and for different industries. Moreover, requirements on BAS related to real-time and operational safety are lower compared to the process and factory automation, and automotive engineering. In addition, most of the required functionality can be covered by similar design patterns. Time and financial resources provided for the design creation are limited for system integrators. In a BAS similar design patterns can be observed particularly in designs for rooms. Moreover, a significant portion of automation functions in a building is performed in rooms and is called room automation (RA). According to that, design of a whole system is subdivided in the designs for autonomous functional units such as rooms. Designs of similar patterns correspond to a similar set of requirements and are in general potential for saving effort in design creation for a whole project by using inheritance. In practice large buildings of highly complex systems may contain up to 20 different room automation designs [Kra06].

Due to the huge variety of use cases and high requirements for real-time and operation safety, the systems of process and factory automation, and automotive engineering are designed individually.

The size of an office communication network varies often in time, since computers are very often added into or removed from the network. Hence, network configuration is modified, whereas automation networks remain in the initial configuration for years, which is created using design tools.

	Complexity			
	No. Devices	Capacity	Requirements	
	Efficiency			
	Patterns	Costs	Service Life	Databases
Building Automation	$< 2.10^4$	small	medium	
	often	medium	10-30 yrs	yes
Home Automation	< 100	small	low	
	rare	small	5 yrs	yes
Process and	< 1000	small	high	
Factory Automation	rare	large	10-30 yrs	partially
Automotive	< 500	small	very high	
Engineering	rare	medium	10 yrs	generable
Office	< 1000	large	low	
Communication	rare	small	10 yrs	no

Table 2.1: Comparison of Different Branches of Automation Systems for: (upper raws) Number of Devices, Network Capacity, and Requirements for Real-time Capability, Operational Safety; (lower raws) Existence of Design Patterns, Design Costs and Mean Investment for a Device, Service Life Time of Devices, Existence of Databases for Managing Design Information [Plo07]

2.2 Engineering of Building Automation Systems

In the literature there exist various approaches for the engineering of a BAS. In [Sch98] the system integrator can assemble various functions to complex BA applications. In [Fis02] a BAS emerges by following a chain of chronologically ordered tasks, called *engineering*. Various stakeholders are involved in these tasks [PDRK11] initiated by requirement elicitation as explained in details in [Run10, RF11, RFHS10, RDFK08]. In the standard ISO 16484-1 [Int10] for BAS these tasks start from the planning task, followed by implementation planning, and commissioning (setting in operation) as illustrated in Figure 2.4. These tasks include steps inter alia

1. choice of the communication systems;
2. planning of systems for building structure elements such as rooms and creation of functional descriptions;
3. choice of software modules on devices, if available; unless, implementation of new software modules, to realize the functional descriptions and services;
4. design of a system network topology and assignment of logical addresses to devices;
5. specification of communication relationships (bindings) among the system components, setting appropriate operation parameters; and
6. commissioning.

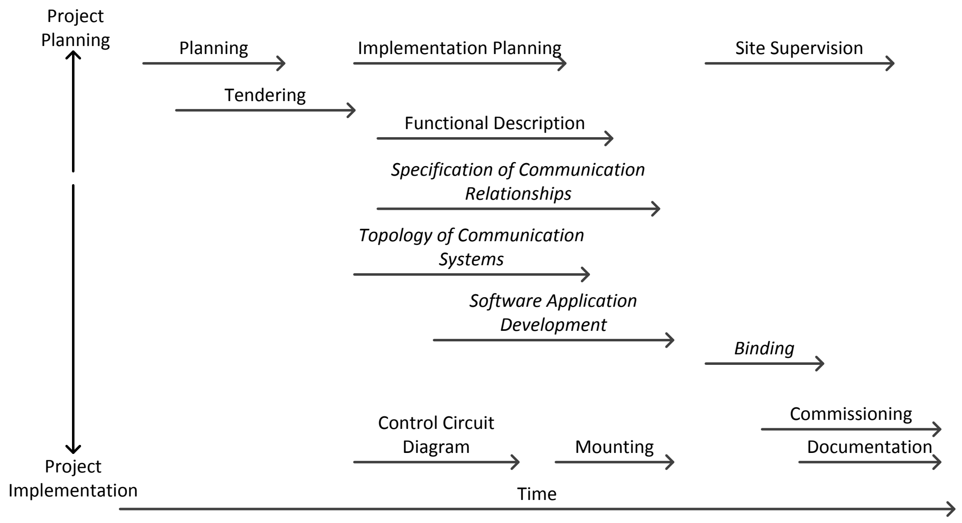


Figure 2.4: Engineering by System Integrators between Planning and Implementation [Fis02]

The tasks directly related to the system configuration are printed in italics in Figure 2.4. System configuration is *software application oriented* and particularly related to component choice or implementation, and component communication.

Configuration of BAS is the focus of this thesis and called *design creation.* Design creation can be performed off-site without need for an existing plant by off-site configuration tools. In this task, scale of the system is presumed, thus realizing devices are selected, applications for devices are developed or existing applications on devices that realize given requirements in the planning task are identified and set in communication relationship. A created design can be downloaded by these tools into the mounted system and devices can be set in operation in a task called commissioning. At this step devices are assigned the previously determined logical addresses, communication relationships and operation parameters.

By a commonly applied installation scenario called the *Engineered System*, the separation of design creation task from commissioning reduces the time spent at the site where the system is mounted. Checking of system operation for correctness and for correct system scaling is thus performed on the mounted system. Particularly, in manufacturer spanning system realizations, the system may not exhibit the desired behavior due to not interoperable function implementations on communication partner devices. This implies that decisions made in the design creation task must be revised. This does not only result in extra financial costs due to retrospective engineering efforts, but also in delay for handover of the system solution.

By additional facts, design creation is a task of a high relevance for a BAS installation: According to [KHT00], the cost for design creation is the highest portion of the total costs of an automation project with 35 percent in its lifecycle. Moreover, problems related to the design creation are according to a study of University of Technology Dresden 70% of the problems encountered in a BAS installation [PV05, KNVT06].

The prefabricated devices of the BAS domain provide various applications that can be activated and configured via device parameters, which brings the advantage that the applications can be reused and application development at the phase of configuration is redundant. Based on this fact device manufacturers develop devices and bring to market as hardware together with embedded applications. This implies that the development of devices and applications (*Domain Engineering*) is independent of the composition of the applications to create designs (*Application Engineering*) as depicted in Figure 2.5. This fact makes a reduction of the time consumed for the configuration per device possible.

Traditionally, industries are designed separately in most buildings and are often realized by separate contractors for different domains, who have employed systems of different manufacturers or even different communication technologies. This is due to the separate historical development of systems from different domains [Fis02]. Hereby, each manufacturer is specialized for a certain application field or industry and most manufacturers cannot provide a complete variety of products that cover all applications that can be required. Moreover, a relatively small number of manufacturers support multiple industries with own product

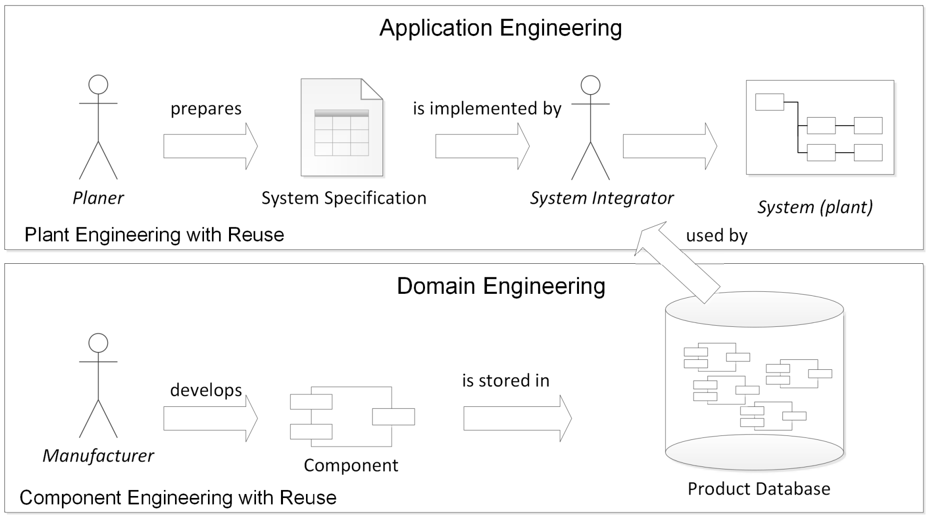


Figure 2.5: Cooperation of Application Engineering and Domain Engineering [Unia]

lines [KNSN05]. Such single manufacturer systems (or homogeneous systems according to [Int04]) are not involved in the interoperability problem among the components; they are however limited to the functionality supported by only the manufacturer's components. As a consequence, only few devices realize each required function, which implies that the number of alternative devices is very limited. On the one hand, this leads to offers for system solutions at relatively high price levels. On the other hand, in case of a system extension due to e.g. modifications in the building structure, the customers are constrained to use additional components of the same manufacturers, which leads to a monopoly of these few manufacturers.

Manufacturer spanning designs are demands of the customer, since by creation of designs with components from different manufacturers, designs of new functionalities can be achieved that can be more suitable than the ones realized by currently known designs. Manufacturer spanning design creation allows furthermore an escape from the monopoly of a certain manufacturer for system extensions or for maintenance. Based on this, it also provides the installed system a significant advantage of being capable of a life time evolution and can decrease the ownership costs [KNSN05, Fis02].

Moreover, a collective use of components such as sensors is possible by a holistic consideration of the system consisting of multiple industries. The main scope hereby is the common usage of components providing information needed by functions which are realized by multiple industries at the same time, instead of designing industries separately by using a single component per function. This approach does not only lead to more economical solutions due to reducing the number of components, but also to use the resources needed for the system operation more efficiently to take an additional advantage of energy-saving potential. An example for a commonly used component is an occupancy sensor that can be used by lighting and heating simultaneously, to turn off the lights and radiator, if the room is unoccupied; and turn them on, if the room is occupied again. By an efficient use of resources, building operation costs can be reduced and new functions can emerge. Realization of this concept with software modules of prefabricated devices is called *innovation by combination* [KDP02, RFK00]. Holistic consideration of industries in the design creation task allows information exchange among the industries which results in a better proportion of uses/costs. In addition, by combining existing applications, new applications can emerge and the variety of applications becomes larger leading to the growth of the market.

For design creation, the interoperability among communicating devices or systems is an essential condition. Systems were proprietary in the past having own protocol implementation of the manufacturers. As a consequence, systems emerged that consist of subsystems from different manufacturers as island solutions that do not cooperate for a common realization of functions. In order to achieve interoperability among devices from different manufacturers, such devices must implement identical communication protocols. This is a minimal and a necessary, however not a sufficient condition for device interoperability. Further conditions for device interoperability are explained in Section 2.5.

2.3 Network Protocols of Building Automation Systems

IEC defined terms from incompatibility to interoperability and interchangeability for allowing devices of systems from different manufacturers to interoperate and cooperate using open network communication protocols.

A *network communication protocol* is an agreement of rules and data formats by which an unambiguous communication of networked devices is possible. The most common network communication model is *OSI Reference Model* [DZ83] which subdivides the communication of implemented open systems in seven layers as presented in Figure 2.6. An overview for the communication protocols of the BAS can be gained in [KNSN05] in which building network protocols for LON platform [Int12a], BACnet [Int12b], KNX [Int06a, Int06b, Int07a, Int07b, Int07c, Int07d, Int07e] and EnOcean [EnO] are compared.

The first three protocols support the transmission media twisted-pair, coaxial cable, power-line, RF, infrared or fiber optics which implement the CSMA medium access mechanism. The openness of all four protocols allows diverse manufacturers to implement own systems. The application and presentation layers are for the design creation of prior importance, since the representation of device application profiles and network variables are conceived in this layer. In functional buildings LON systems are installed more often than KNX sys-

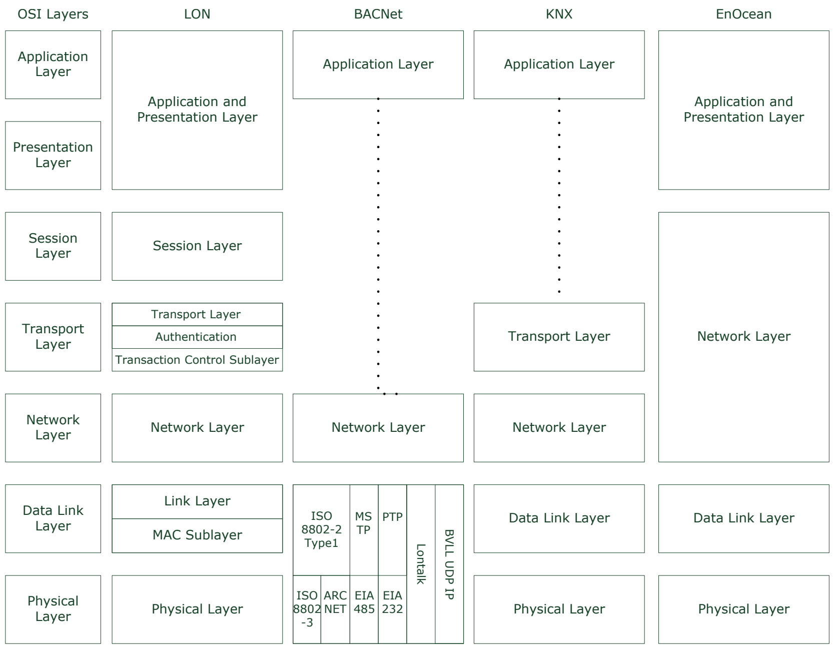


Figure 2.6: Comparison of Various Network Communication Protocols Based on OSI Reference Model [DZ83]

tems and due to the increasing relevance of energy harvesting automation via wireless systems EnOcean systems gain a growing impact.

2.4 Existing Solutions for Design Creation

There exist graphical data flow programming tools that are developed according the standard IEC 1131-3 [Int13] and can implement required functions using a runtime library on the network nodes. This standard is a fundamental guideline for modern graphical PLC programming languages. Currently, the design creation task in the BA domain is supported by system integration tools such as ETS [KNX], LonMaker [Ech], ALEX [Spe], NL220 [New], etc. These tools provide utilities for the creation of a physical and logical network topology, assignment of logical addresses for components, parameterization of devices via device plugins, and composition of applications by selecting software modules provided on the devices and setting them in communication relations via bindings.

For acceleration of design creation in a building project that should consist of ten thousands of network nodes, tool support is provided (e.g. by the tool LonMaker) that allows replication of design schemes for rooms or other building parts (floors, storeys, etc.) with identical automation requirements. Another practical solution for the facilitation of design creation for large projects is provided by NL220 that introduces a pattern library for designs. By the increasing number of projects for which the tool is employed, the number of patterns grows due to the creation of a growing number variants for each initial pattern. Further, an order for the patterns e.g. following the notion of inheritance or evolution of patterns is not followed and the emerging patterns are associated with certain device types [PDRK11] limited to the product lines of a few manufacturers. This fact introduces *the library scaling problem* [Big94] which considerably complicates the management of such a pattern library. A general approach for the library scaling problem is proposed in an early work in [CEG^{+}00]. An approach for requirement engineering of BAS that can generate requirements by using a few generic composition plans that does not consider storing all possible solutions in the library is proposed in [PDRK11].

Currently, many tasks required for the creation of designs are performed by the tool user (the system integrator) manually based on his experience limited to the devices of a few manufacturers. These tasks concern:

- manual choice of devices from manufacturer catalogs that realize the required functions,
- identification of realizing software modules on the chosen devices,
- thereby providing a good distribution of functions onto devices such that the price sum of devices is reasonable,
- determining the input and output data points of the devices (provided on device software modules) that are related to the required application,
- creation of bindings between output and input data points, if implementing devices are interoperable; hereby an evaluation of device interoperability is supported only partially for a chosen pair of output and input data point by comparing the syntax of network variables.

If the existing device variety on the market is considered, the complexity of the design creation task for good quality designs is often higher than a system integrator can manage in a limited amount of time and for limited financial resources. This complexity grows with the growing variety and multiplicity of functional requirements.

Nevertheless, there exist further tools provided by device manufacturers and can automatically perform some of the design creation tasks such as assignment of logical addresses and component composition. These tools, however, create designs with manufacturer-specific product lines and do not support manufacturer and industry-spanning design creation.

Beyond the listed manually performed steps, there are other shortcomings of the existing design creation methods for further aspects considering the design creation in its business process chain:

1. **accordance with the results of the related tasks**: System designs have been created using the information available in the related tasks of engineering without support for common definitions and rules to exchange information. Besides, the results of the individual tasks are used to be exchanged as printed documents. In particular, results of related tasks such as building model made available by the architect and plans for different industries conceived for different locations in a building have solely been handled as such documents, while some of these results have been available as data that can be interpreted by tools (where available) of consequent tasks. Due to the absence of common terms and rules and due to a handling of the results as documents different or even conflicting interpretation of same artifacts by roles involved in the related tasks emerged. This led to a gap between the user requirements and the integration results, for which in turn a cost intensive and elaborate re-engineering has been necessary;

2. **price and quality**: The decisions made in the planning task have often been subjective and the results of planning have been manufacturer specific. Additionally, the industries have been planned separately. As a consequence, designs of different industries are created with multiple fieldbus technologies for which additional amount of cabling have been necessary. Moreover, the potential of using common components among multiple industries has not been used. These two facts have been reason for an added amount of installation costs. Creation of industry spanning designs to obtain economical installations and to better profit from an energy saving potential has not been common.

   Moreover, the variety of devices, thus design solutions have often been limited to the product line of a single manufacturer. Also system integrators have been accustomed to create designs with the devices of a very limited number of manufacturers. This led to a very limited variety of design alternatives, and thus to a very narrow scale for design quality and price sum of devices used in a design;

3. **reusability of the system design solutions**: Due to the lack of tool supported data exchange among the individual tasks as mentioned in 1, associations among the artifacts of the results of the tasks such as planning results and design creation results have not been available in a data model or in a data format. This has resulted in the fact that for similar projects, all tasks had to be performed from the scratch, instead of making the results of previously performed tasks available and reusing them, which have caused high costs and have been highly elaborate for the design engineers. BAS have a high potential for the reuse of existing design creation results, since system designs of the BAS domain often consist of repeatedly used patterns (cf. Section 2.1).

Guidelines that introduce common terms, notions and procedures can allow derivation of new methods to perform the individual tasks of engineering in accordance of all roles involved throughout the life-cycle of a building automation system. Such guidelines can particularly be helpful to eliminate the shortcomings of the existing design creation methods and to provide a basis for coordination among the tasks of engineering.

In University of Technology Dresden in Germany [TU] the research project AUTEG consisting many subprojects was conducted in cooperation with Helmut Schmidt University (University of the Federal Armed Forces Hamburg in Germany) [HSU] that focused on topics such as requirement engineering, planning, intelligent component repository, automated design for building automation systems and control network performance together with partners from the industry. This project was founded by German Federal Ministry of Economics and Technology.

Achieved competency in AUTEG project related to BASs including the specified topics is extended in a follow-up project AUDRAGA that was founded by German Federal Ministry of Education and Research for automated system designs and automated device placement for wireless sensor networks and for hybrid (wired and wireless) systems. Information related to these projects and project partners can be consulted in the web resource [Unia].

2.5 The Device Interoperability Problem

The task of creation of designs with prefabricated devices of multiple manufacturers is confronted with the interoperability problem of the devices that should intercommunicate and cooperate. This requires further properties or agreements beyond the communication standards arranged in profiles for various applications. The properties for device interoperability for communication of two or more devices to achieve a desired functionality are introduced in Table 2.2.

For two devices to be compatible to each other same communication protocol must be used by both devices and they should not interrupt each other's operation. In addition to this, the notion interconnectable requires the use of the same services of the protocol for accessing the variables of the application. If devices are interworkable, they must be able to exchange data and variable definitions must be the same, thus these definitions must syntactically match. Hereby, mapping of variables on the communication system must be the same for both devices. If devices are interoperable, the semantics of the variables and the application functions are the same, so that they can cooperate for one or more distributed applications. Devices are interchangeable, if they possess the same dynamic behavior, which concerns e.g. the update rate of the measured values or the cyclic time of controllers [DS99]. Marco Eichelberg and Klaus Kabitzsch et al. explain further details on device interoperability that can be consulted in [EK10].

	Incompatible	**Compatible**	**Interconnectable**	**Interworkable**	**Interoperable**	**Interchangeable**
Protocol (Layer 1-7)		same	same	same	same	same
Protocol mapping and access to variables			same	same	same	same
Definition of variables				same	same	same
Semantics of variables and application function					same	same
Dynamic behavior						same

Table 2.2: Levels of Device Interoperability [DS99]

2.6 Guidelines for Planning of Room Automation Systems

Besides automation functions in rooms, room spanning or system wide functions such as energy management and optimization functions related to all rooms or to a group of rooms are frequently parts of a room automation system. According to experiences gained in the field of RA, as yet, clients, planers, system integrators have often been unclear about the requirements related to the functions of different tasks of engineering. As a consequence, the concrete realization of the system integrators has been deviating by large from the client's imagination of the system. Clients have frequently been complaining and elaborate reworks have been inevitable (cf. Section 2.4). Approved technical rules for room automation have been hitherto missing; in particular there have been no guidelines to allow the definition of industry spanning and interoperable designs.

Beyond the lack of tool cooperation for different tasks of engineering via data exchange, the major responsible for this situation has been the lack of the definition of useful guidelines for all phases of engineering and operation of BAS, which caused the conception of further standard VDI 3813-1 [The11a] and VDI 3813-2 [The11b] [2]. These standards enable a quality improvement of results, in both planning and execution phases of BAS engineering.

VDI 3813 is the basis for the planned international standard ISO 16484-4:Applications and provides guidelines for RA by focusing on room utilization and room control functions. These guidelines allow:

- the definition of functional subsystems for possible structural elements of a building such as building, floor, corridor, room, etc. according to a shell model as presented in Figure 2.7 with focus on room as a functional

[2]University of Technology Dresden, Germany has made an important contribution to the standard VDI 3813 in cooperation with engineers from the industry.

autonomous system;

- descriptions for common function semantics of a room automation system for all roles involved in the engineering and operation tasks;
- representation for the function semantics and function communication relations in form of a communication schema of function blocks with each function block representing a room automation function with own terms and notions, and connections among function blocks representing communication relations;
- an industry-spanning,
- a technology and manufacturer neutral specification of the room automation functions; and
- the planning of customized systems due to function blocks representing atomic functions.

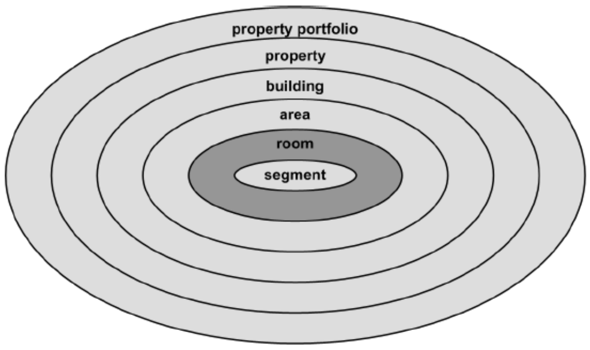


Figure 2.7: Shell Model According to the Standard VDI 3813-1 [The11a]

In [The11b] the definition of function blocks include additionally ports that are represent the data exchanged among components. A port is to be specified with names that are related with names according to conventions implying abstract semantics of the data exchange. This abstract semantics is a useful utility for allowing a specification of software modules of devices that can, as a consequence, interoperate with each other.

Symbols borrowed from the standard VDI 3813 that are used in the examples are presented in Table 2.3. These symbols are used to complete functional schematics of a BAS and do not represent physical end elements in the context of this thesis.

Sensor function block “present detection” in Table 2.3 detects room occupancy by persons automatically and forwards the information that determines either the “present” or the “absent” state to other function blocks, the responses of which depend on room occupancy state. Sensor function block "air temperature measurement" is used to measure air temperature in a room, which is an input information for heating and cooling functions. Sensor function block "brightness measurement (indoor)" is used to measure illuminance in a room and to determine illuminance of daylight. This information is required by lighting and shading functions. Sensor function block "window monitoring" is used to detect the open and close state of a window, which is required particularly in heating and cooling functions [The11b].

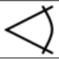
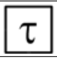
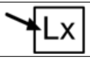
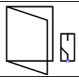
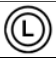
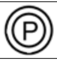
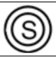
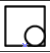
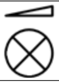
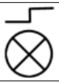
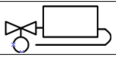
Symbol	Function
	Presence Detection
	Air Temperature Measurement
	Brightness Measurement (Indoor)
	Window Monitoring
	Actuate Light
	Signal Presence
	Select Room Utilization Type
	Adjust Temperature Setpoint
	Light Actuator (Dimming)
	Light Actuator (Switching)
	Control Drive Actuator (Radiator)

Table 2.3: VDI 3813 Symbols Used in the Examples

Operator function blocks "actuate light", "signal presence", "select room utilization type", "adjust temperature setpoint" in Table 2.3 transform a manual user actuation to output information required by application functions as input [The11b].

Actuator function blocks "light actuator (dimming)" and "light actuator (switching)" in Table 2.3 dim and switch lighting equipment respectively as response to input information received from automatic function blocks or from operator function blocks. A light actuator function corresponds to a lighting circuit (also called a pool of light). Actuator function block "control drive actuator" controls valves, air dampers, fans or other control equipment depending on the input from operator or application functions such as air quality control or room temperature control [The11b].

In Figure 2.8 a room automation plan for lighting functions in an office room

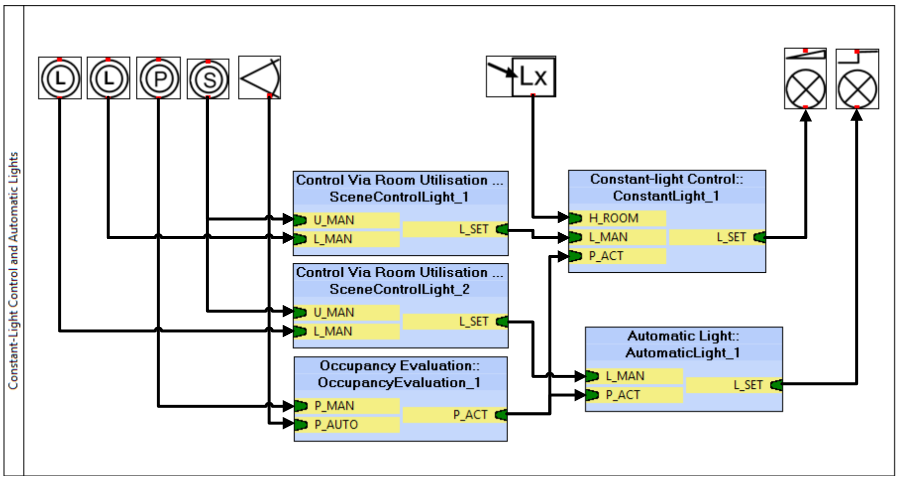


Figure 2.8: Constant-Light Control and Automatic Lights

is illustrated. The plan for lighting consists of two pools of light, in which the first pool provides constant-light control of halogen lamps over the desktop. For comfort and productivity, a constant level of sight in working conditions is provided by dimming the lamps up or down, depending on the changes of the indoor light level by the sunlight shining through the windows or sunblinds. The second pool of light provides the automatic light control of the floor lamp. Both pools need the information about the room occupancy state for the lamps to be switched on or off, which can be detected by an occupancy sensor or toggled by an occupancy push-button. Additionally, the lights are planned to be manually operable by switches or by switching among different scenes via a scene panel.

The automation plan is given in form of a communication schema with manufacturer and BA technology neutral function blocks interconnected via (incoming and outgoing) ports related to the realized functions. Function blocks in an automation plan serve as placeholders for implementing devices of specific platforms and manufacturers, and ports are placeholders for the datapoints of the devices and software modules accordingly. Given a room automation plan, a system design can be created by determining the devices that implement the semantics of the function blocks and ports, and setting them in communication relations via datapoints.

2.7 Quality Requirements on BAS

Quality is defined in ISO 9000 [Int05] as a degree to which a set of inherent characteristics fulfills requirements. The non-fulfillment of a requirement is called non-conformity. Quality requirements on BAS imply the requirements on quality, from which the requirements on the quality of the design creation method can be derived (cf. Figure 2.9). Some of the requirements on a BAS derived from [BBB+04, Spe08, KHT00, MHH07] are:

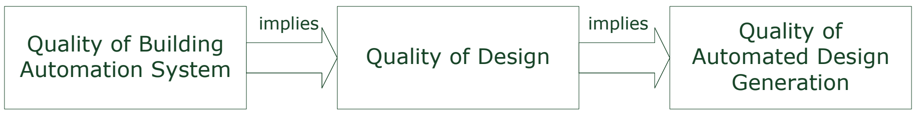

Quality of Building Automation System
implies
Quality of Design
implies
Quality of Automated Design Generation

Figure 2.9: Derivation of Quality Requirements for the Design Generation

- **energy efficiency and costs**: A BAS installation must serve environmental protection. The installed system must be energy efficient and reduce the amount of consumed energy in building, and save costs for building services;
- **comfort**: The comfort of building inhabitants or building users must be guaranteed;
- **safety**: Safe, optimal and a secure operation must be guaranteed;
- **maintenance**: The effort for maintenance must be low and maintenance steps must be schedulable by which a long building life time can be provided. A BAS installation must not be bounded to the product line of a device manufacturer and must be expandable with low effort;
- **acceptance**: A BAS must be accepted by the user for aspects such as usability, ergonomics, aesthetics and installation costs. A comprehensive overview of the operating state must be provided.

2.8 Quality Requirements on Design

This thesis focuses on functional and function based communication aspects of RA designs and this section presents the quality requirements related to these aspects. The requirements on design quality are derived from the requirements on BAS and from the guidelines presented in the standards ISO 16484-1 [Int10] and VDI 3813-1 and VDI 3813-1 [The11a, The11b]. These requirements can be subdivided in requirements related to planning and implementation which are introduced in Section 2.8.1 and 2.8.2 respectively. Hereby, planning refers to the tasks corresponding to planning and project implementation refers to the tasks corresponding implementation illustrated in italics in Figure 2.4.

2.8.1 Quality Requirements Related to Project Planning

A room automation plan reflects the functional requirements specified for a certain room. In the planning task, a complete and correct transformation of functional requirements to room automation plans must be provided. According to this, a room automation plan must meet the following quality requirements, which is essential for the creation of correct and good quality designs:

1. **requirement compliance of planning:** The sum of function blocks in a room automation plan must cover the functional requirements and the specification of each such functional block must be conform to VDI 3813-2 with its ports. For this condition the semantics of specified function blocks and ports must match the semantics of the same elements provided in the standard;

2. **correctness and completeness of function communication relations:** Function blocks must be connected via semantically matching ports. This requires in turn the interoperability of the interconnected function blocks. All communication relations that are necessary and sufficient for the realization of a specific application must be represented as connections among related function blocks.

2.8.2 Quality Requirements Related to Project Implementation

In order to create valid and good quality designs for a given room automation plan which satisfies the quality requirements 1 and 2, additionally, following quality requirements must be met:

3. **requirement compliance of implementation:** A created design must contain devices which implement (or cover) in sum all function blocks specified in a given room automation plan. Each such device must implement software modules that are associated to semantics that match the semantics of the covered function blocks by semantically matching ports to related data points;

4. **correctness and completeness of design:** The devices that cover the interconnecting function blocks must be interoperable to realize the connections among the function block ports by bindings among software module data points for the correctness of a design. This requires in turn correct setting of operation parameters for software modules to be adjusted to the required operation conditions. For the completeness of a design, devices must have all data points connected by bindings that are necessary and sufficient for the realization of the required functions;

5. **design costs:** The price sum of devices used in a design is denoted as design costs for simplification, although further costs for an installation result such as wiring. The design costs have an important impact on the affordability of a design and acceptance by the customers in practice. BAS installations are required to be as economical as possible for which devices of low prices that satisfy the rest of the quality requirements ought to be chosen.

In order to enable a design quality evaluation, the introduced requirements can be quantified. A design can be assigned a degree of fulfillment of the introduced requirements for design quality evaluation. This can be achieved by quantifying the fulfillment of each requirement. Hereby, fulfillment of requirements 1 and 2 is a precondition and is not further considered for a quality evaluation. Moreover, a quality evaluation for individual design components can be conceived to determine components with a low degree of fulfillment of requirements to allow substitutions with better components for improving design quality.

2.9 Quality Requirements on Methods

For achievement of the goals:

- improvement of the design quality and
- development of an improved design creation method

solution methods must fulfill a set of requirements to eliminate the shortcomings of the existing design creation methods (cf. Section 2.4):

- **solution quality:**
  - Creation of solutions that fulfill the *quality requirements* introduced in Section 2.8 at a high degree must be provided. Hereby, definition of additional quality requirements can be desired for the consideration of specific experience or flavor, which must be supported. A large variety of devices from different manufacturers must be considered to allow a wide spectrum of design solution alternatives.
  - It must be possible to *reuse* existing solutions for repeatedly used planning results, which can be the case within the same or different installation projects.
  - *Traceability* of the design creation solutions must also be provided to identify the components of the created designs and guarantee the consistency between specific planning results and the created design results;
- **validity:** The solution method must not only provide good quality results for a specific room automation plan, but also for a variety of plans for allowing statements about its validity and applicability;
- **automation:** By growing size of room automation plans, design creation task becomes more complex, such that a very elaborate and cost intensive work becomes inevitable. Considered a large device variety for better quality solutions, this complexity grows much faster for which an automation

of the design creation task becomes inevitable. This implies the need for conception and implementation of convenient algorithms for which computer interpretable data must be provided;

- **efficiency:** Good quality designs must be created quickly in a limited amount of time. The time required for the design creation method should not be longer than the time spent by the existing manual design creation methods. Design creation should take no longer than 10 minutes per device according to [PDRK11];
- **comprehensibility and reusability:** The evaluation and modification of designs must be possible without knowledge on implementation. Similarly, a flexible extension of quality criteria must also be possible. Furthermore, for ease of use an easy integration of existing design solutions into the building structure elements with similar requirements must be provided;
- **integration:** A seamless integration of the solution method in the business process chain for design engineering must be provided for allowing consistency and a constructive coordination.

2.10 Conclusions

In this chapter engineering for building automation systems is presented. A comparison of various existing communication protocols for building automation systems is given. Further, a state-of-the- art analysis is made for existing solutions for design engineering in the building automation domain. As one of the most important problems concerned in the design engineering, the device interoperability problem is introduced with levels of device interoperability. In addition, existing standards related to planning and implementation of building automation system projects are presented.

Quality requirements on building automation systems are derived with special focus on system design. Since the system design as a task involves project planning and implementation, quality requirements for the tasks of project planning and implementation are derived. Based on these requirements, quality requirements on design engineering methods are identified. A formal definition of the problem and characteristic problem properties, as well as requirements on algorithms are presented in Chapter 3.

Before the approaches proposed in this thesis concepts, methods that rely on using design templates or rule-based approaches have been followed. However, by using optimization algorithms high quality solutions can be obtained according to the optimization criteria which can be flexibly defined by users. Moreover, the design creation task can also be optimized. For the solution of the problem, a solution method including optimization algorithms are used that must be adapted to this specific problem. For the realization

- objective functions and a representation of the problem must be defined;
- approaches for the generation of good initial solutions must be proposed including design generation strategies and
- algorithms that effectively search for high quality design solutions must be proposed.

Following a solution method using optimization algorithms arises some important research questions:

- Which optimization algorithms are best candidates to solve the design creation problem?
- How do the used optimization algorithms converge?
- How is the performance for these algorithms and how can the performance of the algorithms be measured?
- What is the resource consumption (amount of computation time, speed of processor, required memory amount) of the optimization algorithms?

Some early contributions made to application of optimization algorithms related to problems in engineering practice were made in [Kab87]. Answers to these research questions and the solution method followed are presented in the remainder of the thesis.

Chapter 3

The Design Creation Task

3.1 Introduction

The guidelines in VDI 3813 [The11a, The11b] describe industry spanning and manufacturer neutral room automation plans which allow manufacturer and industry spanning design creation with purpose on achievement of good quality designs. There exists yet no method that can automatically create quality oriented designs from room automation plans. An integration of such a design creation method into the engineering process according to the requirement for integration in Section 2.9 can be achieved and designs with improved quality can be obtained by system integrators or yet by planners using the approaches that will be presented in this chapter. Based on this, a new approach for engineering of building automation systems is proposed (cf. Figure 3.1).

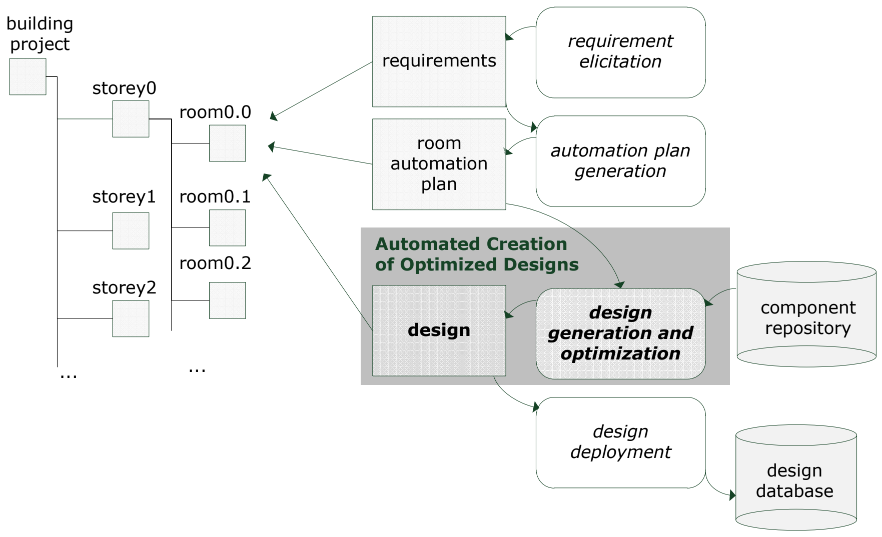


Figure 3.1: Overview of the New Engineering Approach

Available upstream information such as building structure, electrical planning can be made available in an open and CAD tool neutral building information data such as in the IFC data format [Bui] that can be transferred among many building information modeling software tools used to plan different industries by import and export facilities. This information can be used to make preliminary decisions as feasibility constraints for including or excluding some requirements prior to the requirement elicitation. Consequently, the planner can perform dialogs with the customer for the elicitation of requirements in consideration of the preliminary decisions.

Given the elicited requirements ordered in the building structure, room automation plans can be created conform to the definition of abstract function blocks in the standard VDI 3813-2 [The11b]. The availability of a computer interpretable data provided by a suitable representation of requirements allows the generation of room automation plans by appropriate algorithms. The method in [RDK09] that uses the approach of Generative Programming can automatically generate automation plans in a given building structure and for specified requirements, called *automation plan generation*. Hereby, the generated automation plans are available as data where for function blocks the underlying semantics are specified.

The semantics used in the automation plan generation are in accordance with the domain specific semantics defined in a *component repository* [DK11]. In particular, common representation and definition of component semantics are used by the automation plan generation for function blocks, and by the component repository for the devices and software modules. Thus, for a given function block, a component search can be performed for devices and software modules of the desired platforms such as LON, BACNET, KNX and ENOCEAN [EnO] with matching functional and hardware semantics of the components. The search results can be retrieved as computer interpretable data. The component repository allows in addition the evaluation of interoperability among devices, and delivers necessary information for the creation of bindings among the device software modules likewise as computer interpretable data for further processing.

The introduced requirements on solution methods in Section 2.9 can be fulfilled by an efficient method for automated creation of optimized designs that can create designs of quantifiable quality for given room automation plans by generation using computer aid and by performing searches in component repository for interoperable devices that match the specified function blocks. The method for automated creation of optimized designs in Chapter 5 is the main focus of this thesis and aims at automating the engineering task whilst creation of optimized design solution. This method conceives the system design composition model, and strategies and algorithms for design generation and optimization.

3.2 System Design Composition Model

The design generation method conceives the use of components and functions provided by the component repository for the creation of designs that realize the required room automation plans. Beyond the computer interpretable data provided, a system design composition model allows the processing of data relevant for the design generation and matching data from automation plan and component repository used by the algorithms.

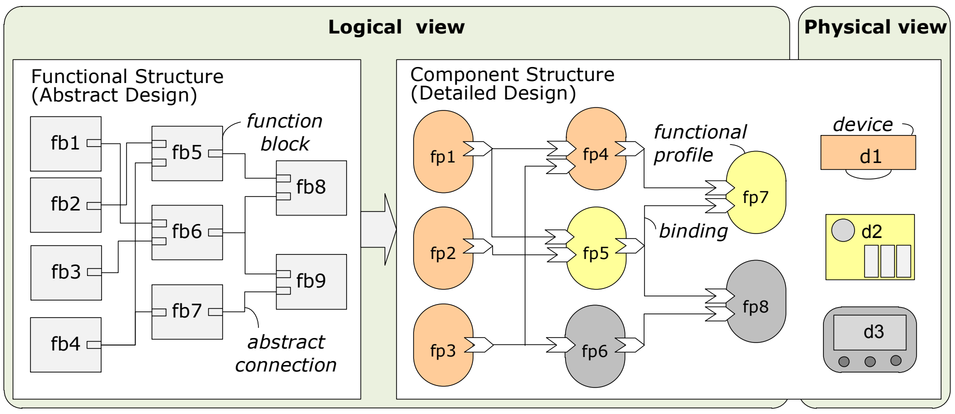


Figure 3.2: Logical and Physical View of System Design Composition Model

The system design composition model can be reflected in logical and physical view (cf. Figure 3.2) to provide an overview for the software and hardware components, as well as the concepts of project planning and project implementation. There exists a component hierarchy among devices, functional profiles and function blocks as can be seen in Figure 3.3. Function blocks are realized by functional profiles and functional profiles are software implementations for BA functions provided on prefabricated devices.

3.2.1 Abstract and Detailed Design Model

Each design is semantically associated to the originating room automation plan. Designs and room automation plans are similar for a consideration of structure. Despite the similarity between a room automation plan and a design realization, there is a significant amount of difference of information detail contained. Based on this reason a room automation plan is called an *abstract design* and the design is called the *detailed design* (cf. Figure 3.2).

Definition 3.2.1. Logical Topology Structure: The logical topology structure of the integrated design approach concerns two design models, each at different level of abstraction: abstract design and detailed design. It is convenient to define relations among the components of the system design composition

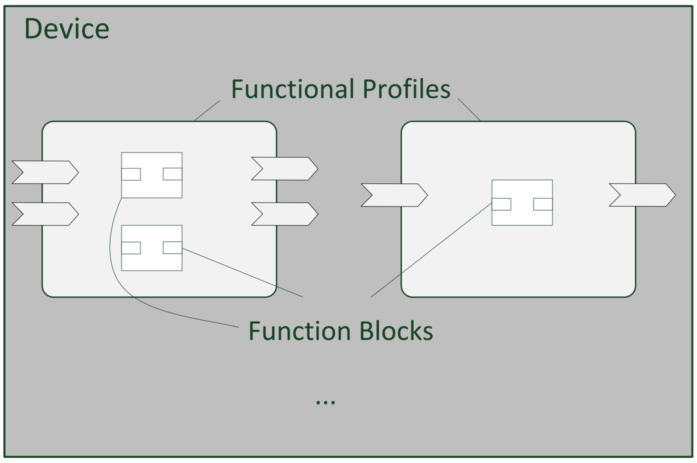


Figure 3.3: Component Hierarchy

model referring to the set theory, to help the identification of problem class in Section 3.3.

Definition 3.2.2. Abstract Design: An abstract design (AD) is a functional communication schema conform to standard VDI 3813-2 representative for all rooms for which the automation function requirements and their communication relationships are identical and it does not represent the automation for a specific room necessarily. Hence, it is a function block-based schema that allows a neutral (technology and manufacturer independent) formalized specification of functional requirements. An abstract design is a set $AD = \{ FB \cup AC \}$ that contains a set of function blocks $FB = \{ fb_1, ..., fb_m \}$ and a set of abstract connections $AC = \{ ac_1, ..., ac_n \}$ with $m \in \mathbb{N}$ denoting the number of functions blocks and $n \in \mathbb{N}$ denoting the number of abstract connections. Each function block $fb_i \in FB$ $(1 \leq i \leq m)$ possesses a set of inports $INPORT = \{ inport_1, ..., inport_o \}$ and a set of outports $OUTPORT = \{ outport_1, ..., outport_p \}$ $(p, o \in \mathbb{N})$. An outport $outport_x$ $(1 \leq x \leq p)$ of a function block fb_i can be connected to an inport of another function block fb_j $(1 \leq j \leq m)$ by an abstract connection in AC.

Definition 3.2.3. Detailed Design: A detailed design DD represents the component communication schema that is created for the realization of an abstract design. The components are functional profiles and their implementing devices, which are manufacturer and technology specific. A DD contains a binding-schema BS corresponding to the logical view and a set of devices $DEV = \{ dev_1, ..., dev_\sigma \}$ $(\sigma \in \mathbb{N})$ corresponding to the physical view (cf. Figure 3.2).

A binding-schema is a set $BS = \{ FP \cup BN \}$ that contains a set of functional profiles $FP = \{ fp_1, ..., fp_r \}$ and a set of bindings $BN = \{ bn_1, ..., bn_q \}$

with $r \in \mathbb{N}$ denoting the number of functional profiles and $q \in \mathbb{N}$ denoting the number of bindings. Each functional profile $fp_k \in FP$ $(1 \leq k \leq r)$ possesses a set of input datapoints $INDPOINT = \{ \mathit{indpoint}_1, ..., \mathit{indpoint}_s \}$ and a set of output datapoints $OUTDPOINT = \{ \mathit{outdpoint}_1, ..., \mathit{outdpoint}_t \}$ $(s, t \in \mathbb{N})$. An output datapoint $\mathit{outdpoint}_u$ $(1 \leq u \leq t)$ of a functional profile fp_k can be connected to an input datapoint of another functional profile fp_l $(1 \leq l \leq r)$ by a binding in BN. Generally, an output data point of a functional profile can be connected to an input data point of another functional profile of the binding-schema by a binding, if the devices of both functional profiles are interoperable with each other for the realized function. A valid binding-schema contains all necessary bindings for the realization of overall functionality of a desired system. Criteria about a valid binding-schema are presented in Section 3.2.2.

A functional profile fp_k is furthermore related to a set of operation modes $\mathit{opmode}_{1,fp_k}, ..., \mathit{opmode}_{z,fp_k}$ $(z \in \mathbb{N})$. Each functional profile fp_k of a binding-schema is assigned an operation mode opmode_{h,fp_k} $(1 \leq h \leq z)$. The set of input datapoints and the set of output datapoints on fp_k vary upon on a chosen opmode_{h,fp_k}.

A device dev_f $(1 \leq f \leq g,\ g \in \mathbb{N} :$ number of devices in detailed design) has a price cost and consists of functional profiles grouped by functional profile types. On each dev_f, a limited number of functional profiles from each functional profile type is provided. This limit is called the *cardinality of a functional profile type on device*, e.g. a lamp actuator device provides 8 functional profiles of type “lamp actuator” $(\mathit{fp\_lamp\_act}_1, ..., \mathit{fp\_lamp\_act}_8)$, where 8 denotes the cardinality of the type “lamp actuator”. The cardinalities are thus device specific and can be different for each functional profile type. There exists for a DD a corresponding AD from which it is created.

Abstract connections can connect many outports to one inport or one outport to many inports; similarly bindings can connect many output datapoints to one input datapoint or one output datapoint to many input datapoints.

3.2.2 Mapping Model

It is of major importance to provide artifacts that are useful for a transition from an abstract design to a detailed design, hence from function blocks to functional profiles and devices and from abstract connections to bindings. The mapping model allows this transition, thus it is an important contribution for the creation of detailed design. In addition, the mapping model is an essential step for fulfilling of the quality requirement *requirement compliance of implementation* (cf. Section 2.8.2).

A VDI 3813-2 conform functional profile of a manufacturer can realize one or more functions depending on its implementation. Thus, in general, many-to-one mappings from function blocks of an abstract design to functional profiles with this property are possible. Choice of specific operation mode for a functional profile determines the set of functions that should be realized by the functional profile. This implies that the functional profile covers a set of function blocks within the abstract design corresponding to this set of functions, which is called a function block combination. In other words, each standard conform functional profile realizes a specific function (block) combination (cf. Definition 3.2.4) for each of its operation modes.

Definition 3.2.4. Function Block Combination: A function block combination is an essential element of the mapping model that represents a group of function blocks. Each of the function blocks in such a group is related to a set of functional and non-functional requirements that can unambiguously be merged to common functional requirements and non-functional requirements. Function block combinations can be mapped to realizing functional profiles in operation modes specific for the merged functional and non-functional requirements.

Definition 3.2.5. Binding Pair Candidate: An abstract connection in abstract design concerns a function block pair. By mapping function blocks on functional profiles, function block pairs correspond to functional profile pairs. Each such functional profile pair (fp_1, fp_2) with $fp_1 \neq fp_2$ is called a *binding pair candidate* (*bpc*). The mapping model contains a set of binding pair candidates BPC, each of which is intended to realize the corresponding abstract connection and hence, intended to be connected to each other by at least one binding. In order words, a binding pair candidate denotes a *required binding.*

Definition 3.2.6. Mandatory Input Data Point: A functional profile may possess a subset of input data points that must be connected by bindings, in order to realize the mapped function block combination in a specific operation mode. Such data points are called *mandatory input data points.*

Definition 3.2.7. Fitting Functional Profiles: Two functional profiles that map certain function blocks of an abstract design are called *fitting functional profiles*, if they form a binding pair candidate and can be connected to each other at their output and input datapoints by bindings to realize the required communication relationships among function blocks implied by abstract connections. This occurs, if the devices of binding pair candidates are interoperable for the mapped function blocks.

Definition 3.2.8. Valid and Complete Binding-Schema: A *valid and complete binding-schema* contains

- only fitting functional profiles (*validity*), and
- functional profiles with all mandatory input data points connected with bindings, whereas for the satisfaction of this criterion, some other functional profiles might necessarily be integrated into the binding-schema,

than the functional profiles that exactly cover the function blocks of the given abstract design (*completeness*).

3.3 Formulation of the Problem

Composition of a detailed design for a single abstract design, that is intended to represent the automation plan of potentially many rooms of the same types (e.g. office rooms, hotel rooms, conference rooms, patient rooms, classrooms, etc.) conceives the highest complexity of the design creation for an entire BAS project. Therefore, it is the core problem and will be called the *design creation problem.*

Input: Given an abstract design AD with function blocks FB, abstract connections AC and available devices DEV on the market with functional profiles FP

$$\begin{aligned} FB &= \{\, fb_1, ..., fb_m \,\} \\ AC &= \{\, ac_1, ..., ac_n \,\} \\ DEV &= \{\, dev_1, ..., dev_p \,\} \\ FP &= \{\, fp_1, ..., fp_o \,\} \end{aligned}$$

, such that each fp_i $(1 \leq i \leq o)$ is pre-programmed on a host device dev'_j $(1 \leq j \leq p)$.

Research Question: Is there a detailed design dd with functional profiles $FP' = \{\, fp_1, ..., fp_v \,\}$ on devices $DEV' = \{\, dev'_1, ..., dev'_w \,\}$ $(FP' \subset FP \wedge DEV' \subset DEV)$ that satisfies the following constraints (cf. Section 2.8.2):

- **requirement compliancy:** each function block fb_g $(1 \leq g \leq m)$ of AD is mapped on a functional profile fp_h $(1 \leq h \leq v)$ that can realize fb_g in a specific operation mode;
- **correctness and completeness:** $\{\, fp_1, ..., fp_v \,\}$ forms a valid and complete binding-schema (cf. Section 3.2.2);

and satisfies the criterion:

- **costs**: the device sum of the prices in detailed design $\Sigma_{l=1}^{w} cost(dev_l)$ is as low as possible, where $cost : DEV \rightarrow \mathbb{R}$ is a function that maps a device on its price cost value.

Definition 3.3.1. Problem Instance: An abstract design AC, which is an input for the research question, is denoted as a problem instance.

3.3.1 Problem properties

It is convenient to identify the most characteristic properties of the design creation problem which play a key role to enable a search for solution approaches to practical problems with similar properties. In addition, based on the problem properties requirements on algorithms can be determined as presented in Section 3.3.2. The problem properties (PP) are:

PP1. **component-based software design:** The design creation problem concerns a *component-based software design* which can specifically called as function block-based design that follows a top-down approach. Devices are preprogrammed as *components-off-the-shelf* (COTS). From the aspect of software design devices are containers for the functional properties that have price costs, which must be considered as a criterion for the creation of optimized function block-based software designs;

PP2. **complexity class:** The problem can be reduced from the bin packing problem [Kar72] from the theory of computational complexity, which is known to be *NP-complete.* The large number of functional profiles preprogrammed on devices available on the market that results from a search in the component repository and the related variety of function block combination assignments for functional profiles increases the number possible combinations for the solution alternatives. This leads to a combinatorial explosion of design alternatives that need to be compared for various comparison criteria; in order to find the optimal design alternatives. An exact search would have an exponential time complexity due to the high degree of freedom in selection among hundreds or even thousands of different manufacturer-specific functional profile alternatives for each function block in a given abstract design. The problem is *a combinatorial optimization problem* (cf. Definition 4.1.1 and 4.1.2). Moreover, the search for valid binding-schemata at minimal price sum of devices plays an additional role on increasing the complexity of the problem;

PP3. **constrained nature, intermeshed structure, vertical and horizontal intercomponent dependency:** This is the most characteristic property of the design creation problem which causes the problem to be distinct compared to all similar problems. There exist two *constraints* for the problem one for the existence of mapping of function blocks on functional profiles (cf. Section 2.8.1) and the other one for the validity of the binding-schema (cf. Section 2.8.2). These constraints additionally make the problem more complex due to a vertical and a horizontal intercomponent dependency:

- **vertical intercomponent dependency:** The functional profiles in a binding-schema that is built by a solution method are required to completely map the function blocks of the given abstract design, which is not a trivial task, since the components exist in a hierarchy (cf. Figure 3.3). Further, the devices are off-the-shelf components; hence they possess preimplemented functional profiles that can realize the function blocks. Devices have in addition a limited capacity for each functional profile type and each device realizes a specific set of functions by its functional profiles. Due to this fact an arbitrary mapping of function blocks onto functional profiles results in infeasible designs. A search for feasible mappings among a high number of alternatives is a highly complex task;

- **horizontal intercomponent dependency:** Obtained feasible mappings of function block onto functional profiles and devices is a necessary, but not a sufficient condition for achievement of feasible binding-schemata. Each binding pair candidate (cf. Definition 3.2.5) related to a binding-schema is required to yield minimum one binding for which the interoperability of the hosting devices is necessary.

  The structure of a binding-schema resembles *pipes and filters architecture* and conceives many-to-one and one-to-many type of intermeshed connections among the functional profiles. By this fact binding-schemata cannot be subdivided as independent sets of functional profile chains, and must be considered as whole. This has an important impact on the suitability of functional profiles in the binding-schema concerning the integration of only fitting functional profiles (cf. Definition 3.2.7). Hence, choice of many feasible function blocks to device mappings yield to pairs of functional profiles which do not fit and many binding-schemata with missing bindings emerge. The search for valid and complete binding-schemata (cf. Definition 3.2.8) among a high number of alternatives is a highly complex task;

PP4. **multiple objectives:** Many detailed design alternatives can be related to a low price sum of devices, however for such designs the constraint for the validity of the binding-schemata (cf. Definition 3.2.8) are often not fulfilled due to the existence of missing bindings due to the lack of interoperability of communication partner devices. Thus, this constraint and the criterion for the cost of the design are conflicting. The design creation problem has further quality criteria and it is a *multi-objective optimization problem*;

PP5. **multi modality**: For the problem there is often not a single optimum, but there are *trade-off solutions* that compete against each other for different criteria. Hence, the problem is *multi modal.*

3.3.2 Requirements on Algorithms

From the problem properties presented in Section 3.3.1 and requirements on a solution method in Section 2.9 following requirements are derived for an algorithm to find solutions to the problem with the specified properties. The requirements on algorithms (RA) are:

RA1. **component-based mapping model and holism:** If the business process in Figure 2.4 considered, algorithms need a data model to perform an automated engineering task by creating detailed designs. Such a data model must contain all necessary elements of the design composition model to enable the evaluation of detailed designs according to the quality criteria as given in Section 2.8.2. Moreover, the data model for the designs must be interoperable with the results of preceding task of planning and the successor task of commissioning considering import/export relationships. More precisely, algorithms must be able to integrate a *component-based mapping model* as presented in Section 3.2.2 to import abstract designs, perform transitions into detailed designs and export these in a form that can be used - as is - by commissioning tools. Therefore, a *holistic approach* in the business process chain concerning the design creation task is required;

RA2. **model-based software design:** The creation of detailed designs is performed outgoing from an abstract design that conceives *reference architecture* in form of a function block-based design, thus it concerns a *model-based design*. In addition, a binding-schema is a *software design*, since functional profiles are software modules that encapsulate rule-based algorithms of BAS functions;

RA3. **heuristics for global search:** There is no known specific solution method of the design creation problem with large number of solution candidates which is NP-complete. Moreover, the properties of the optima are not known. An exact search is not practicable (or intractable), since the problem is related to a very large design space. A *global search* cannot be performed by dividing the problem into subproblems of lower complexity (e.g. by partitioning of the design space in case of branch and bound) and finding the global optimal solution from the local optimal solution of these subproblems. In such cases, metaheuristics can obtain good solutions for a set of defined objectives. In order to achieve this goal, a search must be performed that not only searches better solutions in the neighborhood of good initial solutions, but that is capable to create all feasible solution candidates in the design space by a global search. *Metaheuristics* can be employed to solve this optimization problem based on these properties;

RA4. **handling of constraints as soft-constraints:** Due to the problem property of horizontal intercomponent dependency, it is not possible to prune the search space from infeasible solutions before spanning the complete design space. A chosen algorithm that generates infeasible solutions must allow them to iteratively improve, rather than forbidding them. This can be realized by repairing infeasible solutions, where applicable, or mapping the degree of infeasibility on penalty values during the evaluation of solution candidates;

RA5. **multi-objective combinatorial optimization:** Due to the presence of multiple criteria for solutions, the algorithm must be capable of distinguishing solutions from each other for *multiple objectives* taking the combinatorial nature of the problem in account. Hereby, for a given trade-off solutions a decision making must be provided to choose the best alternatives. Moreover, a flexible integration of further criteria must be provided;

RA6. **goal-oriented search:** Integration of system design composition model presented (cf. Section 3.2) in the methods must be provided for the representation of room automation plans and designs. It must be possible to generate designs and improve them. Hereby, an evaluation of detailed design components must be provided to include more suitable components for *goal-oriented improvements.* Algorithms must integrate operations incorporating problem-specific information for a quick convergence to optimal solutions, rather than performing ineffective blind searches. Hereby, a balance between exploration and exploitation must be provided to avoid from premature local optimal solutions;

RA7. **limited computation time:** Algorithms must produce *short-time-to-market* solutions, which implies that designs of as good quality as possible must be generated in a short period of time with limited computational resources. The solution generation should not take longer than the manual design creation and be efficient (cf. Section 2.4). Algorithms that require computer clusters performing computations for weeks and months are out of consideration for this thesis;

RA8. **generality of solution method:** Algorithms must provide solutions for *different representative problem instances* (cf. Definition 3.3.1) without plan specific adaptations in the solution method. Thus, the solution method must be generic.

3.4 Conclusions

This chapter introduces a briefly new approach for the engineering of BAS on the logical level. The logical level of BAS formed of two logical views the abstract and detailed design models following a top-down approach is presented in Section 3.2.1 along with definitions for elements of both models. Further, the transition from the abstract model to detailed design model, the mapping model is introduced in Section 3.2.2.

A formal definition of the design creation problem that conceives the research question is presented in Section 3.3. Moreover, problem properties are presented that distinguishes the BAS design creation problem from other similar design creation problems in engineering practice. It is concluded that solution methods that involve metaheuristics are candidates for the solution of the problem. Based on the problem properties, requirements on candidate algorithms are identified. Problem properties and requirements on candidate algorithms enable a search in the literature for similar problems and candidate solution methods. In Chapter 4, a literature study for similar problems, algorithms and candidate solution methods are presented.

Chapter 4

Solution Methods for Design Generation and Optimization

4.1 Combinatorial Optimization

Optimization problems encountered in applied engineering fields or in different fields of theory concern the choice of a “best ” configuration or a set of parameters among a countable number of alternatives to achieve certain goals. For a combinatorial optimization problem (COP) the choice which must be made are discrete [AL97].

Definition 4.1.1. Combinatorial Optimization Problem: A *combinatorial optimization problem* is specified by a set of problem instances and it is formulated as a minimization or a maximization problem.

In the literature, task assignment problem, bin-packing problem, 0-1 knapsack problem, set covering problem, vehicle routing problem, traveling salesman problem [Kar72] are examples of combinatorial optimization problems. These problems have different instances depending on some specific details.

Definition 4.1.2. Combinatorial Optimization Problem Instance: A *combinatorial optimization problem instance* can be represented as a pair (S, c), where the solution set S is the set of feasible solutions and the cost function c is a mapping $f : S \to R$. The problem of finding a globally optimal solution $X \in S$ such that $f(X) \leq f(x)$ for all $x \in S$. Moreover, $C = f(X)$ denotes the optimal cost and $P = \{x \in S | f(x) = C\}$ denotes the set of optimal solutions.

4.2 Metaheuristics

For the solution of the design creation problem global search methods based on exact search such as cutting plane and branch and bound algorithms are not suitable (cf. requirements on algorithms in Section 3.3.2). Thus, metaheuristics can be considered, which are applied for practical multi-objective combinatorial optimization (MOCO) problems. Metaheuristics do not guarantee global optimal solutions in large design spaces; however they are applied for practical problems to find near-optimal solutions. Metaheuristics are approximate, usually non-deterministic. They are not problem-specific, but due to the generality, they allow a problem specific adaptation of algorithms. There are various metaheuristics available in the literature which can be classified as:

- single-point search (trajectory) methods vs. population-based search methods,
- nature vs. non nature-inspired,
- constructive vs. improvement methods and
- aggregation-based vs. multi-objective methods.

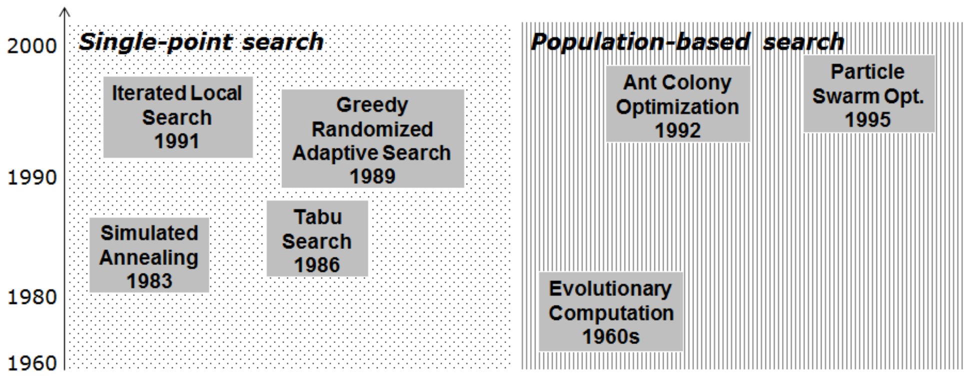

2000
1990
1980
1960
Single-point search
Iterated Local Search 1991
Greedy Randomized Adaptive Search 1989
Simulated Annealing 1983
Tabu Search 1986
Population-based search
Ant Colony Optimization 1992
Particle Swarm Opt. 1995
Evolutionary Computation 1960s

Figure 4.1: A Timeline of Metaheuristics

In trajectory or single-point search methods, an iterative improvement is performed on a single initial solution. Simulated Annealing (SA) [KGV83], Tabu Search (TS) [GM86], Greedy Randomized Adaptive Search (GRASP) [FR89] and Iterated Local Search (ILS) [MOF91] are examples of trajectory methods. Use of a trajectory algorithm brings the advantage that generally lesser computational resources are required than population-based search methods. However, optimization with such algorithms has generally the following disadvantages:

- *a single solution* is generated and improved. However, the design creation problem requires multiple trade-off solutions (cf. Section 2.9) for different objectives;
- the *danger of premature convergence* is higher. Up the hill iterations may not be sufficiently of high performance to escape from local optima.

In contrast to trajectory methods, population-based methods use a number of solutions (*population*) that are intended to be improved simultaneously. Most frequently used examples for population-based methods in applied engineering problems are Evolutionary Computation (EC) [FOW66], [Rec73], [Hol75], Particle-Swarm Optimization (PSO) [KE95], and Ant Colony Optimization (ACO) [Dor92]. Improving multiple solutions simultaneously has the following advantages:

- *simultaneous and continuous improvement*: Posterior to a number of iterations some solutions may converge prematurely; however, there still remain other parallel solutions which can potentially be improved in the next iterations;

- *achievement of multiple solution alternatives*: A population-based method can offer multiple solution alternatives within each iteration.

Some single-point and population-based search methods are depicted in Figure 4.1.

Scientists that proposed nature-inspired metaheuristics mapped phenomena and artifacts as metaphors on algorithms such as EC, ACO and PSO developed in inspiration of biological facts; and Simulated Annealing developed in inspiration of physical processes. Moreover, metaheuristics such as GRASP and ILS are non nature-inspired.

Constructive metaheuristics such as ACO build solutions constructively whilst optimizing them. Improvement metaheuristics such as EC and PSO generally build initially complete solutions and improve them in the process of optimization. Moreover, some metaheuristics can also be designed to optimize in a constructive manner such as EC.

For many real life problems, multiple decision criteria are point of consideration. A lot of combinatorial optimization problems of this kind involve simultaneous optimization of multiple incomparable, conflicting or competing objectives for a given problem. For example, if the 0-1 knapsack problem is considered, with an increasing value of items grabbed that is intended to maximize, the weight of the knapsack increases that is intended to minimize (*conflicting objectives*).

Problems with multiple objectives can be solved by scalarization of the decision vector in the step for evaluation of solution candidates. Each problem solution candidate is assigned a set of decision values preserved in the decision vector. Since to each decision variable a preference-based priority value can be assigned, the priorities form a priority vector and a dot product of the priority vector and the decision vector can be calculated to obtain quality of a solution candidate. Thus, the quality of the optimum is a single value and the method is called *single-objective optimization.* In other techniques the decision vector is not scalarized to a single value, rather, the minimization or maximization is performed at each objective separately, which is called *multi-objective optimization* (cf. Definition 4.2.1).

Definition 4.2.1. Multi-Objective Optimization: Find a vector $x^* = (x_1^*, ..., x_n^*)$ which satisfies the m inequality constraints $g_i(x) \geq 0$, $i \in [1, m]$, p equality constraints $h_i(x) = 0$, $i = [1, p]$, and minimizes the vector function $f(x) = f_1(x), ..., f_k(x)$, where $x = (x_1, ..., x_n)$ is the vector of decision variables. The set of all values that satisfy g_i, h_i defines the feasible region Ω and any point $x \in \Omega$ is a feasible solution.

This technique provides the following advantages:

- *a set of trade-off solutions* as best solutions of each step in an optimization process, instead of a single best solution,
- *a more precise distinction of solutions* in contrast to the single-objective optimization, and thus
- *a better diversity of solutions*: can be achieved that implies decision vectors with different values corresponding to distinct and complementary properties of solutions. These solutions may directly or indirectly interact with each other for combining parts with high quality to form better solutions or may follow a niche to move towards better regions of a search space.
- *allowing achievement of all solutions*: in the design space which is particularly important for a global optimization.

4.3 Examples for Metaheuristics

In this section insights into SA and the TS as examples for single-point; and ACO and EC as examples for population-based search methods are given.

4.3.1 Simulated Annealing

SA is inspired by the annealing process in metals as proposed in [KGV83] and [Cer85] as a search algorithm first to solve combinatorial optimization problems such as VLSI design and the Traveling Salesman Problem (TSP).

In annealing process a metal is heated up to a very high temperature and then slowly cooled down. If the heating temperature is sufficiently high to guarantee a random state of the substance and the cooling process is sufficiently slow to guarantee a thermal equilibrium, then the atoms will form a structure that enables the energy minimum of a perfect crystal. Hence, the strength of the crystal is inverse proportional with the speed for the cooling schedule. A fast cooling process is related to imperfections (or metastable states), which is not desired.

Similarly, SA starts with a high initial number for iterations and converges to near-optimal solutions following an implementation of a slow cooling process. Current energy value of the substance corresponds to the evaluation value of the current solution. SA always accepts improving solutions and also non-improving solutions (uphill moves, in case of minimization). By this property, SA is one of the first algorithms that provides a mechanism to avoid getting stuck in local optima. Hereby, different approaches for the cooling process can be followed such as linear, geometric, logarithmic, etc. An abstract SA algorithm is presented in Algorithm 1.

Algorithm 1: An Abstract SA Algorithm

```
Input: All possible solution components: $S$, initial temperature:
       $TEMPERATURE$
Output: A near-optimal solution: $s$
// number of iterations
$temp = TEMPERATURE$;
// $s$: a random or a known good initial solution
$s = INIT\_SOLUTION()$ ;
while $temp > temperature$ do
    while $inner\_loop\_criterion$ do
        // find a new solution $new\_solution$ in the neighborhood of $s$
        $new\_solution = PERTURB(s)$;
        // optimization is performed in form of minimization
        $\Delta E = COST(new\_solution) - COST(s)$;
        if $\Delta E < 0$ then
            // better solution found
            $s = new\_solution$;
        end
        // accept worse solution with probability
        else if $RANDOM(0,1) > e^{-\frac{\Delta E}{temp}}$ then
            $s = new\_solution$;
        end
    end
    // update temperature
    $temp = COOLING\_SCHEDULE(temp)$;
end
```

4.3.2 Tabu Search

TS is initially proposed in [GM86] and aimed at escaping local optima by using a memory explicitly to store the recent visited solutions generated during the search. This memory is called *tabu list* and is used to avoid revisiting of recently visited solutions and to forbid progress towards them. TS performs searches in the neighborhood of a current solution, where this neighborhood, which is called *allowed neighborhood*, is allowed to contain only solutions that do not exist in the tabu list. The best solution of this restricted neighborhood is set as the current solution and added to the tabu list while removing an existing solution (generally the oldest solution) in the tabu list in order to keep the length of the list (called *tabu tenure*) constant.

For more efficiency solution attributes are rather stored in the tabu list than complete solutions. These attributes can be implemented as moves, solution components, or feature differences between solutions. However, this is related to a loss of information due to the storage of attributes instead of complete solutions. Hence, this can lead to forbidding of solutions which are not yet generated and may potentially be of good quality. *Aspiration criteria* are defined that allows a tabu solution to be considered in the allowed neighborhood. The most common aspiration criterion prefers always better solutions of the search than the current best solution. An abstract TS algorithm is presented in Algorithm 2.

Algorithm 2: An Abstract TS Algorithm

```
Input: All possible solution components: $S$, size of tabu list: $L$
Output: A near-optimal solution: $best\_solution$
// initialize tabu list
$tabu\_list = \emptyset$;
$s = GENERATE\_INITIAL\_SOLUTION()$;
$best\_solution = s$;
while $!termination\_conditions$ do
    $INIT(candidate\_list)$;
    for $all\ new\_solution \in N(s)$ do
        // $N(s)$: solutions in neighborhood of $s$
        if $!CONTAINS(tabu, new\_solution)$
        $||ASPIRATION\_CONDITION(s)$ then
            $candidate\_list = candidate\_list + new\_solution$;
        end
    end
    $new\_solution = CHOOSE\_BEST\_OF(candidate\_list)$;
    if $new\_solution < best\_solution$ then
        $tabu\_list =$
        $FEATURE\_DIFFERENCES(new\_solution, best\_solution)$;
        $best\_solution = new\_solution$;
        $UPDATE(tabu\_list)$;
        $UPDATE\_ASPIRATION\_CONDITION()$;
    end
end
```

4.3.3 Ant Colony Optimization

Foraging behavior of ants inspired Dorigo in his PhD thesis [Dor92] to derive the fundamentals of ACO. Ants find the shortest path from the nest to the food source. Each ant deposits a chemical substance called *pheromone* on the path it travels which attracts the other ants and evaporates in time. As an indirect form of communication among ants, this leads the shortest path to emerge. ACO was first used to solve the TSP problem.

In ACO artificial ants construct complete solutions (*pheromone trails*) iteratively by adding solution components to current partial solutions. For addition in each intermediate phase of the construction, ants are faced to multiple solution component alternatives with different pheromone values assigned. Ants choose a component alternative among many alternatives according to a probability (p_{ij}) proportional to its pheromone value and the profit an ant can possibly achieve.

This probability is usually defined in terms of a state transition, from the state in which the solution component i was chosen to the state in which the solution component j will be chosen, as given in (4.1). τ_{ij} is the pheromone value associated to the addition of component j in this transition. C is the set of solution components which are allowed to be added into the current partial solution, but which are not yet visited. α and β are parameters with which the weights for the pheromone values τ_{ij} and problem-dependent heuristic values η_{ij} can be adjusted in form of a good trade-off. For $\alpha = 0$ the ACO algorithm finds solutions that correspond to a concatenation of partial best solutions which often does not yield to global optima. For $\beta = 0$ search will be guided only by pheromone trails which can lead to the construction of identical suboptimal tours by ants.

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum\limits_{k \in C} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}} \ , \forall j \in C \tag{4.1}$$

$$\tau_{i\pi(l)} = \tau_{i\pi(l)} + \frac{1}{f(\pi)} \ , \forall l \in [1, n] \tag{4.2}$$

$$\tau_{ij} = (1 - \rho)\tau_{ij} \ , \forall i, j \in [1, n] \tag{4.3}$$

In the beginning before ants build complete solutions, pheromone values are identical for all solution components. After the solutions are fully constructed, pheromones values $\tau_{i\pi(l)}$ are updated as given in (4.2) proportional to the quality (f) of the achieved tour π. Hereby, each ant increments the pheromone associated with each chosen solution component i depending on its partial heuristic value within f.

The pheromone values of solution components or paths increase with increasing number of ants visiting them. In order to limit the continuous growing of pheromone values of good solutions and to avoid the convergence towards local optima, pheromone values τ_{ij} are decreased (*evaporation*) by a reduction rate $\rho \in]0,1]$ as given in (4.3) (e.g. after a solution is completed), which is an implementation of forgetting.

Since artificial ants act decentral as agents, some useful centralized actions (*daemon actions*) can optionally be performed, such as application of local search on constructed solutions to ameliorate them or depositing additional pheromone on the components of the best solution. An abstract TS algorithm is presented in Algorithm 3.

Algorithm 3: An Abstract ACO Algorithm

```
Input: All possible solution components: $S$, number of trails per
       iteration: $TRAILS$
Output: Near-optimal solution: $best\_solution$
// initialize the near-optimal solution
$best\_solution = null$;
// initialize solution set
$P = \emptyset$;
$INITIALIZE\_PHEROMONE\_VALUES(T)$;
while $!termination\_conditions$ do
    for all $a \in A$: all ants do
        // construct ant trails by selecting solution components
        // based on pheromone values $T$ and heuristic values $H$
        $solution = CONSTRUCT\_SOLUTION(T, H)$;
        $P = P \cup \{solution\}$;
    end
    // optional daemon activities
    for all $P_i \in P$ do
        $P_i = LOCAL\_SEARCH(P_i)$;
        if $best\_solution == null$
        $||COST(P_i) < COST(best\_solution)$ then
            $best\_solution = P_i$;
        end
    end
    // update pheromones for each $P_i \in P$
    $UPDATE\_PHEROMONE\_ON\_COMPONENTS(T, P)$;
end
```

4.3.4 Evolutionary Computation

EC is an umbrella term for a set of algorithms promoted in the field in Operations Research, which are also called Evolutionary Algorithms (EA). EC is inspired by biological evolution, more precisely by the adaptation of living things or individuals to their environment in order to increase their chance for survival and reproduction.

The field of EC was first established by Evolution Strategies at Technische Universität Berlin, Germany by Rechenberg and Schwefel [Rec73], Evolutionary Programming in San Diego, USA by Fogel [FOW66] and Genetic Algorithms in Michigan, USA by Holland [Hol75], which evolved independently from each other for some decades. In the beginning of 90s Genetic Programming was proposed by Koza [Koz92]. A survey for these algorithms can be found in [DJ06].

In order to achieve the goal of reproduction and survival, the individuals that form a population evolve; hence adapt themselves to the environment during many generations by undergoing phenomena that are characteristic for the evolution. These are cross-over, mutation and selection. Algorithms in the field of EC map these phenomena onto operations that can be applied on problem representations (*solution candidates* or *individuals*) to iteratively find solutions that optimally achieve the intended goals for given problems. An iteration is called in terms of EC a *generation*. The adaptation of individuals to their environment is quantified by the definition objective functions.

For better adaptation to the environment, hence for better values of objective functions, in each generation, two or more individuals (*parents*) may exchange information undergoing *cross-over* by a probability p_C to produce new individuals (*children*). In addition, some individuals undergo a *mutation* by a probability p_M to introduce new and valuable information into the population. Cross-over (also called *recombination*) and mutation operations are called *variation operations*. The created children are evaluated using objective functions. *Selection* determines the individuals of each next *generation* and prevents the number of individuals in a *population* (population size) from growing for dealing with limited computational resources such as memory and computation time. In this process selection operation prefers the individuals with best values of objective functions which constitutes an iterative optimization.

Replacement of parent individuals by children individuals using cross-over and mutation operations is related to the potential risk of losing highly optimized parents. In order to prevent highly optimized individuals from being forgotten, a strategy called *elitism* is developed. This strategy considers both parents and children in the selection operation and gives a further chance to better optimized parents over poorly developed children, in order for them to be reconsidered in a next generation.

Common components of an EC algorithm in general are:

- **representation**: Solutions may be coded in combinatorial optimization problems as bit, integer or enumeration strings, tree or graph structures. Mixed type representations and complex representations are also used depending on characteristics of considered problems. A special care must be taken for the choice of the representation, since the search capability of the algorithm, more specifically, of the variation operations depend on the choice of representation. Algorithms that use too simple representations for complex structured problems which abstract useful problem-specific information, search blindly, thus ineffectively;

- **objective functions**: As a common concept for all metaheuristics , in order to equip algorithms with decision making ability, quantification of solutions is provided in form of objective functions, where each objective function can be minimized or maximized to attain certain goals. They depend on problem-specific parameters and reflect the solution quality quantitatively for each possible parameter constellation. In Multi-Objective Evolutionary Algorithms (MOEA) for each solution a decision vector (or a fitness vector) is conceived with the values of objective functions instead of a scalar value to enable a search for attaining all optimization goals simultaneously;

- **initialization**: A global search can be performed by initial solutions that are well sampled throughout the whole search space; otherwise the algorithm will tend to perform a local search to which highly explorative variation operations can aid to escape from getting stuck in local optima. Some algorithms incorporate special procedures such as a greedy heuristic or single solution based metaheuristics to generate well sampled and preoptimized initial solutions;

- **selection strategy**: Selection of individuals of a next generation may replace parents by children without any condition according to the non-elitist strategy and with the condition that better individuals replace worse ones according to the elitist strategy. There are various selection types such as fitness proportional, ranking-based, tournament, roulette wheel, stochastic universal sampling, truncation. Further details can be found in [ES07]. Most recent MOEAs apply a Pareto dominance-based selection procedure [Deb04];

- **reproduction (variation) strategy**: Improvement of the individuals in a population is achieved by variation operations. Parents that undergo a cross-over operations are determined by a mating (or parent) selection. In elitist MOEAs parents are cloned before performing variation operations. Thus, variation operations are applied only on clones which are children. There are various standard cross-over and mutation operations for string-based representations N-point, uniform cross-over; for permutation representations partially mapped cross-over (PMX), edge cross-over (EX), order cross-over, cycle cross-over etc. [ES07]. Variations determine the current search strategy in a generation, in which a search can be applied between intensification and diversification;

- **termination criteria**: Some practical criteria for the termination of an EA can be the achievement of the global optima given that their fitness values are known, reaching a computation time limit or limit for number of fitness function evaluations, observing poor diversity in the population, or achievement of no improvement in the population for a threshold number of generations.

An abstract EC algorithm is presented in Algorithm 4.

Algorithm 4: An Abstract Elitist EC Algorithm

```
Input: All possible solution components: $S$, population size: $L$,
       mutation probability: $p_M$, cross-over probability: $p_C$
Output: A near-optimal solution: $best\_solution$
// initialize population
$P = \emptyset$;
$P = GENERATE\_INITIAL\_POPULATION()$;
// evaluate objective functions
$EVALUATE(P)$;
$best\_solution = SORT\_BY\_FITNESS\_VALUES(P)$;
while $!termination\_conditions$ do
    // mate parent individuals by probability $p_C$ and clone them
    // cloned parents are called children
    $P' = MATING\_AND\_CLONING(P)$;
    // perform cross-over among children probability $p_C$
    $CROSS\_OVER(P')$;
    // mutate individuals by probability $p_M$
    $MUTATION()$;
    // evaluate objective functions
    $EVALUATE(P')$;
    // select best individuals from $P \cup P'$ and assign them as new
       population
    $P = SELECTION(P \cup P')$;
end
```

4.4 Choice of the Solver Algorithm

Each of the four metaheuristics presented in Section 4.3 can be chosen to solve the design creation problem, since they fulfill the requirement heuristics for global search in Section 3.3.2. Due to the advantages of multi-objective optimization methods and population-based search a multi-objective population-based algorithm can be chosen. In the literature there exist multi-objective ACO, EC and PSO algorithms. Thus a problem-specific adaptation of any of these multi-objective population based algorithms can satisfy also the rest of the requirements. However, most study is done and most practical applications exist for multi-objective EC algorithms. In addition, they are applied in [OPK10a], [ODK09] and [ORK09] to solve the design creation problem effectively.

Thus, Section 4.5 is focused on some multi-objective evolutionary algorithms (MOEA), particularly which follows special approaches to achieve a better diversity and a continuous improvement of solutions.

4.5 Specialized Methods for Diversity Preservation

Elitist MOEA favor solutions with better fitness values which may lead to the emergence of highly fit solution parts (or *genes*) that dominate the population. This phenomenon is called *genetic drift.* This can result in a permanent decrease of the diversity in the population. A phase in optimization emerges, where most or even all of the individuals are identical to each other and the algorithm gets stuck in local optima without any possibility to make further improvements. MOEAs in general rely on comparison of individuals by objective functions value by value based on the notion of Pareto dominance (cf. Definition 4.5.1).

Definition 4.5.1. Pareto Dominance: A vector $u = (u_1, ..., u_k)$ is said to *dominate* a vector $v = (v_1, ..., v_k)$, if and only if, u is partially better (less, in case of minimization) than v, i.e. $\forall i \in [1, k]$, $u_i \leq v_i \wedge \exists i \in [1, k]$: $u_i < v_i$.

There are several MOEAs such as Non-Dominated Sorting Genetic Algorithm 2 and Strength Pareto Evolutionary Algorithm 2 proposed in [DPAM02] and [ZLT01] respectively that consider the integration of an explicit logic in the selection operation to preserve the diversity by incorporating density estimation methods, and a MOGA that preserves diversity implicitly in cross-over operation:

- **Non-Dominated Sorting Genetic Algorithm 2 (NSGA2)**: Population of doubled size $2N$ (union of parents and children) is reduced to N during the selection. First $2N$ individuals are sorted in non-dominated sets of different levels and then N individuals are copied into the population of the next generation beginning with the non-dominated set of the first level. If there are too many individuals at the non-dominated set of the last level that is considered, individuals are sorted according to a crowding-distance metric. After sorting in this level, individuals with the highest crowding-distances are preferred to be included in the new population for preservation of diversity.

- **Strength Pareto Evolutionary Algorithm 2 (SPEA2)**: This algorithm uses both a population and an archive with constant sizes N and M respectively. At each generation the archive A is updated to contain the non-dominated individuals from the union of the population and the archive. If the number of the non-dominated individuals $K < M$, then all non-dominated individuals and $M - K$ best-dominated individuals are copied into the archive of the next generation A'. If $K = M$, then individuals of A are replaced by the non-dominated individuals. If $K > M$, a sophisticated truncation operation is applied on the set of non-dominated individuals using the kth-nearest-neighborhood metric to provide a uniform spread in the set of non-dominated individuals.

- **Cellular Genetic Algorithm for Multi-Objective Optimization (MOCell) [NDL$^+$09]**: *Cellular Genetic Algorithm (cGA)* defines neighborhoods in a population which are isolated from each other [AD08]. Individuals are mated with their cross-over partners only within the same neighborhood by which an information exchange throughout the whole population is avoided. By this approach, different neighborhoods explore different regions of the search space and the problem of genetic drift involving the whole population can be prevented. MOCell is the multi-objective specialization of cGA that uses an archive to store the non-dominated solutions of a current generation.

Both NSGA2 and SPEA2 are applied successfully in a large variety of practical engineering problems. These algorithms both allow information exchange among all individuals in a population in cross-over operations. However, this may also result in genetic drift in the population and only a low number of solutions belonging to the true Pareto front can be achieved. MOCell is reported in [NDL$^+$09] to obtain a better spread of non-dominated solutions toward the true Pareto front than NSGA2 and SPEA2 for instances of example problems used to test multi-objective optimization methods such as Zitzler- Deb-Thiele (ZDT) problems [HHBW06], however its random population update strategy may have negative effects on recombination clusters formed during the evolution of the population on different problems. NSGA2 and SPEA2 are two of the most often applied algorithms in engineering practice.

4.6 Approaches for Real World Problems

In the literature, according to my best knowledge, there is no solution method that solves a problem with properties presented in Section 3.3.1 and at the same time fulfill all the requirements on algorithms given in Section 3.3.2. Thus, in this section, rather solution methods that solve real world multi-objective combinatorial design problems will be considered.

4.6.1 Component-Based Mapping Problems

Erbas proposed in his PhD thesis [Erb06] a solution approach for solving design space exploration problem for multimedia applications, namely an MJPEG encoder and JPEG decoder as a system-on-chip. The problem concerns the mapping of an application model onto an architecture model. The functional behavior of the application is specified independent from the system architecture. Applications are simulated in a framework called Sesame which enables the creation of a mapping layer consisting of virtual processor components and evaluation of solution candidates. These components represent the application processes at the architecture level.

The author claims that number of all possible mappings grows exponentially and designers usually need a subset of good mapping alternatives, which can be achieved by a search for optimal mappings. The fitness criteria considered in the optimization are maximum processing time in the system, total power consumption of the system, and total cost for the architecture. In addition there are several constraints such as: each node has to be mapped onto a single processor, each channel in the application model has to be mapped onto a processor or a memory, and if two communicating nodes are mapped onto the same processor, then the communicating channels between these nodes have to be mapped onto the same processor.

Solutions candidates are encoded as bit strings on which two standard variation operations are applied: one-point cross-over, and bit mutation. For infeasible solutions generated by variations repair operations are applied to obtain feasible individuals by randomly choosing a feasible mapping to handle the constraints as soft-constraints. NSGA2 and SPEA2 algorithms are applied with a population size of 100, mutation probability of 50% and cross-over probability of 80%. The algorithms are run on a PC for 50, 100, 200, 300, 500, 1000 generations, and achieved similar results with NSGA2 at a higher convergence speed than SPEA2. The model used in the tests contains 26 processes, 75-FIFO channels and 9 bus components.

In [MKBR10] a solution approach for the software design of a business reporting system is designed, in which an application model is mapped onto an architecture model. Application models are specified in UML-like diagrams according to the Palladio Component Model specification. Hardware and software nodes in the diagrams contain annotations for costs, performance (processing rates) and reliability, which are considered as the quality criteria of the optimization. It is also possible for software developers to specify resource demands for the software components via annotations, which enable the evaluation of generated solutions. In addition, constraints exist for the range of processing rate of servers, allocation of components on servers and component selection for their feasible option values.

The handled problem instance formulates a search for an optimal system with one web server, a dispatcher and four replicated reporting servers. There exist multiple degrees of freedom for the designs that lay in the choice for processor speed, component selection and allocation of components to the processors. It is assumed that components providing the same interface provide the same functionality and can be exchanged.

An exchange of functional profiles in case of design creation problem 3 that have the same interface (input and output datapoints) do not necessarily provide the same functionality. In contrast, functional profiles that map the same sets of function blocks must be exchanged or substituted to obtain design solution consistent to the given abstract designs. Otherwise, infeasible design solutions emerge and additional computational effort would be necessary to make them feasible.

The type of encoding used is an array of enumerations on which also standard variation operations are applied. NSGA2 is applied with a population size of 60 and optimal results are achieved after 200 generations on a PC. The model used in the tests contains 12 components and 40 tasks.

In Section 4.6.2 two real world MOCO problems are considered which do not satisfy the requirement component-based mapping model and holism (cf. Section 3.3.2) in connection with the problem property component-based software design (cf. Section 3.3.1), but which fulfill the requirement goal-oriented search to a certain extent.

4.6.2 Network Design Problems

Weicker et al. proposed in [WSWW03] an evolutionary multi-objective optimization approach for the problem of base station transmitter placement and frequency assignment. Placement and assignment problems are considered as a single integrated problem for the teletraffic of a 9000 m × 9000 m area in Zürich, Switzerland with a resolution of 500 m for demand nodes and 100 m for transmitter placement.

The base station transmitters are to be placed optimally in order to achieve a good coverage of the desired area with sufficiently strong radio signal. Hereby, each cell of a transmitter must possess a sufficiently number of channels to satisfy all simultaneous demands (calls). The channels should be assigned avoiding interference with channels of neighbor cells or within a cell.

The optimization approach aims at minimizing costs for transmitters, channel interferences and power consumption. Degrees of freedom exist in the number and power of transmitters and assignment of frequency channels. The problem model considered for the experiments consists of 288 demand nodes with 505 calls and 128 frequency channels.

The authors chose NSGA2 and SPEA2 as solver algorithms and also proposed an additional algorithm called Steady-State Evolutionary Algorithm with Pareto Tournaments (stEAPT). Population size (and archive size for SPEA2 algorithm) is 80. Optimization is performed for 800 generations for each of the three algorithms. A problem-specific representation is defined on which problem-specific and goal oriented variations are applied. Variations consist of random and directed mutation operations that remove transmitters or reduce their power to reduce interference, and a cross-over operation that cuts in two selected parents the service area in two halves and recombine the fittest halves. Mutation probabilities applied vary from 60% to 100%, whereas cross-over probabilities vary from 0% to 40%. SPEA2 and NSGA2 turn out to perform better than the proposed stEAPT, particularly regarding the obtained population diversity.

Carrano et al. proposed a problem-specific genetic algorithm for design of electric distribution networks in [CST^{+}06]. In such systems large extensions are required, where these expansions are done in geographically different locations. Expansions which are performed in small system fragments does not consider all combinatorial possibilities of the system and lead to more expensive, less reliable and more lossy system designs than a globally designed one. Hence, the approach is concentrated on the global expansion of a 21-node and a 100-node electric distribution network.

The algorithm used is NSGA2. It considers as objectives the minimization of following design parameters: financial costs due to system installation, maintenance and energy losses, and costs related to system faults (undelivered energy and restarting procedures), energy losses, investment in new facilities and distribution lines, average number of faults and average interruption time in faults. Moreover there exist constraints related to line capacity, voltage level in load buses, graph connectivity and radiality of the active network, quality and reliability index.

A problem-specific representation is defined in form of a string of integers for each substation and a planar tree graph to allow the application problem-specific and goal oriented variations. Individuals of the start population are initialized using a greedy approach, rather than randomly, to prune the design space. Hereby, only a set of neighboring nodes are considered as alternatives for linking to a given node, so that a connection to too distant nodes never occurs.

Variations consist of various problem-specific mutation operations and cross-over operations that sort nodes by costs for a substitution by lower cost nodes, substitute conductors and substations according to their capacity values. Population size is 50 and number of generations is 300. Mutation and cross-over probabilities are 3% and 80% respectively.

4.6.3 Comparison of Solution Methods

In this section, solution approaches in Section 4.6.1 and 4.6.2 are compared according to two criteria:

- They solve similar problems with properties as the design creation problem (cf. Section 3.3.1) and
- fulfill a considerable set of the presented requirements on algorithms (cf. Section 3.3.2).

Solution methods according to these criteria are introduced in Section 4.6.1 and 4.6.2 respectively. The second criterion is as relevant as the first criterion, since by choosing an algorithm further questions arise such as:

- How does the algorithm make a global search possible toward promising regions of design space?
- How good is the algorithm adapted to the problem?
- Is the adaptation sufficiently general for solving any arbitrary problem instance?

In Table 4.1 problems presented in Section 4.6.1 and 4.6.2 are compared according to their properties. All four problems possess properties PP2, PP4 and PP5 (cf. Section 3.3.1) and none of them possesses the property PP3, since a pipes and filters architecture in form of a function-block based design with intermeshed structure, vertical and at the same time horizontal intercomponent dependency cannot be observed.

	PP1	PP2	PP3	PP4	PP5
[Erb06]	+	++	−	++	++
[MKBR10]	+	++	−	++	++
[WSWW03]	-	++	−	++	++
[CST$^+$06]	-	++	−	++	++

Table 4.1: Comparison of Problems in Section 4.6.1 and 4.6.2
++:full, +: partial, -:no similarity

In both of the problems handled in [Erb06] and [MKBR10], component-based software design by mapping of requirements on available resources, more specifically mapping among models of different abstraction levels is considered, in which there exist constraints and multiple conflicting objectives. However, for mapping of application models on architecture models, there exist no constraints related to function realization on the architecture level as in the design creation problem, since processors and memories in [Erb06] and servers [MKBR10] are not preprogrammed COTS. Thus, a mapping can be performed arbitrarily and there is no need to perform a search for them. Moreover, no constraints related to inter-component dependencies on the architecture level exist such as in the case of the interoperability problem (cf. Section 2.5). Hence, these important differences exist, despite the partial similarity to the design creation problem according to the property PP1.

	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8
[Erb06]	+	++	++	++	++	-	++	?
[MKBR10]	+	++	++	++	++	-	++	?
[WSWW03]	-	-	++	++	++	++	++	?
[CST$^+$06]	-	-	++	++	++	++	++	?

Table 4.2: Comparison of Applied Algorithms
++:full, +: partial, -:no satisfaction of requirement; ?:unknown

In [Erb06] and [MKBR10] (cf. Table 4.2) component-based mapping models are integrated which can potentially enable an input-output consistency and an automation in the business process chain (RA1) (cf. Section 3.3.2). However, in none of the both approaches, there is a focused study on this requirement and no statements are made. Still, it can be assumed that the component-based mapping models provide a basis to satisfy this requirement. Furthermore, in both approaches application models as reference architectures are provided to enable a model-based software design (RA2). In [WSWW03] and [CST$^+$06], this is not the case.

In all approaches in Table 4.2 metaheuristics to perform a global search (RA3) are applied for a multi-objective combinatorial optimization (RA5) and for each of them a PC is used for tests (RA7), and no computer clusters. Constraints are handled as soft-constraints (RA4) using repair operations to make infeasible solutions feasible.

Representations used, on which standard variation operations are applied in approaches [Erb06] and [MKBR10], are standard (string-based) representations. In this case, useful problem-specific information is poorly considered in the representation and the algorithms can only perform a blind search. As a consequence, computationally expensive repair operations must be applied on a given infeasible solution to create potentially many feasible solutions and choose a feasible solution randomly, implying that the problem-specific information is widely transparent to the search operations of the algorithm.

An effective search towards promising regions of the search space can be possible by establishing a link between the problem and the algorithm in the variation operations and a goal-oriented search can be possible (RA6). In [WSWW03] and [CST$^+$06] algorithms incorporate problem-specific goal-oriented variations that full this requirement.

In each of [Erb06], [MKBR10] and [WSWW03] the proposed solution method is examined on a single problem instance and in [CST$^+$06] tests are performed for two problem instances.

Further, all of the presented approaches have certain shortcomings:

- Problem-specific goal-oriented variations are either not considered, as in case of [Erb06] and [MKBR10] or they are considered, as in case of [WSWW03] and [CST$^+$06], however they do not deal with problems possessing properties of a component-based software design with intermeshed structure, vertical and horizontal intercomponent dependencies (cf. Section 3.3.1);

- In none of the approaches generality of the solution method has been a point of consideration for which representative problem instances must have been identified and performance of algorithms could be tested (RA8).

4.7 Conclusions

In this chapter a brief introduction to the type of design creation problem, combinatorial optimization is presented in Section 4.1. Regarding the conclusion of Chapter 3 metaheuristics are presented in Section 4.2 and 4.3 that solve combinatorial optimization problems of a high complexity. Further, arguments are made for the choice of the solver algorithm in Section 4.4 and 4.5.

In Section 4.6 existing solution methods, which solve problems with similar properties and which fulfill some of the requirements on algorithms are presented. In Table 4.1 and 4.2 these approaches are compared with each other for their similarity to the properties of the design creation problem presented in Section 3.3.1 and for the fulfilled requirements on algorithms in Section 3.3.2.

None of the presented problems from the literature exactly match the problem properties presented in Section 3.3.1 and they do not fulfill the requirements on algorithms in Section 3.3.2 completely. The solution methods in [Erb06], [MKBR10], [WSWW03] and [CST$^+$06] have important shortcomings for applying on the design creation problem. Hence, a new solution method for the design creation problem is conceived in Chapter 5.

Chapter 5

Automated Creation of Optimized Designs

5.1 Introduction

A goal-oriented optimization process is essential for achieving solutions of high quality. A further impact of such an approach is an algorithm design that considers the information related to the specific problem instance. As many real life design problems in engineering are subjected to a high complexity, consideration of problem-specific information by solving methods aims at providing the advantage of reducing the time complexity, quick convergence and obtaining solutions of improved quality.

There are various concepts followed by a solving method in each of which problem-specific information can be realized. In these concepts different strategies can be followed.

The solution method that will be employed to solve the design creation problem must satisfy the requirements from RA1 to RA8 presented in Section 3.3.2. There are four key concepts to respond these requirements and build the fundamentals of the automated creation of optimized designs:

1. component model focused on the realization of RA1,
2. design evaluation,
3. design generation and
4. design improvement focused on the realization of RA2 to RA8.

5.2 Design Evaluation

For quantification of the suitability of a design as a solution candidate for the problem, a quality measure is incorporated that is determined upon on the design quality criteria. In terms of evolutionary algorithms this measure is called *fitness*. Fitness functions enable a value-based comparison between two solution candidates and guide optimization algorithm to improve solutions.

For the design creation problem following fitness criteria can be derived from design quality criteria given in Section 2.8.2:

- **abstract design compliance:** A solution must represent a binding-schema with device specific functional profiles covering all function blocks in a given abstract design, such that each function block is covered by only one functional profile in a specific operation mode;
- **validity and completeness of binding-schema** All communication partner devices in the design must be interoperable for the function blocks they realize, hence their functional profiles that realize function blocks must build bindings to realize abstract connections among the function blocks (cf. Section 3.2.2). However, some of the communication partner devices may only be interworkable (cf. Table 2.2) with syntactically fitting functional profiles, which do not fit semantically. This can be a consequence of inter alia incomplete specification of devices in the component repository. Thus, for an evaluation it is necessary to distinguish between *interworkable bindings* as bindings among interworkable devices and *interoperable bindings* as bindings among interoperable devices.

  Interoperable bindings are preferred to interworkable bindings, however some communication partner devices may happen not to build interoperable bindings and only interworkable bindings. Automated design creation must also allow such detailed design solutions under consideration of a distinguished evaluation between interoperable and interworkable bindings, rather than forbidding them and ending up with binding-schemata, in which bindings are missing. In a binding-schema, all mandatory input datapoints of functional profiles must possess bindings for a correct realization of abstract design. In some cases, extra functional profiles than the ones that realize the function blocks of the abstract design may need to be added into binding-schema to complete missing bindings;
- **device price costs:** Price sum of devices of a detailed design must be as low as possible. This implies a reasonable price sum for the complete design of a building project and would help customers' conviction by saving him financial resources.

Further criteria that define the optimality level of detailed designs are:

- **demand for device communications**: Bindings created among functional profiles of two different devices (*device external bindings*) mean messages sent between these two devices over the field bus, in case of wired communication e.g. in LON, KNX etc. In contrast, for device internal bindings no message is sent over the field bus which reduces network traffic load. Therefore, realization of function blocks by the functional profiles of a same device is preferred, where possible;

- **configuration and maintenance:** Bindings with functional profiles from different manufacturers (*manufacturer-spanning bindings*) are harder to configure in the commissioning task and to maintain. Designs with lesser manufacturer spanning bindings are preferred.

5.3 Component Model

In order to satisfy the requirement for a component-based mapping model and holism (RA1) (cf. Section 3.3.2) and allow an adaptation of algorithms to the design creation problem, abstract and detailed design models (cf. Section 3.2) are specified in UML class diagrams [1] in Figure 5.1 and 5.2 respectively. Hereby, an object-oriented approach is considered rather than an attribute oriented approach, in order make use of inheritance and encapsulation of the information elements that belong to each other in the context of abstract and detailed design. The abstract design model consists of elements that reflect the room automation planning elements such as function blocks, ports and connections.

The abstract design model is applicable for arbitrary room automation plan specified according to the standard VDI 3813 [The11a, The11b]. The detailed design model can be applied for designs made with LON and EnOcean components, and other similar platforms with protocols that implement the OSI reference model [DZ83] application layer for device application profiles and thus support designs with prefabricated devices and preprogrammed software modules.

A design generation method can generate detailed designs for a given abstract design. An algorithmic mapping can be performed from the abstract design to each such detailed design. In order to provide a matching of mapped components and to emphasize the mapping concept, a comparison of abstract design model and detailed design model is given in Table 5.1, for certain classes for which mappings exist or which exist in the detailed design without a direct counterpart that can be addressed in the abstract design (model components with "-" entries).

Each room automation plan and each related design for a room in broad sense also for a storey, corridor or a hall etc. is unique. This implies that the elements contained such as function blocks and abstract connections, or software modules, bindings and devices are unique in the complete BAS. Creation of each such unique room automation plan and design for large volumes such as buildings with hundreds of rooms individually is a tedious work, particularly if this must be performed with multiple buildings each individually.

[1]A reference to UML can be consulted in [BRJ99]

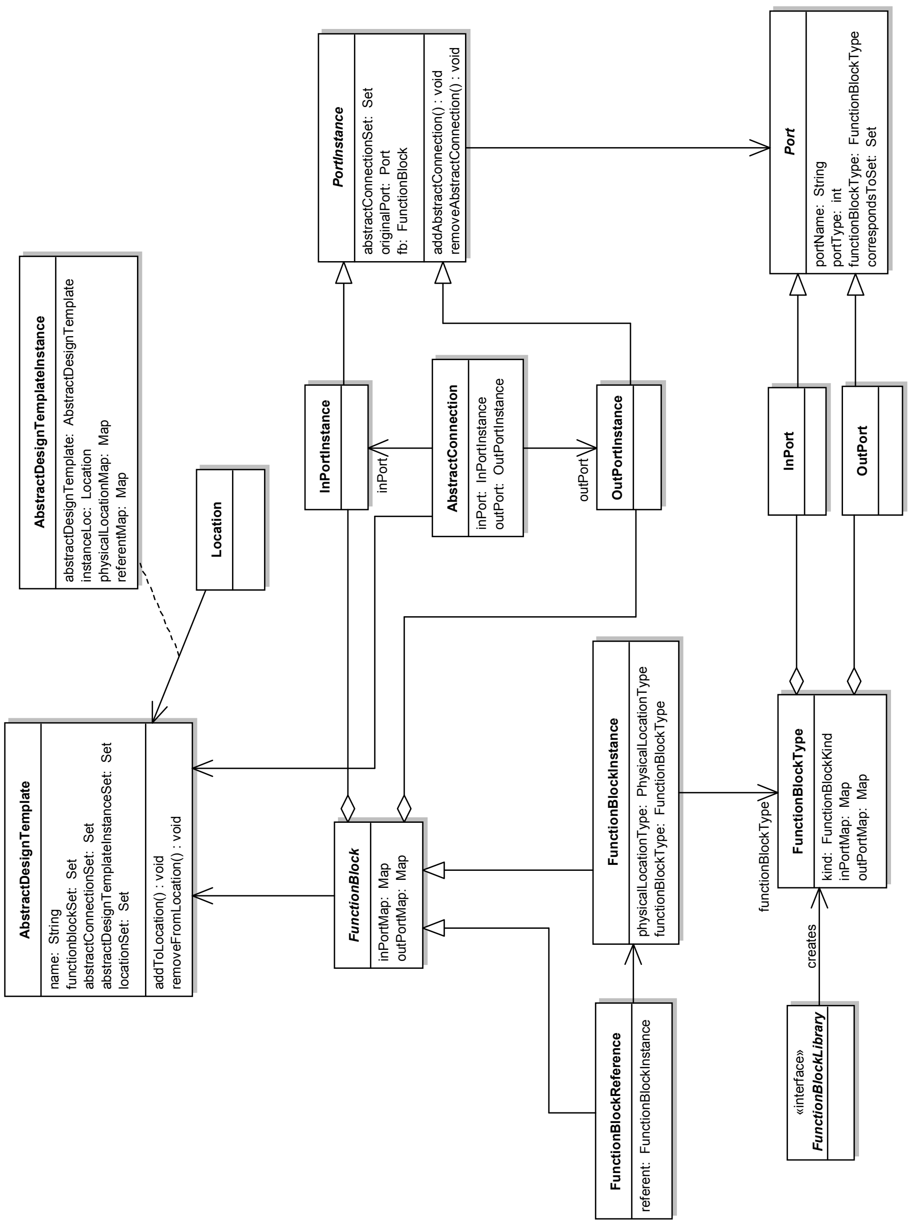


Figure 5.1: Simplified UML Class Diagram of the Abstract Design

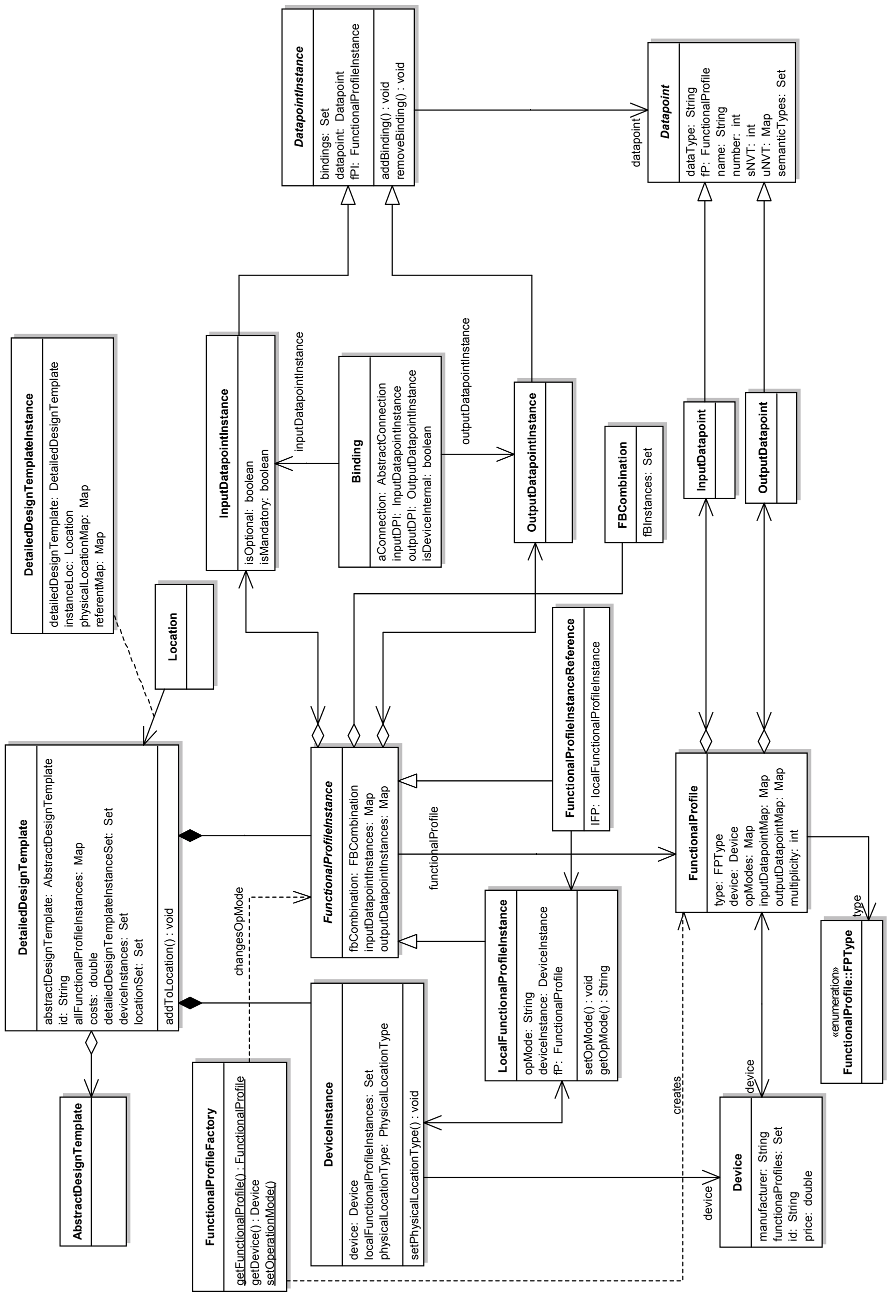


Figure 5.2: Simplified UML Class Diagram of the Detailed Design

According to the quality requirement for reusability (cf. Section 2.9), design templates are conceived for allowing the reuse of room automation plans for similar requirements in different building locations. This approach is based on templates and template instances, and it is provided for all functions in the function portfolio of VDI 3813-2 [The11b]. E.g. if n rooms must be endowed with constant-light control and automatic light functions, an abstract design template (ADT) is created according to the model in Figure 5.1 that contains the automation plan for these functions and assigned to these rooms as abstract design template instances ($ADTI$), where these functions are required. For the realization of the ADTs, a detailed design template (DDT) can be created according to the model in Figure 5.2 and assigned to all n rooms as a detailed design template instance ($DDTI$) with a reference ($detailedDesignTemplate$) pointing to the corresponding DDT and thus by a further reference ($abstractDesignTemplate$) to the related ADT. By this approach the user effort is saved for the creation of $n-1$ similar automation plans and corresponding designs. Further, repeated storage of identical information is prevented. Non identical room endowments that deviate from the original detailed design can be achieved by consequent modifications.

The functions in VDI 3813-2 [The11b] are subdivided in functional categories such as sensor, operating, application and actuator functions, each of which can be modeled as a function block ($FunctionBlockType$ in Figure 5.1) of the related function type. Manufacturer and platform specific functional profiles ($FunctionalProfile$ in Figure 5.2) [2] and the function blocks encapsulate the common properties, parameters, and ports as well as datapoints correspondingly. In order to allow a reuse of these types, instances for function blocks ($FunctionBlock$ in Figure 5.1) and functional profiles ($FunctionalProfileInstance$ in Figure 5.2) can be specified. $FunctionalProfileInstance$ encapsulates the specific operational values.

If abstract designs and detailed designs for a complete building project are considered, the reuse approach allows a saving of high data amounts in the BAS design. The approach applies analogous among ports ($Port$) and port instances, ($PortInstance$ in Figure 5.1) and datapoints ($Datapoint$) and datapoint instances ($DatapointInstance$ in Figure 5.2).

Some functions may commonly be used for a group of automation plans. Typical examples for such functions are sensor functions e.g. outdoor temperature and outdoor luminosity. Such functions are modeled as function blocks assigned to all common using ADTs, in one certain template as a function block instance and in the other templates as a reference ($FunctionBlockReference$ in Figure 5.1). The realizing functional profile ($LocalFunctionalProfileInstance$ in Figure 5.2) of such a function block is analogously used in a certain $DDTI$ and

[2]Functional profile is the term for a software module introduced by the application specifications of LonMark Organization [Lon] and EnOcean Alliance [EnO], which will be used for software module from this stage on.

used as a reference (*FunctionalProfileInstanceReference*) in other *DDTI*s.

A functional profile realizes one or more functions which implies that one or more function blocks are realized on a functional profile. A functional profile in the detailed design can realize various combinations of function blocks depending on its operation value setting. Each functional profile is assigned a function block combination (*FBCombination* in Figure 5.2) (cf. Definition 3.2.4) to denote the implemented function blocks.

5.3.1 Presumptions

For the design generation method to meet the quality requirements introduced in Section sec:designquality some presumptions are made:

- **complete implementation of design requirements in abstract design:** Function blocks in the abstract design completely cover the functional design requirements as well as other hardware requirements. Each function block hereby covers a required function and if desired, with hardware requirements on the device that should realize the function. An abstract design is the unique source of information for the design creation about the requirements on the system to realize. Hence, an abstract design that does not completely cover the requirements will lead to non requirement conform or incomplete detailed designs, or detailed designs with missing bindings;

- **interoperability in abstract design:** Communications of functions are specified via correct abstract connections, which are defined between interoperable function blocks, where the outports and inports are related to the required function and possess compatible port types according to VDI 3813-2 [The11b]. Connections among incompatible port types lead to invalid bindings, thus invalid binding-schemata (cf. Section 3.2.2);

- **correct implementation of functions on devices:** For mapping an abstract design on a detailed design, device and functional profile alternatives exist. For each function block there exists at least one functional profile of a device that realizes the functional requirements and hardware requirements implied by the function block. The functional profile is associated to a set of input and output datapoints that correspond to the semantics implied by the inports and outports of the function block. Such a functional profile can be considered to be conform to standard VDI 3813-2. Functional profile implementations that deviate from the semantics of function blocks in this standard are considered as non-conform and may cause invalid bindings with their binding partner functional profiles. This yields to invalid binding-schemata;

Abstract Design Model	*Comments*	Detailed Design Model	*Comments*
FunctionBlockType	*an unused function block*	FunctionalProfile	*an unused functional profile*
FunctionBlock	*a template specific function block*	FunctionalProfile Instance	*a template specific functional profile*
FunctionBlock Instance	*a used function block with the template assigned to the same location as itself*	LocalFunctional ProfileInstance	*a used functional profile with the template assigned to the same location as itself*
FunctionBlock Reference	*a used function block referenced from another template*	FunctionalProfile InstanceReference	*a used functional profile referenced from another template*
-		FBCombination	*group of used function blocks covered by a used functional profile*
-		Device	*device type of a functional profile*
-		DeviceInstance	*used device in a detailed design template*
AbstractConnection	*connector from an outport instance to an inport instance*	Binding	*connector from an output to an input datapoint instance*
Port	*a port type*	Datapoint	*a datapoint type*
PortInstance	*a used function block's port*	DatapointInstance	*a used functional profile's datapoint*
AbstractDesign Template	*an unassigned abstract design*	DetailedDesign Template	*an unassigned detailed design*
AbstractDesign TemplateInstance	*an abstract design assigned to a location*	DetailedDesign TemplateInstance	*a detailed design assigned to a location*

Table 5.1: Component-Based Mapping between Abstract and Detailed Design Model Types

- **assistance of a component repository:** For an automated retrieval for functional profiles and devices that realize given function blocks and related function semantics assistance of a component repository is provided. This component repository is designed to contain all design relevant data about functional profiles and devices from different device manufacturers world wide and to retrieve this information. Furthermore, it can evaluate interoperability among two given devices for specified functions and deliver all possible bindings among the functional profiles of the devices in a reasonable slice of time. This time requirement is crucial for the realization of the efficiency requirement on solution methods (cf. Section 2.9).

5.3.2 Integration of Component Model

The system design component model allows the definition of room automation plans for varying requirements and resulting designs that can not only be used within the same project, but also in other building projects. Furthermore, graphical visualization tools can integrate the abstract design and detailed design models to illustrate, edit, store the room automation plans and designs to provide a comfortable process of engineering.

In addition, by using the detailed design model the created designs can be imported in diagnosis and performance engineering tools [Plo07] for measuring the network load distribution on nodes. A further use can be provided by importing the designs in design creation and deployment tools for the physical topology of the automation network, commissioning and monitoring such as LonMaker via LNS [Ech] that provides a COM [3] interface or ETS via Falcon [KNX] that provides a DCOM [4] interface. These uses build the fundamentals of a holistic approach to design engineering in its business process model.

5.4 Design Generation

By presentation of all presumptions (cf. Section 5.3.1) and prerequisites (cf. Section 5.2 and 5.3) optimized design detailed designs can be generated for a given abstract design with the first step of design generation of a single detailed design which can be considered as a preoptimization step. The design generation is initiated by identifying alternatives for design components which is realized by searches in the component repository.

Component Retrieval: For function blocks

$$FB = \{fb_1, fb_2, ..., fb_i\} \tag{5.1}$$

[3]COM: Component Object Model
[4]DCOM: Distributed Component Object Model

of a given abstract design AD, manufacturer, device and operation mode specific functional profiles (fp's) can be retrieved as can be seen in (5.2), where ":" represents the relation between a function block and realizing functional profile alternatives.

$$\begin{aligned} fb_1 &: \{fp_{fb_1,1}, fp_{fb_1,2}, ..., fp_{fb_1,\alpha}\} \\ fb_2 &: \{fp_{fb_2,1}, fp_{fb_2,2}, ..., fp_{fb_2,\beta}\} \\ &... \\ fb_i &: \{fp_{fb_i,1}, fp_{fb_i,2}, ..., fp_{fb_i,\zeta}\} \end{aligned} \tag{5.2}$$

The sets of retrievals can be denoted in short form as:

$$\begin{aligned} FP_{fb_1} &= \{fp_{fb_1,1}, fp_{fb_1,2}, ..., fp_{fb_1,\alpha}\} \\ FP_{fb_2} &= \{fp_{fb_2,1}, fp_{fb_2,2}, ..., fp_{fb_2,\beta}\} \\ &... \\ FP_{fb_i} &= \{fp_{fb_i,1}, fp_{fb_i,2}, ..., fp_{fb_i,\zeta}\} \end{aligned} \tag{5.3}$$

5.4.1 Component Search

Some preoptimization steps are necessary for the adaptation of used algorithms to the problem domain for successfully dealing with the problem complexity and to obtain near-optimal detailed design solutions for given abstract designs. These steps are related to the following motivations:

- **correct mapping and coverage of function blocks by functional profiles:** In order to obtain optimized detailed design solutions that are consistent to given abstract designs a correct mapping of function blocks onto manufacturer and device-specific functional profiles is necessary. It is convenient to emphasize the problem property component-based software design at this point, since mappings must consider the fact that on devices preprogrammed functional profiles are implemented with device specific capacity of each for functional profile type. An arbitrary mapping of function blocks onto functional profiles, thus onto devices would lead to detailed design solutions inconsistent to abstract designs, hence to requirements that have to be met.

  Since most devices realize multiple function blocks, correct mapping of function blocks onto realizing devices must be provided. This is possible, only if, the mapped function blocks are realized by the functional profiles of these devices. In addition, devices must optimally cover function blocks of a given abstract design implying that each chosen device covers a maximum possible set of function blocks. On the implementation level, this corresponds to the optimal coverage of function blocks by functional profiles. A suboptimal coverage can on one hand lead to selection of more functional profiles and devices than necessary, which again leads to design solutions of high price costs. On the other hand, there may be more input datapoints that require bindings than actually necessary which leads to incomplete binding-schemata (cf. Definition 3.2.8);

- **substitution of matching functional profiles and devices:** An optimization algorithm will attempt to perform improvements on a created design by substitution of functional profiles and devices by different alternatives. An arbitrary choice of an alternative functional profile or a device for substitution would lead to inconsistency between the given abstract design and the generated detailed designs, hence to non requirement compliant detailed designs.

  Such an approach is inefficient, since additional computational cost intensive operations will be necessarily to regain the consistency by repairing inconsistent (or infeasible) design solutions, only if possible at this posterior stage. If a functional profile that realizes a certain set of function blocks will be substituted by an alternative functional profile as its counterpart, the counterpart must also realize the exact set of function blocks (cf. Definition 5.4.1). *FBCombination* in Figure 5.2 is conceived also to realize the identification of matching counterparts;

- **acceleration of design optimization:** Large number of the component alternatives available in the component repository and related high number of possible component combinations arise a problem of high complexity. In order to reduce the number of the components that can be considered by the optimization algorithm and thus to reduce the problem complexity, preoptimization steps must consider only relevant components for a given problem instance. Therefore, for optimization algorithms, functional profiles other than the ones retrieved as given in (5.2) are irrelevant and are out of consideration for the considered problem instance AD.

  In addition, repeated searches in the component repository for information related to functional profiles as presented in the component model (cf. Section 5.3) after the start of the optimization process would cause an enormous overhead. A book keeping of retrieved information during the search in component repository can accelerate the optimization process until its termination by reducing this overhead to a minimum.

Definition 5.4.1. Matching Functional Profiles: Two functional profiles fp_1 and fp_2 are matching, only if the function block combinations (cf. Definition 3.2.4) fbc_{fp_1} mapped by fp_1 and fbc_{fp_2} mapped by fp_2 are equivalent: $fbc_{fp_1} \equiv fbc_{fp_2}$.

Component Arrangement: In (5.3) some sets of retrieved functional profiles can be identical, since some of the function blocks can be from the same function block types and related to an identical set of requirements. In order to reduce the combinatorial complexity for the following operations, such identical sets of retrieved functional profiles

$$\begin{aligned} FP_{a_1} \equiv FP_{a_2} &\equiv ... \equiv FP_{a_f} \\ FP_{b_1} \equiv FP_{b_2} &\equiv ... \equiv FP_{b_g} \\ ... & \\ FP_{z_1} \equiv FP_{z_2} &\equiv ... \equiv FP_{z_h} \end{aligned} \tag{5.4}$$

with $a_1, a_2, ..., a_f, b_1, b_2, ..., b_g, z_1, z_2, ..., z_h \in [1, i]$ are identified and the corresponding function blocks are grouped in an initial step. Each such group of function blocks is then represented by a place holder function block $fb_{pholder}$ (cf. (5.5)), where "$\xrightarrow{ph}$" represents the relation from a place holder function block to original function blocks in abstract design, which are represented by the place holder function block.

$$\begin{aligned} &fb_{pholder_1} \xrightarrow{ph} \{fb_{FP_{a_1}}, fb_{FP_{a_2}}, ..., fb_{FP_{a_f}}\} \\ &fb_{pholder_2} \xrightarrow{ph} \{fb_{FP_{b_1}}, fb_{FP_{b_2}}, ..., fb_{FP_{b_g}}\} \\ &... \\ &fb_{pholder_e} \xrightarrow{ph} \{fb_{FP_{z_1}}, fb_{FP_{z_2}}, ..., fb_{FP_{z_h}}\}, e \in [1, i] \end{aligned} \tag{5.5}$$

From (5.3) and (5.5) for the union of functional profiles contained in the union set $FP_{fb_1} \cup FP_{fb_2} \cup ... \cup FP_{fb_i}$ with cardinality j, functions block sets can be obtained in (5.6) which can be realized simultaneously by each of these functional profiles. This is possible, since each retrieved functional profile is operation mode specific. Hereby, fb_{unique} denotes a function block with unique functional profile retrieval results hence $\forall\ fb_{unique} \in FB : fb_{unique} \notin \{fb_{pholder_1}, ..., fb_{pholder_e}\}$. Set of function blocks realized by each functional profile fp_p ($p \in [1, j]$) in (5.6) is denoted shortly as $FBSET_p$. In (5.6) "$\diamond$" represents the relation from a functional profile alternative to the place holder function blocks that can simultaneously be realized by the functional profile alternative.

$$\begin{aligned} &fp_1 \diamond \{fb_{pholder_{fp_1,1}}, ..., fb_{pholder_{fp_1,\gamma}}, fb_{unique_{fp_1,1}}, ..., fb_{unique_{fp_1,u_1}}\} \\ &fp_2 \diamond \{fb_{pholder_{fp_2,1}}, ..., fb_{pholder_{fp_2,\delta}}, fb_{unique_{fp_2,1}}, ..., fb_{unique_{fp_2,u_2}}\} \\ &... \\ &fp_j \diamond \{fb_{pholder_{fp_j,1}}, ..., fb_{pholder_{fp_j,\omega}}, fb_{unique_{fp_j,1}}, ..., fb_{unique_{fp_j,u_j}}\} \end{aligned} \tag{5.6}$$

It can be concluded that a functional profile can realize arbitrary combination of the function blocks in its function block set, only if function blocks contained in such a combination set allow a merging of their properties. Merging is possible if

- there is no repetition of function block types (implied room automation functions);
- non-functional requirements related to function blocks do not conflict with each other and can be merged to a common set of non-functional attributes for all contained function blocks. These attributes are:
  - chosen BAS technology: wired, wireless, LON, EnOcean, KNX, etc.,
  - device manufacturer name, device mounting form type (cap rail, surface mounting, etc.), device mounting location (intermediate ceiling, ceiling underside, floor, facade, beside door, wall, etc.), device operating voltage, device price cost ($\leq$, $\geq$, $=$), device transmission medium (twisted pair, power line, etc.), etc.,
  - device group label attribute for a group of function blocks to indicate that the function blocks contained in such a group should be mapped on the same target device, if required, without specification of a concrete device type.

Such a mergeable set of function blocks is denoted as a *mergeable function block combination* and guides the solution method towards optimal solutions by allowing only feasible mappings and solutions, which are consistent to the given abstract designs. A mergeable function block combination is called simply function block combination in the remainder of this thesis.

A set that contains all function block combinations, each of which can be realized by a functional profile is denoted as a function block combinations set ($FBCOMBSET$). For each function block set $FBSET_p$ ($p \in [1, j]$) corresponding to a realizing functional profile fp_p ($fp_p \in \{fp_1, ..., fp_j\}$ in (5.6)), a set of function block combinations $FBCOMBSET_p$ can be obtained as shown in (5.7) by performing merging and power set operations, where "$\overset{combination}{\hookrightarrow}$" represents the relation from a function block set to the set of function block combination sets obtained by building of combinations. Non-mergeable function block sets are not included in a function block combination set.

$$\begin{aligned} FBSET_1 &\overset{combination}{\hookrightarrow} FBCOMBSET_1 \\ FBSET_2 &\overset{combination}{\hookrightarrow} FBCOMBSET_2 \\ &... \\ FBSET_j &\overset{combination}{\hookrightarrow} FBCOMBSET_j \end{aligned} \tag{5.7}$$

Each $FBCOMBSET_p$ is thus a set containing mergeable function block combinations from the power set of $FBSET_p$ with all $2^{nofbs} - 1$ combinations of the function blocks in set $FBSET_p$ ($nofbs$: the number of function blocks in $FBSET_p$). Moreover, in this step, the advantage of reducing the combinatorial complexity for power set calculations by introducing the notion of place holder function blocks can clearly be seen. Beyond the acceleration of calculations, in practical cases, where $nofbs$ is greater than 10, computation lasts noticeably longer and after a certain limit the computation appears not to end. By this approach, for such cases a power set calculation is thus made possible. Furthermore, many non-mergeable function block combinations are prevented from being created and number of merge operations is drastically reduced. From (5.6) and (5.7), (5.8) can be concluded, where "$\overset{realizes}{\hookrightarrow}$" represents the relation between a functional profile alternative and possible function block combinations obtained by combination building, each of which can be realized by the functional profile alternative.

$$\begin{aligned} fp_1 &\overset{realizes}{\hookrightarrow} FBCOMBSET_1 \\ fp_2 &\overset{realizes}{\hookrightarrow} FBCOMBSET_2 \\ &... \\ fp_j &\overset{realizes}{\hookrightarrow} FBCOMBSET_j \end{aligned} \tag{5.8}$$

For each function block combination fbc_r $(r \in [1, k])$

$$ALLFBCOMBS = \bigcup_{o=1}^{j} FBCOMBSET_o \tag{5.9}$$
$$fbc_r \in ALLFBCOMBS$$

all possible mapping functional profile alternatives can be obtained in form of a map in (5.10) from (5.8).

$$\begin{aligned} fbc_1 &\overset{realizedby}{\rightarrow} \{fp_{fbc_1,1}, fp_{fbc_1,2}, ..., fp_{fbc_1,\eta}\} \\ fbc_2 &\overset{realizedby}{\rightarrow} \{fp_{fbc_2,1}, fp_{fbc_2,2}, ..., fp_{fbc_2,\theta}\} \\ &... \\ fbc_k &\overset{realizedby}{\rightarrow} \{fp_{fbc_k,1}, fp_{fbc_k,2}, ..., fp_{fbc_k,\psi}\} \end{aligned} \tag{5.10}$$

In (5.10) $\overset{realizedby}{\rightarrow}$ represents the relation from a function block combination to a set of realizing functional profile alternatives. Function block combinations $ALLFBCOMBS = \{fbc_1, fbc_2, ..., fbc_k\}$ contain place holder function blocks, which can be replaced by original function blocks in (5.5). After the replacement (5.11) is obtained.

Set of all function block combinations $ALLFBCOMBS$ contains after the replacement additional function block combinations and it is referred to as $ALLFBCOMBS' = \{fbc_1, fbc_2, ..., fbc_c\}$ with $c - k \geq 0$ denoting the number of additional function block combinations (cf. 5.11).

$$\begin{aligned} fbc_1 &\overset{realizedby}{\rightarrow} \{fp_{fbc_1,1}, fp_{fbc_1,2}, ..., fp_{fbc_1,\eta}\} \\ fbc_2 &\overset{realizedby}{\rightarrow} \{fp_{fbc_2,1}, fp_{fbc_2,2}, ..., fp_{fbc_2,\theta}\} \\ &... \\ fbc_c &\overset{realizedby}{\rightarrow} \{fp_{fbc_c,1}, fp_{fbc_c,2}, ..., fp_{fbc_c,\psi'}\} \end{aligned} \tag{5.11}$$

From (5.11) functional profiles can be rearranged by implementing devices in a map for each function block combination in (5.12) and (5.13), since functional profiles are device and operation mode specific. In (5.12) "$\overset{realizedby}{\rightarrow}$" represents the relation from a function block combination to a device map *mapdev* with realizing manufacturer-specific devices.

$$\begin{aligned} fbc_1 &\overset{realizedby}{\rightarrow} mapdev_{fbc_1} \\ fbc_2 &\overset{realizedby}{\rightarrow} mapdev_{fbc_2} \\ &... \\ fbc_c &\overset{realizedby}{\rightarrow} mapdev_{fbc_c} \end{aligned} \tag{5.12}$$

For $w \in [1, c]$, each map $mapdev_{fbc_w}$ is denoted in (5.13), where "$\overset{impl.}{\rightarrow}$" represents the relation from a function block combination specific device in a device map *mapdev* from (5.12) to realizing functional profile alternatives preimplemented on the device, each of which realizes the function block combination.

$$\begin{aligned} dev_{fbc_w,1} &\overset{impl.}{\rightarrow} \{fp_{dev_{fbc_w,1},1}, fp_{dev_{fbc_w,1},2}, ..., fp_{dev_{fbc_w,1},\epsilon}\} \\ dev_{fbc_w,2} &\overset{impl.}{\rightarrow} \{fp_{dev_{fbc_w,2},1}, fp_{dev_{fbc_w,2},2}, ..., fp_{dev_{fbc_w,2},\rho}\} \\ &... \\ dev_{fbc_w,d} &\overset{impl.}{\rightarrow} \{fp_{dev_{fbc_w,d},1}, fp_{dev_{fbc_w,d},2}, ..., fp_{dev_{fbc_w,d},\tau}\} \end{aligned} \tag{5.13}$$

All the relations and calculations presented up to here are independent of the connections among function blocks. However, some function block combinations are particularly useful in the design improvement for an accelerated search for functional profile pairs that yield bindings. Such function block combinations are built considering the abstract connections among function blocks and called *connection sensitive function block combinations.* Connection sensitive function block combinations $FBC_{CS} \subset ALLFBCOMBS'$ in (5.14) can be obtained by Algorithm 5, if there exists any.

$$FBC_{CS} = \{fbc_{cs_1}, fbc_{cs_2}, ..., fbc_{cs_{csn}}\} \tag{5.14}$$

In Figure 5.3 an example is presented for the steps performed by Algorithm 5 to help the comprehension. In step 1a and 1b function blocks are determined

Algorithm 5: Algorithm for Calculation Of Abstract Connection Sensitive Function Block Combinations

```
Input: Set of abstract design function blocks: $FB$, map for connection
       pairs in $FB$ with each entry from function block of outgoing side
       of the connection to function blocks of the incoming side: $MAP1$
Output: Set of connection sensitive function block combinations:
        $FBCCS$
// initialize $FBCCS$
$FBCCS = \emptyset$;
// begin step 1a and 1b in Figure 5.3
// collect all function blocks that participate abstract connections from
   incoming side
$INFBS = COLLECT\_ALL\_INCOMING\_FBS(MAP1)$;
// collect all function blocks that participate abstract connections from
   outgoing side
$OUTFBS = COLLECT\_ALL\_OUTGOING\_FBS(MAP1)$;
// collect function blocks which are in none of the abstract connections
   on the outgoing side
$ENDFBS = COLLECT\_ALL\_END\_FBS(INFBS, OUTFBS)$;
// calculate the reverse map for $MAP1$ in which each entry is from
   function block of incoming side of the connection to function blocks of
   the outgoing side
// end step 1a and 1b in Figure 5.3
$MAP2 = REVERSEMAP(MAP1)$;
// step 2-4 in Figure 5.3
for all $endfb \in ENDFBS$ do
    // calculate connection group starting from $endfb$ adding all
       connection partner function blocks from the outgoing side
       recursively and build a sensitive function block combination
    $fbccs = COLLECT\_CONN\_GRP(endfb, MAP2)$;
    // add $fbccs$ to $FBCCS$
    $FBCCS = FBCCS \cup \{fbccs\}$;
end
```

that represent initial data sources (sensor and operation functions) and final data sinks (actuator functions denoted as *endfb* in Algorithm 5) in a given abstract design by navigating through function blocks via their abstract connections from outports to inports and in the opposite direction.

In steps 2 to 4 a stepwise navigation is performed through abstract connections starting from final data sink function blocks (obtained in step 1b) in the set $ENDFBS$ to their connection partner incoming function blocks in the opposite direction of abstract connections. The navigation proceeds until the function blocks determined in step1a are reached. For each data sink function block function blocks visited in the navigation are grouped. Initial data sources obtained in step 1a are not included in the group. In the 4. step the final group emerges that is considered as a connection sensitive function block combination.

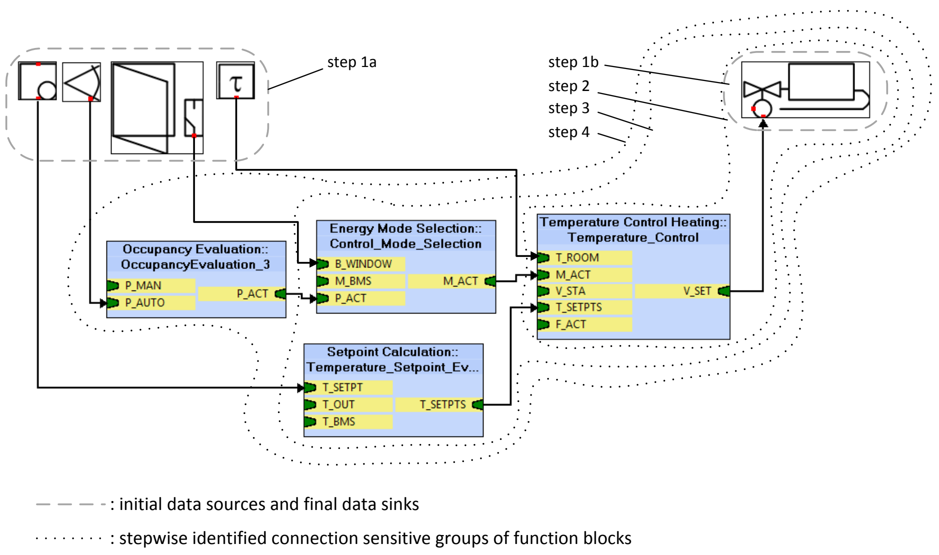


Figure 5.3: Stepwise Identification of Connection Sensitive Function Blocks Groups of a Heating Example

Examples for connection sensitive function block combinations are illustrated (fbc_{cs1}, fbc_{cs2} and fbc_{cs3}) in Figure 5.4 with fbc_{cs3} as the connection sensitive function block combination calculated in Figure 5.3. There exists at least one functional profile for each of these function block combinations that can realize the covered function blocks simultaneously in a specific operation mode. Connection sensitive function block combinations are determined, since a consideration of such combinations can help to accelerate the convergence speed of design improvement algorithms in variation operations by guiding them towards only significant mapping alternatives among function block combinations and functional profiles. Two connection insensitive function block combinations among many other such function block combinations that otherwise would repeatedly be considered by the variation operations are:

$$fbc_{connins_1} = \{fb_1, fb_8\}$$
$$fbc_{connins_2} = \{fb_4, fb_9\}$$

According to the example in Figure 5.4 with fbc_{cs1}, fbc_{cs2} and fbc_{cs3} abstracted, a functional profile that maps $fbc_{connins_1}$ builds a binding pair candidate (cf. Definition 3.2.5) with the functional profile that maps fb_7, which does not

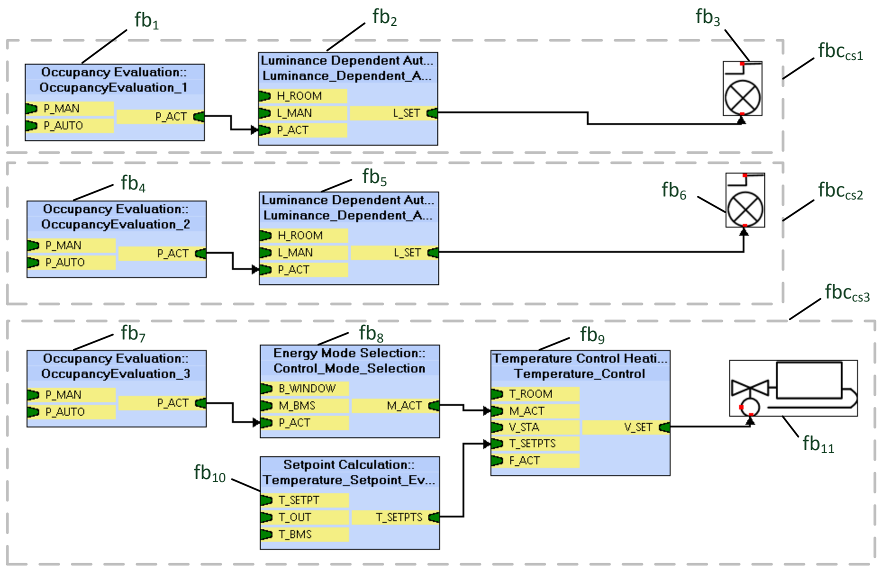


Figure 5.4: Example Connection Sensitive Function Block Combinations for a Lighting and Heating Example in Figure 6.1

achieve the desired binding and in many cases, no binding emerges. Further, such a constellation in this particular example is also semantically incorrect, since according to this scheme, a functional profile that maps fb_1 is required to build bindings with a functional profile that maps its connection partner fb_2 or with another functional profile that maps both fb_2 and its second connection partner fb_3. Connection insensitive function block combinations are not forbidden and are also allowed in variation operations. The reason for this is that there are possibly useful connection insensitive function block combinations e.g. a function block combination that covers fb_1, fb_4 and fb_7, if they were from three different function block types and mergeable. However connection sensitive function block combinations assist the variation operations to achieve a faster convergence to optimal solutions. Hence, both alternatives are considered.

For each function block in the abstract design, all possible function block combination alternatives it participates can be obtained in form of a map in (5.15) from (5.12). In (5.15) $\overset{containedin}{\rightarrow}$ represents the relation between a function block and all function block combinations, each of which contains the function block.

$$\begin{aligned}
&fb_1 \overset{containedin}{\rightarrow} \{fbc_{fb_1,1}, fbc_{fb_1,2}, ..., fbc_{fb_1,\iota}\}\\
&fb_2 \overset{containedin}{\rightarrow} \{fbc_{fb_2,1}, fbc_{fb_2,2}, ..., fbc_{fb_2,\kappa}\}\\
&...\\
&fb_i \overset{containedin}{\rightarrow} \{fbc_{fb_i,1}, fbc_{fb_i,2}, ..., fbc_{fb_i,\chi}\}
\end{aligned} \tag{5.15}$$

Functional profile retrievals and obtained results are saved in a memory to make this information available throughout the design generation and improvement. This prevents repeated retrievals hence reduces overhead and speeds up the overall process.

Yet a proper selection of a set of function block combinations can contain all function blocks of the abstract design FB (cf. (5.1)) without repetition of any function block. In fact, there can exist many such sets of function block combinations that can cover the set of function blocks exactly. Each such set of function block combinations is called a *pattern* in the context of this thesis and should not be confused with the usage of the notion of patterns in Table 2.1. The notion of patterns is highly relevant for the design creation problem, since with this definition a consistency among given abstract designs and generated detailed designs can be provided.

In order to help the comprehension of the relations and calculations from (5.2) to (5.15), the example abstract design in Figure 2.8 is chosen and reillustrated with reduced and enumerated function blocks. For the sake of simplification sets of retrieved functional profiles are also reduced.

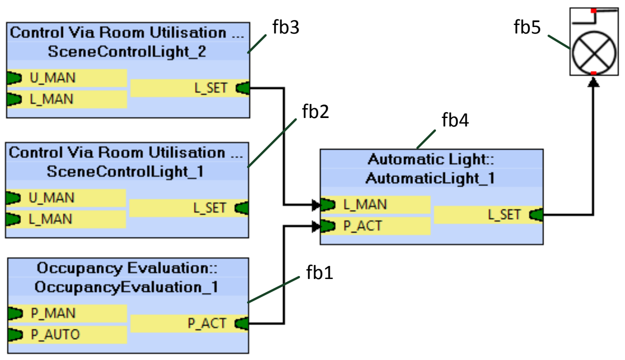


Figure 5.5: Constant-Light Control and Automatic Lights in Figure 2.8 with Reduced and Enumerated Function Blocks

The set of function blocks in Figure 5.5 is:

$$FB = \{fb_1, fb_2, fb_3, fb_4, fb_5\}. \tag{5.16}$$

Component Retrieval for the Example in Figure 5.5: After a search in component repository (5.17) is obtained with device and operation mode specific functional profiles.

$$\begin{aligned} fb_1 &: \{fp_1, fp_2, fp_3, fp_4\} \\ fb_2 &: \{fp_5, fp_6, fp_7\} \\ fb_3 &: \{fp_5, fp_6, fp_7\} \\ fb_4 &: \{fp_1, fp_3, fp_8, fp_9\} \\ fb_5 &: \{fp_2, fp_3, fp_8, fp_{10}\} \end{aligned} \tag{5.17}$$

Component Arrangement for the Example in Figure 5.5: It is observed that the sets of functional profiles for fb_2 and fb_3 are identical, thus these function blocks are grouped in (5.18) and refered to as a place holder function block $fb_{pholder}$.

$$fb_{pholder} \overset{ph}{\to} \{fb_2, fb_3\} \tag{5.18}$$

From (5.17) and (5.18) function block sets can be obtained in (5.19) which can be realized simultaneously by each of these functional profiles.

$$\begin{aligned}
fp_1 &\diamond \{fb_1, fb_4\}\\
fp_2 &\diamond \{fb_1, fb_5\}\\
fp_3 &\diamond \{fb_1, fb_4, fb_5\}\\
fp_4 &\diamond \{fb_1\}\\
fp_5 &\diamond \{fb_{pholder}\}\\
fp_6 &\diamond \{fb_{pholder}\}\\
fp_7 &\diamond \{fb_{pholder}\}\\
fp_8 &\diamond \{fb_4, fb_5\}\\
fp_9 &\diamond \{fb_4\}\\
fp_{10} &\diamond \{fb_5\}
\end{aligned} \tag{5.19}$$

As it can clearly be observed, functional profiles can cover a different number of function blocks which is a consequence of the device manufacturers' preferences on the implementation. After building power sets of function block sets in (5.19) and merging, (5.20) is obtained, which provides every possible mapping of function block combinations to device and operation mode specific functional profiles. Function blocks fb_2 and fb_3 referred by $fb_{pholder}$ cannot be merged together to common set of properties due to the repetition of function block types of functions (both function blocks are from type "Control Via Room Utilization"), thus the function block combination $\{f_2, f_3\}$ cannot be created.

$$\begin{aligned}
fbc_1 &= \{fb_1\} & &\overset{realizedby}{\to}\{fp_1, fp_2, fp_3, fp_4\}\\
fbc_2 &= \{fb_1, fb_4\} & &\overset{realizedby}{\to}\{fp_1, fp_3\}\\
fbc_3 &= \{fb_1, fb_5\} & &\overset{realizedby}{\to}\{fp_2, fp_3\}\\
fbc_4 &= \{fb_1, fb_4, fb_5\} & &\overset{realizedby}{\to}\{fp_3\}\\
fbc_5 &= \{fb_{pholder}\} & &\overset{realizedby}{\to}\{fp_5, fp_6, fp_7\}\\
fbc_6 &= \{fb_4\} & &\overset{realizedby}{\to}\{fp_1, fp_3, fp_8, fp_9\}\\
fbc_7 &= \{fb_4, fb_5\} & &\overset{realizedby}{\to}\{fp_3, fp_8\}\\
fbc_8 &= \{fb_5\} & &\overset{realizedby}{\to}\{fp_2, fp_3, fp_8, fp_{10}\}
\end{aligned} \tag{5.20}$$

Due to the simplicity of this example the advantage of performing the operations so far on a reduced set of function blocks cannot be shown. The number of function blocks that yield same retrieval results is only two and this would not cause a high combinatorial complexity in the power set calculation. In Chapter 6, design improvement step for Example2 profits from this advantage by achieving the optimal solutions much faster than when the approach is not followed.

After replacement of $fb_{pholder}$ by fb_2 and fb_3 in (5.20), an increased number of function block combinations in (5.21) is obtained.

$$\begin{aligned}
&fbc_1 = \{fb_1\} && \overset{realizedby}{\rightarrow} \{fp_1, fp_2, fp_3, fp_4\}\\
&fbc_2 = \{fb_1, fb_4\} && \overset{realizedby}{\rightarrow} \{fp_1, fp_3\}\\
&fbc_3 = \{fb_1, fb_5\} && \overset{realizedby}{\rightarrow} \{fp_2, fp_3\}\\
&fbc_4 = \{fb_1, fb_4, fb_5\} && \overset{realizedby}{\rightarrow} \{fp_3\}\\
&fbc_5 = \{fb_2\} && \overset{realizedby}{\rightarrow} \{fp_5, fp_6, fp_7\}\\
&fbc_6 = \{fb_4\} && \overset{realizedby}{\rightarrow} \{fp_1, fp_3, fp_8, fp_9\}\\
&fbc_7 = \{fb_4, fb_5\} && \overset{realizedby}{\rightarrow} \{fp_3, fp_8\}\\
&fbc_8 = \{fb_5\} && \overset{realizedby}{\rightarrow} \{fp_2, fp_3, fp_8, fp_{10}\}\\
&fbc_9 = \{fb_3\} && \overset{realizedby}{\rightarrow} \{fp_5, fp_6, fp_7\}
\end{aligned} \tag{5.21}$$

After application of (5.12) for the example, (5.22) is obtained.

$$\begin{aligned}
&fbc_1 \overset{realizedby}{\rightarrow} mapdev_{fbc_1}\\
&fbc_2 \overset{realizedby}{\rightarrow} mapdev_{fbc_2}\\
&\dots\\
&fbc_9 \overset{realizedby}{\rightarrow} mapdev_{fbc_9}
\end{aligned} \tag{5.22}$$

An example for the maps $mapdev_{fbc_1}$ is given in (5.23), in which the indexes are simplified corresponding to the retrieval results.

$$\begin{aligned}
&dev_1 \overset{impl.}{\rightarrow} \{fp_1, fp_3\}\\
&dev_2 \overset{impl.}{\rightarrow} \{fp_2, fp_4\}\\
&dev_3 \overset{impl.}{\rightarrow} \{fp_1, fp_2\}
\end{aligned} \tag{5.23}$$

Furthermore, (5.24) is obtained to denote the participated function block combinations for each function block of the example.

$$
\begin{aligned}
fb_1 &\overset{containedin}{\rightarrow} \{fbc_1, fbc_2, fbc_3, fbc_4\} \\
fb_2 &\overset{containedin}{\rightarrow} \{fbc_5\} \\
fb_3 &\overset{containedin}{\rightarrow} \{fbc_9\} \\
fb_4 &\overset{containedin}{\rightarrow} \{fbc_2, fbc_4, fbc_6, fbc_7\} \\
fb_5 &\overset{containedin}{\rightarrow} \{fbc_3, fbc_4, fbc_7, fbc_8\}
\end{aligned}
\tag{5.24}
$$

The information present in (5.21), (5.22), (5.23) and (5.24) is particularly crucial for the design improvement. Storage of this information in corresponding data structures not only allows an acceleration of the improvement process, but also provides the variation operations a utility to access the required information for the functionality and in the required form (data structure) without the need to perform any transformation of information representation.

The application of preoptimization steps guarantees correct mappings and coverage of function blocks by functional profiles and the information on mappings can guide optimization algorithms towards near-optimal solutions.

5.4.2 Generation Approaches

After retrieval of all realizing functional profiles from different devices and manufacturers for each function block in the abstract design, there emerges a huge variety functional profile alternatives with which a high number detailed designs can be built (cf. Figure 5.6). The design creation can be performed by generation constructively or by a complete mapping of the required room automation plan provided on a design, or by substituting devices or functional profiles within an existing design. The latter approach is handled in Section 5.5 and the first two approaches is discussed in this section.

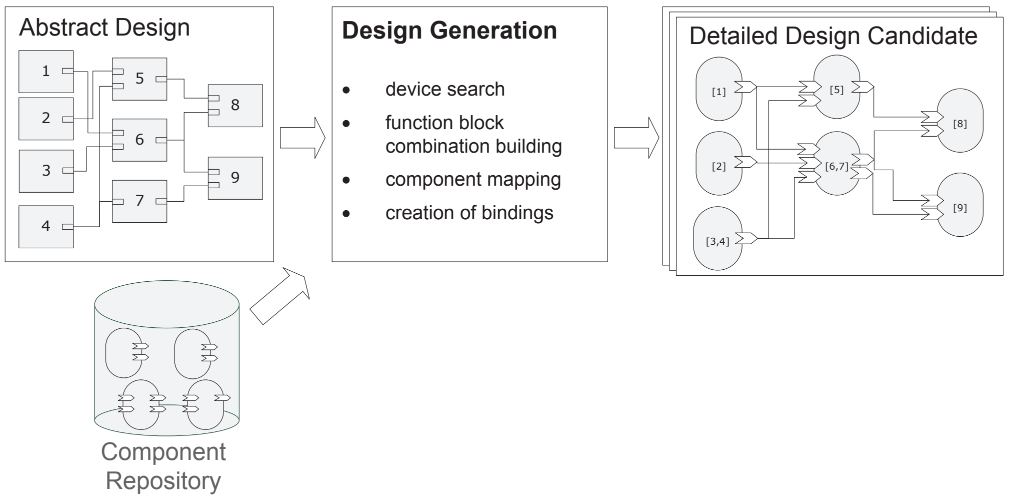


Figure 5.6: Design Generation

For the decision, which function block combination is to map by which functional profile and the order of consideration of function block combinations, there exist two approaches, constructive and holistic (improving) approach:

- **constructive approach:** This approach conceives the generation of a single solution constructively, component-by-component, while optimizing. Thus, in each step, evaluation of each chosen component for its measure of suitability in the so far constructed design, a local view for the evaluation (or a partial evaluation) is considered. Evaluation of the complete design, a global view for the evaluation is out of consideration in the constructive approach.

  Initially a functional profile is chosen that realizes a function block combination according to a design quality criterion, e.g. choose functional profile of the cheapest realizing device. In the next step, connection partners of the function blocks in the first function block combination are considered. Function block combinations of the connection partner function blocks are identified and they are mapped onto functional profiles, if possible from the previously chosen device, unless from the cheapest possible device. In this step binding pair candidates are built.

  On one hand, choice of the functional profiles from the cheapest devices may not necessarily yield to valid bindings. On the other hand, due to the enormous number of functional profile alternatives, it is not practicable to evaluate interoperability for all possible functional profile alternatives to find a fitting functional profile for the functional profile chosen in the first step, that is at the same time on a cheapest possible device. Choosing the first fitting functional profile or a randomly chosen fitting functional profile may lead to expensive, suboptimal detailed design solutions. Moreover, even if a state is reached in which most of the function block combinations are realized by functional profiles of cheap devices and binding pair candidates build valid bindings, if no fitting functional profile can be found for the last considered function block combination, all previous decisions must be reviewed. In worst case the procedure has to be restarted from very beginning. Another constructive approach can be applied by adapting the approach used in ACO, where ants travel around a given graph to find the optimal path from the source node to the final node.

In this approach, there must be a fixed set of nodes, each of which is a functional profile that satisfies a function block combination in the abstract design. However, there are multiple function block combination alternatives to which a single function block may belong. In order to enable such an approach, there is need to span a graph that connects all possible decisions for functional profile alternatives that satisfy all possible function block combination alternatives. This implies that there is not a fixed set of nodes and multiple decision graphs can be constructed instead of a single one as in the case of ACO solving the TSP. Since there can be thousands of functional profile alternatives for a function block, each such graph can contain an extremely high number of nodes. Depending on the number of different function block combination alternatives, thus number of different patterns (cf. Section 5.4.1) the number of such graphs can also end up to be very high. Such a problem-specific setup procedure for the solution method is itself very complex;

- **holistic randomizing approach:** This approach is more intuitive than the constructive approach and is used by the problem-specific multi-objective evolutionary algorithm to generate the initial population. This approach conceives a complete mapping of a given abstract design onto the detailed design alternatives. The approach is subdivided into two different approaches: holistic randomizing approach by precalculation of all possible patterns and without precalculation of all possible patterns;

  - **holistic randomizing approach by precalculation of all possible patterns:** In this approach, all possible sets of function block combinations that can cover the set of function blocks exactly, thus all possible patterns are calculated.

    The problem of obtaining exact covering sets of a given set is called the *exact cover problem* [Kar72]. In the literature there exists a variety of solution methods that solve the exact cover problem. The most known of these methods is the Algorithm X by Donald Knuth, which is a recursive, non-deterministic, depth-first, backtracking algorithm. The algorithm requires the entry of the input data in form of a matrix with column headers representing the elements of input set and row headers the subsets of the collection. The matrix is called the sparse matrix, since for each row node that contains the node represented by the column header a “1” is inserted and rest places in the matrix remain empty. The column and raw nodes are doubly linked as circular links via pointers. By using a technique called Dancing Links that can uncover and cover the nodes without deletion, Algorithm X finds all the solutions for an exact cover problem for a given input set and a collection of subsets. The algorithm performs an exhaustive search and has successfully been used to solve puzzle problems such as pentomino and sudoku to efficiently compute all possible solutions. A detailed explanation of the algorithm is presented in [Knu00].

In order to adapt the dancing links algorithm to the problem of obtaining all patterns exactly cover FB, a sparse matrix is prepared with function blocks and function block combinations (cf. Table 5.2). Each cell denotes a pair of function block in (5.1) and function block combination in (5.10). Each such cell is filled with “1”, if the function block combination contains its paired function block.

	fbc_1	fbc_2	...	fbc_k
fb_1				
fb_2				
...				
fb_i				

Table 5.2: Sparse Matrix for Obtaining All Patterns (unfilled)

By applying the dancing links algorithm, all patterns can be obtained. Experiments done for different abstract designs have shown that patterns are computed very quickly (in milliseconds) using a contemporary PC. Application of this approach allows a calculation of the variety of detailed design solution candidates, which can give an idea about the complexity of a considered problem instance.

For a problem instance with the number of different patterns M and an EA with population size N, three approaches can be followed for these cases:

* $M = N$: an individual is generated corresponding to each pattern starting from pattern with the lowest cardinality (maximum covering) proceeding with patterns of lower cardinalities in order.
* $M < N$: individuals are generated considering each pattern as in the first case. Following a rotational order, after all patterns are considered, the design generation restarts from the pattern with the lowest cardinality and proceeds similarly.

    * $M > N$: individuals are generated similar as in the first case corresponding to N patterns and $M - N$ patterns remain unvisited in the initialization. Optimization algorithm must be able to generate remaining patterns.

  The number of all possible function block combinations obtained after power set calculations can be very high for some abstract designs and the sparse matrix can be very large, not only due to the high number of contained columns and rows, but particularly due to the high number of contained 1's. In such situations the dancing links algorithm can become inconvenient. Performing calculations on a reduced set of function blocks and thus function block combinations can also decrease the combinatorial complexity for dancing links algorithm, only if such a reduction significantly reduces the number of function blocks to be used in calculations, e.g. down to half of the original number of function blocks;

  – **holistic randomizing approach without precalculation of all possible patterns:** In this approach, function block combinations in (5.11) are randomly chosen ensuring that all function blocks in (5.1) are exactly covered by the chosen function block combinations. There emerges a pattern of a random cardinality from which a detailed design solution candidate can be created. The approach is realized by Algorithm 6.

Hereby, in each pattern there is a set of function block combinations, each of which can be mapped to a realizing functional profile alternative. In this step, a random functional profile is chosen. By iterative application, all functional block combinations are mapped by chosen functional profiles.

For a list of all selected functional profiles of a solution candidate, a detailed design can be built (cf. Definition 3.2.3) by determining all binding partner functional profiles (binding pair candidates), connecting functional profiles by bindings where possible, and forming a device list from the chosen functional profiles.

Algorithm 6: Algorithm of Holistic Randomizing Approach Without Precalculation of All Possible Patterns

```
Input: All possible function block combinations: $FBC$, set of abstract
       design function blocks: $FB$
Output: A pattern as a set of function block combinations: $PAT$
// initialize a temporary function block combination set $FBCTEMP$
$FBCTEMP = FBC$;
// initialize set of pattern
$PAT = \emptyset$;
// initialize set of function blocks which are already added
$ADDEDFB = \emptyset$;
while $ADDEDFBs! = FBs$ do
    // choose a random function block combination
    $fbc = CHOOSE\_RANDOM\_FBC(FBCTEMP)$;
    // add the chosen function block combination to the pattern that is
       being built
    $PAT = PAT \cup \{fbc\}$;
    // add function block set $FB_{fbc}$ of $fbc$ to $ADDEDFB$
    $ADDEDFB = ADDEDFB \cup FB_{fbc}$;
    // remove all elements in function block combinations that contain
       any of $ADDEDFB$
    $FBCTEMP = REMOVE\_FBCS(ADDEDFB)$;
end
```

For ease of comprehension of Algorithm 6 the example in Figure 5.5 is reconsidered with related function block combinations in (5.21) and mappings of function blocks to function block combinations in (5.24). For this example iterations of the Algorithm 6 are presented in Table 5.3 starting by the initialization of $FBCTEMP$ with the function block combination set $\{fbc_1, ..., fbc_9\}$. Set of function blocks that are already added ($ADDEDFB$) is initialized as an empty set. Column fbc represents the randomly chosen function block combinations in each iteration from 1 to 4 from the current $FBCTEMP$. Column FB_{fbc} represents the set of function blocks covered by fbc that can be obtained from (5.21). Column $fbc's\ to\ remove$ represents the function block combinations in $FBCTEMP$ that realize any of the function blocks in FB_{fbc}. These function block combinations can be obtained from (5.24). Column $FBCTEMP$ represents current content of the function block combination set in an iteration after the removals. Column $ADDEDFB$ represents the function blocks which are already added. In the fourth and the last iteration, the set $ADDEDFB$ is identical to the set of function blocks $FB = \{fb_1, ..., fb_5\}$ and the loop stops. The union of function block combinations in Column fbc forms the pattern PAT as the result of the algorithm. As can be seen, this algorithm supports an exact cover of the function blocks of a given abstract design by suitable choices of function block combinations.

Iter.	$\boldsymbol{fbc}$	$\boldsymbol{FB_{fbc}}$	$\boldsymbol{fbc's\ to\ remove}$	$\boldsymbol{FBCTEMP}$	$\boldsymbol{ADDEDFB}$
init.	-	-	-	$\{fbc_1, ..., fbc_9\}$	$\emptyset$
1	fbc_3	$\{fb_1, fb_5\}$	$fbc_1, fbc_2, fbc_3, fbc_4, fbc_7, fbc_8$	$\{fbc_5, fbc_6, fbc_9\}$	$\{fb_1, fb_5\}$
2	fbc_9	$\{fb_3\}$	fbc_9	$\{fbc_5, fbc_6\}$	$\{fb_1, fb_3, fb_5\}$
3	fbc_6	$\{fb_4\}$	$fbc_2, fbc_4, fbc_6, fbc_7$	$\{fbc_5\}$	$\{fb_1, fb_3, fb_4, fb_5\}$
4	fbc_5	$\{fb_2\}$	fbc_5	$\emptyset$	$\{fb_1, fb_2, fb_3, fb_4, fb_5\}$

Table 5.3: Example Iterations of Algorithm 6 for Randomly Chosen Function Block Combinations

The holistic approach can often lead to initialization of solutions with missing bindings and/or an expensive price sum of devices. A normally distributed randomization of the initialization process aims to start at different regions of the search space.

Another approach can consider a start with preoptimized solutions by providing a mapping of function blocks onto functional profiles of cheapest possible devices or a mapping of function blocks onto devices with maximum possible coverage. This would lead to a start with a less number of devices. However, these devices can not necessarily be interoperable for the realization of the mapped functions, hence there would emerge many missing bindings. Creation of preoptimized solutions with interoperable devices cannot be performed by considering each device separately, since interoperability involves a pair of devices and their functional profiles used in specific operation modes. There emerges a very high number of operation mode specific functional profile pairs for evaluation of interoperability. This can turn the preoptimization step to an exhaustive search, which must be avoided. In addition, it cannot be concluded that the initialization with such preoptimized devices would enable an optimization start at promising regions of the search space considered globally for all objectives simultaneously.

A mix of the holistic randomized and holistic preoptimizing approaches can create a start population by creating half of the individuals with one approach and the other half with the other approach. Such a mixed approach has been experimented. Since individuals created by the latter approach have a far better start than the individuals created by the first approach, they were not as competitive as the preoptimized individuals and were quickly dominated, hence vanished from the population during the steps of design improvement. The remaining preoptimized individuals have shown only little improvement and lead to invalid solutions with many missing bindings, since these solutions were located in a very tiny region of the search space and modifications could not lead them to move to other interesting regions of the search space in a reasonable number of generations. Therefore, use of such an approach turns the global search into a local search and a local search aims at achieving local optimal solutions. However, the requirement for a global search (cf. Section 3.3.2) must be fulfilled. By considering these facts and avoiding the mentioned risks in the holistic randomizing approach with precalculation of all possible patterns, the concentration is on the holistic randomizing approach without precalculation of all possible patterns. The main course of design optimization is in the design improvement step in Section 5.5 and it follows the design generation.

5.5 Design Improvement

Under consideration of motivations listed in Section 5.4.1, algorithms defined for the optimization must overcome certain problems to improve the generated detailed designs towards near-optimal solutions. These algorithms are integrated into the variation operations of the optimization algorithm. Variation operations must deal with the problems introduced in Section 5.5.1 by approaching to these problems fulfilling the derived requirements.

Problems of engineering practice are similar to these presented problems to some extent, thus they are presented with specific paradigms in the building automation domain, as well as in a general theoretical consideration to provide a perspective to the readers who are searching for effective solution methods for their own problems in other problem domains.

5.5.1 Problems and Requirements

The design generation following the holistic approach allows an exact coverage of the function blocks by the initially generated detailed designs which plays an important role for abstract design compliancy, however this does not guarantee the compliancy. Functional profiles that exactly cover the function blocks must also possess input and output datapoints that realize the inports and outports of the covered function blocks. Although, this condition is stated in Section 5.3.1 as a presumption, functional profile implementations of some manufacturers deviate from the function block definitions in the standard VDI 3813-2 [The11b]. E.g. functional profiles of such manufacturers match functional semantics of function blocks, however matching of datapoints to ports may not be possible. If no standard conform alternatives for such components are provided, invalid detailed designs must be taken in account due to missing bindings in generated binding-schemata. This is a problem related to the implementation of components or incorrect definitions in the component data source and therefore, it is not addressed by the variation operations.

Problems or challenges that need to be specified by the variation algorithms of the multi-objective evolutionary optimization method after the generation of initial solutions and at intermediate steps of the optimization can be subdivided into two problems, *domain specific problems* and *problems related to variation algorithms*:

- **domain specific problems:**
  - **missing bindings:** Connection partner function blocks covered by different functional profiles imply bindings between these functional profiles. However, in some cases no bindings can be created due to the existence of non fitting functional profiles (cf. Definition 3.2.7). Additionally, some mandatory input datapoints of a functional profile (cf. Definition 3.2.6) in a generated binding-schema may not possess bindings. This can occur for a functional profile, even if it can build bindings for all binding pair candidates (cf. Definition 3.2.5), in which it participates. Non fitting functional profiles and functional profiles with missing bindings at mandatory input datapoints may both be encountered, if there are function block combinations in a pattern chosen to build an individual, that can only be mapped to such functional profiles;
  - **expensive devices:** The price sums of devices of the initially generated solutions are usually high. Although, device prices of the optimal solution are not known, variation algorithms must tend to generate new solutions by replacement of expensive devices of detailed designs with cheaper devices;
  - **low coverage of functional profiles:** Devices in a binding-schema may cover lesser function blocks than they actually can. This results in more devices to realize an abstract design than necessary. Variation operations must identify such cases and reduce the number of devices of such a detailed design by replacing such devices by alternative devices that can cover more function blocks than existing devices. Reducing the number of devices can lead to the reduction of the price sum of devices and device external bindings (cf. Section 5.2). However, the alternative devices may not be interoperable with their communication partners and that leads to an increase of the number of missing bindings.

  At any step of the optimization, the listed problems may occur in a solution candidate, and in worst case, they may occur at the same time. It is clear that variations, which perform blind searches based on random replacement of functional profiles and devices, are ineffective and would have a low chance to approach to near-optimal solutions considering the huge variety of solution candidates. Variation operations that are designed to directly address each of these problems will have to solve with certain problems in order to perform a more effective search;

- **problems related to variation algorithms:**

- **information sharing and social structure:** Information sharing among the individuals of a population in an evolutionary algorithm is performed by cross-over operations. Hereby, an individual can mate any other individual in the population and this can make a contribution to generate further improved solutions by combining best parts of both individuals which helps a quick convergence of the optimization. Hence, the social structure hereby is the population itself. However, in intermediate generations less optimized solutions may be attracted to better optimized solutions and tend to improve towards the attraction by receiving highly improved solution parts or building blocks and genetic drift occurs. Due to genetic drift, diversity in the population decreases and the evolution process starts revisiting same few different solutions over many iterations without considerable improvement;

  practical consideration: Functional profiles of interoperable devices and devices with low price costs or with a high coverage of functional profiles in improved individuals are highly fit individual parts and attraction points for less improved individuals in a population. Inclusion of such highly fit individual parts will cause less improved solutions to be more similar to better improved solutions and in worst case to be identical. As a result, variety of functional profiles and devices in a population sinks. If this occurs before the achievement of near-optimal solutions, the optimization algorithm will get stuck in premature solutions;
- **accessibility of solution candidates in the search space:** Poorly designed variation algorithms can generate often infeasible solutions inconsistent to given problem instances which requires application of computational cost intensive repair operations. This is due to the lack of problem specific information that must be provided in the variation algorithms. In this case, algorithms turn to perform a blind search, which is very inefficient and highly non deterministic. As a result, only poor quality solutions can be achieved in a limited computation time and final solutions of different runs vary from each other at a high rate.

  Furthermore, a poor construction of variation algorithms cannot reach at different regions of the search space and thus cannot exhibit a good exploration behavior. In worst case, improvements are limited to the nearest neighborhood of the initial solutions.

  Another important issue is the size of moves that variation algorithms can consider. If only small moves are performed from one solution candidate to another, created solutions may be dominated by so far better optimized solutions and this fact leads to a decrease of the number of competitive solutions which very probably yields to the prevention of current best solutions from further improvement. Same result can be encountered, if only large moves are performed. Permanent application of large moves can achieve a premature local optima that can cause genetic drift and prevent further improvements. Similar problems have been addressed by the SA algorithm;

practical consideration: Variation algorithms that generate infeasible solutions which are solutions with functional profiles and devices that do not cover function blocks of the given abstract design are considered as poorly designed. Such infeasible solutions are achieved, if function blocks are mapped to functional profiles which can not realize them. Replacement of a single functional profile is considered as smallest possible move and replacement of multiple functional profiles is considered as a large move. Thus, size of a move is determined by the number of replaced functional profiles. Permanent application of small moves or very large moves may yield to achievement of premature solutions that cannot be further improved;

– **calculations of component statistics:** For generation of improved solutions some calculations on components can be performed to replace current individual parts with better fitting components in form of a local search. An intensive search for better suiting components would slow down the algorithm drastically and can introduce premature local optima as a result of a genetic drift. In worst case, an NP time complexity can be encountered;

  practical consideration: In order to replace non fitting functional profiles in an individual with fitting functional profiles (cf. Definition 3.2.7), a search can be performed to identify non fitting functional profiles and replace them with other functional profile pairs and potentially with fitting functional profiles. However, it may be necessary to perform binding evaluation with all candidates for the replacement in worst case. Since, there exists a very high number of candidates, such an attempt would result in an exhaustive search that must be avoided in variation operations. This case may be encountered also at an exhaustive search for cheaper devices or devices with a better functional profile coverage and must be avoided as well.

From these problems, following requirements on the design of variation algorithms can be derived: Variations must address all of the problems and must be able to

RV1. identify functional profiles with missing bindings and replace them with different functional profiles,

RV2. identify expensive devices in a design and replace them with other devices covering the same function blocks, RV3. identify devices that can cover more function blocks than the ones they cover currently and improve the coverage,

RV4. identify candidates for replacement and coverage improvement concerning functional profiles and devices in RV1-3 quickly,

RV5. keep consistency between the given abstract design and solution candidates, on which they are applied, and avoid infeasible solutions,

RV6. reach any solution of the search space by performing different size of moves and by introducing diversity.

A multi-objective evolutionary algorithm that possesses variation algorithms that fulfill these requirements can effectively solve the presented problems.

5.5.2 Variations

Variation operations of an Evolutionary Algorithm (EA) attempt to improve solutions iteratively by exploration in the search space. Search space, as it is generally defined, contains all components from which solutions can be created, including also solutions potentially with irrelevant components for a given problem-instance. Such components are functional profiles and devices that do not cover any of the function blocks in a given abstract design. Solutions that contain at least one such component is considered an *infeasible solution*. In order to introduce clarity to the definition of search space and to indicate the search regions for variation operations, two nested subspaces are defined within the search space: *solution space* and *component space* as illustrated in Figure 5.7 in form of a Venn diagram.

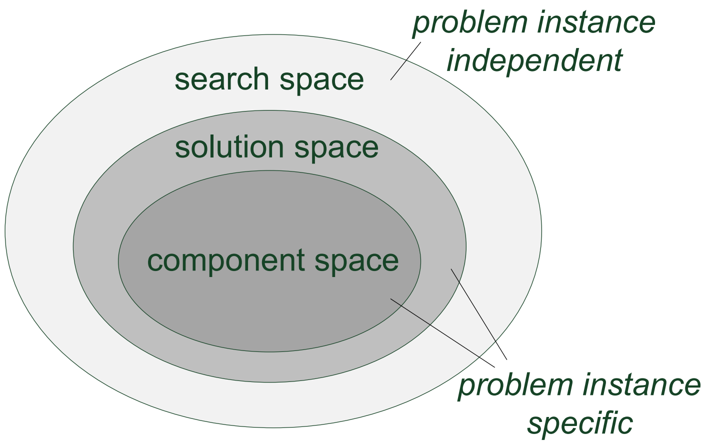


Figure 5.7: Search Space, Solution Space and Component Space

Solution space is the space of all possible problem instance specific detailed design candidates with each detailed design candidate matching the covering function blocks of a given abstract design. Solution space implicitly contains the functional profiles and devices which build such detailed design candidates. Component space is the space of functional profiles and devices contained by detailed design candidates in a specific iteration of the optimization algorithm, thus in terms of EA it corresponds to a population.

In an EA exploration or diversification can be achieved by introducing new individuals to the population. Such individuals contain parts that do not exist in a current population, thus other devices and functional profiles that do not exist in the component space. Exploitation or intensification can be achieved by introducing good fitting components according to some problem specific criteria. Two variation strategies are introduced that can be classified by their exploring and exploiting nature:

- **component space variation (CSV):** aims to modify the distribution of devices and functional profiles among the individuals without introducing other devices and functional profiles than the ones that exist in the component space (thus without diversification in the gene pool). Hereby, the main objective is to explore the component space;
- **solution space variation (SSV):** aims to introduce other devices and functional profiles than the ones that exist in the component space for exploration in the solution space (to perform intensification in the gene pool).

In order to realize requirement RV1-3, three rankings are proposed to identify certain components in a solution candidate SC: $ranking_1$, $ranking_2$ and $ranking_3$.

Hereby, SC contains a binding-schema BS and a corresponding set of devices $DEVS$. BS contains a set of functional profiles FP.

$$\begin{aligned} FP &= \{fp_{bs,1}, fp_{bs,2}, ..., fp_{bs,z}\} \\ DEVS &= \{dev_{sc,1}, dev_{sc,2}, ..., dev_{sc,y}\} \end{aligned} \tag{5.25}$$

Binding pair candidates in which a functional profile $fp_{bs,i}$ ($i \in [1,z]$) participate $BPC_{fp_{bs,i}}$ denote the bindings that $fp_{bs,i}$ is required to build for the realization of abstract connections. A metric m_1 can quantify binding requirement realization rate of an $fp_{bs,i}$ that realizes a set of bindings $BIN_{fp_{bs,i}}$ in BS by the proportion: number of realized bindings divided by number of required bindings. Another metric m_2 can quantify functional profile coverage rate of a device $dev_{sc,j}$ ($j \in [1,y]$) in BS by the proportion: number of used functional profiles of $dev_{sc,j}$ ($FP_{dev_{sc,j}}$) divided by number of all functional profiles in BS.

$$\begin{aligned} m_1 &= \frac{|BIN_{fp_{bs,i}}|}{|BPC_{fp_{bs,i}}|} \\ m_2 &= \frac{|FP_{dev_{sc,j}}|}{z} \end{aligned}$$

- *Ranking*$_1$**:** Functional profiles are sorted in an ascending order according to the metric m_1.This ranking list intentionally contains also functional profiles that satisfy all required bindings, since variation operations that make use of this ranking must still be able to achieve improvements in advanced stages of design improvement with individuals that possess valid and complete binding-schemata.
- *Ranking*$_2$**:** Devices are sorted in a descending order according to the price costs.
- *Ranking*$_3$**:** Devices are sorted in an ascending order according to the metric m_2.

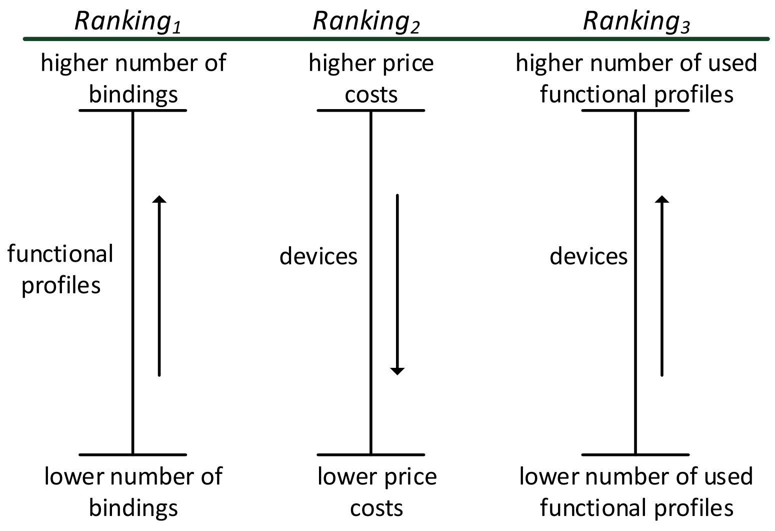


Figure 5.8: Rankings Used in Variation Operations

An overview for the introduced rankings is depicted in Figure 5.8 in combination with sorting directions illustrated by arrows. The conceived variation algorithms are all based on a definition for matching functional profiles (cf. Definition 5.4.1). Only matching functional profiles are exchanged in a cross-over operation and a functional profile can only be substituted by a matching functional profile. The holistic design generation approach allows the initialization of detailed design solutions consistent to a given abstract design. By this precondition for the variation algorithms, the consistency is preserved, infeasible solutions are avoided (cf. RV5). For replacements of functional profiles with matching other functional profiles, the obtained map in (5.11) can be used, in order to avoid retrievals from component repository and accelerate the optimization process.

Variation algorithms can be grouped as semi-directed (SD), random (R) and directed variations (D) with SD, R and D denoting prefixes for individual variation algorithms. Cross-over and mutation algorithms are abbreviated with C and M correspondingly.

In a semi-directed variation, components that should be replaced are determined by using one of the presented rankings, however the replacing components are chosen randomly. The use of rankings helps the algorithm to increase the probability for the intended specific improvement and the random choice of replacing components avoids an exhaustive search, thus replacements can be performed quickly.

Random variation is intended to introduce more diversity in a population by choosing components for replacements randomly. In a directed variation, component that should be replaced are determined by using ranking as in a semi-directed variation, however replacing components are chosen by an efficient search among a small number of possible candidates, where applicable. Such an approach for the variation operations specific to the design creation problem has been presented and successfully tested in [OPK10a].

Mutation Operations:
Mutation operations from SD-M1 to R-M4 are considered that can be applied on an individual with functional profiles

$$FPINDV = \{fpindv_1, fpindv_2, ..., fpindv_m\}$$

chosen for the mutation according to the mutation probability p_M:

- **SD-M1-Missing Bindings:** A number (rn) of functional profiles with the lowest values of m_1 in $ranking_1$ is randomly chosen and substituted by another randomly chosen matching functional profile using the map (5.11). The number rn is chosen randomly from the range $[1, \frac{m}{2}]$, where m is half of the number of functional profiles in the individual rounded down (variation strategy SSV).

  Goal: Non fitting functional profiles can be replaced by fitting functional profiles to improve the validity and completeness of the individual by allowing both small and larger moves in the solution space without changing the pattern of the individual.

- **SD-M2-Larger FBC:** One of the functional profiles $fpindv_i$ ($i \in [1, m]$) that maps the function block combination fbc_{fpindv_i} mapped by with the lowest value of m_1 in $ranking_1$ is randomly chosen and substituted by another randomly chosen functional profile fp' that maps the function block combination $fbc_{fp'}$ using the map (5.11), such that $fbc_{fp'}$ is a set of higher cardinality than fbc_{fpindv_i}, if there exists any such $fbc_{fp'}$, such that $fbc_{fp'} \cap fbc_{fpindv_i} \neq \emptyset$ (cf. Figure 5.9). In order to support design consistency, the set of covered additional function blocks by fbc_{fpindv_i} is calculated:

$$\begin{aligned} FB_{add} &= fbc_{fp'} \setminus fbc_{fpindv_i} \\ &= \{fb_{add,1}, fb_{add,2}, ..., fb_{add,n}\} \end{aligned}$$

Functional profiles of the individual

$$fpind_{rem,1}, fpind_{rem,2}, ..., fpind_{rem,p}$$

that map function block combinations

$$fbc_{fpind_{rem,1}}, fbc_{fpind_{rem,2}}, ..., fbc_{fpind_{rem,p}} \tag{5.26}$$

containing $fb_{add,j}$ ($j \in [1, n]$) are removed from the individual. The union of function block sets covered by all function block combinations in (5.26) is FB_{union}. At this step there are no functional profiles in the individual that cover the function blocks of the set difference

$$FB_{diff} = FB_{union} \setminus FB_{add}.$$

Additional functional profiles are added into the individual using the maps (5.15) and (5.11). Hereby, for each function block in FB_{diff} function block combinations of cardinality one is chosen in (5.15) (variation strategy SSV).

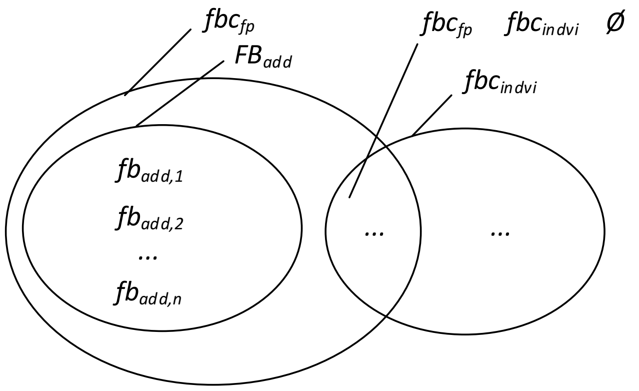


Figure 5.9: Venn Diagram for the Relationship between Function Block Sets of Function Block Combinations $fbc_{fp'}$ and fbc_{fpindv_i} in SD-M2

Goal: Non fitting functional profiles can be replaced by fitting functional profiles to improve the validity and completeness of the individual by changing the pattern of the individual. The existence of non fitting individuals may be due to an inappropriate coverage of multiple function blocks by function block combinations of such functional profiles. In addition, an exploration for different patterns in the solution space is performed that can guide the optimization to near-optimal solutions.

- **SD-M3-Smaller FBCs:** One of the functional profiles $fpindv_i$ ($i \in [1, m]$) that maps the function block combination fbc_{fpindv_i} with a cardinality greater than one ($|fbc_{fpindv_i}| > 1$) corresponding to the lowest value of m_1 in $ranking_1$ is randomly chosen and substituted by other randomly chosen functional profiles $fp'_1, .., fp'_o$ with function block combinations of cardinality one ($\forall l \in [1..o]\ fbc_{fp'_l}$) using maps (5.15) and (5.11), such that each $fbc_{fp'_l}$ is a strict subset of fbc_{fpindv_i} ($fbc_{fp'_l} \subset fbc_{fpindv_i}$), if there exist any such fbc_{fpindv_i} (variation strategy SSV).

  In Figure 5.10 function block combination fbc_{fpindv_1} of an example functional profile $fpindv_1$ consists of two function blocks fb_a and fb_b. For the removal of $fpindv_1$ by substitution, fb_a and fb_b are mapped separately to two other functional profiles ($fp'1$ and $fp'2$ respectively) with function block combinations $fbc_{fp'1}$ and $fbc_{fp'2}$.

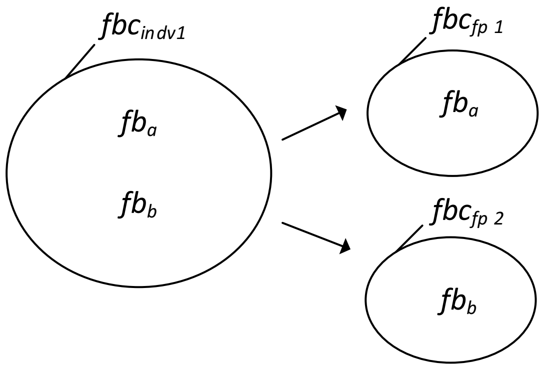


Figure 5.10: Venn Diagram for the Relationship between Function Block Sets of Example Function Block Combinations fbc_{fpindv_1}, $fbc_{fp'1}$ and $fbc_{fp'2}$ in SD-M3

  Goal: Similar as the goal of SD-M2 with the difference that the existence of non fitting individuals may be due to the coverage of single function blocks by function block combinations of such functional profiles.

- **SD-M4-Device Price Cost:** One of the devices with the highest price cost $dev_{exp} \in \{dev_{exp_1}, dev_{exp_2}, ..., dev_{exp_q}\}$ that covers functional profiles

$$FP_{dev_{exp}} = \{fp_{exp,1}, fp_{exp,2}, ..., fp_{exp,r}\}$$

  mapping function block combinations

$$FBC_{exp} = \{fbc_{fp_{exp},1}, fbc_{fpexp,2}, ..., fbc_{fpexp,r}\}$$

  is randomly chosen in $ranking_2$. Each function block combination $fbc_{fp_{exp},g}$ ($g \in [1, r]$) is attempted to map on a functional profile of the cheapest device using maps in (5.12) and (5.13). If there exist multiple devices with the lowest price cost, one of them is randomly chosen (variation strategy SSV).

Goal: The aim is to replace devices with high price costs by devices with lower price costs covering the same set of function blocks to obtain a lower sum of price costs in the individual. In addition, unvisited devices and functional profiles can be introduced to the component space by this kind of exploration in the solution space.

- **SD-M5-Device Coverage:** One of the devices

  $$dev_{lowc,f} \in \{dev_{lowc_1}, dev_{lowc_2}, ..., dev_{lowc_s}\}$$

  with the lowest functional profile coverage rate ($f \in [1, s]$) that covers functional profiles

  $$FP_{dev_{lowc,f}} = \{fp_{lowc,1}, fp_{lowc,2}, ..., fp_{lowc,t}\}$$

  is randomly chosen in $ranking_3$. Each functional profile $fbc_{fp_{lowc},d}$ ($d \in [1, t]$) is mapped to matching functional profiles of other devices than $dev_{lowc,f}$ using the maps (5.12) and (5.13) (variation strategy SSV).

  Goal: The aim is to replace devices with high price costs are replaced by devices with lower price costs covering the same set of function blocks to obtain a better sum of device functional profile coverage rate in the individual. This action has the potential to increase the number of device internal bindings and (not necessarily) also to reduce the sum of device price costs. In addition, unvisited devices and functional profiles can be introduced to the component space by this kind of exploration in the solution space.

- **SD-M6-Functional Profile Coverage:** A number (rn) of functional profiles with the lowest values of m_1 in $ranking_1$ is randomly chosen and each such functional profile with function block combination fbc is substituted by randomly chosen functional profiles with function block combinations from the set of connection sensitive function block combinations in (5.14). Similar to SD-M2 with the difference that replacing functional profiles map connection sensitive function block combinations (variation strategy SSV).

  Goal: Similar as SD-M2 with the difference that the integration of functional profiles with connection sensitive function block combinations can better accelerate the search for choosing the function block combinations towards valid binding-schemata. This is particularly useful for abstract designs that contain a very large number of function blocks causing an exponential growth of function block combinations. This variation operation achieves a faster convergence towards optimal solutions by considering a reduced number of mapping alternatives among function block combinations and functional profiles.

- **D-M1-Device Price Cost:** One of the devices with the highest price cost $dev_{exp} \in \{dev_{exp_1}, dev_{exp_2}, ..., dev_{exp_q}\}$ that covers functional profiles

  $$FP_{dev_{exp}} = \{fp_{exp,1}, fp_{exp,2}, ..., fp_{exp,r}\}$$

  mapping function block combinations

  $$FBC_{exp} = \{fbc_{fp_{exp},1}, fbc_{fpexp,2}, ..., fbc_{fpexp,r}\}$$

  is randomly chosen in $ranking_2$. Each function block combination $fbc_{fp_{exp},g}$ ($g \in [1, r]$) is attempted to map on a functional profile of the cheapest device in $ranking_2$. If there exist multiple devices with the lowest price cost, one of them is randomly chosen (variation strategy CSV).

  Goal: Similar as SD-M4 with the difference that the focus is on improvement of an individual for the sum of device price costs. This is done by mapping function block combinations on the functional profiles of a reduced number of devices without introducing new components to the component space.

- **D-M2-Device Coverage:** One of the devices

  $$dev_{lowc,f} \in \{dev_{lowc_1}, dev_{lowc_2}, ..., dev_{lowc_s}\}$$

  with the lowest functional profile coverage rate ($f \in [1, s]$) that covers functional profiles

  $$FP_{dev_{lowc,f}} = \{fp_{lowc,1}, fp_{lowc,2}, ..., fp_{lowc,t}\}$$

  is randomly chosen in $ranking_3$. Each functional profile $fbc_{fp_{lowc},d}$ ($d \in [1, t]$) is attempted to map to matching functional profiles of other devices of the individual than $dev_{lowc,f}$ using the maps (5.12) and (5.13) starting from the devices with the highest functional profile coverage rate and proceeding with devices of lower functional profile coverage rates. Thus, devices other than $dev_{lowc,f}$ in $ranking_3$ are visited in the reverse order (variation strategy CSV).

  Goal: Similar as SD-M5 with the difference that the focus is on improvement of an individual for the device functional profile coverage rate by a better distribution of function blocks to devices, thus for an increase of device internal bindings and (not necessarily) a reduction of the sum of device price costs. This is done by mapping function block combinations on the functional profiles of a reduced number of devices without introducing new components to the component space.

- **R-M1** to **R-M4:** These mutations operations are similar to SD-M1 to SD-M4 with the difference that the functional profiles in R-M1 to R-M3 and devices in R-M4 are chosen without consideration of rankings (variation strategy SSV).

  Goal: These mutation operations aim to introduce more diversity into the component space, in order to attain the same goals in SD-M1 to SD-M5 by performing an exploration in the solution space, which is particularly useful to avoid premature local optima.

Mutation operations SD-M1, SD-M2, SD-M3 and SD-M6 that make use of $ranking_1$ choose functional profiles without missing bindings, if applied on individuals with valid and complete binding-schemata in advanced stages of design improvement, in order to still contribute to the improvement of designs. In such cases, SD-M1, SD-M2 and SD-M3 are expected to cause same effects on improvement as R-M1, R-M2 and R-M3.

Cross-Over Operations:
Cross-over operations from SD-C1 to R-C2 are considered that can be applied on two mating individuals ind_1 and ind_2 to combine parts of both individuals to construct better optimized individuals without introducing new components to the component space:

- **SD-C1-Missing Bindings:** A random number of functional profiles (rn) of ind_1 with missing bindings (for values of $m_1 < 1$) (rn chosen in a range from 1 up to half of the number of functional profiles in ind_1) starting from the functional profiles with the lowest values of m_1 and proceeding with higher values are chosen. If there are no functional profiles with missing bindings in ind_1, rn functional profiles are chosen. These functional profiles are exchanged with the matching functional profiles of ind_2 (variation strategy CSV).

  Goal: Non fitting functional profiles in ind_1 can be replaced by fitting functional profiles of ind_2 and vice versa. This is possible, since a non fitting functional profile in ind_1 can fit in ind_2. Hence, after this operation, both ind_1 and ind_2 can be improved for the validity and completeness by an intensification in the component space without changing the patterns of ind_1 and ind_2.

- **SD-C2-Device Coverage:** Functional profiles of ind_1 covered by a random chosen device from the set of devices with the lowest functional profile coverage rate in $ranking_3$ are chosen and exchanged with the matching functional profiles of ind_2, if there exist such functional profiles in ind_2 (variation strategy CSV).

  Goal: By performing this operation, functional profiles can be covered by a reduced number of devices in both ind_1 and ind_2, since functional profiles on devices with low device functional profile coverage rate in ind_1 can be mapped onto devices with higher functional profile coverage rate in ind_2 to obtain an improvement similar as in SD-M5.

- **SD-C3-Device Price Cost:** Functional profiles of ind_1 covered by a random chosen device from the set of devices of the highest price cost in $ranking_2$ are chosen and exchanged with the matching functional profiles of ind_2, if there exist such functional profiles in ind_2 (variation strategy CSV).

  Goal: By performing this operation, functional profiles can be covered by devices with lower price costs in both ind_1 and ind_2, since functional profiles on devices with high price costs in ind_1 can be mapped onto devices with lower price costs in ind_2 to obtain an improvement similar as in SD-M4.

- **SD-C4-Binding Exchange:** All fitting functional profile pairs in ind_1 are exchanged with functional profile pairs consisting of matching functional profiles. In a second step, the same operation is performed outgoing from the fitting functional profile pairs in ind_2. (variation strategy CSV).

  Goal: Same as the goal for SD-C1 with the difference that this operation is connection-based and performed based on bindings rather than on functional profiles and devices.

- **R-C1** to **R-C2:** These cross-over operations are similar to SD-C1 and SD-C2 with the difference that the selection of functional profiles in R-C1 and devices in R-C2 are random (variation strategy CSV).

  Goal: These cross-over operations aim to introduce more diversity into the component space, in order to explore for better optimized individuals in the component space, which is also useful to avoid premature local optima.

In total, a list of 18 variation operations are conceived as summarized in Table 5.4.

Variation Type	Operation	Variation Strategy	Improvement Criterion	Pattern Change
Mutation	SD-M1	SSV	bindings	no
	SD-M2		bindings	yes
	SD-M3		bindings	yes
	SD-M4		device price cost	no
	SD-M5		device coverage rate	no
	SD-M6		functional profile coverage rate	yes
	R-M1 to R-M4		similar as SD-M1 to SD-M4	
	D-M1	CSV	device price cost	no
	D-M2		device coverage rate	no
Cross-Over	SD-C1		bindings	no
	SD-C2		device coverage rate	no
	SD-C3		device price cost	no
	SD-C4		bindings	no
	R-C1 to R-C2		similar as SD-C1 to SD-C2	

Table 5.4: Summary of Variation Operations

All variation operations require additional recovery operations that remove bindings of the concerned functional profiles after exchanges or modifications and attempt to create new bindings. The recovery operations consequently map new functional profiles onto devices (cf. Section 5.6.2).

5.5.3 Application Strategies

Variation operations presented in Section 5.5.2 can be grouped for different application strategies to determine the group that provides the best performance. Each such group of variation operations must be able to reach any point in the solution space. In particular, it is interesting to observe the performance of variation operations with respect to their exploration and exploitation capacity. These application strategies are:

- **R:** application of the set of only random variation operations, R-M1 to R-M4;
- **SD/R/D:** application of the set of random, semi-directed and directed variation operations presented in Table 5.4.

At each of these strategies, variation operations are applied in each generation in the order of a rotating list. Hence, each variation operation once applied, can be reapplied in the optimization process or even in a single generation, if it is its turn. The rotating list is global for the optimization process and the application rates for each variation operation depends on the mutation and cross-over probability, hence on the number of the individuals that are mutated and the number of individuals that perform cross-over operations.

5.6 Realization of the Approach

5.6.1 Objective Functions

Objective functions can be derived from the fitness criteria in Section 5.2. By using the design generation approach in Section 5.4 and variation operations in Section 5.5.2 that are designed to keep the consistency between solutions and abstract design, abstract design compliancy for the detailed designs is guaranteed in the design generation as well as in the design improvement. Hence, objective functions are designed corresponding to the remaining criteria: validity and completeness of binding-schema, device price costs, demand for device communications, and configuration and maintenance.

For a detailed design solution candidate that contains the set of functional profiles FP and devices $DEVS$ in (5.25), six objective functions are designed:

- **validity and completeness of binding-schema:**

  $F_1 = 1 - |BIN|/|BPC|$, with $|BIN|$: is related to number of bindings and $|BPC|$: number of binding pair candidates,

  $F_2 = (|MAN| - |RMAN|)/|MAN|$, with $|MAN|$: number of mandatory functional profile input datapoints and $|RMAN|$: number of input datapoints in MAN that possess bindings,

  $F_3 = |EX|/|BPC| + 1 - |BIN|/|BPC|$, with $|EX|$: number of excessive bindings at functional profile input datapoints;

- **device price costs:**

  $F_4 = \sum_{i=1}^{y} price(dev_i)/maxprice$, with $y = |DEVS|$ and $maxprice$: maximum possible device price costs that emerges by mapping each function block on a separate device with the highest possible price cost;

- **demand for device communications:**

  $F_5 = |BINEX|/|BPC| + 1 - |BIN|/|BPC|$, with $|BINEX|$: number of device external bindings;

- **configuration and maintenance:**

  $F_6 = |MSB|/|BPC| + 1 - |BIN|/|BPC|$, with $|MSB|$: number of manufacturer-spanning bindings (cf. Section 5.2).

$|BIN|$ is calculated in distinction for interoperable and interworking bindings (cf. Table 2.2). Each interoperable binding is evaluated with the value 1, whereas each interworking binding is evaluated with the value $1-1/(|BPC|+1)$. All objective functions are designed for a minimization and normalized to enable comparisons. The presented objective functions allow an evaluation and thus comparison of individuals among each other.

5.6.2 Individual Representation

For the evaluation of individuals using the objective functions in Section 5.6.1 a representation is required that contains all necessary design elements such as:

- functional profiles with operation modes together with input and output datapoints, attribute for implementing devices of functional profiles, attributes for indicating whether an input datapoint is mandatory;
- bindings with attribute, whether a binding is device internal;
- devices with attribute for price costs, manufacturer and platform.

Design elements with identical information content can exist in multiple individuals, hence information contained in each individual must appear in a unique representation for distinction and allowing a general reuse of information. Same functional profiles, datapoints and devices can be contained in multiple individuals. For distinguishing such design elements, instances are conceived which are unique for each and every individual which point to the unique information contained. Explicitly, functional profile instance, device instance and datapoint instance are conceived and point to functional profile, device and datapoint respectively. This approach is realized by the component model in Figure 5.2. The representation used for the design generation and improvement is a direct representation based on this component model and consists of a set of used functional profile instances in the binding-schema, a set of device instances and the abstract design for which the individual is generated.

Choice of such a direct representation model with redundant elements allows the application of sophisticated variation operations as presented in Section 5.5.2, since all the necessary information required for such a variation concept is made available to the operations. Further, number and complexity of recover operations that must be performed to update the solution candidates after modifications by variation operations are reduced. For the chosen individual representation, after an exchange or a substitution of a functional profile, bindings of the removed functional profiles are also removed. Further a new functional profile in a modified individual attempts to build new bindings with its partners by the notion of binding pair candidate (cf. Definition 3.2.5). By application of this recover operation on all new functional profiles, a new binding-schema emerges. In a last step an update of device list is calculated for the modified individual via the set of functional profiles, which is relatively simple, since the functional profiles are device specific. With the specified cardinality for each functional profile on its implementing device (cf. Definition 3.2.3), this step conceives the calculation of the number of devices required to completely cover the functional profiles and map the functional profiles to implementing devices within cardinality limits.

5.7 Automated Design Creation For A Building

So far the concentration of this thesis has been on creation of designs for a room for its being the task with the highest complexity in the creation of a BAS design. However, if a design creation for a building in practice is considered, there are further complementary issues of concern: room spanning control, flexible rooms, technology spanning designs and preferences for mapping of function blocks to devices.

5.7.1 Room Spanning Control

Function blocks of certain locations may be required to communicate with function blocks across many different rooms. The most common example for such a communication is between the sensor function blocks placed on the roof or at a facade (e.g. wind velocity measurement, precipitation detection) and application function blocks required in abstract designs for many different rooms. In such designs connections are from each of these sensor function blocks, each of which exists only once in the BAS design for all other communication partners.

On the detailed design level this implies that e.g. a single sensor device for wind velocity measurement placed on the roof communicates with all other controller and actuator devices across many rooms in the building. This kind of a one-to-many communication relation from the abstract design of a room to abstract designs of many other rooms is represented by using reference function block instances as mentioned in Section 5.3. Hence, such sensor function blocks appear only once in their own abstract designs and they are represented via reference function blocks in other abstract designs, which correspond to local functional profile instance and functional profile instance reference on the detailed design level.

Installation location for sensor devices in room automation are frequently the same as the location where the function is required, apart from some minor examples as explained in the case of sensor devices placed on the roof or at the facades. In contrast, controller and actuator devices may be required to install in other locations than where their functions act. Typical examples for such devices are light actuators which are mounted in sub-distributors (by rail mounting). Their functional profiles are not used locally in the detailed design assigned to the installation location, but in detailed designs of other rooms of e.g. a storey. Each time a functional profile is required, a different functional instance is used from the device in sub-distributor location.

5.7.2 Flexible Rooms

In current functional buildings buildings room layouts of floors are mostly not conceived to be flexibly modifiable. However, number and dimension of the rooms are desired to be modified flexibly and economically. In some of the modern buildings, room layout of floors can be flexibly changed e.g. in an office building from a number of individual offices to an open-plan office in time of operation. In whole area of a floor a number of areas are defined as the base for locations where the automation plant should be installed. These areas are divided into a number of segment grids with each segment being the smallest functional unit (cf. Figure 2.7). The segment grids are individually grouped into rooms. Rooms that are built by grouping of segment grids are separated from each other by partition walls.

The possibility of such a modification of room layout of a floor up to the load-bearing walls has a prior advantage that in case of a room layout modification, there is no necessity to change mounting location of the existing room automation devices in segment grids and the desired functionality is achieved by adapting the system configuration (by modifying the bindings and occasionally parameter settings of functional profiles), explicitly making the necessary changes in the corresponding abstract and detailed design. In Figure 5.11 an example floor plan can be seen, for which in Figure 5.12 areas B1, B2 and B3 with each containing a different number of segments and therein 12 rooms are defined.

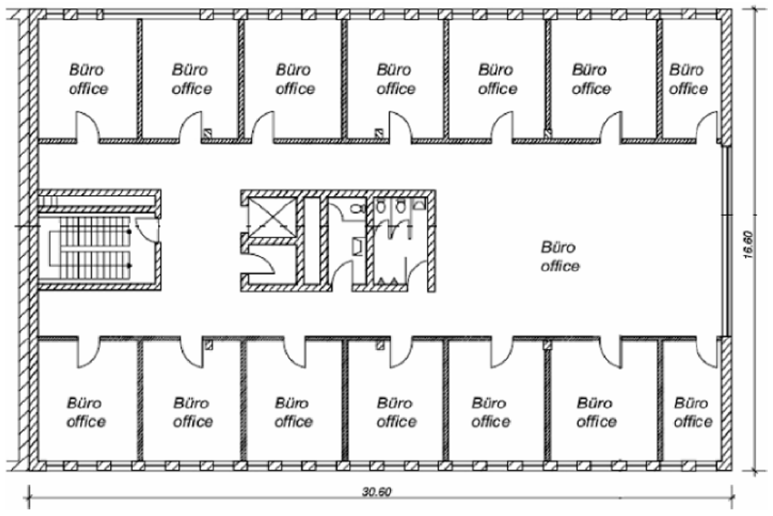


Figure 5.11: An Example Floor Plan from the Standard VDI 3813-1 [The11a]

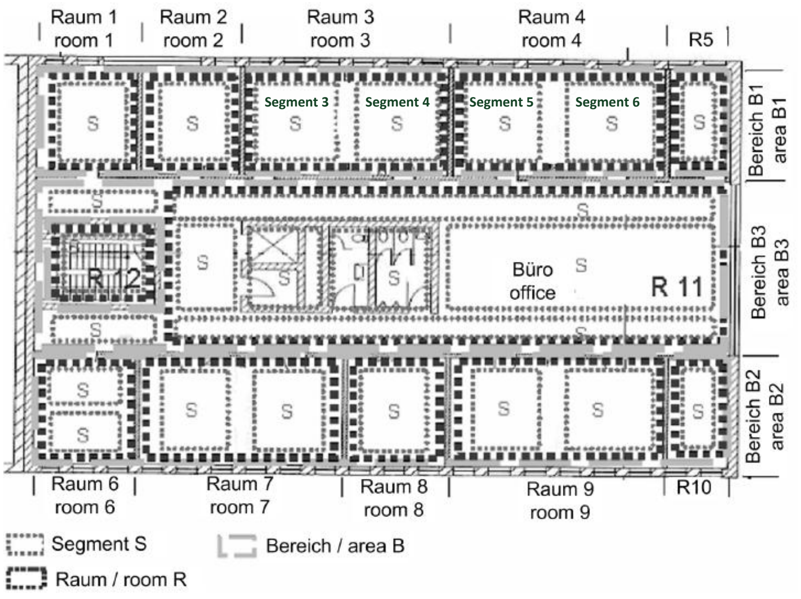
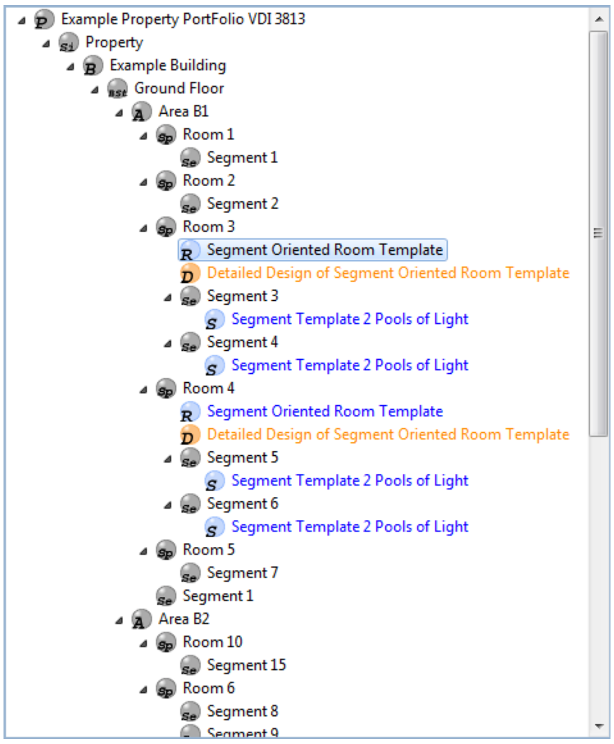


Figure 5.12: Floor Plan with Segments Oriented (Flexible) Rooms from the Standard VDI 3813-1 [The11a]

A segment typically contains one or more pools of light, a window with sunblind and window contact, and a radiator, if they exist, which are related to the realization of the functions light actuator, sunshade actuator, window monitoring and control driver actuator. The approach of the thesis to enable segment-oriented designs conceives an abstract design for such a segment that contains function blocks which realize these functions. In segments, frequently a repeatedly used set of functions are allocated. Hence, segments, each of which contains an identical set of function blocks is represented by one unique abstract design template (*segment abstract design template*).

Further, there exist functions which belong to such a room and which are not allocated in a segment (the *difference template*). These functions are application (e.g. occupancy evaluation, constant-light control, automatic solar control, temperature control heating, etc.), operator and display functions (e.g. actuate light, select room utilization type), and some sensor functions (e.g. room brightness measurement, room air temperature measurement, presence detection) and they are defined in an abstract design template for the room with each represented as a function block.



```
Example Property PortFolio VDI 3813
  Property
    Example Building
      Ground Floor
        Area B1
          Room 1
            Segment 1
          Room 2
            Segment 2
          Room 3
            Segment Oriented Room Template
            Detailed Design of Segment Oriented Room Template
            Segment 3
              Segment Template 2 Pools of Light
            Segment 4
              Segment Template 2 Pools of Light
          Room 4
            Segment Oriented Room Template
            Detailed Design of Segment Oriented Room Template
            Segment 5
              Segment Template 2 Pools of Light
            Segment 6
              Segment Template 2 Pools of Light
          Room 5
            Segment 7
          Segment 1
        Area B2
          Room 10
            Segment 15
          Room 6
            Segment 8
            Segment 9
```

Figure 5.13: Project Structure for VDI 3813 Example Building

In Figure 5.13 the design project structure related to the example building in standard VDI 3813 with areas, segments and rooms is presented. Hereby, “Room 3” is formed by grouping segments “Segment 3” and “Segment 4”, and “Room 4” is formed by grouping segments “Segment 5” and “Segment 6” in area “B1”.

After grouping segments to a room, one can determine which segment abstract design templates should be used and how many times each such template is required in the corresponding grouping. A room abstract design template emerges by providing the number of required segment abstract design templates and abstract connection from each segment abstract design template to the difference template. The room abstract design template can then be used by the automated design creation approach to create optimized detailed designs.

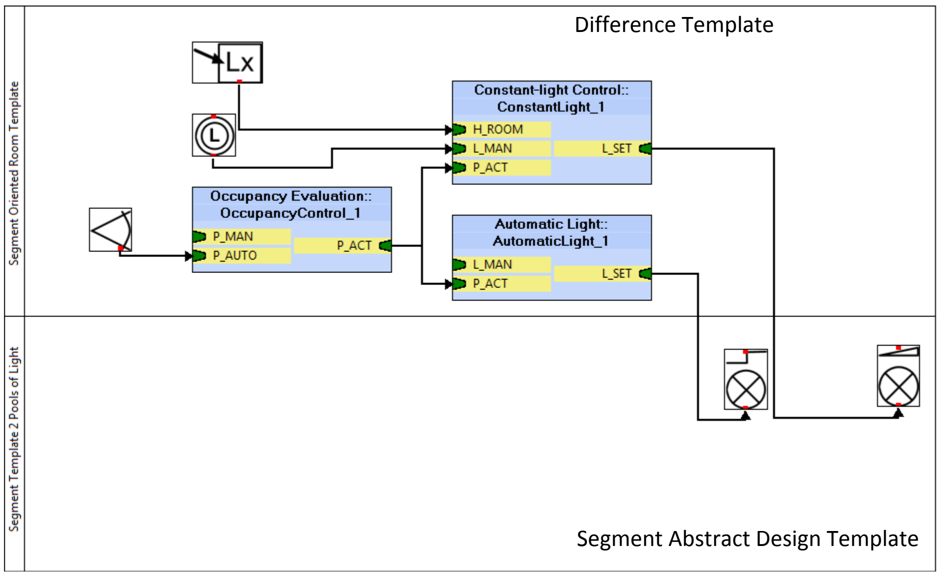

Difference Template
Segment Oriented Room Template
Segment Template 2 Pools of Light
Lx
L
Constant-light Control:: ConstantLight_1
H_ROOM
L_MAN
P_ACT
L_SET
Occupancy Evaluation:: OccupancyControl_1
P_MAN
P_AUTO
P_ACT
Automatic Light:: AutomaticLight_1
L_MAN
P_ACT
L_SET
Segment Abstract Design Template

Figure 5.14: Abstract Design Template for Segment Oriented Rooms in Figure 5.12 and Figure 5.13

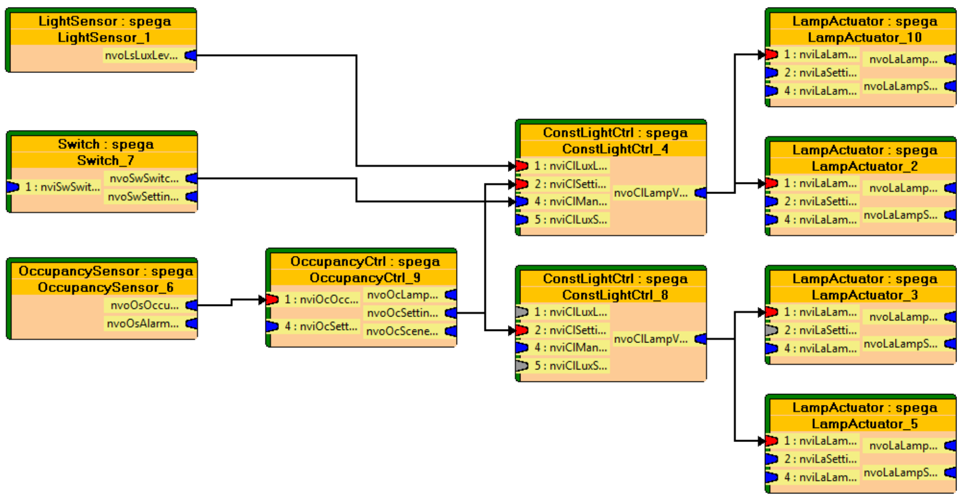


Figure 5.15: Detailed Design Template - Binding-Schema for Segment Oriented Rooms in Project Structure in Figure 5.13

In Figure 5.14 a segment oriented room abstract design template is given that should realize constant-light control and automatic lights functions in segment oriented rooms “Room 3” and “Room 4” in Figure 5.13. Thus, this template is assigned in project structure in Figure 5.13 to “Room 3”, “Segment 3” and “Segment 4”, and “Room 4”, “Segment 5” and “Segment 6”. Since the segment abstract design template in Figure 5.14 is assigned to e.g. segments “Segment 3” and “Segment 4” in room “Room 3”, thus it is assigned twice in sum in this room and contains two function blocks, a realizing detailed design for the abstract design in Figure 5.14 must realize twice each of these two function blocks (in sum four function blocks). All four function blocks build abstract connections corresponding to the room abstract design template in Figure 5.14. In Figure 5.15 the detailed design is presented that realizes the segment oriented abstract design in Figure 5.14 and that is assigned to rooms “Room 3” and “’Room4’.

5.7.3 Technology Spanning Designs

Another important issue is the integration of components from different technologies using different communication protocols within a detailed design. This also concerns such components that communicate with each other to realize the functions of an industry or also industry spanning-functions. Since in such a constellation messages in different protocols are exchanged among devices, gateways are required to translate such messages from sender devices to receiver devices.

On the logical level, this is realized i.e. by an adapter functional profile of the gateway device. For a binding from a LON functional profile to an EnOcean functional profile, adapter functional profile of a LON-EnOcean gateway device is integrated to translate the network variables of the LON device to EnOcean device and to enable a correct communication.

The approach for enabling technology spanning designs conceives specifying technology preference for function blocks in an abstract design as a non-functional requirement, if desired. In the design generation step, function block combinations are built by grouping only the function blocks of an identical technology preference, which are covered by functional profiles of the devices from the same technology. Similarly, the design improvement step, in candidate detailed designs, improvement of devices and functional profiles are performed by isolating components of different technologies. In variation operations functional profiles of a technology are substituted or exchanged only by other functional profiles of the same technology.

An example for a technology spanning system that realizes functions of industries heating and lighting is presented in Figure 5.16 and 5.17. In Figure 5.16 function blocks that are planned for a realization using EnOcean technology are marked with the letter “E” and LON technology with the letter “L”. Detailed design in Figure 5.17 that realizes the abstract design in Figure 5.16 consists of functional profiles from EnOcean and LON technologies correspondingly. A complete list of these functional profiles is given in Table 5.5 with mapped function blocks.

Communication from EnOcean devices to LON devices is realized via a EnOcean-LON gateway that provides adapter functional profiles for this purpose. For example, the EnOcean functional profile “7” in Figure 5.17 communicates with the LON functional profiles “3” and “8” via adapter functional profile of the EnOcean-LON gateway “12”.

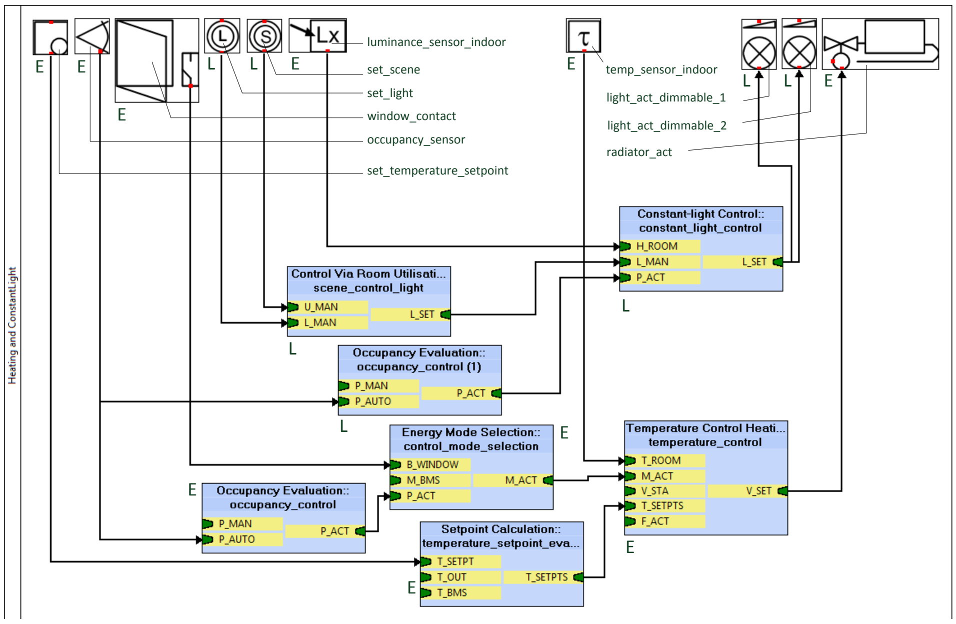

Heating and ConstantLight
luminance_sensor_indoor
set_scene
set_light
window_contact
occupancy_sensor
set_temperature_setpoint
temp_sensor_indoor
light_act_dimmable_1
light_act_dimmable_2
radiator_act
Constant-light Control:: constant_light_control
H_ROOM
L_MAN
P_ACT
L_SET
Control Via Room Utilisati... scene_control_light
U_MAN
L_MAN
L_SET
Occupancy Evaluation:: occupancy_control (1)
P_MAN
P_AUTO
P_ACT
Energy Mode Selection:: control_mode_selection
B_WINDOW
M_BMS
P_ACT
M_ACT
Temperature Control Heati... temperature_control
T_ROOM
M_ACT
V_STA
T_SETPTS
F_ACT
V_SET
Occupancy Evaluation:: occupancy_control
P_MAN
P_AUTO
P_ACT
Setpoint Calculation:: temperature_setpoint_eva...
T_SETPT
T_OUT
T_BMS
T_SETPTS
E
L
Lx
τ

Figure 5.16: Abstract Design for a Technology Spanning System - Heating and Constant-Light Control

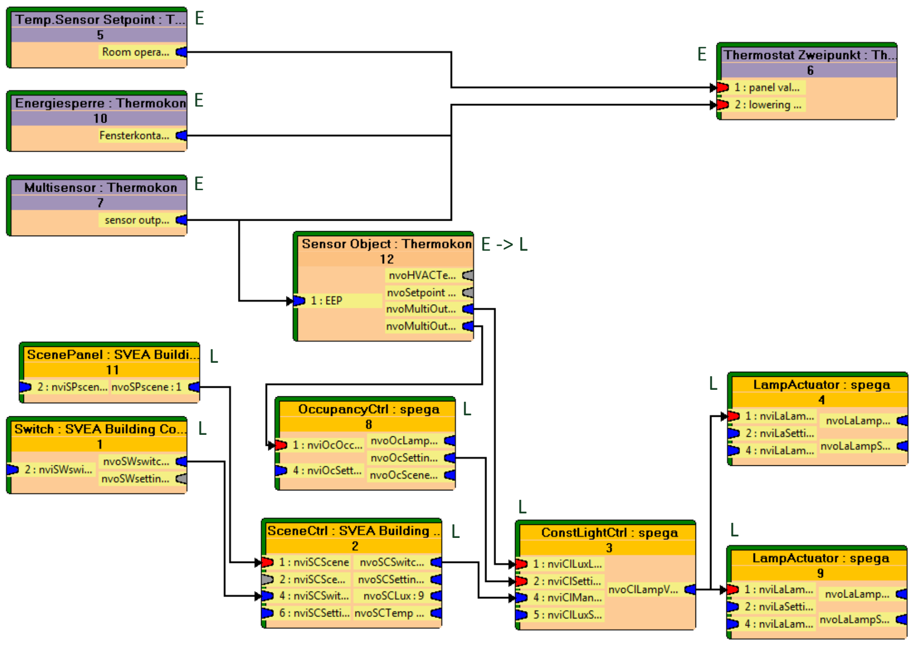


Figure 5.17: Detailed Design Realizing the Abstract Design in Figure 5.16 - Binding-Schema

Functional Profile	Function Block Combination
5	set_temperature_setpoint temp_sensor_indoor
10	window_contact
7	occupancy_sensor luminance_sensor_indoor
11	set_scene
1	set_light
8	occupancy_control (1)
2	scene_control_light
3	constant_light_control
6	control_mode_selection temperature_setpoint_eva. occupancy_control temperature_control radiator_act
4	light_act_dimmable_1
9	light_act_dimmable_2

Table 5.5: Mappings among Functional Profiles in Figure 5.17 and Function Blocks in Figure 5.16 by Function Block Combinations

5.7.4 Preferences for Mapping of Function Blocks to Devices

Another important issue of creation of BAS designs in application practice conceives special preferences for mapping of function blocks to devices. A planer may specify preferences for some function blocks in an abstract design to be

- from a specific and the same technology: some technologies can be preferred for providing components with more reasonable device price costs or due to law for protection of historical buildings e.g. wireless BAS for the famous Opera Building Semper Opera in Dresden;
- from a specific and the same manufacturer: some device manufacturers that possess geographically close offices to the building of installation can be preferred for ease of reachability in case of maintenance in the life-cycle of an installed BAS;
- or to be mapped onto the same device (also without specifying any particular device type): some group of function blocks can be desired to be mapped onto the same device that is planned to be installed at a specific location in a room e.g. function blocks presence detection and brightness measurement can be planned to be realized by a same device that should be installed at the ceiling. Further, function blocks that belong to different control loops (e.g. light actuators for separate pools of light) are desired to be mapped on separate functional profiles or separate devices (the latter case is often required in case of a realization with wireless automation devices).

These preferences are defined as non-functional requirements attached to each of such function blocks in a considered abstract design. In the design generation step, during a component search, only devices and functional profiles from the specified technologies and/or from the specified device manufacturers are retrieved from the component repository.

Furthermore, for function blocks that are desired to map onto a same device a non-functional requirement can be defined as an attribute to mark such function blocks. The design generation and improvement steps consider this requirement and create detailed designs that contain devices each mapping all such function blocks simultaneously by its functional profiles, where applicable.

An example for such a mapping preference is given in Example 2 that is used for performance tests in Section 6.3. In Appendix A in Table A.2 the device *EasySens SR-MDS 24V Wireless-Ceiling-Multi-Sensor_2* covers function blocks *Occupancy Sensor right*, *Temperature Sensor right*, *Luminance Sensor right* and the device *EasySens SR-MDS BAT Wireless-Ceiling-Multi-Sensor_13* covers function blocks *Occupancy Sensor left*, *Temperature Sensor left*, *Luminance Sensor left* which belong to the abstract design given in Figure A.3 and A.4.

Thus, function blocks that are desired to be mapped on same devices that should be installed at the specific locations in the room “right” and “left” receive non-functional requirements in the step of abstract design creation that point three of these function blocks (“left” function blocks) onto an anonymous but same device for “left” installation location and point the other three of the function blocks (“right” function blocks) to another anonymous but same device for “right” installation location.

In case, if such non-functional requirements are not defined, detailed designs of same objective function values would arise that contain various mappings of function blocks on devices e.g. *Occupancy Sensor right*, *Temperature Sensor left*, *Luminance Sensor right* onto *EasySens SR-MDS 24V Wireless-Ceiling-Multi-Sensor_2*, which are incorrect, since this device after an installation cannot detect presence and measure room brightness at the “right” location and measure room temperature at the “left” location in the room.

5.8 Further Uses and Applicability of the Approach

The approach for automated creation of optimized designs has various important uses beyond the uses mentioned in Section 5.3.2:

- **documentation and traceability:** Many building automation installations have a lack of documentation. Project documentation is often poor. In a project with missing documentation of particularly binding-schemata and mapping of functions to functional profiles and devices, an enormous additional effort by system integrators is required in case of maintenance in the life-cycle of a BAS installation. Especially, new components may be required for integration that should realize additional functions or some old broken and malfunctioning components might have to be exchanged. In time traceability reduces and it cannot be determined with reasonable effort which components were integrated in the past for the realization of which functions, thus for the realization of which requirements.

  Especially in case of a BAS installation with wireless components such as EnOcean and ZigBee, bindings are assigned by pressing special learn buttons on devices and a detailed design is then not necessary for such technologies to build a functioning system. However, due to the lack of a detailed design, such systems are poorly documented. The proposed component model and mapping approach overcomes these lacks by providing relations in the business process model of a BAS from the requirement elicitation to the generated detailed designs that are e.g. in case of LON technology ready for commissioning;

- **assistance for testing component standard conformity:** Automated design creation can also be used as a utility for testing conformity of device functional profile implementations to the standard VDI 3813-2, future ISO

16848-4. By the use of the component mapping concept which is presented in form of a comparison in Table 5.1, functional profile implementations of device manufacturers can be compared to the function block specifications in the standard and elements that cannot be mapped in manufacturer device functional profiles (e.g. functional profiles that implement a number of room automation functions, but do not completely implement ports of function blocks in the standard in datapoints) can be identified. Such functional profiles can be classified as non standard conform and device manufacturers would gain information about the implementation elements that cause a violation of standard conformity. This can assist device manufacturer to easily review their implementations to enhance their products.

Further, the approach is

- **installation location independent:** there is no presumption made on the installation location (functional buildings and rooms) of devices, and the approach applies for a realization in any kind of room and functional building;
- **industry independent:** there is no presumption made on a support for functions of specific industries exclusively such as lighting, HVAC, etc., and the approach applies for all industries;
- **manufacturer and platform independent:** there is no presumption made on specific device manufacturers or platform technologies such as LON, EnOcean, KNX, and the approach applies for all device manufacturers that provide device functional profile implementations conform to VDI 3813-2 designed for different technologies. Further, the approach also applies for preferred manufacturers and/or technologies, if there exist any;
- **dual modal:** an optimized set of devices and an optimized set of functional profiles can also be computed for abstract designs that contain function blocks with and without abstract connections.

5.9 Conclusions

In this chapter the proposed solution method to the design creation problem as the core problem of the thesis (cf. Chapter 3) is presented, which is together with the validation in Chapter 6 the main contribution of this thesis.

For the design creation problem criteria for design evaluation is presented in Section 5.2. Further, the component problem is conceived in Section 5.3 that allows design evaluation, the automated design creation and a seamless integration of the solution method in the business process model for BAS engineering.

In Section 5.4 and 5.5 methods and algorithms are presented for the creation of optimized designs which is the main focus of this chapter. In these sections problem-specific information is integrated in the operations of multi-objective evolutionary algorithms that search for optimized solutions to enable an effective search and to achieve solutions of high quality.

In Section 5.4 design generation methods are presented with possible alternatives that create detailed designs consistent to the input abstract designs. In Section 5.5 a focus is made on variation operations of MOEAs that enable a goal-oriented search using rankings based on evaluation of candidate solution components. Further, strategies for the application of variation operations are presented.

In Section 5.6 objective functions are presented which are essential for the evaluation of detailed designs for comparisons among solution candidates, thus for the design improvement. Based on this, a problem-specific individual representation is conceived that enables the application of goal-oriented variation operations and a seamless integration of the detailed design solutions in the business process model of BAS.

Beyond the problems confronted in the design creation for a room, there are further issues of the BAS engineering practice which must be considered in the design for a whole building. These are presented in Section 5.7 with solutions. These issues are room spanning, control, flexible rooms, technology spanning designs and preferences for mapping of function blocks to devices. Further, it is possible to provide assistance to device manufacturers to test their component implementations for standard conformity and to determine possible lacks related to implementations.

The proposed method for the automated creation of optimized designs fulfill all the requirements on algorithms (cf. Section 3.3.2) as can be seen in Table 5.6.

	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8
[Erb06]	+	++	++	++	++	-	++	?
[MKBR10]	+	++	++	++	++	-	++	?
[WSWW03]	-	-	++	++	++	++	++	?
[CST$^+$06]	-	-	++	++	++	++	++	?
Proposed Method [OPK10a]	++	++	++	++	++	++	++	++

Table 5.6: Comparison of Solution Methods in Table 4.2 including the Proposed Solution Method for Design Creation Problem

For the validation of the concepts handled in this chapter, implementations are prepared and tested for different representative problem instances using alternative algorithms and evaluated in Chapter 6 by an empirical analysis.

Chapter 6

Validation and Performance Analysis

6.1 Validation Method

For the realization of the concepts of the solution method for the design creation problem presented in Chapter 5 and testing, whether these concepts satisfy the requirements on algorithms as presented in Section 3.3.2, a validation method must be conceived and applied for a performance comparison of algorithms based on empirical results.

The validation method conceives:

- **performance metrics:** to quantitatively determine algorithms that obtain best convergence to optimal solutions and that achieve a good exploration in the solution space;
- **a choice of representative problem instances:** to determine the capabilities, strength and weaknesses, thus possible limitations of the algorithms in different use cases in terms of performance metrics and computation time;
- **performance tests:** to determine the performance of algorithms by performance metrics for representative problem instances at different MOEA parameter settings and to suggest algorithms with parameter settings that achieve best performances for different problem instances.

6.2 Performance Metrics

Performance of MOEAs are generally measured considering the criteria for *quality of the solutions* and *exploration capability* of the optimization method. The latter criterion conceives both the *spread of the solutions* and the *extent of the* *solutions* reached by the variation operations, thus range of values covered by the solutions.

In the literature, there does not exist a single metric that can measure the performance of an MOEA simultaneously for all of these criteria. According to these three criteria performance metrics can be defined:

- **generational distance** $GD(P1, P0)$**:** This metric (cf. Formula 6.1) is used to measure the quality of the solutions of a set $P1$ by using their objective function values with respect to a reference set $P0$ by calculation Euclidean distances. Hence, smaller values of GD is always better, since this implies higher quality of the solutions contained in $P1$ [VL00];
- **spread** $S(P1, P0)$**:** This metric (cf. Formula 6.2) is used to determine the spread of the obtained non-dominated solutions P1 and can be used for problems with more than two objectives [ZJZ+06]. For this reason it is called *generalized spread.* First, the distance from a point to each solution's nearest neighbor is calculated [ZJZ+06][DPAM02]. Then, extreme points in the true Pareto front $P0$ are calculated, which is used for the calculation of Euclidean distances between the extreme solutions and the boundary solutions in P1. A smaller value of S is always better, since this implies that all extreme points are in $P1$ and the $P1$ solutions are uniformly distributed;
- **coverage** $C(P1)$**:** This metric (cf. Formula 6.3) is used to measure the maximum extent of the solutions $P1$ in each dimension, thus for each objective function to estimate the range that $P1$ spans [ZDT00]. A higher value implies a better extent and therefore it is always better.

Hereby, solution set $P1$ is the non-dominated set of solutions, whereas $P0$ is the true Pareto front. An explanation for the achievement of P0 is presented in Section 6.3.3 in tests using the original and cloned repository. Generational distance can be calculated by the formula in (6.1), where $p = 2$ and d_i is the Euclidean distance between each fitness vector of $P1$ solutions and the fitness vector of the nearest solution in $P0$. The formula implies $P1 \equiv P0 \Leftrightarrow GD = 0$. In the formula for the calculation of the spread $S(P1, P0)$ in Formula 6.2 $\{e_1, e_2, ..., e_m\}$ are extreme solutions in $P0$.

GD metric is the most directly related metric to the convergence behavior of an MOEA algorithm and thus for the test results, it is the most important metric. S and C metrics help to observe the distribution of solutions and the exploration behavior of algorithms.

$$GD(P1, P0) = \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{|P1|} \tag{6.1}$$

$$S(P1,P0) = \frac{\sum_{i=1}^{m} d(e_i,P1) + \sum_{X\in P1} |d(X,P1) - \overline{d}|}{\sum_{i=1}^{m} d(e_i,P1) + |P1|\overline{d}},$$

$$d(X,P1) = \min_{Y\in P1, Y\neq X} \|F(X) - F(Y)\|^2, \tag{6.2}$$

$$\overline{d} = \frac{1}{|P1|}\sum_{X\in P1} d(X,P1)$$

$$C(P1) = \sqrt{\sum_{1}^{n} \max\{\|x_i - y_i\|; x_i, y_i \in P1\}} \tag{6.3}$$

Moreover, the computation time spent in the optimization is also important statistics for making arguments on the efficiency of the solution method.

6.3 Example Abstract Designs and Performance Tests

6.3.1 Criteria for Choosing Example Abstract Designs

Beyond the problems mentioned in Section 5.5.1, an analysis of design creation using different abstract designs leads to the conclusion that the abstract designs can be classified in three categories with respect to the challenges the automated design creation must deal with:

- **high number of function block types and/or high number of function blocks:** Calculations performed in the preoptimization step in Section 5.4 may have to deal with abstract designs of a high number of function block types and/or a high number of function blocks, which may result in long computation times before the start of the design improvement step. Moreover, in the design improvement step, the search for functional profiles and devices that optimally cover the function blocks and allow the construction of valid and complete binding-schemata may be a big challenge for the variation operations to confront due to the high number of component combinations, particularly from different manufacturers;
- **highly intermeshed abstract connections:** A high number of many-to-one, one-to-many type of abstract connections among function blocks as well as multiple one-to-many connections with shared function blocks on the "to-side" are particularly difficult to map to bindings, since in such cases functional profiles of binding pair candidates are parts of many other binding pair candidates (cf. Definition 3.2.5), and bindings must yield

from each such binding pair candidate in the presence of a high number of component combinations, particularly from different manufacturers. This implies a high horizontal intercomponent dependency (cf. Section 3.3.1);

- **high and varying rate of functional profile coverage:** Varying number of function blocks can be covered by functional profiles, implying that same function blocks can repeatedly be contained in many different function block combinations, thus in different patterns. This may also result in long computation times in the preoptimization step as pointed in the first criterion. Further, in the design improvement step solutions that yield to global optima cannot be reached in a reasonable time, if the variation operations cannot effectively explore toward these solutions.

It should be emphasized that these challenges and the problems presented in Section 5.5.1 are independent of the location for designs, whether it is a room in a huge office complex with several building parts, a room in a hospital, or a room at a school.

6.3.2 Example Abstract Designs

By considering the criteria for choosing example abstract designs for performance tests, performance tests are involved with following three abstract design examples:

- **Example1:** This is the same abstract design as in Figure 2.8 and an example for the industry lighting. It concerns a relatively high number of function block types (11) and a moderate size of function blocks (13). It contains many-to-one and one-to-many type of abstract connections among function blocks. The target platform is chosen to be LON and the maximum functional profile coverage rate in this example is moderate (2).
- **Example2:** This is an industry-spanning example (with the complete form in Appendix A in Figure A.3 and A.4) with functions of both industries lighting and heating. It concerns a relatively high number of function block types (13) as well as a high number of function blocks (46). It contains many-to-one and one-to-many type of abstract connections. The target platform is chosen to be EnOcean and the maximum functional coverage rate in this example is high (5). For the sake of simplicity, a reduced form of this example with same number of function block types and 19 function blocks is illustrated in Figure 6.1.
- **Example3:** Similar to Example1, this example (cf. Figure 6.2) is from the industry lighting. It concerns a relatively low number of function block types (5) and a moderate size of function blocks (19). It contains many-to-one and one-to-many type of abstract connections as well as multiple one-to-many connections with shared function blocks on the "to-side".

The target platform is chosen to be LON and the maximum functional coverage rate in this example is low (2).

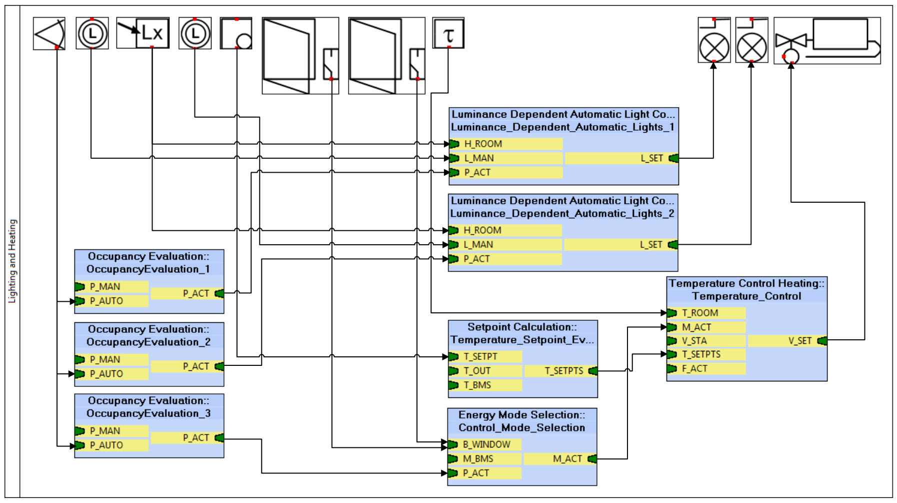


Figure 6.1: Lighting and Heating in Simplified Form

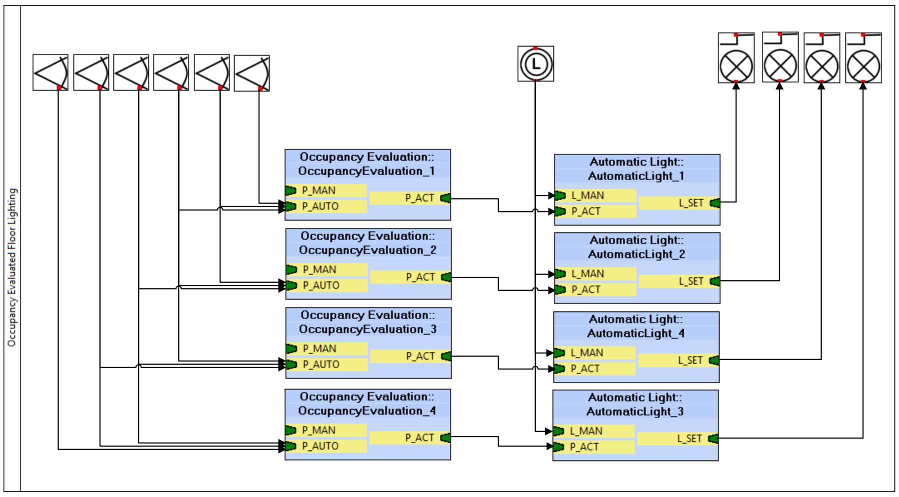


Figure 6.2: Occupancy Evaluated Floor Lighting

Symbols used in these examples are presented in Table 2.3 and they do not represent physical system elements. The abstract connections from and to these symbols do not represent wires, but a reflection of physical communication to the logical level of design via network variables which are not modified by these symbols.

6.3.3 Performance Tests

As a choice of representative problem instances example abstract designs in Section 6.3.2 with two component repositories (original and cloned) are considered. The purpose for performance testing with two repositories is to find optimal solutions using the original repository with an analogy of searching needles in the hay and to search for the optimal solutions in a much larger repository, thus searching the needles in a hay of a much greater volume. The aim is to be able to make conclusions on the performance of design improvement algorithm that they can achieve good results in both cases and the obtained solutions are near-optimal, and can be used as well optimized BAS designs according to the identified objective functions in Section 5.6.1.

The used original component repository [DK11] consists of 118 devices and 277 functional profiles from LON and EnOcean technology. Devices in the original repository are 100 times cloned to form the cloned repository. The cloned component repository contains also the original devices and functional profiles. In the cloning operation properties and capacities of the original devices and functional profiles are worsened to form their clones:

- operation modes of functional profiles are randomly reduced: to enlarge the range of the first, second and third objective function values;
- price costs of devices are randomly made higher: to enlarge the range of the fourth objective function values;
- device names are changed: to enlarge the range of the fifth objective function values;
- functional profile multiplicities of the devices are changed: to increase the variety of fourth and fifth objective function values;
- device manufacturers are changed by choosing them randomly among a set of 100 additional device manufacturers: to enlarge the range of the sixth objective function values.

Performance tests consist of two series of simulations using two MOEAs NSGA2 and SPEA2 in combination with the variation application strategies R and SD/R/D in Section 5.5.3:

- **tests using the original repository:** Automated creation of optimized designs method is applied on Example1 to 3 using all four combinations of MOEAs and variation application strategies NSGA2-R, NSGA2-SD/R/D, SPEA2-R, SPEA2-SD/R/D at a parameter setting population size $P = 40$, mutation probability $p_M = 0.6$ and cross-over probability $p_C = 0.6$. For each of the four algorithms and the parameter setting 30 runs for 300 generations are obtained. The true Pareto front $P0$ for each example is computed as the set of all non-dominated solutions obtained from all runs with the algorithms and parameter settings for the respective example using the original repository. In the design generation and improvement steps

  - 54 devices and 205 functional profiles are considered by using Example 1;
  - 40 devices and 40 functional profiles are considered by using Example 2;
  - 53 devices and 203 functional profiles are considered by using Example 3 which are relevant for the search for optimized detailed designs, thus which belong to the solution space.

The results contain

  - plots for average generational distances $GD(P1, P0)$ from the non-dominated solutions $P1$ of each run to $P0$ per generation;
  - plots for average spread $S(P1, P0)$ per generation;
  - plots for average coverage $C(P1)$ per generation and
  - an optimal detailed design example from the true Pareto front of each of the three example abstract designs Example1 to 3 in Appendix A;

- **tests using the cloned repository:** Automated creation of optimized designs method in Chapter 5 is applied on Example1 to 3 using the algorithms NSGA2-SD/R/D and SPEA2-SD/R-D. The reason for not considering the remaining R algorithms is explained in the results for tests using the original repository. The objective function value ranges are larger compared to the tests using the original repository, thus analysis for different parameter setting is significant to apply using the cloned repository. Hence, the algorithms are applied for analysis with different population sizes, different mutation and cross-over probabilities. For each of the two algorithms and for each parameter setting 30 runs for 1000 generations are obtained. The results contain plots for average generational distance, average spread and average coverage as in the tests using the original repository, where the true Pareto front $P0$ for each example is computed as the set of all non-dominated solutions obtained from all runs with the algorithms and parameter settings using the cloned repository. Further, the results contain near-optimal detailed design examples for Example2 and Example3 in Appendix A. In the design generation and improvement steps

  - 4730 devices and 18830 functional profiles are considered by using Example 1;

- 2853 devices and 2853 functional profiles are considered by using Example 2;
- 4278 devices and 17199 functional profiles are considered by using Example 3 which are relevant for the search for optimized detailed designs, thus which belong to the solution space.

Test Results Using the Original Repository:

Example 1: The true Pareto front $P0$ for this example consists of 7 non-dominated solutions. All four algorithms very frequently (in the major part of the runs) obtain one or more solutions in $P0$. The SD/R/D algorithms obtain more solutions in $P0$ than R algorithms. Plots for GD, S and C contain four observation points for each algorithm in generations 50, 100, 200 and 300. Although all values are discrete with respect to the generation number, the observation points are connected with lines to illustrate the tendency of each algorithm by growing number of generations.

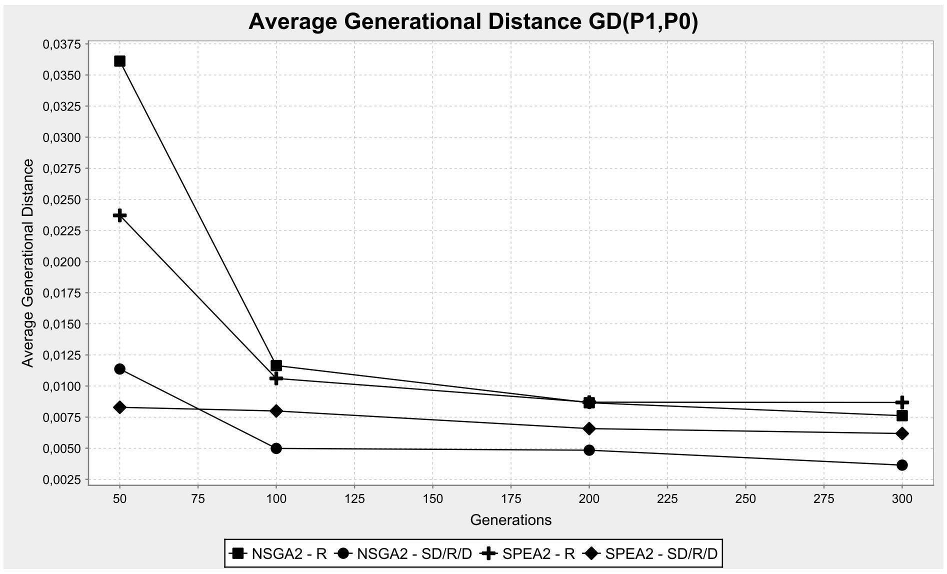


Figure 6.3: Generational Distance for Example 1

As can be seen in Figure 6.3, SD/R/D algorithms obtain already in generation 50 much better results than R algorithms. In the next 50 generations R algorithms manage to obtain closer results to SD/R/D algorithms, however, SD/R/D algorithms converge overall clearly better. Moreover, NSGA2-SD/R/D can move all its non-dominated solutions closer to $P0$ than SPEA2-SD/R/D, hence NSGA2-SD/R/D achieves for this example the best performance for convergence.

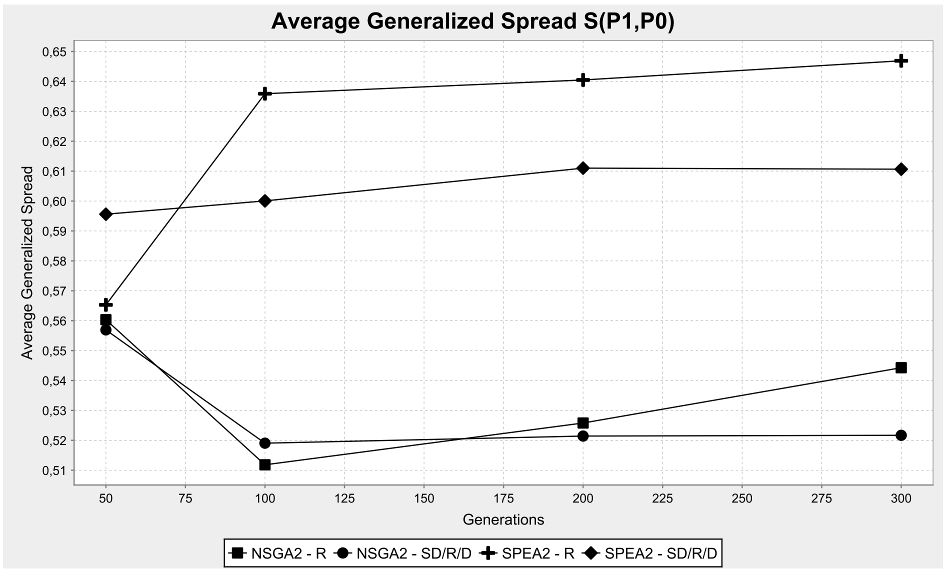


Figure 6.4: Spread for Example 1

In Figure 6.4, NSGA2-SD/R/D algorithm obtains the most stable and overall the best performance for spread followed by NSGA2-R which achieves a better spread value between 60. and 160. generations, but it achieves worse values for spread due to loss of uniformity in P1.

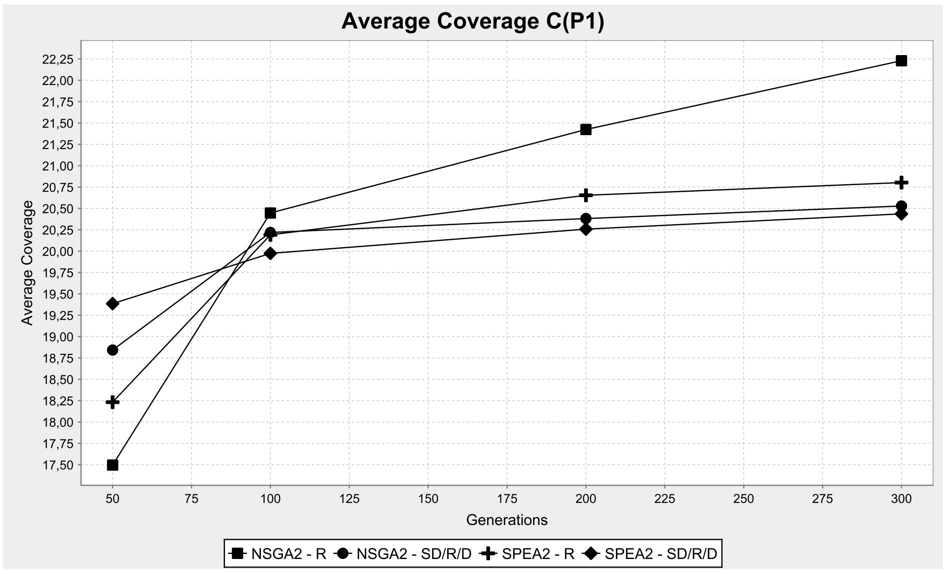


Figure 6.5: Coverage for Example 1

In Figure 6.5, R-algorithms cover a larger volume of the objective space than

SD/R/D algorithms, which is not surprising, since the used variation operations explore randomly towards any region of the solution space rather than regions with near-optimal solutions. NSGA2-R obtains the best coverage value.

Conclusion: NSGA2-SD/R/D is the algorithm with the best performance for this example, since it shows the best convergence behavior, most stable and best spread values and good coverage values.

Example 2: The true Pareto front $P0$ for this example consists of 3 non-dominated solutions. SD/R/D algorithms always obtain this solution between 20 and 100 generations. R algorithms obtain it rarely and between 250 and 300 generations.

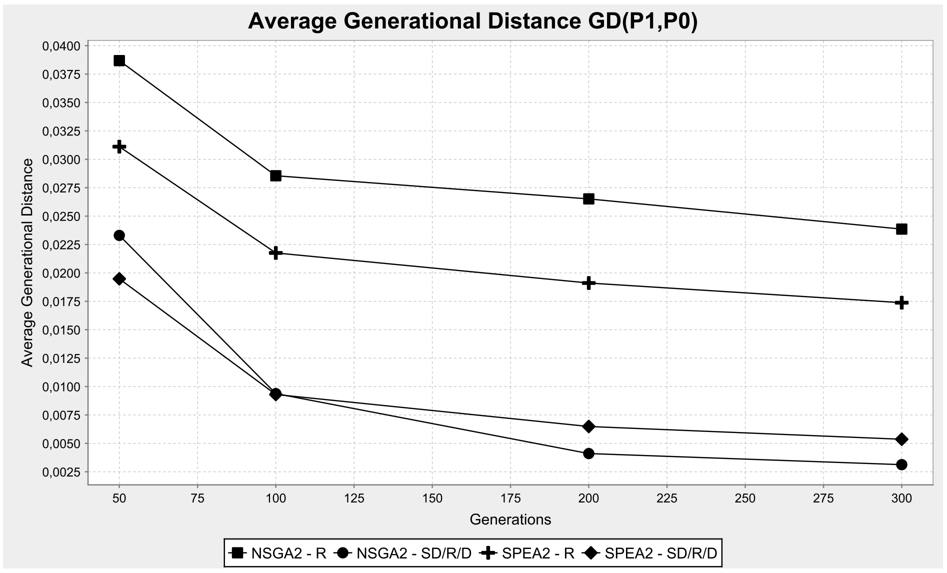


Figure 6.6: Generational Distance for Example 2

In Figure 6.6, SD/R/D algorithms converge clearly better than R algorithms. NSGA2-SD/R/D achieves the best convergence, however SPEA2-SD/R/D follows it very closely in all observation points.

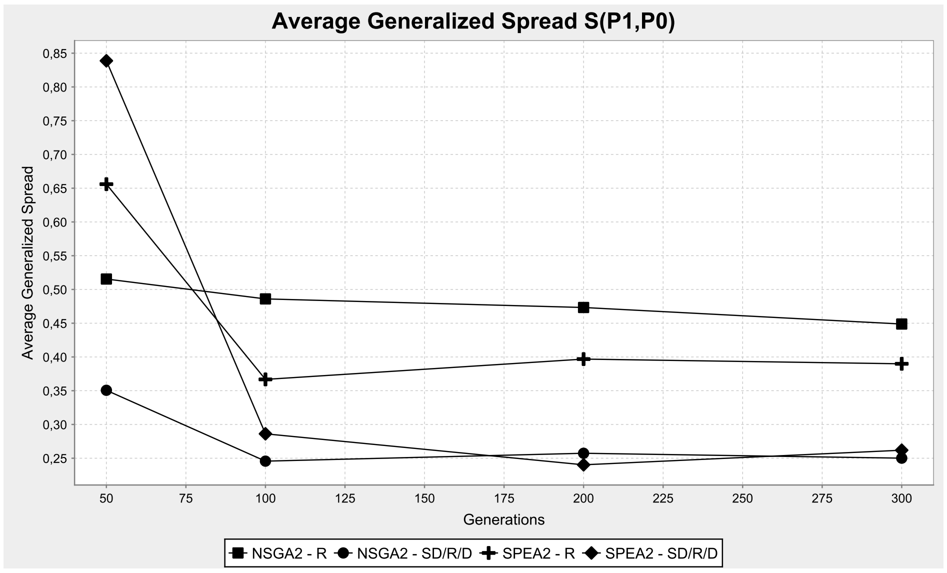


Figure 6.7: Spread for Example 2

In Figure 6.7, SD/R/D algorithms obtain overall the best results for spread. NSGA2-SD/R/D achieves in average the best results in the end population, however between 100. and 200. generations the spread worsens indicating a loss of uniformity in P1. SPEA2-SD/R/D shows a similar loss of uniformity of solutions between 200. and 300. generations and however NSGA2-SD/R/D reaches the best value for spread in the 300. generation.

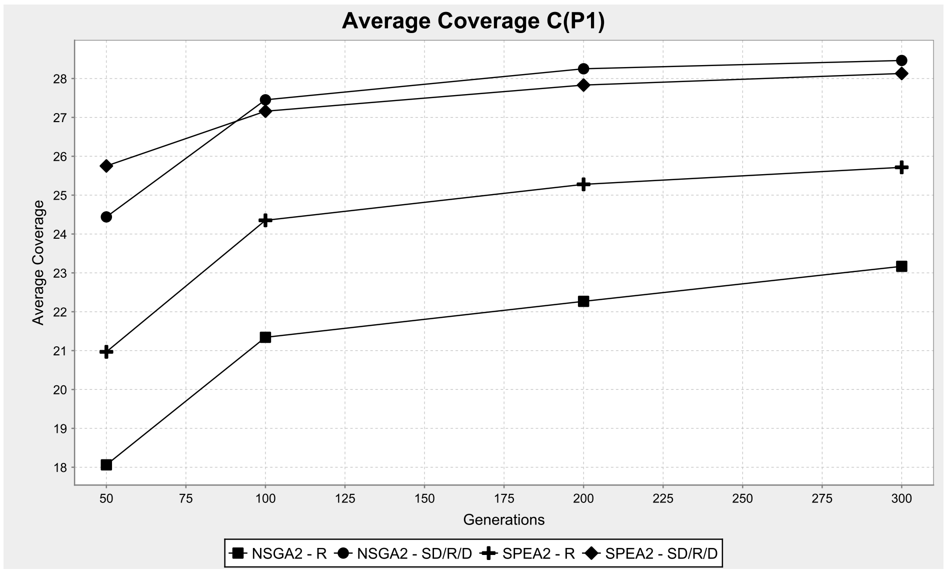


Figure 6.8: Coverage for Example 2

In Figure 6.8, SD/R/D algorithms obtain the best results for coverage. NSGA2-SD/R/D shows overall the best performance for coverage.

Conclusion: NSGA2-SD/R/D is the algorithm with the best performance for this example, since it shows the best convergence behavior, best final spread value and the best coverage value.

Example 3: The true Pareto front $P0$ for this example consists of 17 non-dominated solutions. All four algorithms frequently obtain at least one of these solutions between 100 and 300 generations.

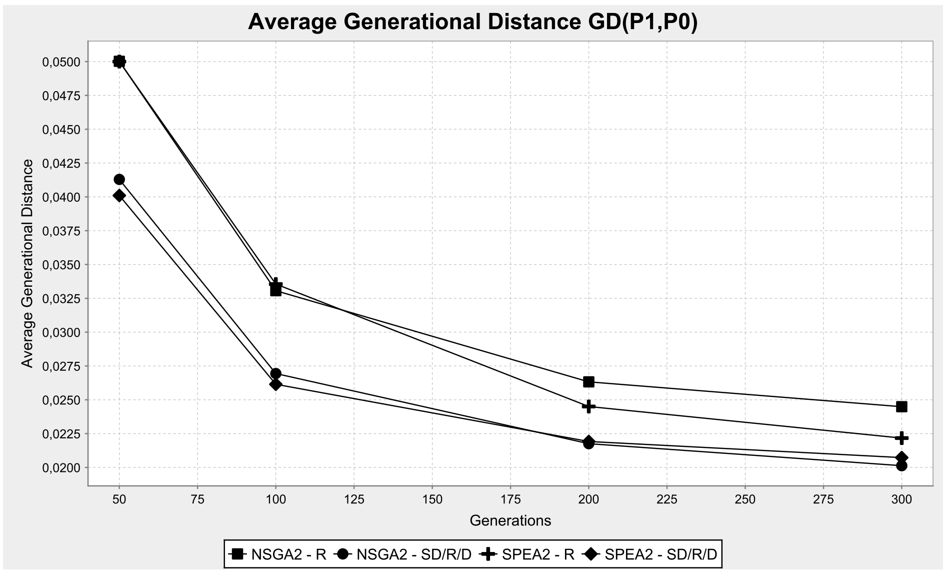


Figure 6.9: Generational Distance for Example 3

In Figure 6.9, SD/R/D algorithms converge slightly better than R algorithms. NSGA2-SD/R/D achieves the best convergence, however SPEA2-SD/R/D converges very closely in all observation points.

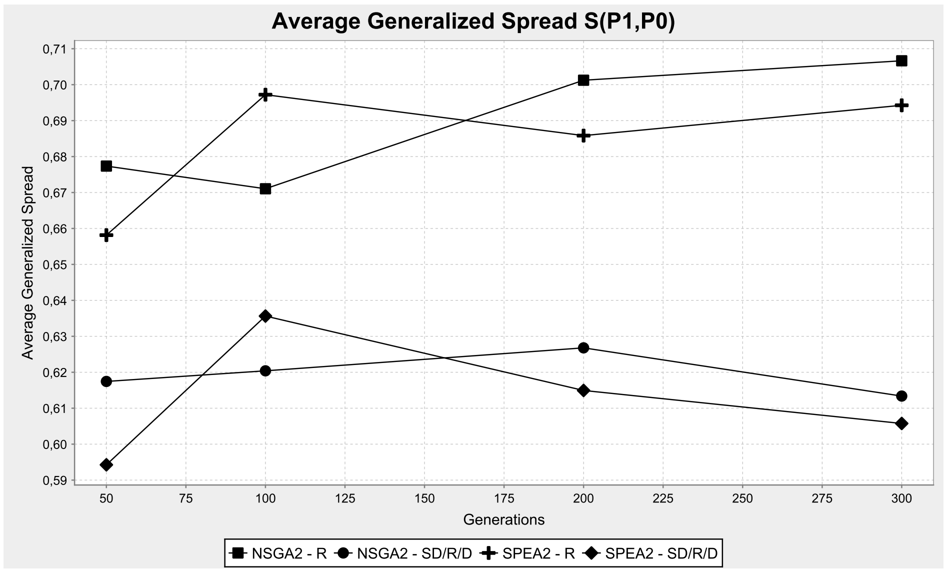


Figure 6.10: Spread for Example 3

In Figure 6.10, SD/R/D algorithms obtain the best results for spread in all generations. NSGA2-SD/R/D shows the most stable performance and SPEA2-SD/R/D obtains the best values.

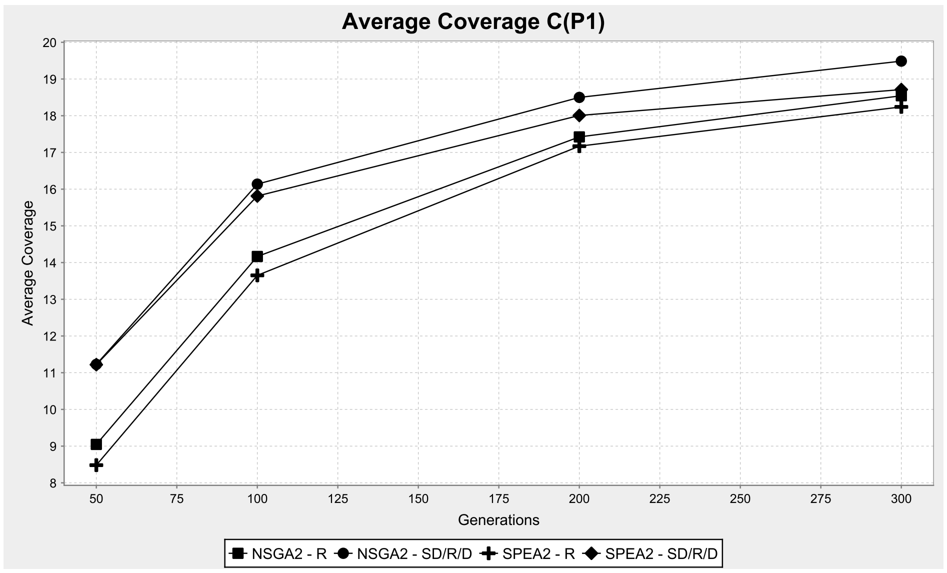


Figure 6.11: Coverage for Example 3

In Figure 6.11, SD/R/D algorithms achieve the best results for coverage. NSGA2-SD/R/D obtains overall the best coverage.

Conclusion: NSGA2-SD/R/D is the algorithm with the best performance for this example, since it shows the best convergence behavior, the most stable performance for spread and the best coverage values. However SPEA2-SD/R/D obtains very similar results and achieves the best overall performance for spread.

According to the results of the tests using the original repository, for all three examples SD/R/D algorithms obtain the best results for GD metric and overall competitive results for S and C metrics. Thus, SD/R/D algorithms are considered in the performance tests using the cloned repository for population size, cross-over probability and mutation probability analysis. Test results using the cloned repository are presented in Section 6.3.4, Section 6.3.5 and Section 6.3.6.

6.3.4 Population Size P - Analysis

Results of this analysis are obtained by tests using population size values $P = 40, 60, 80$ at a constant probability value for cross-over $p_C = 0.6$ and mutation $p_M = 0.6$. For SPEA2-SD/R/D algorithm size of archive is chosen to be identical to the population size P.

Example 1:

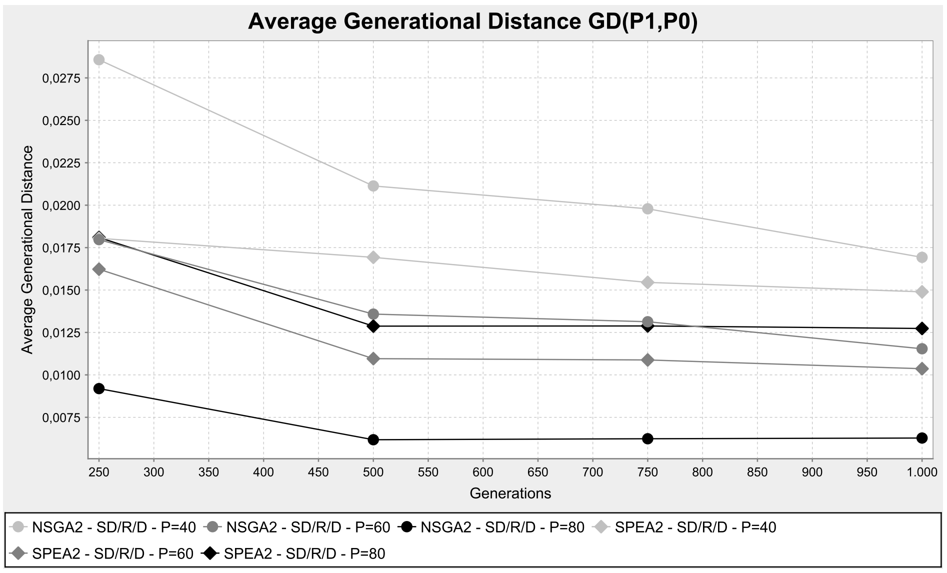


Figure 6.12: Generational Distance by P for Example 1

In Figure 6.12 NSGA2-SD/R/D for $P = 80$ achieves the best performance in all observed generations. An increase of population size for NSGA2-SD/R/D algorithm in this example achieves better results, whereas it is not the case for SPEA2-SD/R/D. It achieves its best results at an intermediate population size $P = 60$.

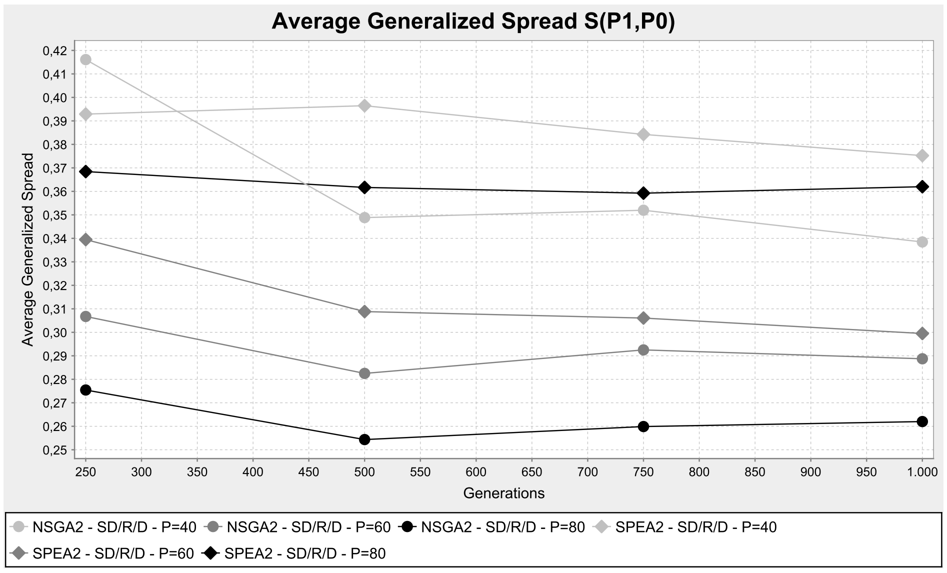


Figure 6.13: Spread by P for Example 1

In Figure 6.13 NSGA2-SD/R/D for $P = 80$ achieves the best values for spread.

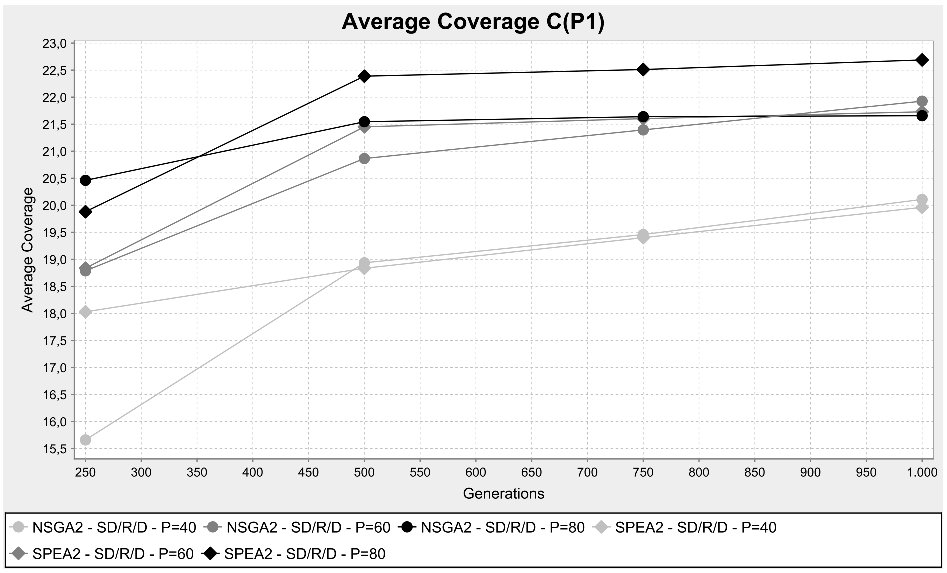


Figure 6.14: Coverage by P for Example 1

In Figure 6.14 SPEA2-SD/R/D for $P = 80$ achieves the best values for coverage after 350 generations. NSGA2-SD/R/D for $P = 80$ also achieves a good performance for coverage.

Conclusion: NSGA2-SD/R/D for $P = 80$ achieves the best performance for this example, since it shows the best convergence and spread values. It is followed by SPEA2-SD/R/D for $P = 60$, since it obtains similar detailed designs, good values for spread and also a good performance for coverage.

Example 2:

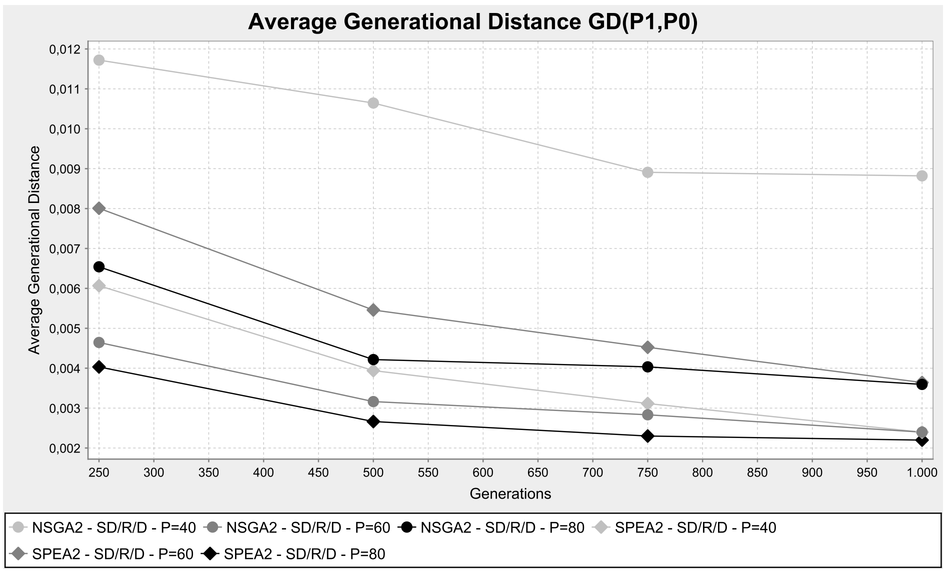


Figure 6.15: Generational Distance by P for Example 2

In Figure 6.15 SPEA2-SD/R/D for $P = 80$ achieves overall the best convergence overall followed by SPEA2-SD/R/D for $P = 40$ and NSGA2-SD/R/D for $P = 60$.

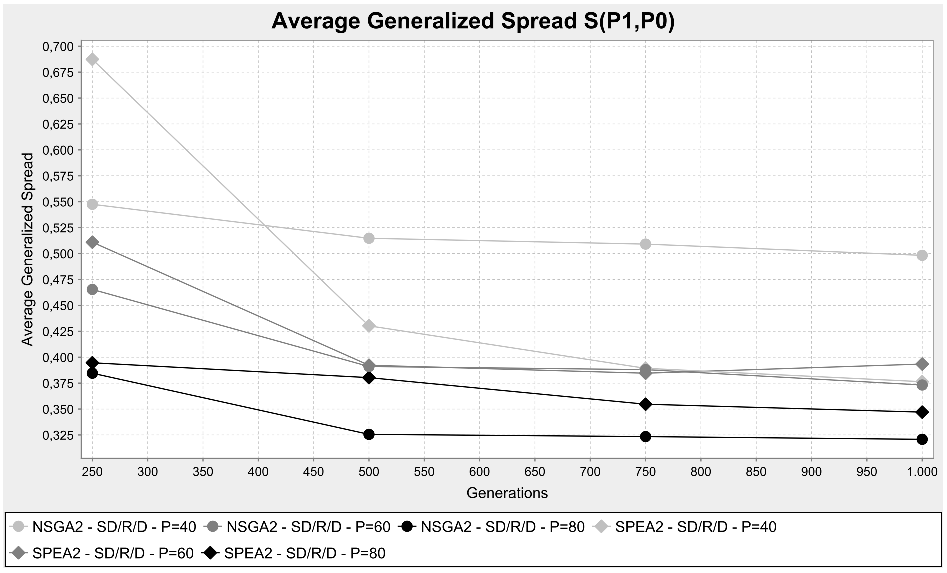


Figure 6.16: Spread by P for Example 2

In Figure 6.16 NSGA2-SD/R/D for $P = 80$ achieves the best performance for spread at all observed generations followed closely by SPEA2-SD/R/D for $P = 80$.

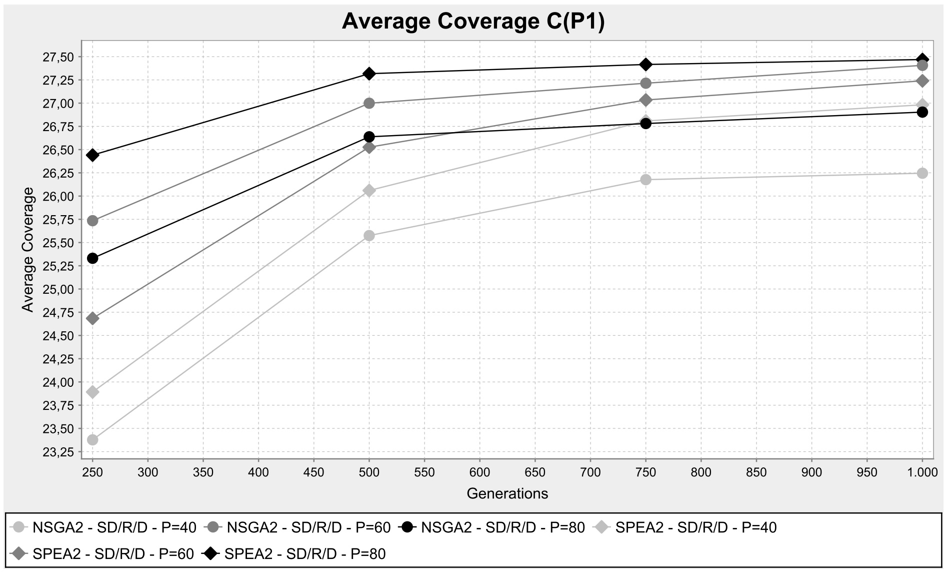


Figure 6.17: Coverage by P for Example 2

In Figure 6.17 SPEA2-SD/R/D for $P = 80$ achieves the best performance with NSGA2-SD/R/D for $P = 60$ and SPEA2-SD/R/D for $P = 60$ following closely.

Conclusion: SPEA2-SD/R/D for $P = 80$ achieves the best performance for this example, it achieves the best convergence, good spread and best coverage values.

Example 3:

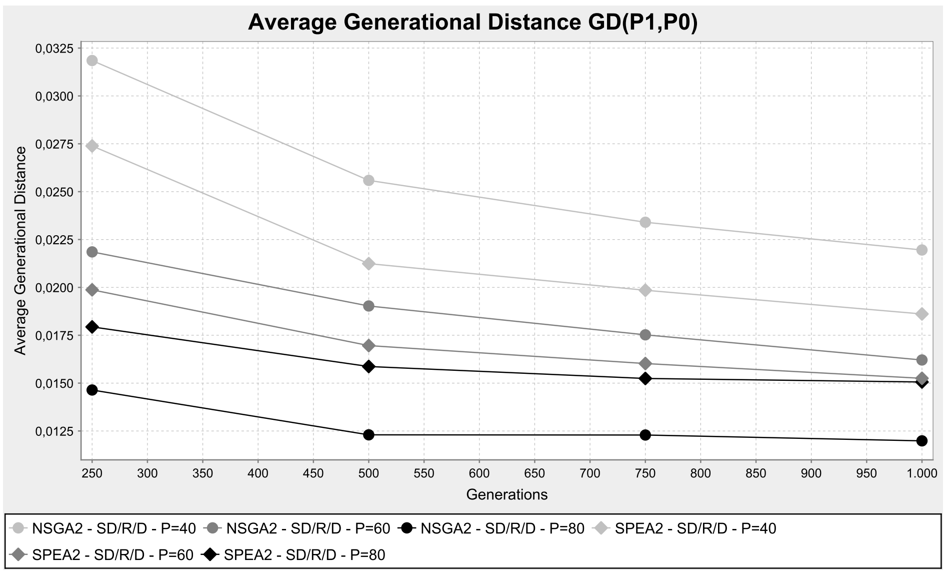


Figure 6.18: Generational Distance by P for Example 3

In Figure 6.18 NSGA2-SD/R/D for $P = 80$ achieves the best convergence values. It is followed by SPEA2-SD/R/D for $P = 80$, SPEA2-SD/R/D for $P = 60$ and NSGA2-SD/R/D for $P = 60$.

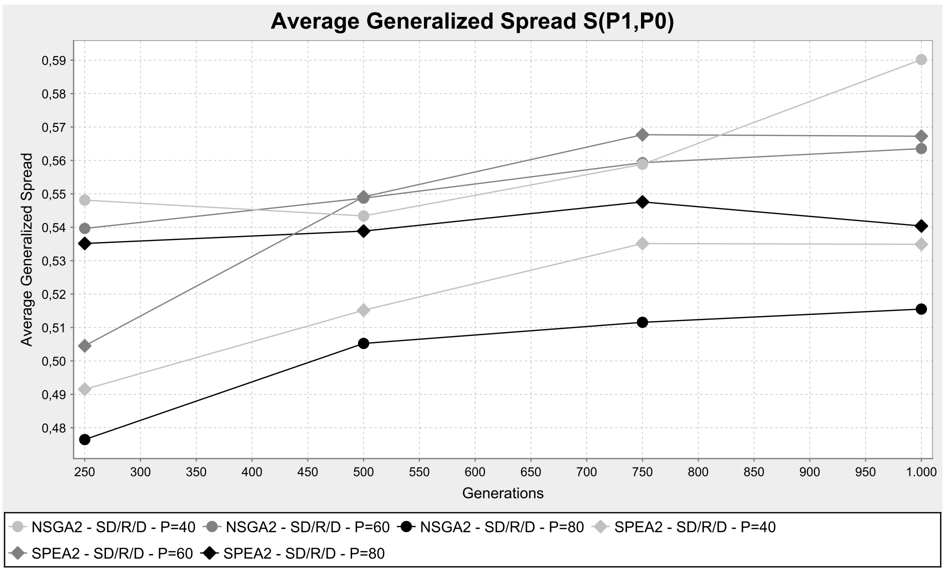


Figure 6.19: Spread by P for Example 3

In Figure 6.19 NSGA2-SD/R/D for $P = 80$ achieve the best performance for spread with SPEA2-SD/R/D for $P = 40$ as second best.

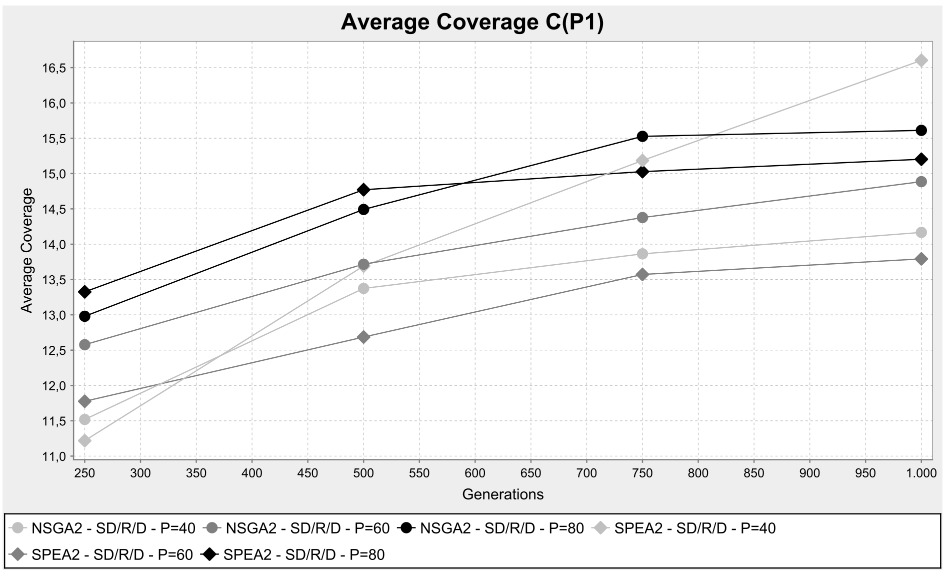


Figure 6.20: Coverage by P for Example 3

In Figure 6.20 SPEA2-SD/R/D for $P = 40$ achieves the best performance for coverage. NSGA2-SD/R/D and SPEA2-SD/R/D for $P = 80$ achieve also good coverage performances.

Conclusion: NSGA2-SD/R/D for $P = 80$ achieves overall the best performance, since it achieves the best values for convergence and spread, and also a good coverage performance.

6.3.5 Cross-Over Probability p_C - Analysis

Results of this analysis are obtained by tests using cross-over probability values $p_C = 0.4, 0.6, 0.8$ at constant values for population size $P = 60$ and for mutation probability $p_M = 0.6$. For SPEA2-SD/R/D algorithm size of archive is chosen to be identical to the population size P.

Example 1:

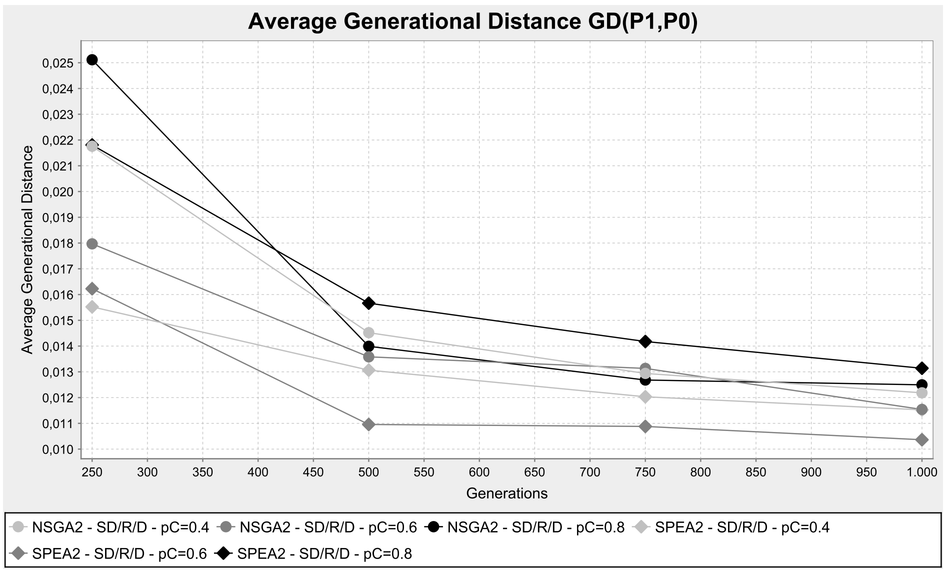


Figure 6.21: Generational Distance by p_C for Example 1

In Figure 6.21 SPEA2-SD/R/D for $p_C = 0.6$ achieves overall the best performance after 310. generation followed by NSGA2-SD/R/D for $p_C = 0.6$.

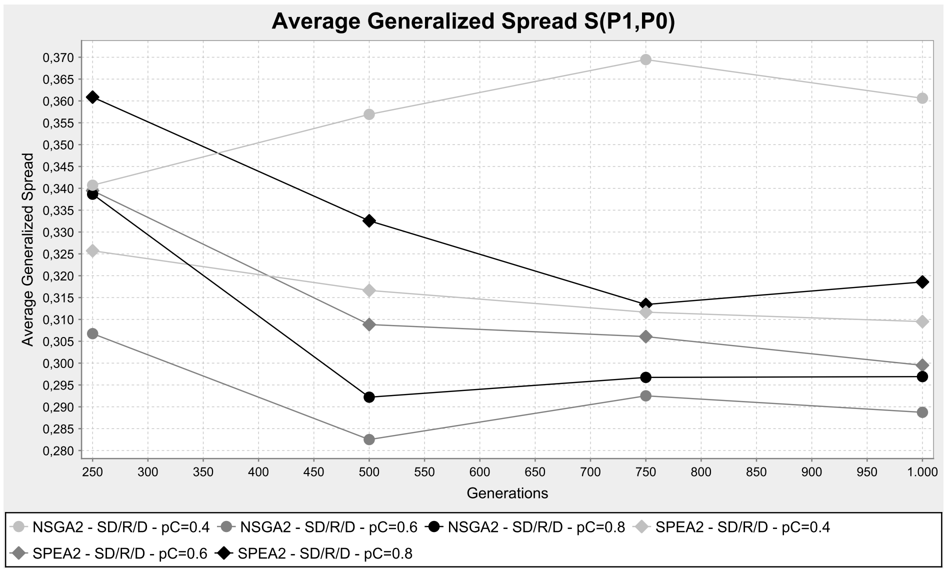


Figure 6.22: Spread by p_C for Example 1

In Figure 6.22 NSGA2-SD/R/D for $p_C = 0.6$ achieves the best performance for spread with some deterioration. SPEA2-SD/R/D for $p_C = 0.6$ also achieves a good and a more stable performance for spread.

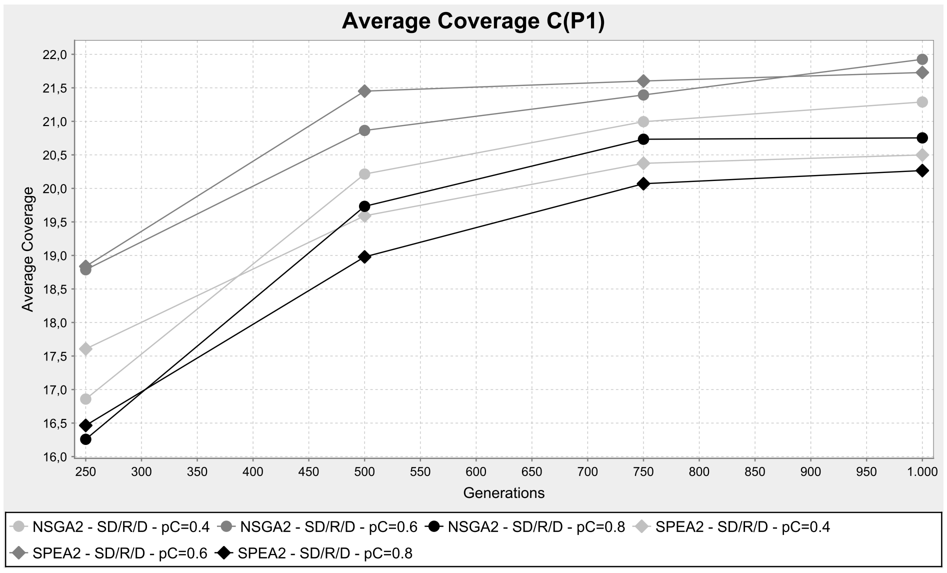


Figure 6.23: Coverage by p_C for Example 1

In Figure 6.23 NSGA2-SD/R/D and SPEA2-SD/R/D for $p_C = 0.6$ achieve two of the best performance results for coverage with NSGA2-SD/R/D or

$p_C = 0.6$ achieving the best coverage values after 880. generation.

Conclusion: SPEA2-SD/R/D for $p_C = 0.6$ achieves overall the best performance, since it obtains the best performance result for convergence, good and stable performance for spread, and a good performance for coverage.

Example 2:

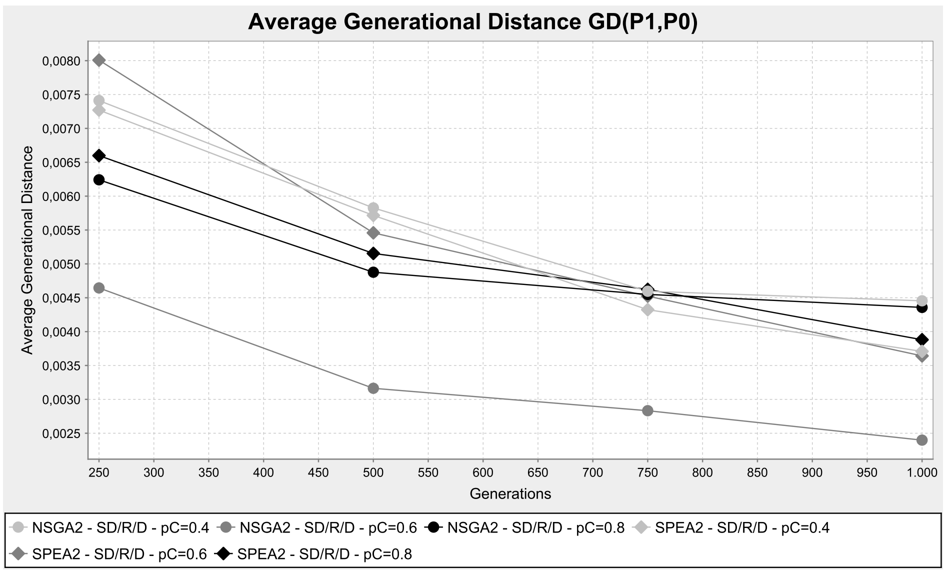


Figure 6.24: Generational Distance by p_C for Example 2

In Figure 6.24 NSGA2-SD/R/D for $p_C = 0.6$ achieves the best performance for convergence with a great difference compared to its nearest competitors.

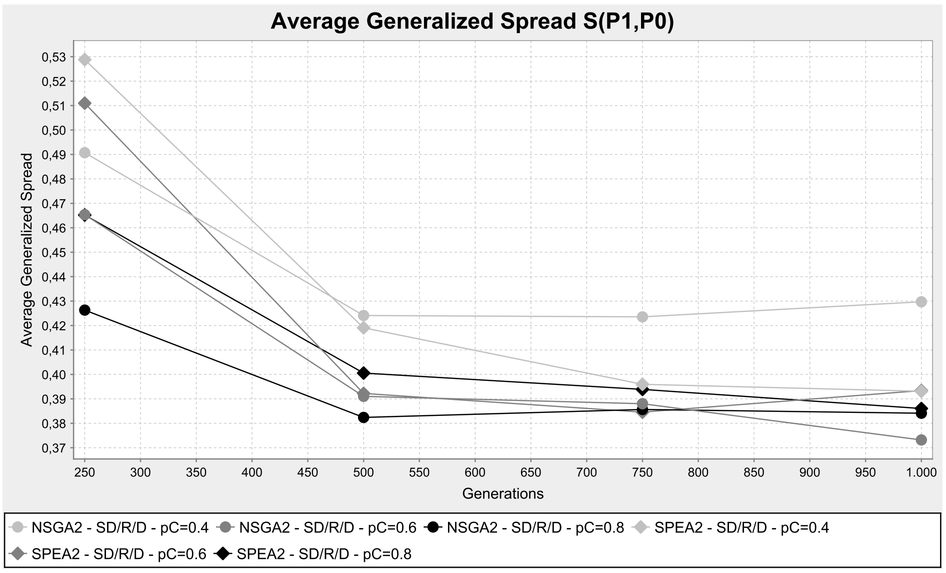


Figure 6.25: Spread by p_C for Example 2

In Figure 6.25 NSGA2-SD/R/D for $p_C = 0.6$ achieves the best performance for spread followed closely by its competitors.

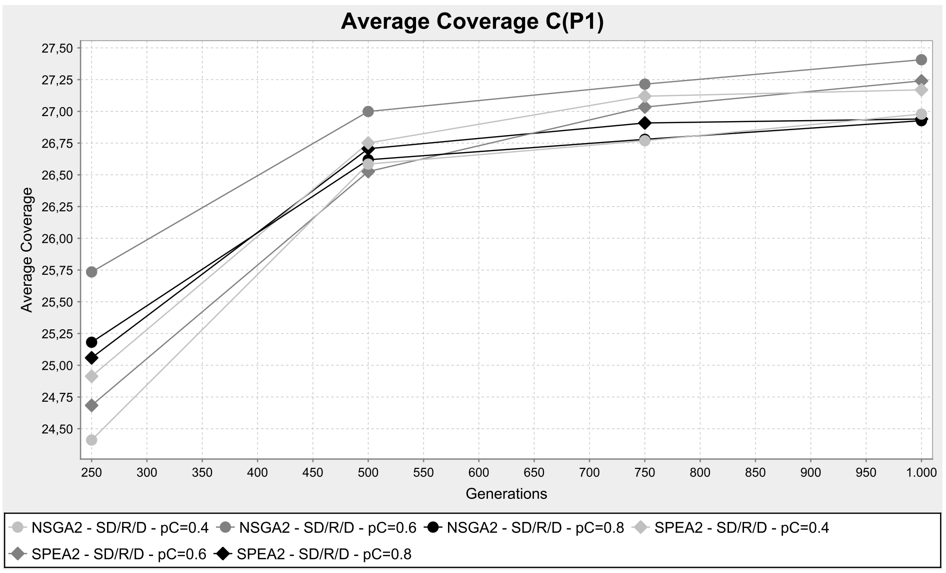


Figure 6.26: Coverage by p_C for Example 2

In Figure 6.26 NSGA2-SD/R/D for $p_C = 0.6$ achieves the best overall performance for coverage.

Conclusion: NSGA2-SD/R/D for $p_C = 0.6$ achieves best overall performance, since it achieves the best results for convergence, spread and coverage.

Example 3:

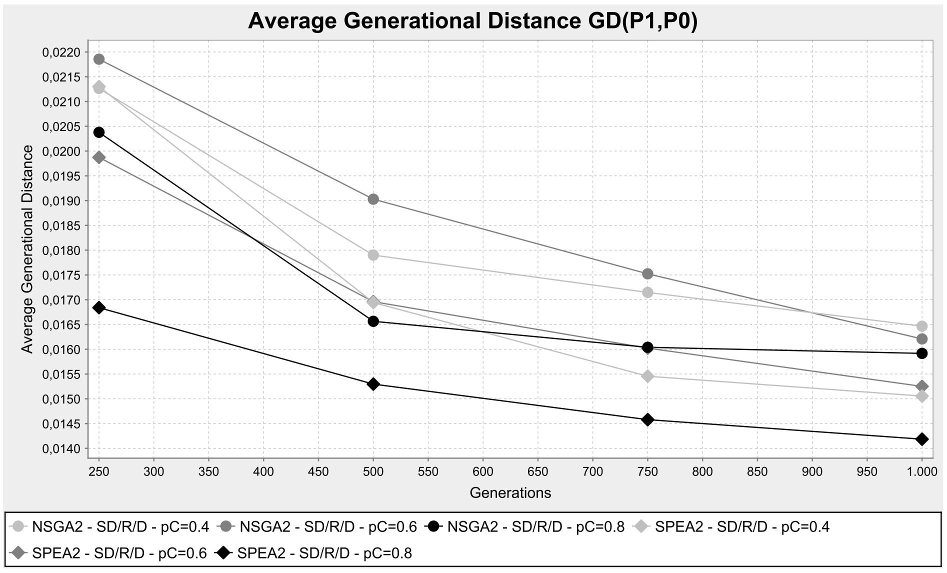


Figure 6.27: Generational Distance by p_C for Example 3

In Figure 6.27 SPEA2-SD/R/D for $p_C = 0.8$ achieves the best performance for convergence.

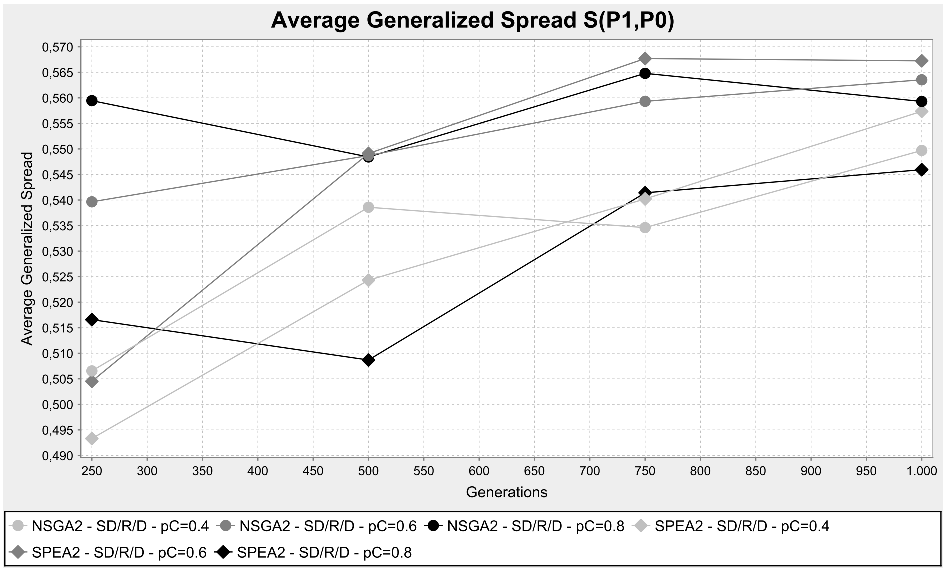


Figure 6.28: Spread by p_C for Example 3

In Figure 6.28 SPEA2-SD/R/D for $p_C = 0.8$ achieves the best performance for spread. However, its uniform distribution worsens after 500. generation despite an improving convergence (cf. Figure 6.27).

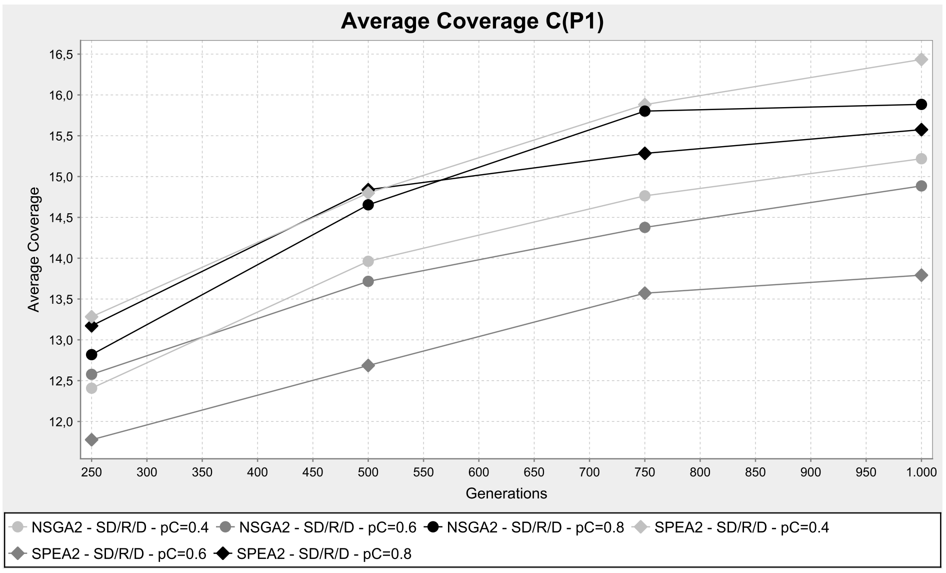


Figure 6.29: Coverage by p_C for Example 3

In Figure 6.29 SPEA2-SD/R/D for $p_C = 0.4$ achieves the best performance for coverage followed by NSGA2-SD/R/D for $p_C = 0.8$ and SPEA2-SD/R/D for $p_C = 0.8$.

Conclusion: SPEA2-SD/R/D for $p_C = 0.8$ achieves the best overall performance, since it achieves the best convergence for convergence and spread, and a good performance for coverage.

6.3.6 Mutation Probability p_M - Analysis

Results of this analysis are obtained by tests using mutation probability values $p_M = 0.2, 0.4, 0.6$ at constant values for population size $P = 60$ and for mutation probability $p_C = 0.6$. For SPEA2-SD/R/D algorithm size of archive is chosen to be identical to the population size P.

Example 1:

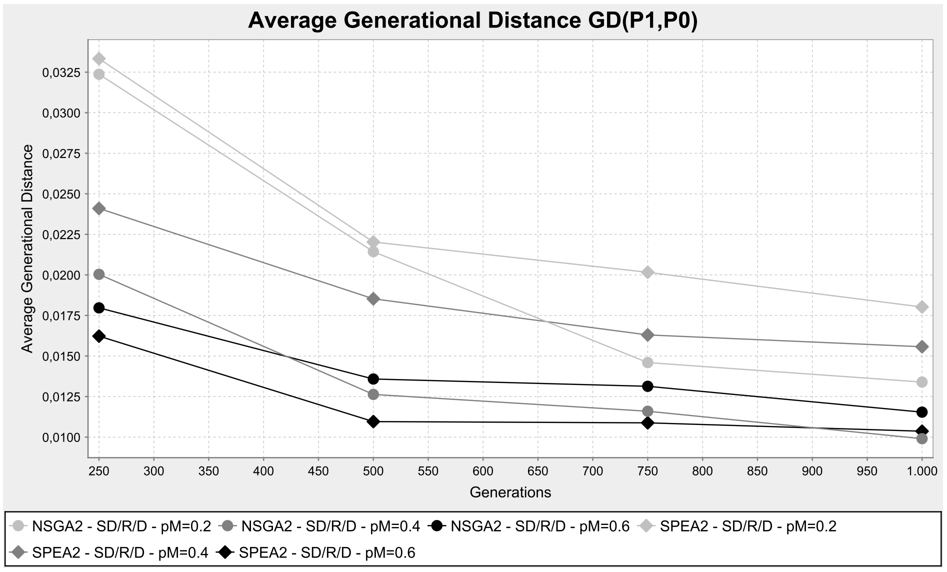


Figure 6.30: Generational Distance by p_M for Example 1

In Figure 6.30 NSGA2-SD/R/D for $p_M = 0.4$ achieves the best performance for convergence followed closely by SPEA2-SD/R/D for $p_M = 0.6$.

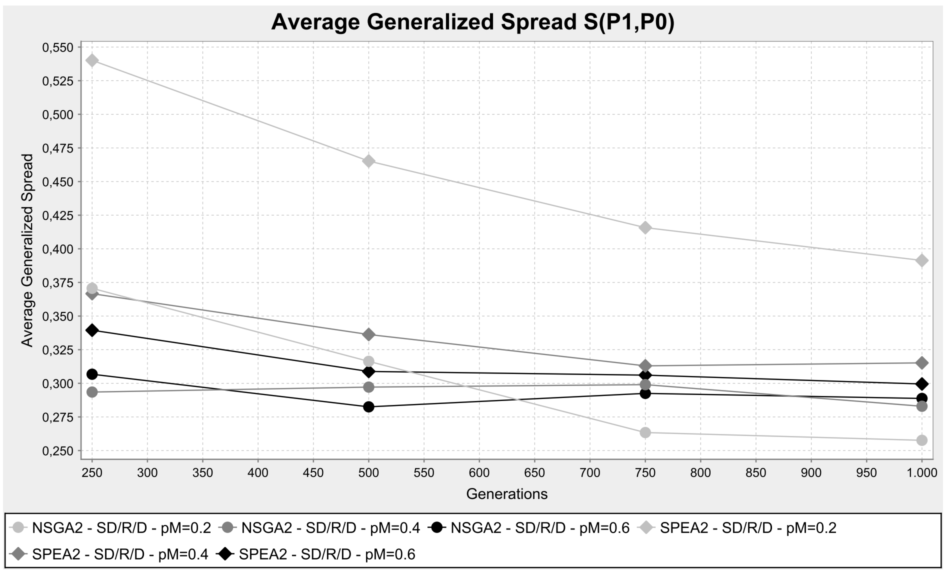


Figure 6.31: Spread by p_M for Example 1

In Figure 6.31 NSGA2-SD/R/D for $p_M = 0.2$ achieves the best performance for spread following a stable decreasing course.

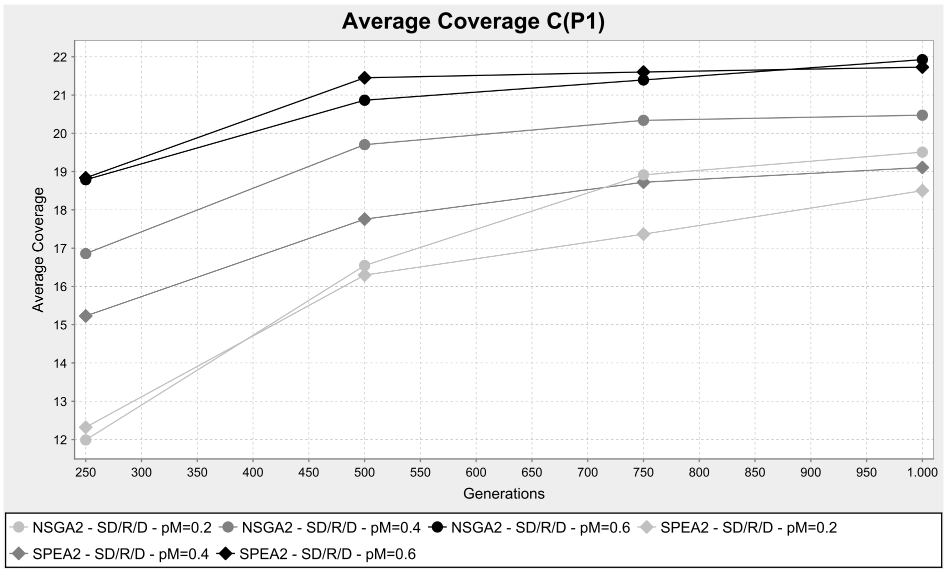


Figure 6.32: Coverage by p_M for Example 1

In Figure 6.32 NSGA2-SD/R/D and SPEA2-SD/R/D for $p_M = 0.6$ achieve the best performances for coverage with NSGA2-SD/R/D for $p_M = 0.6$ slightly in front.

Conclusion: NSGA2-SD/R/D for $p_M = 0.4$ and SPEA2-SD/R/D for $p_M = 0.6$ achieve overall the best performances, since NSGA2-SD/R/D for $p_M = 0.4$ achieves the best performance for convergence, a good performance for spread and coverage. SPEA2-SD/R/D for $p_M = 0.6$ achieves the second best performance for convergence slightly behind NSGA2-SD/R/D for $p_M = 0.4$, but it achieves a better performance for coverage.

Example 2:

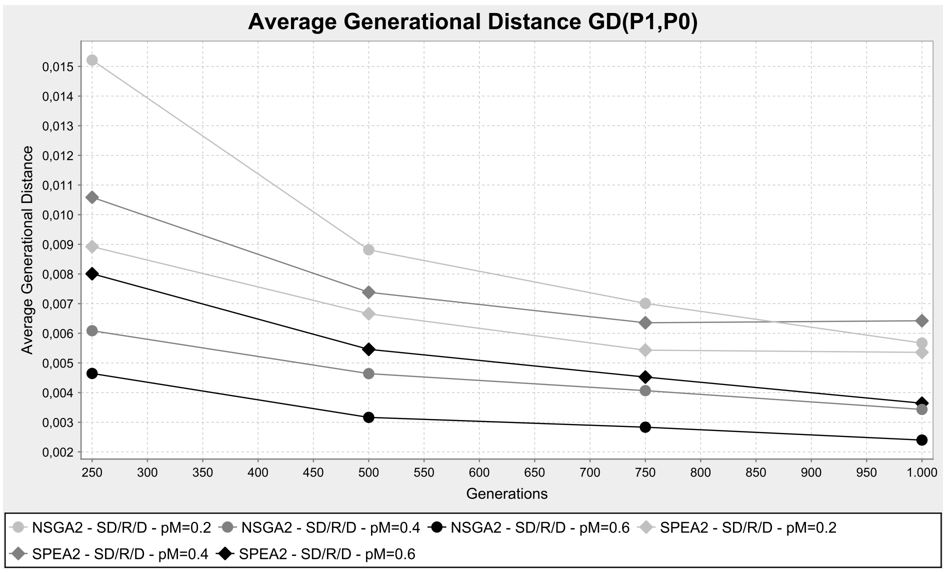


Figure 6.33: Generational Distance by p_M for Example 2

In Figure 6.33 NSGA2-SD/R/D for $p_M = 0.6$ achieves the best performance for convergence followed by NSGA2-SD/R/D for $p_M = 0.4$ and SPEA2-SD/R/D for $p_M = 0.6$.

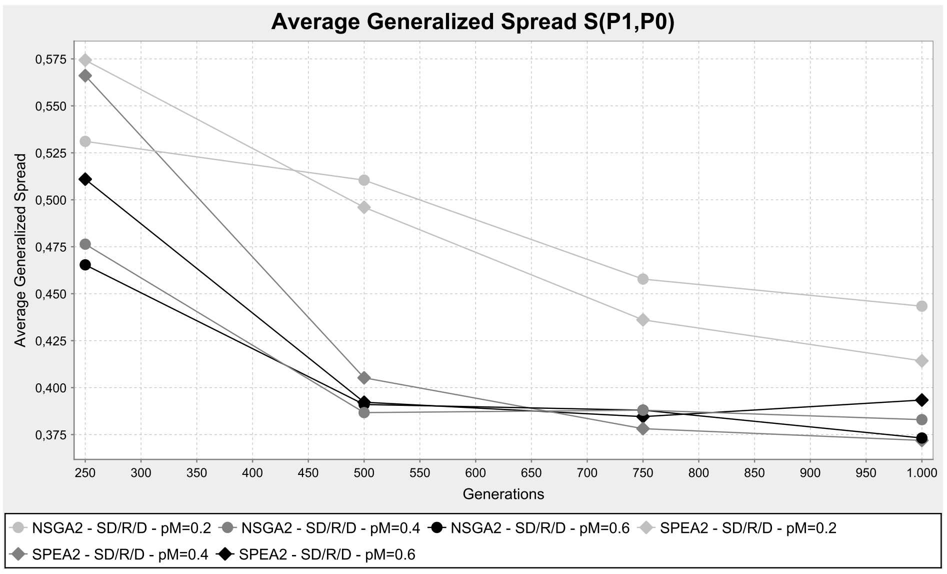


Figure 6.34: Spread by p_M for Example 2

In Figure 6.34 SPEA2-SD/R/D for $p_M = 0.4$ and NSGA2-SD/R/D for $p_M =$

0.6 achieves two best performances for spread. SPEA2-SD/R/D for $p_M = 0.4$ exhibits a greater improvement between 250. and 1000. generations.

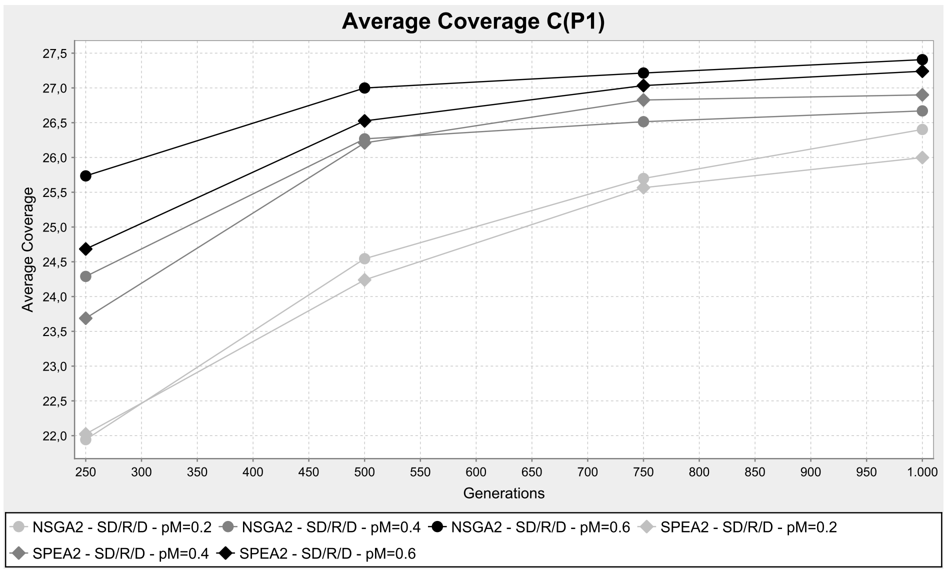


Figure 6.35: Coverage by p_M for Example 2

In Figure 6.35 NSGA2-SD/R/D for $p_M = 0.6$ achieves the best performances for coverage with SPEA2-SD/R/D for $p_M = 0.6$ slightly behind.

Conclusion: Overall NSGA2-SD/R/D for $p_M = 0.6$ achieve the best performance, since it achieves the best performance for convergence, spread and coverage.

Example 3:

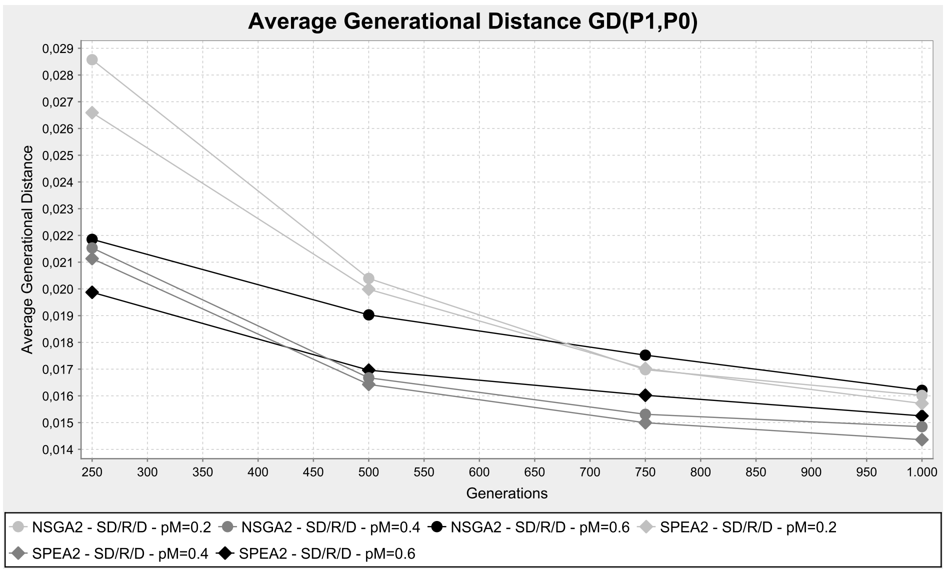


Figure 6.36: Generational Distance by p_M for Example 3

In Figure 6.36 SPEA2-SD/R/D for $p_M = 0.4$ achieves the best performances for convergence with NSGA2-SD/R/D for $p_M = 0.4$ slightly behind.

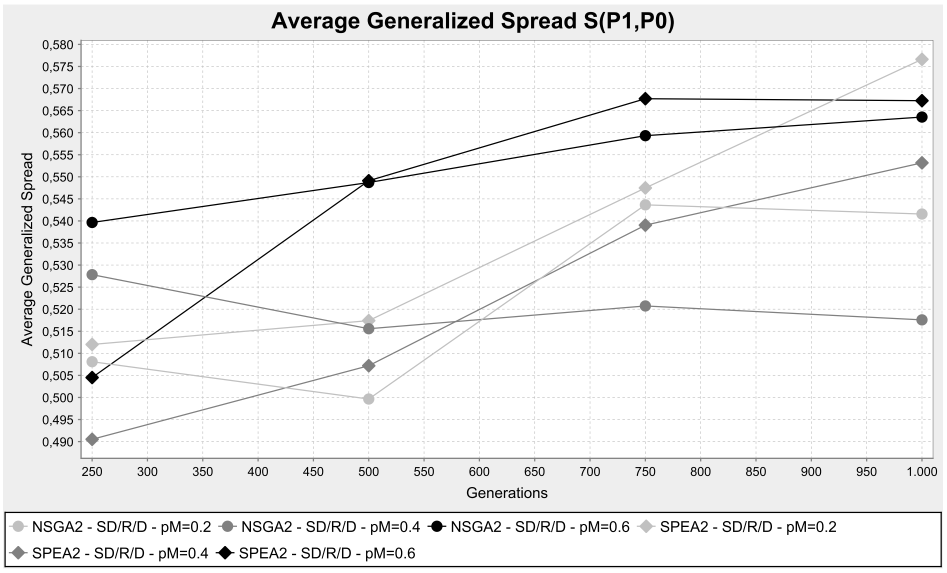


Figure 6.37: Spread by p_M for Example 3

In Figure 6.37 NSGA2-SD/R/D for $p_M = 0.4$ achieves the best performance for spread with least deterioration.

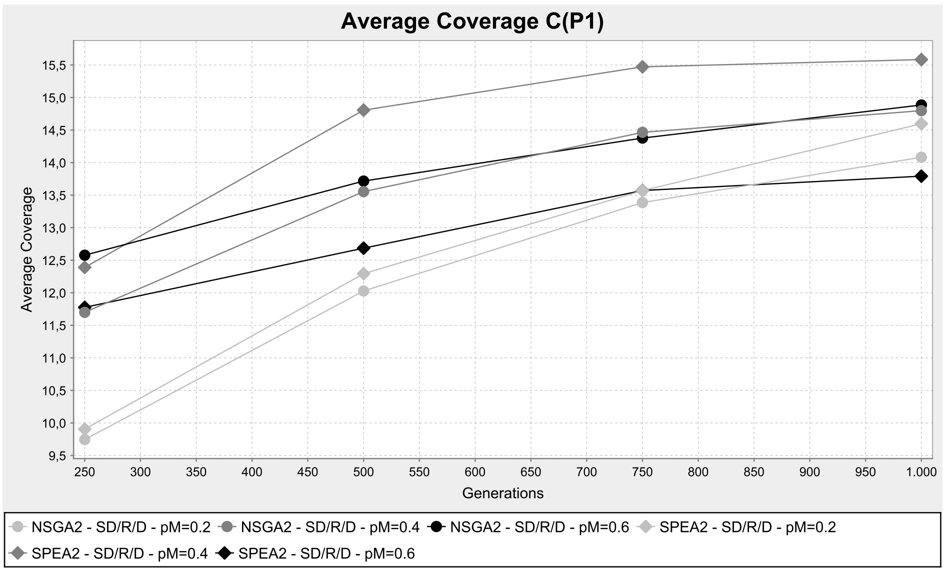


Figure 6.38: Coverage by p_M for Example 3

In Figure 6.38 SPEA2-SD/R/D for $p_M = 0.4$ achieves the best performance for coverage followed by NSGA2-SD/R/D for $p_M = 0.6$ and $p_M = 0.4$.

Conclusion: SPEA2-SD/R/D and NSGA2-SD/R/D for $p_M = 0.4$ achieve overall best performances, since SPEA2-SD/R/D for $p_M = 0.4$ achieves the best performance for convergence and coverage and NSGA2-SD/R/D for $p_M = 0.4$ achieves the second best performance for convergence and best performance for spread.

6.3.7 Discussion for Optimization Results and Example Designs

If test results in Section 6.3.4, 6.3.5 and 6.3.6 are considered, the tests do not concern all triples of chosen parameter settings for P, p_C and p_M applied with the two algorithms NSGA2-SD/R/D and SPEA2-SD/R/D. For examples Example1 to 3, algorithms and parameter settings that achieve good performances are identified.

For Example1 one of the best performances is achieved with the algorithm NSGA-SD/R/D and parameter setting $P = 80$, $p_C = 0.6$ and $p_M = 0.6$. For Example2 one of the best performances is achieved with the algorithm NSGA2-SD/R/D and parameter setting $P = 60$, $p_C = 0.6$ and $p_M = 0.6$. For Example3 one of the best performances is achieved with the algorithm NSGA-SD/R/D and parameter setting $P = 80$, $p_C = 0.6$ and $p_M = 0.6$. For test results with the cloned repository in Appendix A these algorithms and parameter settings are chosen.

The tests concern 3 examples for 3 parameter analysis, hence 9 cases for a comparison between NSGA-SD/R/D and SPEA2-SD/R/D algorithms, if generational distance is considered, which are Example 1 to 3 for P, p_C and p_M. In these 9 cases, NSGA-SD/R/D achieves the best performance in 5 cases and SPEA2-SD/R/D achieves the best performance in 4 cases. Hence, with the problem-specific adaptation, these algorithms obtain almost indistinguishable performances. Both algorithms apply a goal-oriented search using rankings in contrast to NSGA-R and SPEA2-R algorithms and hence achieve better performances. Thus, it can be concluded that an adequate problem-specific adaptation of a chosen algorithm is crucial, if good performances and high quality results are desired.

Further, the test results imply that an increase of population size, cross-over probability or mutation probability does not always lead to the best performance e.g. if results for Example2 with the cloned repository in Figure 6.15 are considered, a higher population size does not always lead to the best performance results, since NSGA2-SD/R/D for $P = 60$ achieves better results than NSGA2-SD/R/D for $P = 80$. For all three examples Example1 to 3 intermediate rate 0.6 for p_C achieves good performances. This is related to the reduced genetic drift. Further, the highest mutation probability $p_M = 0.6$ achieves good performances for all three examples.

In Appendix A, in case of Example1 with tests using the cloned repository individual runs achieve frequently some of the results in the true Pareto front. Due to this fact for Example1 in Figure A.1 a detailed design example for the original repository is presented in Figure A.2 with mappings in Table A.1 and in order to prevent a repetition, a detailed design example for the cloned repository for Example1 is omitted. Further, optimal detailed design examples obtained using the original repository that realize Example2 in Figure A.3 and A.4, and Example3 in Figure A.6 are presented in Figure A.5 and A.7 respectively. Mappings for Example2 and 3 among the abstract designs and detailed designs are presented in Table A.2 and A.3 respectively. In Appendix A in tables devices and functional profiles are enumerated in each detailed design they belong. The numbers of devices and functional profiles are attached to their names as suffixes.

In case of Example2 and 3 for tests with the cloned repository some optimal results are achieved. In Figure A.8 and A.9 two of the achieved near-optimal detailed designs are presented with abstract design to detailed design mappings in Table A.4 and A.5 respectively.

If the optimal detailed design example in Figure A.5 with mappings in Table A.2 and the near-optimal detailed design example in Figure A.8 with mappings in Table A.4 are compared, mainly 4 devices 7, 17, 22, 24 in Table A.4 are different from the devices 3, 4, 10, 16 in Table A.2 with identical function block combinations. Since devices 7, 17, 22, 24 are copies with higher device price costs, sum of device price costs for detailed design in Table A.4 is approximately 1,84 % higher than sum of device price costs in Table A.2 which can be considered as a very good result. It should also be remarked that some of the runs with the cloned repository also achieve the optimal detailed design result in Table A.4.

The BAS technology for realization of the Example2 in case of both tests using the original and cloned repository is EnOcean. In order to consider only EnOcean devices in design generation and improvement steps, function blocks are assigned an attribute value EnOcean for the attribute of the non-functional requirement technology platform.

Moreover, some function blocks are desired to be on separate devices or same devices as explained in Section 5.7.4. Such function blocks are assigned a corresponding non-functional requirement in form of an attribute as presented in Table 6.1.

Function Block	**Group Attribute Device**
S Window left	dev1
S Wall left	dev2
S Window right	dev3
S Wall right	dev4
Set Temperature Setpoint left	dev5
Set Temperature Setpoint right	dev6
Temperature Sensor right	dev7
Luminance Sensor right	
Occupancy Sensor right	
S Window middle	dev8
S Wall middle	dev9
Occupancy Sensor left	dev10
Luminance Sensor left	
Temperature Sensor left	

Table 6.1: Preferences for Device Mappings by Group Attributes

As can be observed function blocks in Table 6.1 are mapped onto the devices as specified by the non-functional requirements, which implies correct mappings on function blocks to devices. In case, if these non-functional requirements are not specified, the design generation and improvement steps allow the creation of detailed designs with incorrect mappings e.g. function blocks “Luminance Sensor left”, “Occupancy Sensor right” and “Temperature Sensor right” onto the same device. Mapping of groups of function blocks “Occupancy Control right1”, “Control Mode Selection right”, “Temperature Setpoint Evaluation right”, “Temperature Control right”, “Radiator right” and “Occupancy Control left1”, “Control Mode Selection left”, “Temperature Setpoint Evaluation left”, “Temperature Control left”, “Radiator left” onto same devices is quickly achieved by using the variation operation SD-M6 in Section 5.5.2.

If the optimal detailed design example for Example3 in Figure A.7 with mappings in Table A.3 and the near-optimal detailed design example in Figure

A.9 with mappings in Table A.5 are compared, Table A.5 consists of 1 device more than Table A.3 and sum of price device costs are approximately 10 % higher, which can be considered as a good result. This is due to an extra device and the contained cloned device number 9 in Table A.5 that is 87 Euros more expensive than its original number 3 in Table A.3. Moreover, on device number 5 in Table A.3 multiple functions are mapped, which are mapped on multiple devices in Table A.5. Also for this example individual runs in tests with the cloned repository achieve some of the optimal detailed designs obtained in tests with the original repository.

6.3.8 Resource Consumption

In order to validate the suitability of the proposed solution method in Chapter 5 for the requirement of limited computation time in Section 3.3.2, an analysis for average resource consumption in the example of NSGA2-SD/R/D algorithm is performed. This requires information on processor type and speed, amount of used memory, amount of computation time and number of fitness evaluations (number of generations times the number of population size) is necessary. The algorithm is run using the Example2 on two different computer configurations with dual core (Intel Core 2 Duo 2 GHz) and quad-core (Intel i7-3770 3.40 GHz) processors respectively. Size for java heap space for the individual runs is chosen to be 1GB.

System	**Component Repository**	**Fitness Evaluations**	**T**
Processor: Intel Core 2 Duo 2.0 GHz Ram: 3GB OS: Windows	original	12000	< 1 min.
		18000	< 1 min.
		24000	1 min.
	cloned	40000	6 min.
		60000	9 min.
		80000	12 min.
Processor: Intel Core i7-3770 3.40 GHz Ram: 12 GB OS: Windows	original	12000	< 1 min.
		18000	< 1 min.
		24000	< 1 min.
	cloned	40000	3 min.
		60000	4 min.
		80000	5 min.

Table 6.2: Average Resource Consumption for Example 2

With computer configurations competitive detailed design solutions in Table 6.2 are obtained that include good near-optimal solutions and some optimal solutions. T represents average time consumption for computation. For each of 12 cases in Table 6.2 results of component search as presented in Section

5.4 and device interoperability evaluations have been cached by performing a first run, since these steps are related to an overhead caused by an access to the component repository and this overhead must be minimal for an analysis of time consumption of computation performed in design generation and improvement steps.

Although the results obtained using the first computer configuration needed more time for computation, the requirement for limited computation time in Section 2.9 could still be fulfilled as in case of the results obtained using the second computer configuration.

6.3.9 Parallelism

It has been observed that results from multiple runs can be considered to obtain a better set of optimized solutions containing more optimal solutions and better near-optimal solutions than an individual run can obtain, however this requires a significant amount of additional time, if these runs were performed sequentially.

In order to obtain benefits from the multi-core architecture and memory capacity of the computer configurations and to achieve better solutions at the same time, multiple runs (e.g. 5 runs) can be performed in parallel in a comparable computational time as the computation time required for an individual run.

A number of runs can be started simultaneously and allowed to run in parallel. After aggregation of all optimization results of these runs to a common set of solutions for each problem-instance, a non-dominated set from all runs can be obtained. This set can contain more optimal solutions and better near-optimal solutions. This approach has been tested using the faster computer configuration in Table 6.2 for 5 simultaneous runs.

It has been observed that the aggregated set of non-dominated solutions contained more optimal solutions and better near-optimal solutions than individual runs. Hence, an introduction of such an algorithm independent parallelism can contribute on achievement of better results for the optimization, which can be applied independent from any given problem.

6.4 Optimization Framework

Many real world problems of engineering practice are related to design synthesis and optimization. These problems concern different instances depending on the design components and parameters involved in their specifications. For solution of the problems, optimization frameworks meeting certain requirements are deployed rather than hard-coded and problem-specific adaptations of algorithm implementations. Such optimization frameworks are designed based on a distinction of conceptual components such as problem specification, algorithm choice, and presentation of results which cooperate via common interfaces. This allows an extension of a conceptual component independently from the rest and provides design and code reuse. This is particularly necessary, if

- there are different design problems or problem instances that are required to be solved;
- the problems are required to be solved with different algorithms along with different parameter settings and operations to search for an algorithm with the best performance;
- the optimization results and solutions for different problems and algorithms must be presented;
- and if a setup of a design synthesis and optimization environment with a relatively little effort is required.

For obtaining high quality optimization results, any chosen solving algorithm should be adapted to a given problem by using a convenient problem representation and algorithm operations designed for this representation. Thus, in order to meet these requirements, a framework design must allow the definition of problem-specific extensions e.g. problem-specific representation and problem-specific operations. These extensions must be plugged into the framework by minimal implementation effort.

Second, it should be possible to approach given problems with different solving algorithms; each time, by only choosing the algorithm and its problem-specific operations. Moreover, a generic extension of existing algorithms should be allowed to special variants according to the framework design specification.

Third, presentation of the optimization results and solutions should also be plugged into the framework via generic interfaces.

Current optimization frameworks are focused on evolutionary algorithms such as ECJ [ECJ], EVA2 [Unib], and JCLEC [VRZ$^+$07] exist, which provide implementation of problem independent optimization operations (e.g. crossover, mutation, selection, etc.) designed for general purpose representation schemes such as string and tree-based representations etc. In these frameworks, as benchmark problems, mostly standard function optimization problems such as ZDT function series, Rosenbrock and Rastrigin function; or multiplexer, regression and parity problems [HHBW06] based on general representations are integrated.

BA design creation problem is a constrained multi-objective combinatorial optimization problem for assigning one-to-many function blocks to software modules and also one-to-many functional profiles to interoperable devices including generation of a complex logical network with intermeshed connections. Thus, the problem (cf. Section 3.3.1 and Figure 3.2) and its representation (cf. Section 5.6.2) is much more complex than in the case of function optimization or tree representation based problems.

The existing frameworks do not provide representations and algorithms with operations that can be as useful as ready-to-use implementations to solve the design creation problem. The integration of a problem-specific representation and operations into the existing frameworks require deep modifications in their component hierarchies. Definition of different operation variants of the same operation type and switching between different variants in runtime according to an algorithm is not provided by design of these frameworks. The proposed framework is designed to support these features.

6.5 Framework Design

The optimization framework design consists of parts that are provided for distinct tasks. Information exchange among the tasks is performed via defined interfaces that are common to both communication sides. Hence, problem specification with related setup operations, algorithm logic, program flow, display of results and solutions, and export of solutions are all decoupled from each other in design and implementation.

6.5.1 Components and Interfaces

The tasks performed by the framework [OPK10b] are accomplished by diverse components that are integrated in Java packages. The components are implemented as Java classes and interfaces; and can be separated into three groups for belonging to the optimization framework and information flow direction:

1. *input components* (gray boxes left to *Optimization Control* in Figure 6.39) do not belong to the optimization framework core, provide though necessary information for the design creation and optimization;

2. *framework core components* (white boxes) cooperate with each other and with input and output components forming a reusable and extendable framework design for the application of diverse algorithms to solve different design problems;

3. *output components* (gray boxes right *to Optimization Control*) do not belong to the optimization framework core, provide though useful interfaces to external libraries, the display of optimization results and solutions, an automated performance analysis and the export of solutions in a file directory structure.

The input component *Component Repository* provides components for solution candidates that can be accessed using a Java interface; and can be retrieved and stored by the component *Problem-Specific Setup.* Evaluations for interoperability among devices can also be performed by using this interface. The component *Problem* specifies the design problem in form of an abstract design given according to the problem component model. In the problem component model, both abstract design and solution representation of the room automation design are defined (cf. Section 5.6.2).

The core component *Problem-Specific Setup* is designed to decouple input components from the optimization by caching relevant data for the given design problem. *Problem-Specific Setup* is used to build solution candidates in the optimization. Furthermore, this component can perform some precalculations (cf. Section 5.4) for number of components and variety present in the component repository for the estimation of solution space dimension.

In *Metaheuristics-Specific Setup*, parameters and preferred algorithm type are read from the framework console GUI based on Java Swing components (cf. Section 6.5.3). These parameters are e.g. for an EA, population size, mutation probability, etc. Depending on the choice of algorithm, a specific workflow type is instantiated with corresponding specific algorithm components e.g. population, optimization operation types such as cross-over, mutation, selection type, etc.

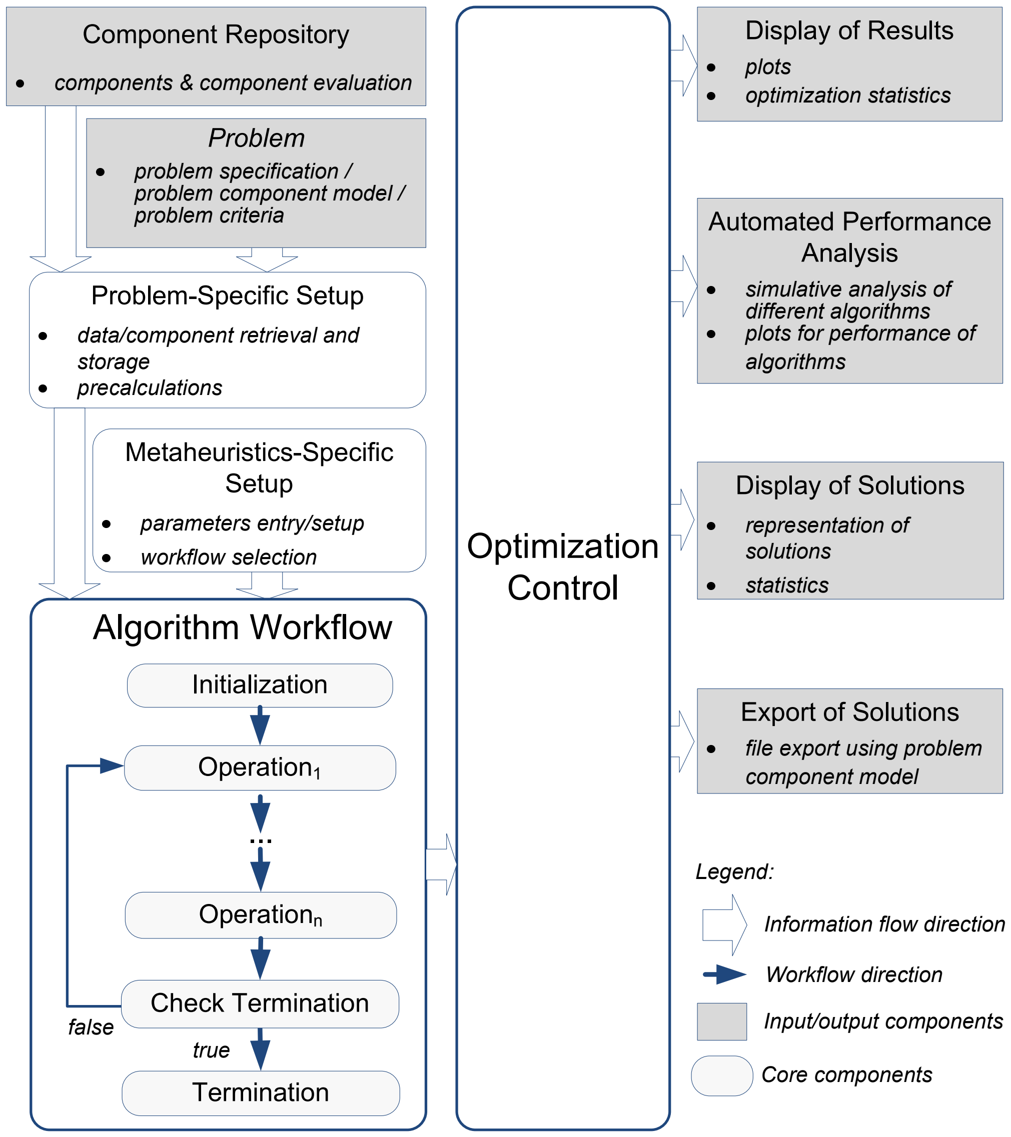


Figure 6.39: Optimization Framework Components and Control Flow

Optimization Control runs the chosen algorithm without knowing it starting with the initialization step and proceeding with the optimization iteratively until a termination criterion is satisfied. It is also possible to run the optimization control for multiple set of runs, having each set using a different algorithm, to obtain statistical results, plotting statistics for a comparison of performances of different algorithms as done in Section 6.3.

The output component *Display of Results* is designed to provide plots for convergence of criteria separately, as well as a lower triangle matrix of criterion pairs (cf. Figure 6.44). The latter is designed for the existence of multiple criteria for observing convergence towards Pareto front. To draw plots, the Java chart library JFreeChart [GM] is used. Plots that depict performances of algorithms in tests in Section 6.3 are prepared using this library.

By *Automated Performance Analysis* multiple algorithms can be set to run for a given number of times and parameter settings. This is particularly useful for performing empirical analysis of different algorithms and obtaining statistical results. The results e.g. for many runs, average of decision maker's fitness calculations of best solutions per generation can be plotted for each algorithm for a comparison of convergence properties associated to the considered algorithms (cf. Section 6.3).

Another output component *Display of Solutions* can present a user chosen optimization solution from the result set using a listener interface in the framework GUI based on texts or alternatively by using an interface in a GUI with graphical representation. *Export of Solutions* can be used to export a chosen solution as specified problem representation in a file.

6.5.2 Workflow Model

The framework is designed to be a white box framework for design and code reuse. The framework core components can flexibly be extended for new algorithms by means of the inheritance and a new problem can be defined with minimal implementation effort.

In Figure 6.40 the component model of the framework is presented in UML. Hereby, the output components and detailed hierarchy of algorithm workflows for single and multi-objective optimization are excluded for conciseness. The input components are illustrated in gray and all problem-specific components are marked by the prefix *Specific* in class names.

Program flow is directed by *OptimizationControl*. First, it performs an algorithm and a problem-specific setup which particularly contains the selection of the solver algorithm and problem-specific representation, and operations. In a next step, *OptimizationControl* initializes solution candidates. *OptimizationControl* controls then an algorithm workflow via the abstract class *AlgorithmWorkflow* without knowing the chosen concrete algorithm. An algorithm workflow contains methods to call constituent operations of a solver algorithm e.g. cross-over, mutation, selection operations of an EA and a method that prescribes the order of call for these operations. This can be simply done by overriding the iterate() method of *AlgorithmWorkflow*. By providing this flexibility, operations of many algorithms can be defined in own workflow classes including a preferred order of execution.

The workflow of a general algorithm such as an EA *EvolutionaryAlgorithmWorkflow* is modeled as a child of *AlgorithmWorkflow* and provides default definitions of standard EA operations e.g. *performCrossover()* performs binary tournament selection to determine crossover partners, *performMutation()* choses mutant candidates for a given mutation probability randomly, *evaluate()* assigns weighted sum based fitness values to the solution candidates, and *performSelection()* selects the solution candidates of the next generations that have the best fitness values. A special algorithm workflow can be specialized to implement different algorithm variants. The children *NSGA2Workflow* and *SPEA2Workflow* inherit most operations from the parent class *EvolutionaryAlgorithmWorkflow* and override only their specialized operations.

For problems with complex representations that require repair operations for recovering solution candidates after crossover and mutation, an abstract method *performUpdate()* is provided and has to be implemented depending on the structure of the specific problem representation. Recovering of solution candidates for the automated creation of optimized designs for BAS is presented in Section 5.6.2.

SpecificSetup determines a list of problem-specific cross-over, mutation operations that are related to the problem specific representation. For example, for the design creation problem problem-specific crossover operations or mutations (cf. Section 5.5.2) can be applied that select one or several functional profiles and replace them by other ones with the matching set of covered function blocks. Any further problem-specific crossover and mutation can be designed and added to the lists *crossoverPerformers* and *mutators*. *EvolutionaryAlgorithmWorkflow* or any of its child classes can select an operation from these lists in *performCrossover()* and *performMutation()* randomly or according to some strategy.

OptimizationControl can display and export solutions by using the implementation of interfaces *Exportable* and *Displayable* in an *Individual*. The displayer and exporter methods can then be called via *AlgorithmWorkflow*'s handle of *Population* that aggregates a set of individuals.

The framework is implemented in Java [Ora] programming language following the introduced design concepts and tested for design creation and optimization for examples of room automation. Many different algorithms can be integrated with minimal implementation efforts by means of decoupled optimization tasks. Particularly, the generic algorithm workflow concept, and the simple integration of problem-specific representation and operations point out the flexibility of framework design based on code and design reuse. A GUI is also conceived and implemented as presented in Section 6.5.3.

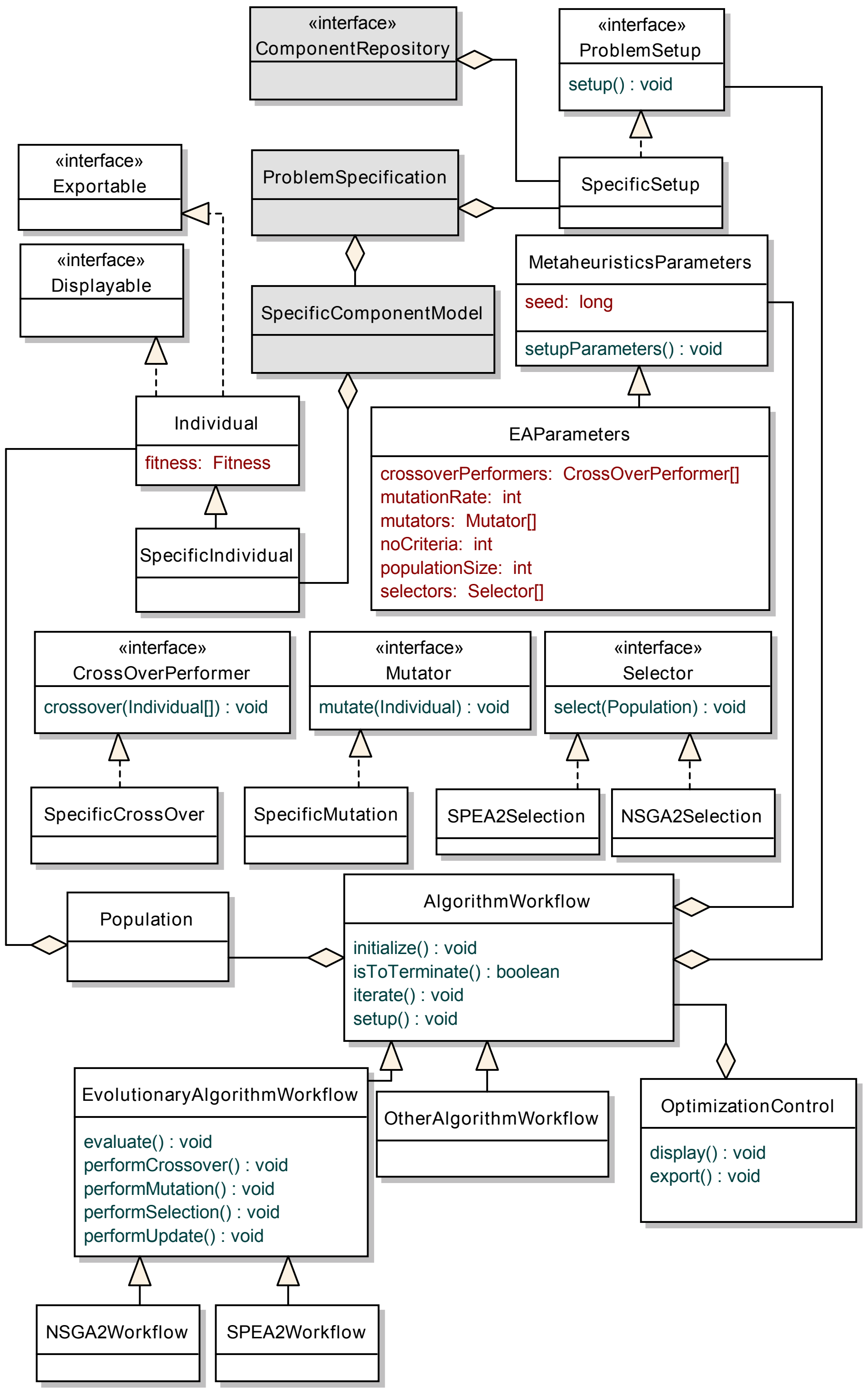
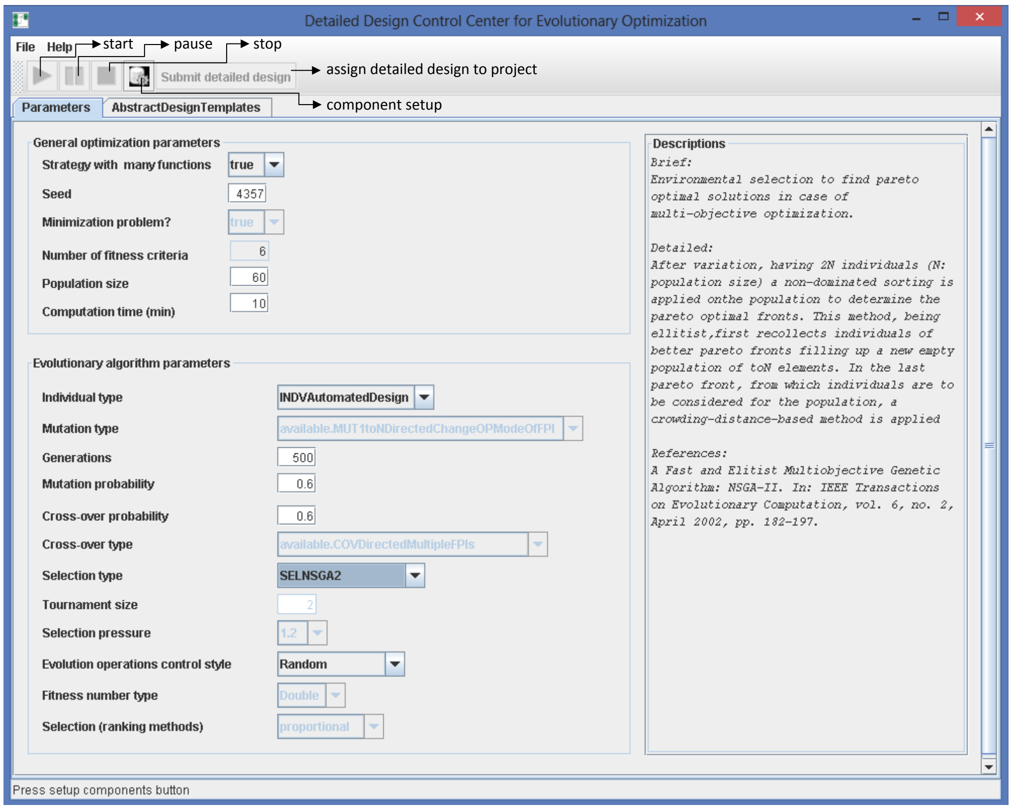


Figure 6.40: A Simplified UML Model of the Framework

6.5.3 Optimization Control By Graphical User Interface

The GUI that is used to compute optimized detailed design solutions allows the setting of required parameters of a chosen algorithm (cf. Figure 6.41). A component search process is performed flowed by all necessary calculations for a design generation (cf. Section 5.4.2) (component setup) activated by the button for setup. During this process algorithm parameters can be modified on GUI, if default parameters should be modified. After completion of component setup process, optimization process can be started by the start button. During the optimization a running algorithm can be paused, stopped, restarted or resumed. Such an approach for the control of algorithms is inspired by the GUI for optimization framework ECJ [ECJ].


Detailed Design Control Center for Evolutionary Optimization
File
Help
start
pause
stop
Submit detailed design
assign detailed design to project
Parameters
AbstractDesignTemplates
component setup

General optimization parameters

Strategy with many functions	true
Seed	4357
Minimization problem?	true
Number of fitness criteria	6
Population size	60
Computation time (min)	10

Evolutionary algorithm parameters

Individual type	INDVAutomatedDesign
Mutation type	available.MUT1toNDirectedChangeOPModeOfFPI
Generations	500
Mutation probability	0.6
Cross-over probability	0.6
Cross-over type	available.COVDirectedMultipleFPIs
Selection type	SELNSGA2
Tournament size	2
Selection pressure	1.2
Evolution operations control style	Random
Fitness number type	Double
Selection (ranking methods)	proportional

Descriptions

Brief:
Environmental selection to find pareto optimal solutions in case of multi-objective optimization.

Detailed:
After variation, having 2N individuals (N: population size) a non-dominated sorting is applied onthe population to determine the pareto optimal fronts. This method, being ellitist,first recollects individuals of better pareto fronts filling up a new empty population of toN elements. In the last pareto front, from which individuals are to be considered for the population, a crowding-distance-based method is applied

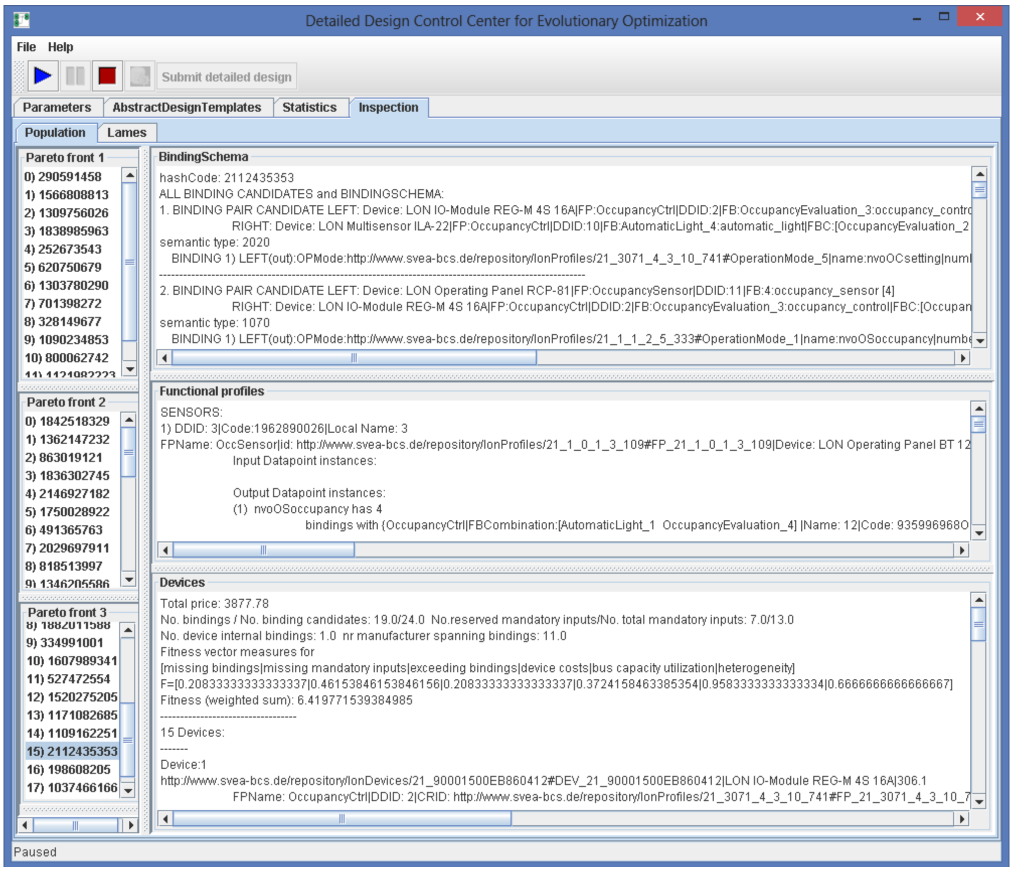
References:
A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. In: IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, April 2002, pp. 182-197.

Press setup components button

Figure 6.41: Main Control Window for Setting of Algorithm Parameters

If a visualization of the optimized solutions before termination is desired, the algorithm can be paused, solutions can be inspected (cf. Figure 6.42), and then resumed or restarted. This is particularly significant, if a high number of generations are specified and for many generations no improvement is observed, i.e. if the run should be interrupted before reaching the generation number limit.

After termination or a halt of the algorithm in an intermediate generation by stop button, an individual can be selected and submitted to the project as a detailed design of the given abstract design. In this connection, in the panel in Figure 6.42 individuals are sorted in Pareto fronts starting from the first front to the last. Solely, for the visualization and not for a consideration in algorithms, individuals in each front are sorted according to weighted sum of their objective values.



```
Pareto front 1
0) 290591458
1) 1566808813
2) 1309756026
3) 1838985963
4) 252673543
5) 620750679
6) 1303780290
7) 701398272
8) 328149677
9) 1090234853
10) 800062742

Pareto front 2
0) 1842518329
1) 1362147232
2) 863019121
3) 1836302745
4) 2146927182
5) 1750028922
6) 491365763
7) 2029697911
8) 818513997

Pareto front 3
9) 334991001
10) 1607989341
11) 527472554
12) 1520275205
13) 1171082685
14) 1109162251
15) 2112435353
16) 198608205
17) 1037466166
```

```
BindingSchema
hashCode: 2112435353
ALL BINDING CANDIDATES and BINDINGSCHEMA:
1. BINDING PAIR CANDIDATE LEFT: Device: LON IO-Module REG-M 4S 16A|FP:OccupancyCtrl|DDID:2|FB:OccupancyEvaluation_3:occupancy_contro
            RIGHT: Device: LON Multisensor ILA-22|FP:OccupancyCtrl|DDID:10|FB:AutomaticLight_4:automatic_light|FBC:[OccupancyEvaluation_2
semantic type: 2020
  BINDING 1) LEFT(out):OPMode:http://www.svea-bcs.de/repository/lonProfiles/21_3071_4_3_10_741#OperationMode_5|name:nvoOCsetting|num
---------------------------------------------------------------------------------------------
2. BINDING PAIR CANDIDATE LEFT: Device: LON Operating Panel RCP-81|FP:OccupancySensor|DDID:11|FB:4:occupancy_sensor [4]
            RIGHT: Device: LON IO-Module REG-M 4S 16A|FP:OccupancyCtrl|DDID:2|FB:OccupancyEvaluation_3:occupancy_control|FBC:[Occupan
semantic type: 1070
  BINDING 1) LEFT(out):OPMode:http://www.svea-bcs.de/repository/lonProfiles/21_1_1_2_5_333#OperationMode_1|name:nvoOSoccupancy|numbe

Functional profiles
SENSORS:
1) DDID: 3|Code:1962890026|Local Name: 3
FPName: OccSensor|id: http://www.svea-bcs.de/repository/lonProfiles/21_1_0_1_3_109#FP_21_1_0_1_3_109|Device: LON Operating Panel BT 12
            Input Datapoint instances:

            Output Datapoint instances:
            (1)  nvoOSoccupancy has 4
                        bindings with {OccupancyCtrl|FBCombination:[AutomaticLight_1  OccupancyEvaluation_4] |Name: 12|Code: 9359969680

Devices
Total price: 3877.78
No. bindings / No. binding candidates: 19.0/24.0  No.reserved mandatory inputs/No. total mandatory inputs: 7.0/13.0
No. device internal bindings: 1.0  nr manufacturer spanning bindings: 11.0
Fitness vector measures for
[missing bindings|missing mandatory inputs|exceeding bindings|device costs|bus capacity utilization|heterogeneity]
F=[0.20833333333333337|0.46153846153846156|0.20833333333333337|0.3724158463385354|0.9583333333333334|0.6666666666666667]
Fitness (weighted sum): 6.419771539384985
----------------------------------
15 Devices:
-------
Device:1
http://www.svea-bcs.de/repository/lonDevices/21_90001500EB860412#DEV_21_90001500EB860412|LON IO-Module REG-M 4S 16A|306.1
            FPName: OccupancyCtrl|DDID: 2|CRID: http://www.svea-bcs.de/repository/lonProfiles/21_3071_4_3_10_741#FP_21_3071_4_3_10_7
```

Figure 6.42: Panel for Selection of Individuals

For observation of improvement of the solutions online plots are provided that visualize the fitness values of solutions according to a weighted sum of objective functions by generations for a specified priority vector as depicted in Figure 6.43. A selected solution in the panel in Figure 6.42 is automatically pointed in the weighted sum plot in Figure 6.43 to allow a visual comparison of the selected individuals to the other in the plot. Further, in a matrix of plots as depicted in Figure 6.44 with each plot illustrating the improvement of each pair of objective functions, a multi-objective behavior of the algorithm can be observed.

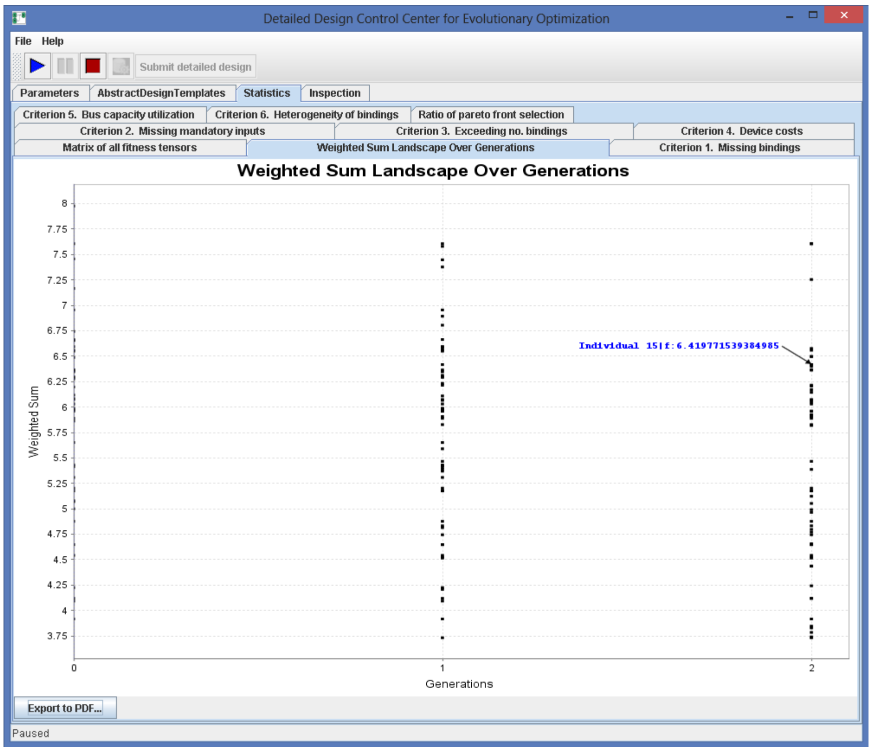

Detailed Design Control Center for Evolutionary Optimization
File
Help
Submit detailed design
Parameters
AbstractDesignTemplates
Statistics
Inspection
Criterion 5. Bus capacity utilization
Criterion 6. Heterogeneity of bindings
Ratio of pareto front selection
Criterion 2. Missing mandatory inputs
Criterion 3. Exceeding no. bindings
Criterion 4. Device costs
Matrix of all fitness tensors
Weighted Sum Landscape Over Generations
Criterion 1. Missing bindings
Weighted Sum Landscape Over Generations
Weighted Sum
8
7.75
7.5
7.25
7
6.75
6.5
6.25
6
5.75
5.5
5.25
5
4.75
4.5
4.25
4
3.75
0
1
2
Generations
Individual 15|f:6.419771539384985
Export to PDF...
Paused

Figure 6.43: View for Scalarized Objective Function Values

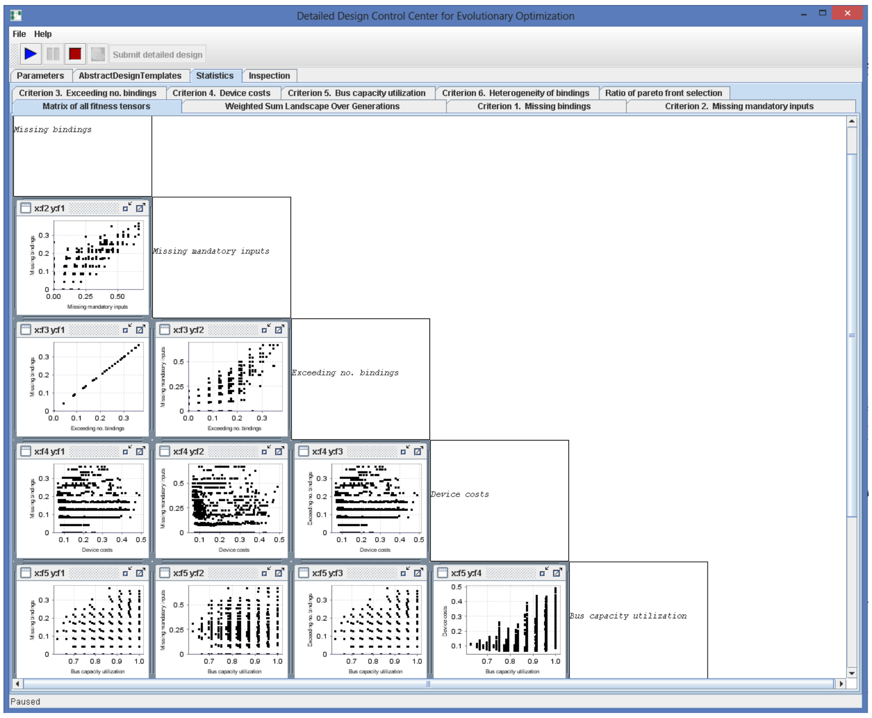

Detailed Design Control Center for Evolutionary Optimization
File Help
Submit detailed design
Parameters
AbstractDesignTemplates
Statistics
Inspection
Criterion 3. Exceeding no. bindings
Criterion 4. Device costs
Criterion 5. Bus capacity utilization
Criterion 6. Heterogeneity of bindings
Ratio of pareto front selection
Matrix of all fitness tensors
Weighted Sum Landscape Over Generations
Criterion 1. Missing bindings
Criterion 2. Missing mandatory inputs
Missing bindings
Missing mandatory inputs
Exceeding no. bindings
Device costs
Bus capacity utilization
xcf2 ycf1
xcf3 ycf1
xcf3 ycf2
xcf4 ycf1
xcf4 ycf2
xcf4 ycf3
xcf5 ycf1
xcf5 ycf2
xcf5 ycf3
xcf5 ycf4
Paused

Figure 6.44: Multi-Objective View of the Optimization Process

6.6 Conclusions

For the validation of concept in Chapter 5 implementations are done and an optimization framework for the engineering of BAS emerged in Section 6.4 and 6.5. This framework is used to perform an empirical analysis of the proposed algorithms in Section 5.4 and 5.5 using two different MOEAs, NSGA2 and SPEA2 with component repositories of a small and a large size. Many algorithm runs are performed for performance tests and for analysis with different population sizes, cross-over and mutation probabilities.

In order to prove the genericity of the approaches in Chapter 5 algorithms with different parameter settings are applied using three types of problem instances as identified in Section 6.3. As a result, algorithms all achieved near-optimal and in some cases optimal detailed design solutions for all three problem instance examples, which have been handled in the previous research projects [Unia] together with the partners from the BAS industry and exist in the BAS engineering practice. Particularly, Semi-Directed/Random/Directed (SD/R/D) algorithms that apply goal-oriented approaches achieved better results than Random (R) algorithms.

Further, it has been validated that well-optimized design solutions can be created in a limited time span using a slow and a faster computer configuration by applying the solution method for automated design creation in the presence of ten thousands of design relevant candidate components for practicable examples from the building automation domain. Results presented in Section 6.3.8 have shown that algorithms could be run from the start multiple times for parameter settings with which good optimization results obtained with a slow or a faster the computer configuration to obtain multiple sets of optimization solutions P1's, from which as many optimal results as possible can be obtained, still in a reasonable time.

Chapter 7

Conclusions

In this thesis a solution method for problem of automated creation of optimized building automation system designs is presented including tests for validation of algorithms proposed. First, insights to the design engineering in the domain of building automation are given in its business process model. Shortcomings of the current design creation approaches are presented that are limited to system components from a few manufacturers. Further, in the current situation the existing optimization potential for the design engineering and for the quality of the created design solutions are addressed and the respective design quality requirements as well as requirements on design engineering methods are derived.

A solution that requires device manufacturers to follow new standards to ensure interoperability of their own components with components of different manufacturers would presumably not gain an expected acceptance. An alternative solution is followed that conceives a search for the devices that can interoperate with each other for the realization of a given building automation plan. For the realization of the alternative solution there exist yet guidelines in the standard VDI 3813 (planned to be a part of ISO 16484) that allows a unified identification of the functionality and logical interfaces of components from device manufacturers. With the available components on the market, the functionalities and the logical interfaces of which can unambiguously be described according to this standard, a component search is enabled for the realization of a given set of functional requirements.

The design engineering is then identified in Chapter 2 as a problem of a high complexity, if all available components of device manufacturers are considered. In this context the high complexity is related with the existence of ten thousands of available candidate components for the design engineering as well as with the problem of device interoperability particularly among the devices of different device manufacturers. Based on these facts, quality requirements on a building automation system, quality requirements related to project planning and project implementation are presented. Further, a formal description of the design creation problem is given and the most characteristic problem properties are emphasized in order to distinguish the problem from similar design problems of the engineering practice. Based on the problem properties and quality requirements on project planning and project implementation, requirements for candidate algorithms are identified.

With the problem properties given, the type of the problem in theory of Computational Intelligence is identified in Chapter 3 as multi-objective combinatorial optimization. Under consideration of the problem type and requirements on algorithms, a state-of-the-art for candidate algorithms is presented in Chapter 4 , followed by four solution methods for various design problems of the engineering practice. These practical problems and the corresponding solution methods are then compared with each other for their similarity to the properties of the design creation problem. Similarly, proposed solution methods are compared to each other for the requirements on algorithms. It is concluded that none of the solution methods fulfill the requirements on algorithms completely. Thus, the conclusion states that a new solution method for the design creation problem is required.

A new solution method is proposed in Chapter 5 that allows an automated creation of optimized designs. For the solution method an automated interoperability check for communication partner devices is provided. Four major contributions of the solution method conceive the criteria for the *evaluation* of designs and design components including objective functions, the *component model* for abstract and detailed designs, alternative methods for *design generation* and *design improvement* including goal-oriented variation operations. Moreover, realization of further practical concepts related to the automated design creation for a building is emphasized. Further uses and applicability of the solution method is also presented.

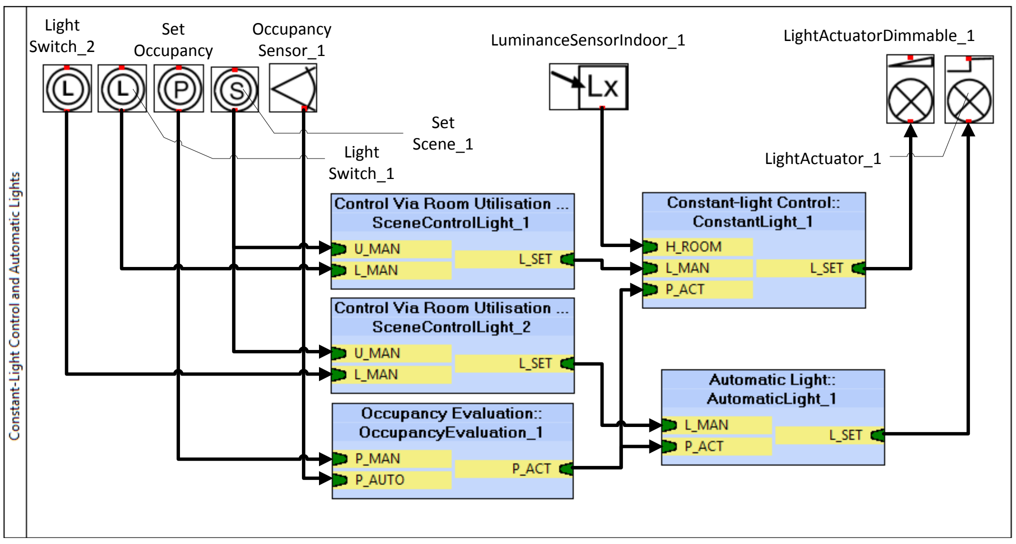
For the validation of the solution method, a validation method in Chapter 6 is conceived that involves a test platform including performance metrics and representative problem instance examples. An empirical analysis for the solution method is conducted using the proposed algorithms which are tested for their performances under consideration of performance metrics and the quality of the design solutions they produce. Considered problem instances and example design solutions are presented in Appendix of Designs.

A generic optimization framework is conceived in Chapter 6 that enables a seamless integration of the problem component model and algorithm alternatives for various metaheuristics and their operations by means of an algorithm workflow-based and modular design. Graphical user interfaces for setting optimization algorithms, for observing design improvement process by objective function plots and for consideration of any intermediate and final design solutions of the optimization process are also implemented and attached to the optimization framework via generic interfaces. Tests for validation are performed using this generic optimization framework.

Tests conducted for three problem instance examples have been performed using a component repository that consists of about 120 devices (original component repository) and a larger component repository that is 100 times larger than the original component repository (cloned repository). Tests with both sizes of repositories have shown that the proposed algorithms can achieve optimal design solutions or solutions with high quality which are very close to optimal solutions, in case of all three problem instance examples. If the practical relevance of the used examples, performance of the algorithms and the quality of design solutions are considered, it can be concluded that the proposed solution method is highly practicable and efficient. Further, it makes a very important contribution for a holistic design engineering in the business process model for building automation systems.

Appendix A

Appendix of Designs



Constant-Light Control and Automatic Lights
Light Switch_2
Set Occupancy
Occupancy Sensor_1
LuminanceSensorIndoor_1
LightActuatorDimmable_1
Set Scene_1
Light Switch_1
LightActuator_1
Control Via Room Utilisation ... SceneControlLight_1
U_MAN
L_MAN
L_SET
Control Via Room Utilisation ... SceneControlLight_2
U_MAN
L_MAN
L_SET
Occupancy Evaluation:: OccupancyEvaluation_1
P_MAN
P_AUTO
P_ACT
Constant-light Control:: ConstantLight_1
H_ROOM
L_MAN
P_ACT
L_SET
Automatic Light:: AutomaticLight_1
L_MAN
P_ACT
L_SET

Figure A.1: Example 1 - Abstract Design - Constant-Light Control and Automatic Lights

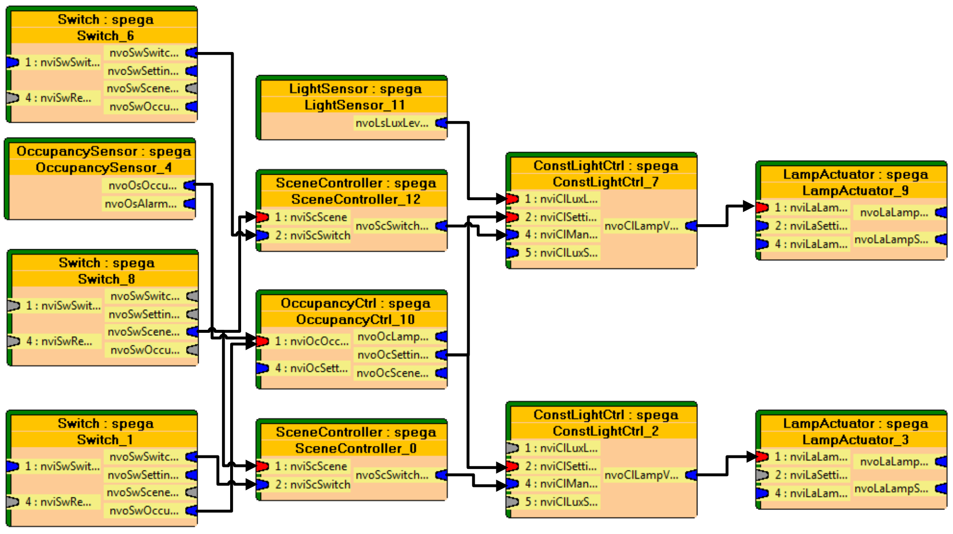


Figure A.2: Example 1 - A Detailed Design from the True Pareto Front - Binding-Schema

Device	Used Functional Profile	Function Block Combination
econtrol lumina RDA2-UN lamp actuator dimmable 2x_1	LampActuator_3	LightActuator_1
	ConstLightCtrl_2	AutomaticLight_1
	SceneController_0	SceneControlLight_2
	SceneController_12	SceneControlLight_1
	LampActuator_9	LightActuatorDimmable_1
	ConstLightCtrl_7	ConstantLight_1
econtrol lumina MS3 EB multisensor_2	OccupancySensor_4	OccupancySensor_1
	LightSensor_11	LuminanceSensorIndoor_1
econtrol lumina T6 binary switch with Constant Light Control_3	OccupancyCtrl_10	OccupancyEvaluation_1
	Switch_1	LightSwitch_2 Set_Occupancy
	Switch_6	LightSwitch_1
	Switch_8	SetScene_1
Sum of device price costs = 753.00 Euros		

Table A.1: Example 1 - Detailed Design - Devices, Functional Profiles and Function Block Mappings from Abstract Design in Figure A.1 to Detailed Design in Figure A.2

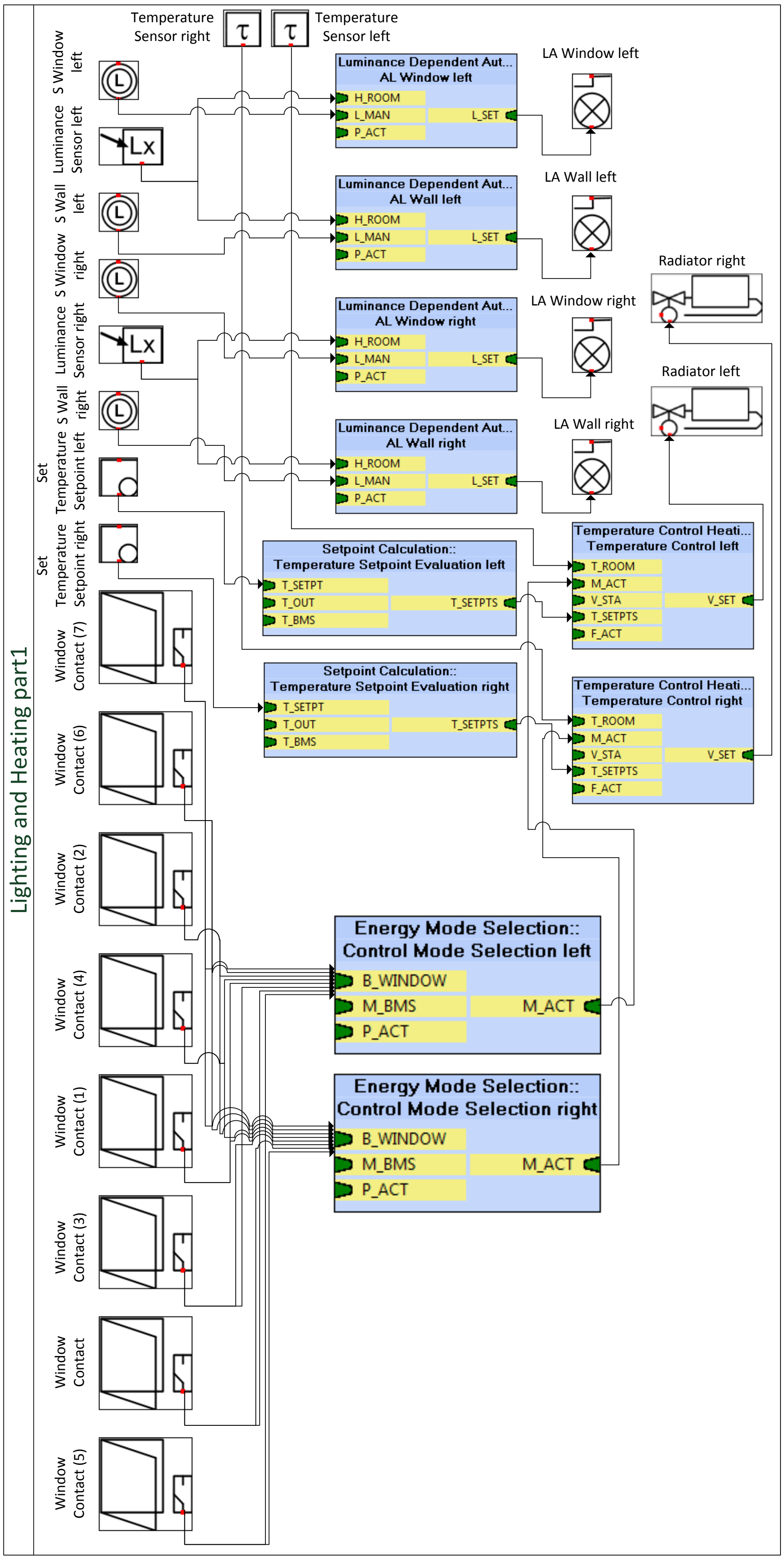

Lighting and Heating part1
Temperature Sensor right
Temperature Sensor left
S Window left
Luminance Sensor left
S Wall left
S Window right
Luminance Sensor right
S Wall right
Set Temperature Setpoint left
Set Temperature Setpoint right
Window Contact (7)
Window Contact (6)
Window Contact (2)
Window Contact (4)
Window Contact (1)
Window Contact (3)
Window Contact
Window Contact (5)
Luminance Dependent Aut... AL Window left
Luminance Dependent Aut... AL Wall left
Luminance Dependent Aut... AL Window right
Luminance Dependent Aut... AL Wall right
H_ROOM
L_MAN
P_ACT
L_SET
LA Window left
LA Wall left
LA Window right
LA Wall right
Radiator right
Radiator left
Setpoint Calculation:: Temperature Setpoint Evaluation left
Setpoint Calculation:: Temperature Setpoint Evaluation right
T_SETPT
T_OUT
T_BMS
T_SETPTS
Temperature Control Heati... Temperature Control left
Temperature Control Heati... Temperature Control right
T_ROOM
M_ACT
V_STA
T_SETPTS
F_ACT
V_SET
Energy Mode Selection:: Control Mode Selection left
Energy Mode Selection:: Control Mode Selection right
B_WINDOW
M_BMS
P_ACT
M_ACT

Figure A.3: Example 2 - Abstract Design - Lighting and Heating Part1

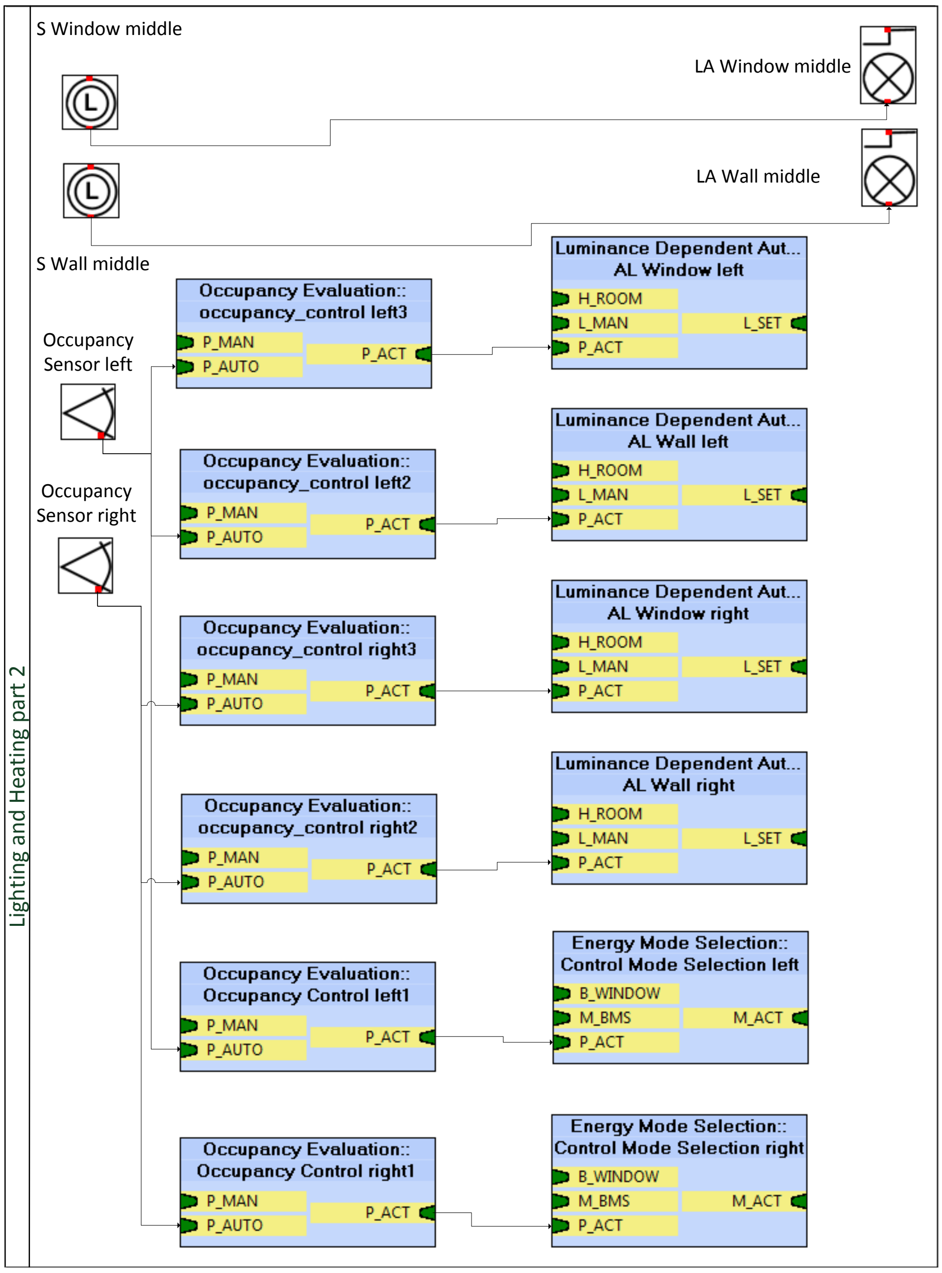

Lighting and Heating part 2
S Window middle
LA Window middle
LA Wall middle
S Wall middle
Occupancy Sensor left
Occupancy Sensor right
Occupancy Evaluation:: occupancy_control left3
P_MAN
P_AUTO
P_ACT
Occupancy Evaluation:: occupancy_control left2
P_MAN
P_AUTO
P_ACT
Occupancy Evaluation:: occupancy_control right3
P_MAN
P_AUTO
P_ACT
Occupancy Evaluation:: occupancy_control right2
P_MAN
P_AUTO
P_ACT
Occupancy Evaluation:: Occupancy Control left1
P_MAN
P_AUTO
P_ACT
Occupancy Evaluation:: Occupancy Control right1
P_MAN
P_AUTO
P_ACT
Luminance Dependent Aut... AL Window left
H_ROOM
L_MAN
L_SET
P_ACT
Luminance Dependent Aut... AL Wall left
H_ROOM
L_MAN
L_SET
P_ACT
Luminance Dependent Aut... AL Window right
H_ROOM
L_MAN
L_SET
P_ACT
Luminance Dependent Aut... AL Wall right
H_ROOM
L_MAN
L_SET
P_ACT
Energy Mode Selection:: Control Mode Selection left
B_WINDOW
M_BMS
M_ACT
P_ACT
Energy Mode Selection:: Control Mode Selection right
B_WINDOW
M_BMS
M_ACT
P_ACT

Figure A.4: Example 2 - Abstract Design - Lighting and Heating Part2

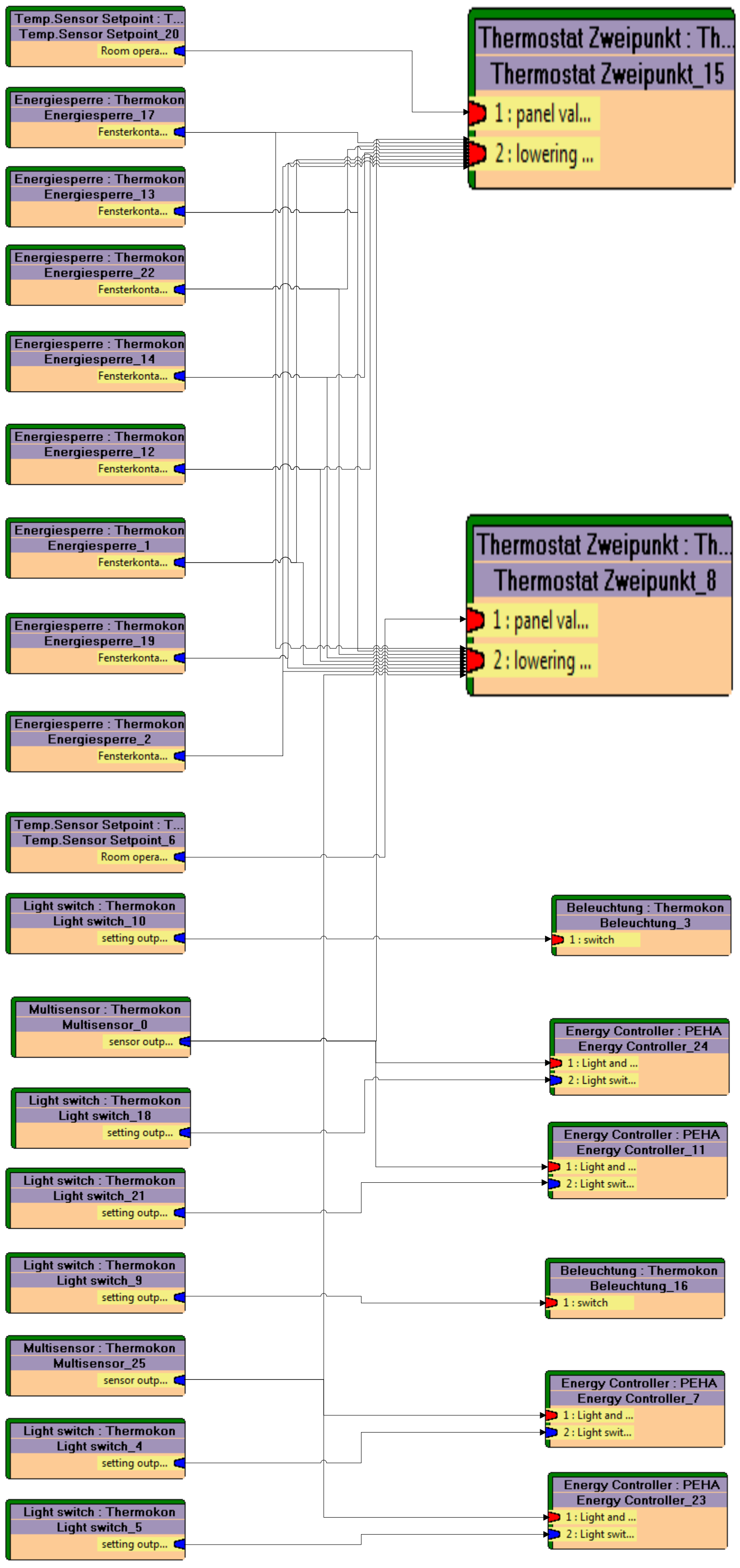


Figure A.5: Example 2 - A Detailed Design from the True Pareto Front - Binding-Schema

Device	**Used Functional Profile**	**Function Block Combination**
EasySens Wireless Switch Mini 2-Channel Light aluminium_1	Light switch_18	S Wall left
EasySens SR-MDS 230V Wireless-Ceiling-Multi-Sensor_2	Multisensor_25	Luminance Sensor right Temperature Sensor right Occupancy Sensor right
EasySens Wireless Switch Mini 2-Channel Light pure white glossy_3	Light switch_5	S Wall right
EasySens SRC-DO Lighting 230V_4	Beleuchtung_16	LA Window middle
EasySens SRW01 Window Contact_5	Energiesperre_2	Window Contact
EasySens SRW01 Window Contact_6	Energiesperre_17	Window Contact (4)
SENSOLUX Energy Controller 1-Channel_7	Energy Controller_23	occupancy_control right2 AL Wall right LA Wall right
EasySens SRW01 Window Contact_8	Energiesperre_13	Window Contact (3)
EasySens SRC-DO 230V Typ 4 Heating On/Off_9	Thermostat Zweipunkt_15	Occupancy Control left1 Control Mode Selection left Temperature Setpoint Evaluation left Temperature Control left Radiator left
EasySens SRC-DO Lighting 230V_10	Beleuchtung_3	LA Wall middle
EasySens SRW01 Window Contact_11	Energiesperre_14	Window Contact (6)
EasySens SR04P Wireless Room Temperature Sensor_12	Temp.Sensor Setpoint_20	Set Temperature Setpoint left
EasySens SR-MDS BAT Wireless-Ceiling-Multi-Sensor_13	Multisensor_0	Occupancy Sensor left Luminance Sensor left Temperature Sensor left
EasySens Wireless Switch Mini 2-Channel Light anthracite_14	Light switch_9	S Window middle
EasySens Wireless Switch Mini 2-Channel Light aluminium_15	Light switch_4	S Window right
EasySens Wireless Switch Mini 2-Channel Light pure white glossy_16	Light switch_21	S Window left
EasySens SRW01 Window Contact_17	Energiesperre_22	Window Contact (5)
EasySens SR04P Wireless Room Temperature Sensor_18	Temp.Sensor Setpoint_6	Set Temperature Setpoint right
EasySens SRW01 Window Contact_19	Energiesperre_1	Window Contact (1)
EasySens SRC-DO 230V Typ 4 Heating On/Off_20	Thermostat Zweipunkt_8	Occupancy Control right1 Control Mode Selection right Temperature Setpoint Evaluation right Temperature Control right Radiator right
EasySens SRW01 Window Contact_21	Energiesperre_19	Window Contact (2)
EasySens SRW01 Window Contact_22	Energiesperre_12	Window Contact (7)
SENSOLUX Energy Controller 1-Channel_23	Energy Controller_11	occupancy_control left3 AL Window left LA Window left
SENSOLUX Energy Controller 1-Channel_24	Energy Controller_7	occupancy_control right3 AL Window right LA Window right
EasySens Wireless Switch Mini 2-Channel Light aluminium_25	Light switch_10	S Wall middle
SENSOLUX Energy Controller 1-Channel_26	Energy Controller_24	occupancy_control left2 AL Wall left LA Wall left
Sum of device price costs = 2450.66 Euros		

Table A.2: Example 2 - Detailed Design - Devices, Functional Profiles and Function Block Mappings from Abstract Designs in Figure A.3 and Figure A.4 to Detailed Design in Figure A.5

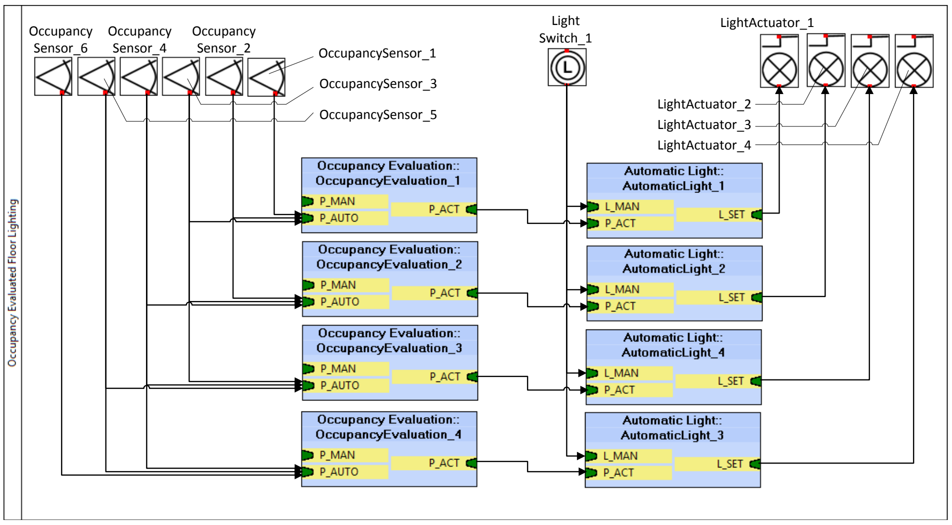

Occupancy Evaluated Floor Lighting
Occupancy Sensor_6
Occupancy Sensor_4
Occupancy Sensor_2
OccupancySensor_1
OccupancySensor_3
OccupancySensor_5
Light Switch_1
LightActuator_1
LightActuator_2
LightActuator_3
LightActuator_4
Occupancy Evaluation:: OccupancyEvaluation_1
Occupancy Evaluation:: OccupancyEvaluation_2
Occupancy Evaluation:: OccupancyEvaluation_3
Occupancy Evaluation:: OccupancyEvaluation_4
P_MAN
P_AUTO
P_ACT
Automatic Light:: AutomaticLight_1
Automatic Light:: AutomaticLight_2
Automatic Light:: AutomaticLight_4
Automatic Light:: AutomaticLight_3
L_MAN
P_ACT
L_SET

Figure A.6: Example 3 - Abstract Design - Occupancy Evaluated Floor Lighting

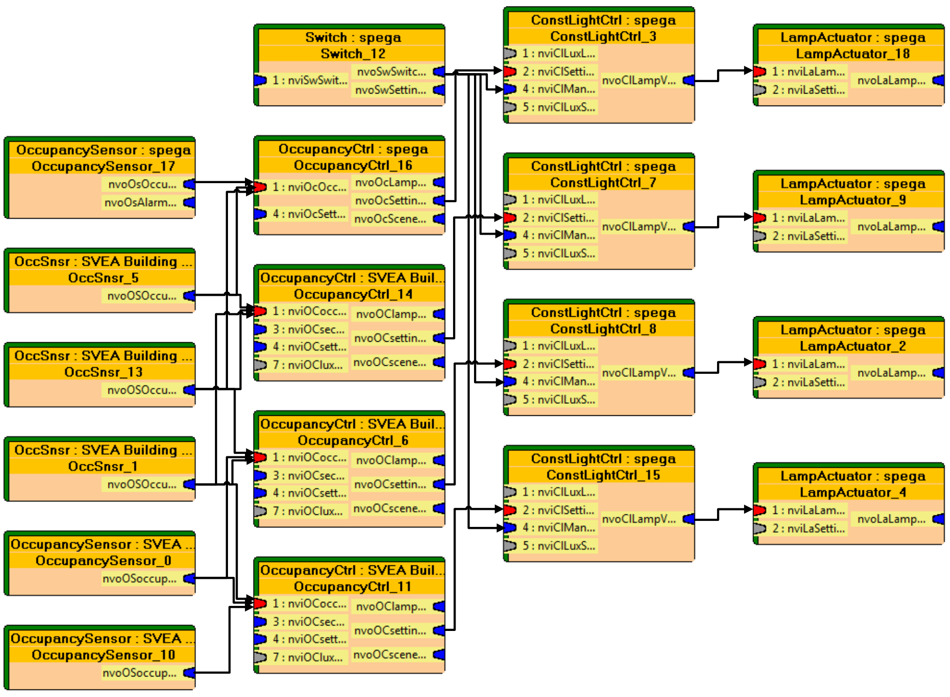


Figure A.7: Example 3 - A Detailed Design from the True Pareto Front - Binding-Schema

Device	Used Functional Profile	Function Block Combination
LON System-M pushbutton 2x_1	OccupancySensor_0	OccupancySensor_5
econtrol lumina RSA4 lamp actuator 4x_2	LampActuator_2	LightActuator_3
	LampActuator_4	LightActuator_4
	LampActuator_9	LightActuator_2
	LampActuator_18	LightActuator_1
econtrol sistema RC2 (Constant Light Control)_3	ConstLightCtrl_7	AutomaticLight_2
	ConstLightCtrl_8	AutomaticLight_4
	ConstLightCtrl_15	AutomaticLight_3
LON System-M pushbutton 2x_4	OccupancySensor_17	OccupancySensor_1
econtrol lumina MS3 EB multisensor_5	OccupancyCtrl_16	OccupancyEvaluation_1
	ConstLightCtrl_3	AutomaticLight_1
	Switch_12	LightSwitch_1
	OccupancySensor_10	OccupancySensor_6
LON System-M occupancy sensor_6	OccupancyCtrl_14	OccupancyEvaluation_2
	OccSnsr_5	OccupancySensor_2
LON System-M occupancy sensor_7	OccSnsr_1	OccupancySensor_4
	OccupancyCtrl_11	OccupancyEvaluation_4
LON System-M occupancy sensor_8	OccupancyCtrl_6	OccupancyEvaluation_3
	OccSnsr_13	OccupancySensor_3
Sum of device price costs = 846.30 Euros		

Table A.3: Example 3 - A Detailed Design - Devices, Functional Profiles and Function Block Mappings from Abstract Design in Figure A.6 to Detailed Design in Figure A.7

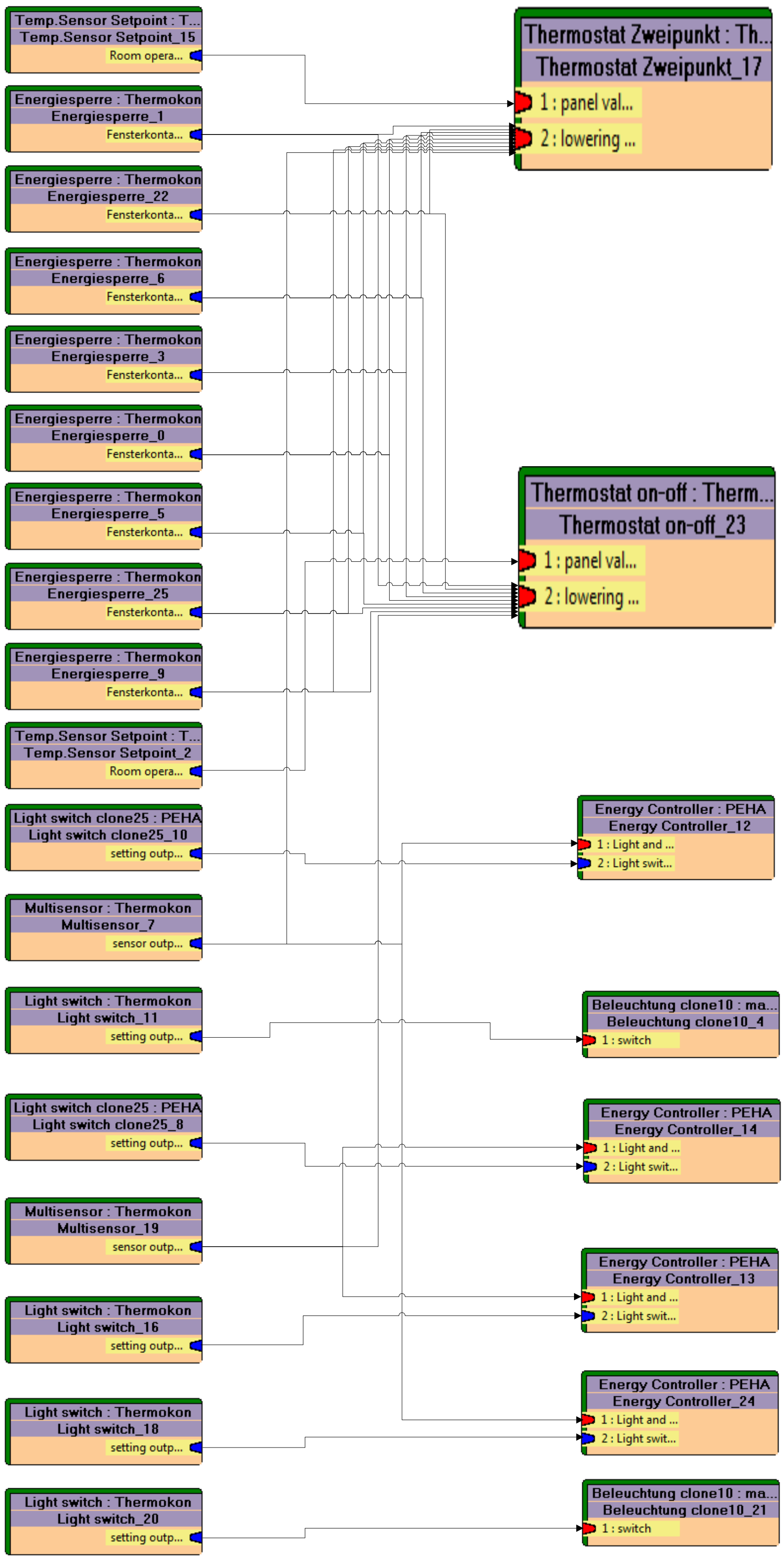


Figure A.8: Example 2 with Cloned Repository - A Detailed Design from the Last Generation of A Randomly Chosen Run with NSGA2-SD/R/D $P = 60$, $p_C = 0.6$, $p_M = 0.6$ - Binding-Schema

Device	Used Functional Profile	Function Block Combination
EasySens SRC-DO 24V Typ 4 Heating On/Off_1	Thermostat on-off_23	Occupancy Control right1 Temperature Setpoint Evaluation right Temperature Control right Control Mode Selection right Radiator right
EasySens SRW01 Window Contact_2	Energiesperre_25	Window Contact (4)
SENSOLUX Energy Controller 1-Channel_3	Energy Controller_14	occupancy_control right2 AL Wall right LA Wall right
EasySens SRW01 Window Contact_4	Energiesperre_9	Window Contact (1)
EasySens SRC-DO 230V Typ 4 Heating On/Off_5	Thermostat Zweipunkt_17	Occupancy Control left1 Control Mode Selection left Temperature Control left Temperature Setpoint Evaluation left Radiator left
SENSOLUX Energy Controller 1-Channel_6	Energy Controller_24	occupancy_control left2 AL Wall left LA Wall left
EasySens Wireless Switch Mini 2-Channel Light aluminium clone25_7	Light switch clone25_8	S Wall right
EasySens SRW01 Window Contact_8	Energiesperre_22	Window Contact (7)
SENSOLUX Energy Controller 1-Channel_9	Energy Controller_13	occupancy_control right3 AL Window right LA Window right
EasySens SRW01 Window Contact_10	Energiesperre_0	Window Contact (3)
SENSOLUX Energy Controller 1-Channel_11	Energy Controller_12	occupancy_control left3 AL Window left LA Window left
EasySens Wireless Switch Mini 2-Channel Light aluminium_12	Light switch_16	S Window right
EasySens SRW01 Window Contact_13	Energiesperre_5	Window Contact (5)
EasySens SRW01 Window Contact_14	Energiesperre_6	Window Contact (2)
EasySens Wireless Switch Mini 2-Channel Light anthracite_15	Light switch_11	S Window middle
EasySens Wireless Switch Mini 2-Channel Light aluminium_16	Light switch_18	S Wall left
EasySens SRC-DO Lighting 230V clone10_17	Beleuchtung clone10_4	LA Window middle
EasySens SR04P Wireless Room Temperature Sensor_18	Temp.Sensor Setpoint_15	Set Temperature Setpoint left
EasySens SR-MDS 24V Wireless-Ceiling-Multi-Sensor_19	Multisensor_7	Luminance Sensor left Occupancy Sensor left Temperature Sensor left
EasySens SRW01 Window Contact_20	Energiesperre_1	Window Contact
EasySens SRW01 Window Contact_21	Energiesperre_3	Window Contact (6)
EasySens SRC-DO Lighting 230V clone10_22	Beleuchtung clone10_21	LA Wall middle
EasySens SR-MDS 24V Wireless-Ceiling-Multi-Sensor_23	Multisensor_19	Occupancy Sensor right Luminance Sensor right Temperature Sensor right
EasySens Wireless Switch Mini 2-Channel Light aluminium clone25_24	Light switch clone25_10	S Window left
EasySens SR04P Wireless Room Temperature Sensor_25	Temp.Sensor Setpoint_2	Set Temperature Setpoint right
EasySens Wireless Switch Mini 2-Channel Light aluminium_26	Light switch_20	S Wall middle
Sum of device price costs = 2495.80 Euros		

Table A.4: Example 2 with Cloned Repository - Detailed Design - Devices, Functional Profiles and Function Block Mappings from Abstract Design in Figure A.3 and Figure A.4 to Detailed Design in Figure A.8

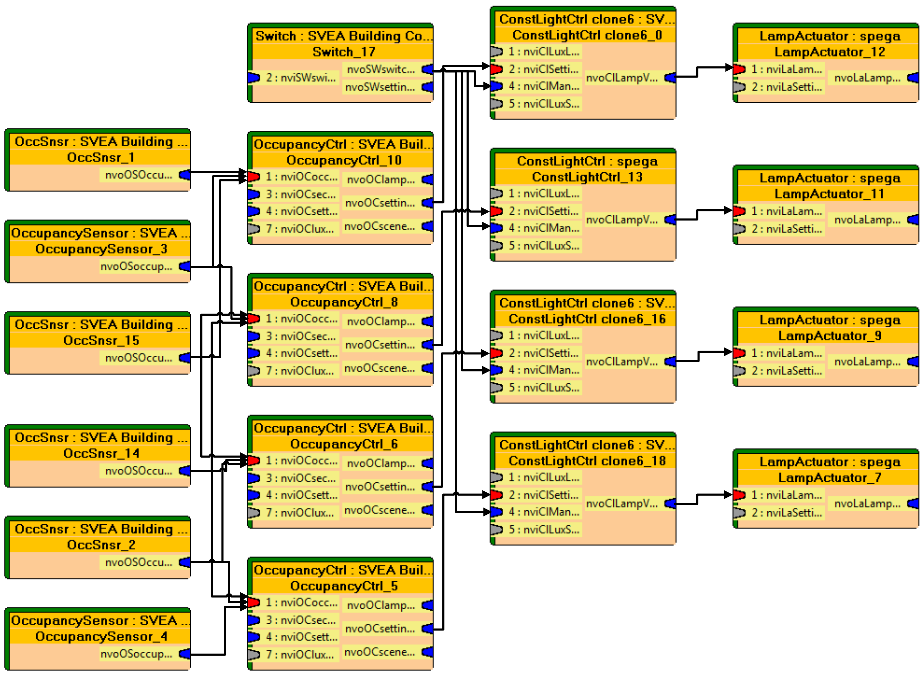


Figure A.9: Example 3 with Cloned Repository - A Detailed Design from the Last Generation of A Randomly Chosen Run with NSGA2-SD/R/D $P = 80$, $p_C = 0.6$, $p_M = 0.6$ - Binding-Schema

Device	**Used Functional Profile**	**Function Block Combination**
LON System-M pushbutton 2x_1	OccupancySensor_4	OccupancySensor_1
	Switch_17	LightSwitch_1
LON System-M occupancy sensor_2	OccupancyCtrl_5	OccupancyEvaluation_1
	OccSnsr_2	OccupancySensor_2
econtrol lumina RSA4 lamp actuator 4x_3	LampActuator_11	LightActuator_3
	LampActuator_12	LightActuator_4
	LampActuator_7	LightActuator_1
	LampActuator_9	LightActuator_2
LON System-M occupancy sensor_4	OccupancyCtrl_8	OccupancyEvaluation_3
	OccSnsr_15	OccupancySensor_4
LON System-M occupancy sensor_5	OccupancyCtrl_10	OccupancyEvaluation_4
	OccSnsr_1	OccupancySensor_6
econtrol lumina T6 binary switch with Constant Light Control_6	ConstLightCtrl_13	AutomaticLight_4
LON System-M pushbutton 2x_7	OccupancySensor_3	OccupancySensor_5
LON System-M occupancy sensor_8	OccupancyCtrl_6	OccupancyEvaluation_2
	OccSnsr_14	OccupancySensor_3
econtrol sistema RC2 (Constant Light Control) clone6_9	ConstLightCtrl clone6_18	AutomaticLight_1
	ConstLightCtrl clone6_0	AutomaticLight_3
	ConstLightCtrl clone6_16	AutomaticLight_2
Sum of device price costs = 930.80 Euros		

Table A.5: Example 3 with Cloned Repository - A Detailed Design - Devices, Functional Profiles and Function Block Mappings from Abstract Design in Figure A.6 to Detailed Design in Figure A.9

Bibliography

[AD08] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[AL97] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons Ltd., 1997.

[BBB+04] S. Baumgarth, E. Bollin, M. Büchel, B. Fromm, A. Karbach, D. Otto, H. Pärschke, P. Ritzenhoff, G.-P. Schernus, F. Sokolik, F. Tiersch, and W. Treusch. *Digitale Gebäudeautomation*. Springer, 2004.

[BDR00] A. Bauer, A. Döderlein, and P. Rössler. Fieldbus Systems for Home and Building Automation. *it + ti - Informationstechnik und Technische Informatik*, 42(4):17–23, 2000.

[Big94] T. J. Biggerstaff. The library scaling problem and the limits of concrete component reuse. In *Third International Conference on Software Reuse: Advances in Software Reusability*, pp. 102–109. 1994.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, 1999.

[Bui] BuildingSMART International Ltd. International home of openBIM. `http://www.buildingsmart-tech.org`. Accessed: 19/02/2013.

[CEG+00] K. Czarnecki, U. W. Eisenecker, R. Glück, D. Vandevoorde, and T. L. Veldhuizen. Generative Programming and Active Libraries. In *Selected Papers from the International Seminar on Generic Programming*, pp. 25–39. Springer-Verlag, London, UK, 2000.

[Cer85] V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[CST$^+$06] E. Carrano, L. Soares, R. Takahashi, R. Saldanha, and O. Neto. Electric distribution network multiobjective design using a problem-specific genetic algorithm. *IEEE Transactions on Power Delivery*, 21(2):995 – 1005, 2006.

[Deb04] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 111 River Street, Hoboken, NJ 07030, USA, 2004.

[DJ06] K. A. De Jong. *Evolutionary Computation*. MIT Press, Cambridge, MA, 2006.

[DK11] H. Dibowski and K. Kabitzsch. Ontology Based Device Descriptions and Device Repository for Building Automation Devices. In *EURASIP Journal on Embedded Systems*, volume 2011, pp. 3:1–3:17. Hindawi Publishing Corp., 2011.

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms*. Ph.D. thesis, Politecnico di Milano, Italy, 1992.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Trans. Evol. Comp*, 6(2):182–197, 2002.

[DS99] D. Dietmar and H.-J. Schweinzer. *LON-Technologie Verteilte Systeme in der Anwendung*. Hüthig, 1999.

[DZ83] J. D. Day and H. Zimmermann. The OSI reference model. *Proceedings of the IEEE*, 71(12):1334–1340, 1983.

[Ech] Echelon Corporation. LonMaker Integration Tool. `http://www.echelon.com/products/tools/integration/lonmaker`. Accessed: 19/02/2013.

[ECJ] ECJ. A Java-based Evolutionary Computation Research System. `http://www.cs.gmu.edu/~eclab/projects/ecj`. Accessed: 19/02/2013.

[EK10] M. Eichelberg and K. Kabitzsch et al. *Interoperabilität von AAL-Systemkomponenten. Teil 1: Stand der Technik*. VDE VERLAG, Berlin, 2010.

[EnO] EnOcean Alliance. `http://www.enocean-alliance.org`. Accessed: 19/02/2013.

[Erb06] C. Erbas. *System-Level Modeling and Design Space Exploration for Multiprocessor Embedded System-on-Chip Architectures*. Ph.D. thesis, University of Amsterdam, Netherlands, 2006.

[ES07] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series).* Springer, 2007.

[Fis02] P. Fischer. *Analyse und Bewertung von Kommunikationssystemen in der Gebäudeautomation.* Ph.D. thesis, Vienna, University of Technology, Vienna, Austria, 2002.

[FOW66] L. Fogel, A. Owens, and M. Walsh. *Artificial intelligence through simulated evolution.* Wiley, Chichester, WS, UK, 1966.

[FR89] T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67 – 71, 1989.

[GM] D. Gilbert and T. Morgner. JFreeChart: A Java chart library. Available: `http://www.jfree.org/jfreechart/`.

[GM86] F. Glover and C. McMillan. The general employee scheduling problem: an integration of MS and AI. *Comput. Oper. Res.*, 13(5), 1986.

[HHBW06] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. Second edition, 1992.

[HSU] HSU. Helmut Schmidt University (University of the Federal Armed Forces Hamburg, Germany). `http://www.hsu-hh.de`. Accessed: 19/02/2013.

[Int04] International Organization for Standardization. ISO16484:2 Building automation and control systems (BACS) – Part2: Hardware, 2004.

[Int05] International Organization for Standardization. ISO 9000 Quality management systems – Fundamentals and vocabulary, 2005.

[Int06a] International Organization for Standardization. ISO14543:3-1 Information technology – Home Electronic Systems (HES) Architecture – Part 3-1: Communication layers – Application layer for network based control of HES Class 1, 2006.

[Int06b] International Organization for Standardization. ISO14543:3-2 Information technology – Home Electronic Systems (HES) Architecture – Part 3-2: Communication layers – Transport, network and general parts of data link layer for network based control of HES Class 1, 2006.

[Int07a] International Organization for Standardization. ISO14543:3-3 Information technology – Home Electronic Systems (HES) Architecture – Part 3-3: User process for network based control of HES Class 1, 2007.

[Int07b] International Organization for Standardization. ISO14543:3-4 Information technology – Home Electronic Systems (HES) Architecture – Part 3-4: System management – Management procedures for network based control of HES Class 1, 2007.

[Int07c] International Organization for Standardization. ISO14543:3-5 Information technology – Home Electronic Systems (HES) Architecture – Part 3-5: Media and media dependent layers – Power line for network based control of HES Class 1, 2007.

[Int07d] International Organization for Standardization. ISO14543:3-6 Information technology – Home Electronic Systems (HES) Architecture – Part 3-6: Media and media dependent layers – Network based on HES Class 1, twisted pair, 2007.

[Int07e] International Organization for Standardization. ISO14543:3-7 Information technology – Home Electronic Systems (HES) Architecture – Part 3-7: Media and media dependent layers – Radio frequency for network based control of HES Class 1, 2007.

[Int10] International Organization for Standardization. ISO16484:1 Building automation and control systems (BACS) – Part1: Project specification and implementation, 2010.

[Int12a] International Organization for Standardization. ISO14908:1 Information technology – Control network protocol – Part1: Protocol stack, 2012.

[Int12b] International Organization for Standardization. ISO16484:5 Building automation and control systems (BACS) – Part5: Data communication protocol, 2012.

[Int13] International Electrotechnical Commission. IEC 61131-3 ed3.0 Programmable controllers - Part 3: Programming Languages, 2013.

[Kab87] K. Kabitzsch. *Mikrorechner in der Automatisierungspraxis. Ausgewählte Probleme der Software- und Hardwaregestaltung*. Akademie-Verlag, Berlin, 1987.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, eds., *Complexity of Computeter Computations*. Plenum Press, 1972.

[KDP02] K. Kabitzsch, D. Dietmar, and G. Pratl. *LonWorks-Gewerkeübergreifende Systeme*. VDE VERLAG, Berlin, Offenbach, 2002.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pp. 1942–1948 vol.4. IEEE, 1995.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[KHT00] W. Kriesel, T. Heimbold, and D. Telschow. *Bustechnologien für die Automation, Vernetzung, Auswahl und Anwendung von Kommunikationssystemen*. Hüthig, Heidelberg, 2000.

[KNSN05] W. Kastner, G. Neugeschwandtner, S. Soucek, and H. M. Newman. Communication Systems for Building Automation and Control. *Proceedings of the IEEE*, 93(6):1178–1203, 2005.

[Knu00] D. E. Knuth. Dancing Links. In J. Davies, B. Roscoe, and J. Woodcock, eds., *Millennial Perspectives in Computer Science*, pp. 187–214. Palgrave Macmillan, England, 2000.

[KNVT06] K. Kabitzsch, J. Naake, V. Vasyutynskyy, and S. Theiss. Untersuchung zum Fernzugriff auf Automatisierungstechnik. *atp - Automatisierungstechnische Praxis*, 7(48), 2006.

[KNX] KNX Association. ETS4 Engineering Tool Software. `http://www.knx.org/knx-tools/ets4`. Accessed: 19/02/2013.

[Koz92] J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[Kra06] H. Kranz. *BACnet Gebäudeautomation 1.4*. Promotor Verlags- und Förderungsges. mbH, 2. edition, 2006.

[Lon] LonMark International. Guidelines and Specifications. `http://www.lonmark.org/`. Accessed: 19/02/2013.

[MHH07] H. Merz, T. Hansemann, and C. Hübner. *Gebäudeautomation Kommunikationssysteme mit EIB/KNX, LON und BACnet*. Hanser, 2007.

[MKBR10] A. Martens, H. Koziolek, S. Becker, and R. Reussner. Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pp. 105–116. ACM, New York, NY, USA, 2010.

[MOF91] O. Martin, S. W. Otto, and E. W. Felten. Large-Step Markov Chains for the Traveling Salesman Problem. *Complex Systems*, 5:299–326, 1991.

[NDL+09] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal on Intelligent Systems*, 24(7):726–746, 2009.

[New] Newron System. NL220-TE LonWworks Manager Tool. `http://www.newron-system.com/NL220`. Accessed: 19/02/2013.

[ODK09] A. C. Oezluek, H. Dibowski, and K. Kabitzsch. Automated Design of Room Automation Systems by using an Evolutionary Optimization Method. In *14th International Conference on Emerging Technologies and Factory Automation ETFA 2009*, pp. 1–8. 2009.

[OPK10a] A. C. Oezluek, J. Ploennigs, and K. Kabitzsch. Designing Building Automation Systems Using Evolutionary Algorithms with Semi-Directed Variations. In *2010 IEEE International Conference on Systems, Man, and Cybernetics SMC 2010*. 2010.

[OPK10b] A. C. Oezluek, J. Ploennigs, and K. Kabitzsch. A Generic Framework for Synthesis and Optimization of System Designs in the Example of Building Automation Systems. In *2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4. 2010.

[Ora] Oracle. Java. `http://www.oracle.com/technetwork/java`. Accessed: 19/02/2013.

[ORK09] A. C. Oezluek, U. Ryssel, and K. Kabitzsch. Multi-Objective Combinatorial Optimization for Designing Room Automation Systems by Using Evolutionary Algorithms. In *35th Annual Conference of IEEE on Industrial Electronics. IECON '09.*, pp. 3335 –3340. 2009.

[PDRK11] J. Ploennigs, H. Dibowski, U. Ryssel, and K. Kabitzsch. Holistic Design of Wireless Building Automation Systems. In *IEEE Conference on Emerging Technologies Factory Automation*, pp. 1–9. 2011.

[Plo07] J. Ploennigs. *Control Network Performance Engineering - Qualitätsorienterter Entwurf von CSMA-Netzwerken der Automation.* Jörg Vogt Verlag, 1. aufl. edition, 2007.

[PV05] J. Ploennigs and V. Vasyutynskyy. AMES-Umfrage in der Zielgruppe "Gebäudeautomation", 2005.

[RDFK08] S. Runde, H. Dibowski, A. Fay, and K. Kabitzsch. Integrated Automated Design Approach for Building Automation Systems. In *IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 1488–1495. 2008.

[RDK09] U. Ryssel, H. Dibowski, and K. Kabitzsch. Generation of Function Block Based Designs using Semantic Web Technologies. In *IEEE Conference on Emerging Technologies and Factory Automation*, pp. 1–8. 2009.

[Rec73] I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution.* Fromman-Hozlboog Verlag, Stuttgart, 1973.

[RF11] S. Runde and A. Fay. Software Support for Building Automation Requirements Engineering - An Application of Semantic Web Technologies in Automation. *IEEE Transactions on Industrial Informatics*, 7(4):723–730, 2011.

[RFHS10] S. Runde, A. Fay, A. Heidemann, and P. Schmidt. Engineering der Automation im Kontext der Bauplanung: Praxis, Defizite und Ansätzte zur Verbesserung. *atp - Automatisierungstechnische Praxis*, 10:36–47, 2010.

[RFK00] T. Rauscher, P. Fischer, and K. Kabitzsch. An Inter-Industry Interoperability Concept for Fielbus Objects. *it + ti - Informationstechnik und Technische Informatik*, 42(4):38–44, 2000.

[Run10] S. Runde. *Wissensbasierte Engineeringunterstützung automatisierter technischer Systeme am Beispiel der Gebäudeautomatisierung.* Ph.D. thesis, University of Federal Armed Forces Hamburg, Germany, 2010.

[Sch98] B. Schuermann. Structure and Design of Building Automation Systems. In *Proceedings of the International Conference on New*

Information Technologies in Science, Education, Telecommunication and Business, Crimea, Ukraine. 1998.

[Spe] Spelsberg Gebäudeautomation GmbH. ALEX Designer Suite. `http://www.spega.com/produkte/highlights/alex-designer-suite`. Accessed: 19/02/2013.

[Spe08] J. Spelsberg. Weissbuch Gebäudeautomation, 2008.

[The99] The Association of German Engineers (VDI). VDI 3687 Selection of fieldbus systems by evaluating their performance characteristics for industrial application, 1999.

[The11a] The Association of German Engineers (VDI). VDI 3813:1 Building automation and control systems (BACS) - Fundamentals for room control, 2011.

[The11b] The Association of German Engineers (VDI). VDI 3813:2 Building automation and control systems (BACS) - Room control functions (RA functions), 2011.

[TU] TU Dresden. University of Technology Dresden, Germany. `http://www.tu-dresden.de/en`. Accessed: 19/02/2013.

[Unia] University of Technology Dresden, Germany. Automated Design Of Building Automation Systems. `http://www.ga-entwurf.de`. Accessed: 19/02/2013.

[Unib] University of Tübingen, Germany. A Java based framework for Evolutionary Algorithms. `http://www.ra.cs.uni-tuebingen.de/software/JavaEvA`. Accessed: 19/02/2013.

[VL00] D. A. V. Veldhuizen and G. Lamont. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pp. 204–211. 2000.

[VRZ$^+$07] S. Ventura, C. Romero, A. Zafra, J. Delgado, and C. Hervas. JCLEC: a java framework for evolutionary computation. *Soft Computation*, 12(4):381–382, 2007.

[WSWW03] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer. Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment. *IEEE Transactions on Evolutionary Computation*, 7(2):189–203, 2003.

[ZDT00] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8:173–195, 2000.

[ZJZ$^+$06] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 892–899. 2006.

[ZLT01] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In K. C. Giannakoglou and D. T. Tsahalis and J. Périaux and K. D. Papailiou and T. Fogarty, ed., *Evolutionary Methods for Design Optimization and Control with Application to Industrial Problems*. Athens, Greece, 2001.

Index