A Generic Configuration Tool and its Application to Vehicle Configuration

Fabian Kazmaier
Ein generisches Konfigurations-Werkzeug und seine Anwendung auf Konfigurierung von Kraftfahrzeugen

A Generic Configuration Tool and its Application to Vehicle Configuration

Student: Fabian Kazmaier
Supervisor: Apl. Prof. Peter Struss
Advisor: Apl. Prof. Peter Struss
Submission Date: 17. September 2012
I assure the single handed composition of this master’s thesis only supported by declared resources.

Ich versichere, dass ich diese Master’s Thesis selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwandt habe.

______________  _________________
Fabian Kazmaier  Place, Date
Acknowledgements

I express my sincere gratitude to my supervisor and advisor Prof. Dr. Peter Struss for his entire support. I also thank Dr. Oskar Dressler for his advice. I warmly thank Michael Blazek and eGym for the support. I am thankful to Florial Eibl and Alexandros Verikios for their help with the prototypes.
Abstract

This Master’s thesis deals with generic software for configuration. Component-based systems are spread throughout industry and configuration became more and more popular in the last decades, since mass production is changing towards customization. This led to an increased complexity for customers in terms of decision making, resulting in extensive and time consuming effort for the customer. Not a lot of effort has been made towards supporting the user in his decisions. This work deals with the question how the user can be supported in his process of decision making and how tools can support this process. In the course of this thesis a generic framework for configuration was developed, with an emphasis on general user goals. In the first part of the thesis challenges are described and requirements for a configuration tool are determined. Current web-based configurators do not meet these requirements. An approach has been developed to apply constraint satisfaction on configuration and a software specification has been created. A generic framework has been implemented and two domain specific configurators for vehicle configuration and training planning have been created and evaluated. The results show that supporting the user in his decision making process is feasible and that the chosen approach offers numerous ways for supporting users in various domains.
# Contents

Acknowledgements ........................................................................................................... i

Abstract ............................................................................................................................... ii

List of Figures ...................................................................................................... vi

List of Tables....................................................................................................................... vii

1 Introduction ...................................................................................................................... 8
   1.1 Motivation .................................................................................................................. 8
   1.2 Solution Idea .............................................................................................................. 9
   1.3 Results ....................................................................................................................... 9
   1.4 Outline ...................................................................................................................... 10

2 Problem and Requirement Analysis .............................................................................. 11
   2.1 User Information ....................................................................................................... 11
   2.2 Component-based Systems ..................................................................................... 11
   2.3 Decision Making ..................................................................................................... 12
   2.4 Flexible Control ...................................................................................................... 12
   2.5 Knowledge Management ....................................................................................... 12
   2.6 Generic Configuration Framework ......................................................................... 13
   2.7 Summary Requirements ......................................................................................... 13

3 State of the Art ............................................................................................................... 15
   3.1 Vehicle Configurators ............................................................................................. 15
      3.1.1 Market Overview .............................................................................................. 16
      3.1.2 Conclusion ....................................................................................................... 19
   3.2 Configuration Support: Research and Systems ...................................................... 19
   3.3 Constraints .............................................................................................................. 21
   3.4 Constraint Satisfaction ........................................................................................... 21

4 Conceptualization of the Configuration Task ............................................................... 22
   4.1 Generic Concepts .................................................................................................... 23
   4.2 Vehicle Concepts .................................................................................................... 26
List of Figures

7.1 Domain Characteristics ........................................................................................................60
  7.1.1 Domain Knowledge ............................................................................................................60
  7.1.2 Global Software Control ...................................................................................................63

7.2 Evaluation ................................................................................................................................65
  7.2.1 Validity ..............................................................................................................................65
  7.2.2 Achievements ....................................................................................................................67

8 Training Planning Configurator .................................................................................................70
  8.1 Domain Characteristics ..........................................................................................................70
     8.1.1 Domain Knowledge .........................................................................................................70
     8.1.2 Global Software Control ................................................................................................72
  8.2 Evaluation ...............................................................................................................................73
     8.2.1 Validity ............................................................................................................................73
     8.2.2 Achievements ..................................................................................................................75

9 Conclusion and Future Work .......................................................................................................76
  9.1 Extension of the prototypes ....................................................................................................77
  9.2 Knowledge Management ........................................................................................................77
  9.3 Extension of the CSP Model .................................................................................................78
  9.4 Explanations ...........................................................................................................................78

Appendix ........................................................................................................................................79
  Domain Knowledge Vehicle Configuration ..............................................................................79
  Constraints Generated for Vehicle Configuration .................................................................86
  Domain Knowledge Training Planning ..................................................................................91
  Constraints Generated for Training Planning ........................................................................95
  Training Planning Test Run Results .......................................................................................97
  Glossary .......................................................................................................................................98
  Bibliography ............................................................................................................................100
List of Figures

Figure 1 - SSD of the Generic Concepts ........................................................................................................... 25
Figure 2 - Conceptualization Vehicle Configuration .......................................................................................... 26
Figure 3 - Conceptualization Training Planning ............................................................................................... 27
Figure 4 - Aggregate Component ....................................................................................................................... 28
Figure 5 - Goal Structure .................................................................................................................................... 29
Figure 6 - Components of the Generic Configuration Framework ..................................................................... 32
Figure 7 - Subsystem Decomposition ................................................................................................................ 34
Figure 8 - High-Level Use Cases ....................................................................................................................... 36
Figure 9 - Refinement of Create Configuration use case ..................................................................................... 37
Figure 10 - Refinement of Create Domain Specific Configurator Use Case .......................................................... 38
Figure 11 - Conceptual Flow of Events ................................................................................................................ 39
Figure 12 - Sequence Diagram for Vehicle Configuration ................................................................................ 41
Figure 13 - Sequence Diagram for Training Planning .......................................................................................... 43
Figure 14 - Market Overview Constraint Systems ................................................................................................ 45
Figure 15 – Simplified Structure .......................................................................................................................... 52
Figure 16 - Architecture Subsystem Configurator ............................................................................................... 55
Figure 17 – Key Steps of the Configuration Process .......................................................................................... 57
Figure 18 - Communication AgendaProcessor and Constraint System ............................................................... 58
Figure 19 - Communication VariableCreator and Constraint System .................................................................. 59
Figure 20 - Workflow ConfigurationCreator ...................................................................................................... 59
Figure 21 - Overview Screen GUI ....................................................................................................................... 63
Figure 22 - Refinement of the General Goal ......................................................................................................... 64
Figure 23 - List of Suitable Cars .......................................................................................................................... 64
Figure 24 - Description of the Chosen Car ............................................................................................................. 65
List of Tables

Table 1 - Evaluation Web-Based Vehicle Configurators .............................................18
Table 2 - General Goals of Vehicle Configuration .....................................................60
Table 3 - Refinement of Sub Goals ...........................................................................61
Table 4 - Components for Vehicle Configuration .......................................................62
Table 5 - Refinement of General Training Goals .......................................................70
Table 6 - Refinement of Unit Goals ...........................................................................71
1 Introduction

1.1 Motivation

Configuration takes place in many different domains, however often concealed from a customer point of view. Junker (2006) describes configuration as follows: “the task of composing a customized system out of generic components”. Component-based systems are spread throughout industry. Prominent representatives of component-based systems are passenger cars. The cars of a manufacturer consist of many standard components, which are also shared throughout product lines but every combination with further, specialized parts forms a unique car with its own characteristics. Moreover, the customer is allowed to select from a wide range of colors, electronic equipment, and optional features to further customize the vehicle. This is resulting in an explosion of complexity since there are thousands of components in a car and dependencies between them are not obvious and hard to understand for customers. Let us assume a family father wants a vehicle for transportation of his whole family, he wants a car that is safe, which has enough space for luggage, has requirements on design, optional features like an MP3-compatible sound system and a car which he is able to afford. This customer is facing an immense task. There are many manufacturers; moreover, each of the manufacturers also has various models. He needs to look at each model if it fits his general goals concerning space, safety etc. Also, he does not really have exactly specialized general goals. Rather there are vague expectations. At the same time, he needs to consider prices and select the optional features, colors and interior design etc. Obviously, this is a complex, extensive and time consuming task.

Therefore, tools are required to enable the customer to help and guide this process. The idea is to explicitly model the customer’s goals, for example that he wants the car to be family-friendly, or he wants it to have a lot of space for luggage. Secondly, the family father should be able to state properties and restrictions the new car should have or adhere to. These could, for example, include the planned annual mileage or a maximum price.

There are other examples where tools can help users to make decisions when dealing with complex and time consuming configuration tasks. The training planning in a gym represents a similar set of problems like finding a car. A trainee wants to conduct a training that satisfies certain goals, which might be losing weight, gaining muscles, or gaining necessary strength for a specific sport. Since most people are not fitness professionals, these goals are also vague, the trainee might not know how exactly he should train. Additionally, every trainee has a set of individual properties like
gender, body-mass-index, a heuristic proxy for human body fat based on an individual's weight and height, former injuries or time constraints. In contrary to the vehicle configuration, the focus in training planning is more an automatic process of finding a suitable training plan given the customer’s goals and properties. User interaction, like picking specific exercises might not be desired, since the user does not have the necessary knowledge, and subjective decisions like colors of a car are not present in this domain. The major difference between the two example domains is the degree of user interaction for decision making, vehicle configuration is rather user interactive and training planning tends to automatic decision making. Currently, there are generally no tools available that can support the user in complex configuration tasks as described before. The objective of this study is therefore to analyze such interactive configuration tasks, which are characterized by the requirements to represent and exploit user specific goals and properties and to specify and implement a generic tool, for such terms.

1.2 Solution Idea

There needs to be a tool that supports the user in complex, extensive and time consuming configuration tasks as explained in the introduction. The tool shall allow the user to state his requirements. The goals might be of general nature and may be resolved in a refinement structure. Also properties and restrictions shall be available. Configurations shall be represented as component-based systems. Constraints shall define the relations between goals and components. The search for a suitable solution for the user requirements shall be modeled as constraint satisfaction problem. A constraint system shall be used to solve the constraint satisfaction problem in order to provide the user valid configurations. In order to support various application domains the tool shall be designed in a generic, domain independent way. Domain knowledge needs to be represented by constraints that describe the relations between goals and components and the dependencies of goals and components.

1.3 Results

A generic framework for configuration has been created within this study. Two prototypes have been implemented for domains with different characteristics. The vehicle configurator represents a classic component-based system with lots of components that have dependencies. The prototype showed how decision guidance can be provided by connecting the components to a goal structure and enabling the user to state his goals. The training planning configurator represents a component-
based system as well, but the emphasis is on a large refinement structure of the general goals that leads to concrete goals and eventually to components. The training planning configurator shows the possibilities of automatic configuration according to requirements of the user.

1.4 Outline

The following outline gives an overview of the structure of this thesis and gives a brief description of each chapter. It is recommended to start reading from the beginning since every chapter depends on the previous.

- **Chapter 2** states and illustrates the challenges of configuration for the context of this thesis and identifies requirements derived from the challenges and the motivation statement.

- **Chapter 3** comprises the state of the art of currently available web-based vehicle configurators of large car manufacturers. An overview of research in the field of configuration is given and an introduction on constraints and constraint satisfaction is presented.

- **Chapter 4** contains the conceptualization of vehicles and training planning. Derived by these conceptualizations the generic concepts are derived. Then a general approach how constraint satisfaction problems can be applied to configuration is presented.

- **Chapter 5** presents the software specification for the generic configuration framework. Design goals are identified and an architecture is proposed. Afterwards the sub systems are described in detail. For the constraint system is also given a market overview and a interface definition

- **Chapter 6** describes the implementation of the framework. The scope for the prototypes is defined and the concrete approach how the CSP shall be modeled is introduced. Used technology is described and the system design is presented.

- **Chapter 7** comprises the vehicle configuration prototype. It is described how the domain knowledge has been modeled and which features have been implemented in the prototype. At the end the validity is proven and an evaluation is presented.

- **Chapter 8** presents the prototype for training planning. The knowledge library is presented and differences to the knowledge modeling of the vehicle configurator are described. Then the functionality of the prototype is described. Afterwards the validity is proven and the prototype is evaluated.

- **Chapter 9** summarizes the content of this study and describes the results. Conclusions are drawn and areas of future work are suggested.
2 Problem and Requirement Analysis

In this section, challenges for configuration tasks are analyzed. The examples from the introduction are taken on and further discussed. Eventually requirements for new tools that aim at supporting the user in complex configuration tasks are derived, and an overview of the requirements is given at the end. For readability reasons the challenges have been separated in six different categories.

2.1 User Information

Remember the father of a family who is looking for a new car for his family, and have a closer look at his activities. The first step in the process of finding the best car for his purposes is an analysis of his goals. That leads to the necessity that there needs to be the possibility for the user of stating goals the configuration should satisfy. The goal “Family Car” is a vague, abstract description of a car. We need a refinement of what the abstract goal means in detail. The goal could imply that storage space is important, enough seats for at least four people are necessary, and that safety is more important than performance. Storage space could be divided into trunk space, space for legs, strollers, and so on. This leads to a refinement structure that can be resolved into more concrete, feasible sub goals. The first functional requirement for a new tool is therefore a refinement structure for goals. Besides goals, a user also may have properties; properties are information about the user that may add constraints to the configuration. When the family car is also used for longer weekend trips the annual mileage could be an important property of the user for the configuration. The fuel consumption of the car is related to the mileage, and the family father needs to take the running costs of the car into account. Therefore, the second functional requirement is the consideration of user properties. User restrictions are explicitly specified information about unwanted aspects in a configuration. User restrictions shall be functional requirement three.

2.2 Component-based Systems

In the section above we described that the user should be able to state his requirements. In order to obtain a configuration meeting these requirements, it needs to be modeled what a configuration looks like. A configuration is a set of interconnected components. A car, for instance, consists of many components like engine, tires, audio system and so on. Some of these components are of elementary type from a user’s perspective like the engine and do not need to be further refined.
Therefore, a hierarchy of components needs to be available for a fine-grained system representation. This leads to functional requirement four.

2.3 Decision Making

Consider again the family car, we assume that the goals are now stated clearly or have been refined into sub-goals. The user now does not want to look at all available cars himself. The tool should support him whenever it is possible. Therefore, the tool should provide different methods for decision making. Whenever the user does not want to or does not have the knowledge for a profound decision, an automatic selection should take over. In this way, the user is not confused by complex decisions and is supported in gaining a desired configuration. For instance, all cars with only two seats are obviously not suited as a family car and should, therefore, be disregarded automatically. This leads to functional requirement five, an automatic decision making mechanism. For subjective decisions or for expert usage decisions, it should also be possible to explicit select valid options, e.g. over choosing the colors of the car or for optional features like the MP3-compatible sound system. The manual decision making mechanism is functional requirement five.

2.4 Flexible Control

The control of the configurator workflow needs to be flexible for best support of different application domains. Flexible means there should be various possibilities like when to ask the user for choices, how to display options or how to handle options that are no longer available, when retraction of choices is possible or how to determine when a configuration is satisfied. The adaption should not be time extensive nor imply higher complexity of the configurator.

2.5 Knowledge Management

There needs to be a structured way for representing knowledge. In particular, dependencies among goals, the relations between goals and components as well as interdependencies among components need to be represented. Constraints over goals and components need to be managed. The knowledge base has to be created by domain experts who usually do not have a background in constraint programming or even computer science. This leads to functional requirement seven.
2.6 Generic Configuration Framework

As already stated in the introduction, there are a lot of application domains for configurators that can benefit from a tool that supports the user with complex configurations. That indicates a general need, thus the approach shall be designed in a domain independent way with a generic, domain-independent structure to allow easy adaption to various domains. The generic approach shall then be specialized to the needs of a particular application domain. This leads to functional requirement eight. Since the approach is to model configuration as a constraint satisfaction problem, a constraint system is necessary, that enables the desired functionality. This represents functional requirement nine.

2.7 Summary Requirements

New tools for configuration should meet the requirements presented in this section.

FR1: Goals with structure
The user shall be able to state his goals. There is also a structure for the refinement of goals necessary, i.e. a goal may imply or exclude another goal, or a goal may be a conjunction of disjunctions.

FR2: User Properties
The user shall be able to state properties that can influence the configuration. Properties are attributes of the user of the system which change infrequent like his annual mileage or his BMI.

FR3: User Restrictions
The user shall be able to state restrictions that can influence the configuration. Restrictions are parameters like the maximum purchase price of a car, unwanted components, or the length of the training.

FR4: Component Structure
A part-of hierarchy of components needs to be available for a fine grained system representation.

FR5: Automatic Decision Maker
An automatic configuration mode shall be available. Choices shall be selectable by an automatic mechanism implemented in the configurator. In contrast to the interactive mode, the user is not
asked to choose any options of choices. All choices shall be made automatically by the decision making mechanism. Different modes for automatic selection shall be possible, like random or heuristics. For the initial prototype implemented in this thesis an exemplary mechanism shall be implemented.

FR6: Manual Decision Maker
An interactive configuration mode shall be available. Choices shall be explicitly selectable by the user, i.e. during the configuration process, the user is asked step by step about choices that are necessary to obtain a complete configuration. Only valid options i.e. those respecting the domain knowledge, should be available for user selection. A specific application may also use both, manual decisions and automatic choices at different stages in the configuration process.

FR7: Representation of Domain Knowledge
There needs to be a way to store domain knowledge permanently. Domain specific components, goals, choices and the constraints over them need to be gathered and saved in a way readable for the constraint system.

FR8: Creation of a Generic Configuration Framework
A generic framework for configuration shall be created. It shall implement the stated requirements before. The generic framework must be able to support the creation of a specific application system by specialization of the generic classes and populating these specialized classes.

FR9: Constraint System
In order to be independent of a specific constraint system, the necessary functionality has to be defined and an interface has to be defined and implemented that allows all necessary operations to fulfill the requirements above. For the prototype a suitable constraint system has to be selected to provide proper functionality.

NFR1: Performance
Sufficient performance for interactive configurations is necessary. The user should not have to wait more than five seconds on any operation. Also for large configurations, say, configurations with over 300 components and over 100 goals and over 1000 constraints, the memory and computation time shall be less than 30 seconds.
NFR2: Modularity
There shall be a three layer architecture separating GUI, Configurator and Constraint System. All interfaces need to be defined properly to ensure encapsulation. The goal is to keep every layer exchangeable to ensure the highest flexibility for future work.

NFR3: Extendibility
There needs to be an easy way to integrate new goals, new components, new choices and new constraints. In this context, “easy” means that developers wanting to add new items should be able to focus on the implementation of classes realizing the item’s functionality. They should not have to worry too much about the integration of the new item into the rest of the tool.

NFR4: Usability
The configuration process needs to have a transparent structure the user is able to understand. Navigation to previous steps needs to be easy accessible. The flow of steps should be arranged in an easy understandable way for the user. Invalid components shall not be displayed.

3 State of the Art

This chapter contains an overview of currently available vehicle configurators including a conclusion. Then an overview of research in the field of configuration is given. The chapter is closed by an introduction to constraints and constraint satisfaction.

3.1 Vehicle Configurators

An evaluation of current vehicle configuration systems has been performed at the beginning of the study in order to assess current functionality. Web-based configurators of major vehicle manufacturers have been evaluated.
### 3.1.1 Market Overview

<table>
<thead>
<tr>
<th>Brand</th>
<th>Possibility of stating Goals, Properties or Restrictions</th>
<th>Usability</th>
<th>Decision Guidance</th>
<th>Effectiveness</th>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audi</td>
<td>Goals: / Properties: / Restrictions: Purchase Price, Lease Rate, Tax Rate, CO₂ Output, Car Type</td>
<td>The user can retract his input to any previous step. Only in the model selections there are problems with the retraction. Unavailable choices are vanished.</td>
<td>There is no effective search for a car. The user can only use some basic filters when he chooses a model.</td>
<td>The price is always available as also the exterior and the interior image of the vehicle and the price of every part of the car. User can also see at any point the technical information of the car. He can also search if there are cars available, save his choice or contact a sales assistant.</td>
<td>The design really good. The user can see in every step all the available choices. It would be better if all the models were presented and not only the series.</td>
</tr>
<tr>
<td>BMW</td>
<td>Goals: / Properties: / Restrictions: Purchase Price, Lease Rate, CO₂ Output</td>
<td>Generally not user-friendly. There is no concrete flow in the steps and there is no continue button. Difficult to navigate through the configurator.</td>
<td>There is no effective search for a car. The user can only use some basic filters when he chooses a model.</td>
<td>The car and equipment price is always available as also the exterior and the interior image of the vehicle. User can also see at any point the technical information of the car. He can also search if there are cars available, save his choice or contact a sales assistant.</td>
<td>The user cannot see in every step all the available parts that he has to select. Very simple design. Really complicated structure. However, there was a nice presentation of all the models available.</td>
</tr>
<tr>
<td>Citroen</td>
<td>Goals: / Properties: / Hard to understand currently available options. Sometimes the user is obliged to select</td>
<td>/</td>
<td>The car and equipment price is always available as also the exterior and the interior image of the vehicle. User can also</td>
<td>The design is nice but the configurator is extended in the whole screen which causes usability problems.</td>
<td></td>
</tr>
<tr>
<td>Restrictions: /</td>
<td>something in order to go further. Also the buttons “Continue” and “Back” are not well positioned.</td>
<td>see at any point the technical information of the car. He can also search if there are cars available, save his choice or arrange a Test-Drive</td>
<td>The steps are also not well positioned. Nice presentation of the car.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GM</strong></td>
<td><strong>Goals: /</strong></td>
<td>The car selection is really usable. The typical configurator is generally not user-friendly. There is no concrete flow in the steps and there is no continue button. Difficult to navigate through the configurator.</td>
<td>The design is really nice and aesthetic. However, the user is not well handled.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Properties:</strong> /</td>
<td></td>
<td>The user can select from a variety of useful filters that help him find his ideal cars of the company. The unavailable options are vanished.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Restrictions:</strong></td>
<td></td>
<td>The car and equipment price is always available as also the exterior and the interior image of the vehicle. User cannot see at every point technical data for the car</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase Price, Car Type, # Seats, Horsepower, Towing Capacity, Storage Capacity</td>
<td></td>
<td>The user is asked if he wants a row of questions about the purpose and usage of the car and there is a suggestion of three cars at the end.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Toyota</strong></td>
<td><strong>Goals: Indirect goals through questionnaire</strong></td>
<td>It is really easy to navigate through the configurator and retract your choices. However, it is difficult for the user to see in which step the configurator is.</td>
<td>The graphics are really nice. But when you build the car the design is very simple and the graphics of the car are bad.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Properties:</strong> /</td>
<td></td>
<td>The user is asked if he wants a row of questions about the purpose and usage of the car and there is a suggestion of three cars at the end.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Restrictions:</strong> Purchase Price, MPG, # Seats</td>
<td></td>
<td>The car and equipment price is always available as also the exterior and the interior image of the vehicle. User can see at every point technical data for the car and also many other details. The user can then contact a dealer.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


### Volkswagen

**Goals:** /  
**Properties:** /  
**Restrictions:** /

It is really easy to navigate through the configurator and retract your choices. The unavailable choices are vanished which is not really usable.

The car and equipment price is always available as also the exterior and the interior image of the vehicle. User can see at every point technical data for the car and also many other details. The user can then contact a dealer or arrange a Test-Drive.

All the models are presented together. The user cannot see in every step all the choices offered.

<table>
<thead>
<tr>
<th>Volkswagen</th>
<th>Goals: /</th>
<th>Properties: /</th>
<th>Restrictions: /</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>It is really easy to navigate through the configurator and retract your choices. The unavailable choices are vanished which is not really usable.</td>
<td>/</td>
<td>The car and equipment price is always available as also the exterior and the interior image of the vehicle. User can see at every point technical data for the car and also many other details. The user can then contact a dealer or arrange a Test-Drive.</td>
</tr>
</tbody>
</table>

Table 1 - Evaluation Web-Based Vehicle Configurators
3.1.2 Conclusion

There are approaches like the Toyota configurator, which aim at supporting the user in his decisions by using a questionnaire, but the configurator is not realized consequently enough to pose additional benefit since only the search part of the configuration process is covered. The configurators of GM, Toyota, Audi and BMW provide basic filters for decision guidance. Filters are a good way for eliminating unwanted options. Albeit there is no support in gaining help what options could be desirable. Generally it can be concluded that none of the evaluated configurators fulfills the requirements determined in chapter 2.7. Some approaches also lack basic features in usability, effectiveness and design. This proves that new tools with extended functionality are required.

3.2 Configuration Support: Research and Systems

This section gives an overview of general related research on configuration and existing solutions. Previous research on configuration, like by (Mittal, 90) focused on how to model configuration as a constraint satisfaction problem (CSP) and how to create dynamic constraints. A widely used example for a CSP is Sudoku. In Japanese su means digit and doku, unique (Choco 2012). The rules for the game are really simple: an 81 cells square grid is divided in 9 smaller blocks of 9 cells (3 x 3). Some of the 81 fields are filled with one digit. The aim of the puzzle is to fill in the other cells, using digits except 0, such as each digit appears once and only once in each row, each column and each smaller block. Obviously these rules can be easily translated into constraints. The constraints, the domain knowledge of Sudoku, are these rules:

- Each column must contain the digits from 1 to 9
- Each row must contain the digits from 1 to 9
- Each of the nine 3×3 boxes must contain the digits from 1 to 9

These rules can easily be modeled as constraints and are able to represent the domain knowledge of Sudoku. It will be shown later, that this is also possible for configuration. (Soininen, 97) focused on a general ontology for configuration and (Knoblock, 2001) focused on applying CSPs on data from mixed sources. We want to take the approach of modeling configuration as CSP as a basis for this work but do not focus on the creation of dynamic constraint constraints, rather want to present a general approach how interactive configurations can be modeled, similar to the work of Soininen. Interactive configurations also may contain different
stages in the configuration process like introduced by Knoblock. Therefore poses the research above the fundamentals for this study.

Ulrich Junker gives a summary on configuration in the Handbook of Constraint Programming (Junker, 2006). He explains that many different techniques have been applied to configuration problems because of the differences in the structure of the systems to be configured. There is rule-based reasoning, model-based reasoning and case-based reasoning for a general differentiation. The model-based approach uses techniques like description logic, constraint programming, resource-models and also answer-set programming. The first configurators were rule-based; nowadays the model-based approach constraint programming (CP) seems most promising according to Junker.

Constraint programming is the study of computational systems based on constraints. The idea of constraint programming is to solve problems by stating constraints (conditions, properties) which must be satisfied by the solution. In 1996 (Freuder, 96) said: “Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.” CP addresses the complete solution space of possible configurations without leaving out valid configurations. CP is able to discover unforeseen interactions between different components and constraints due to propagation, search and learning algorithms. However, the difficulties in classic CP methods are the appropriate presentation for the complex knowledge of the configurations domain, the reasoning about an unknown number of components, the handling of user preferences and the explanation of the configuration (Junker, 2006). Explanations are an important part of configurations. Explanation in this context means that the information about derived decisions or decisions that have been made by the user or the system is recorded and available to the user. If the user is made aware why components are contained in his configuration or why components may not be available in his configuration he gains knowledge about domain specific dependencies and is therefore able to understand his configuration. I.e. a training configuration for a trainee with excessive overweight will not contain to go running because the stress on his knee joints is too high. If running is just omitted in his training plan he will not understand the dependency to his weight. If there is an explanation he understands that he has to start losing weight in order to get a training plan that contains running. However, the representation of this information is not trivial, since only some decisions are relevant to the user and information about other decisions may only confuse him.
3.3 Constraints

A constraint is a relation over one or several variables, each having a finite domain. Reconsider Sudoku, a rule in Sudoku is that each column must contain the digits from 1 to 9. In this example each of the fields in the column represents a variable. A constraint restricts the valid values of a variable. In this case the valid values for a field are the digits from 1 to 9. This represents one of the constraints in Sudoku. For modeling the whole game more constraints would be necessary. Generally constraints express partial information about the variables it restricts. Constraints hold information about the domain they are used in.

In the choco constraint system (Choco, 2012) constraints are described with the following attributes:

- Typically a constraint on (say) two variables X,Y can be used to infer a constraint on X given a constraint on Y and vice versa, constraints are declarative, i.e. they specify what relationship must hold without fixing a computational procedure to enforce that relationship
- Constraints are additive, i.e. the order of composition of constraints does not matter, all that matters at the end is that the conjunction of constraints is in effect
- Constraints are rarely independent, typically the set of all constraints shares variables.
- Constraints arise naturally in most areas of human endeavor. The three angles of a triangle sum to 180 degrees, the sum of the currents flowing into a node must equal zero, the position of the scroller in the window scrollbar must reflect the visible part of the underlying document, these are some examples of constraints which appear in the real world. Thus, constraints are a natural medium for people to express problems in many fields.

3.4 Constraint Satisfaction

Constraint programming allows solving combinatorial problems modeled by a Constraint Satisfaction Problem. Formally, a CSP is defined (Choco, 2012) by a triplet (X, D, C):

- Variables: $X = \{X_1, X_2, ..., X_n\}$ is the set of variables of the problem
- Domains: $D$ is a function which associates to each variable $X_1$ its domain $D(X_1)$, i.e. the set of possible values that can be assigned to $X_1$.
- Constraints: $\{C_1, C_2, ..., C_n\}$ is the set of constraints. A constraint $C_j$ is a relation defined on a subset $X = \{X_1^j, X_2^j, ..., X_n^j\}$ of variables which restricts the possible tuples of values $(v_1^j, ..., v_n^j)$ for these variables:
Remark the father of the family who wants to buy a new car. We define two variables $X_1, X_2$. $X_1$ is defined as his general goal, a family-friendly car. We define a Boolean domain, containing true and false. Therefore the general goal family-friendly may have the value true or false. The second variable $X_2$ is defined as a car that is suitable for family use. The domain, again, is defined as Boolean. Now we want to express as a constraint, that if the general goal $X_1$, family-friendly, is activated, the car, $X_2$, is a valid option. Therefore we define the constraint $C_1$: $X_1, true \Rightarrow X_2, true$. This is an example of a constraint in vehicle configuration.

Like defined by (Mittal, 90), the task can then be described as: “finding consistent assignment of values to a fixed set of variables given some constraints over these variables”. The approach presented in this thesis applies constraint satisfaction to configuration.

4 Conceptualization of the Configuration Task

The conceptualization of the task of configuration treated in this thesis has been performed with UML static structure diagrams (SSD). SSDs show the static structure of a model, that is, elements that exists, their internal structure and how the elements are related. The conceptual diagrams represent concepts from the real world and the relationships between them. Derived by the analysis in chapter 2 a structure with generic concepts has been derived. The generic concepts are depicted in figure 1. After creation of the generic concepts, a conceptualization for the configuration of vehicles has been carried out. In order to proof the generic character of the framework a second application domain has been analyzed. The second domain is the training planning in a gym. Both domains are very suitable for configuration since both domains contain a user who wants to have a customized product out of standard components. In terms of vehicles components may be wheels, engines or models. In terms of training planning components may be exercises, training equipment or training sessions. The structures of the domains are very different and therefore allow a good indication of the genericity of the framework that shall be created. Then the attempt was made to assign a generic concept to each of the domain specific concepts. Multiple iterations have been necessary until the generic concepts have been able to model all domain specific concepts. Remark for reading the SSDs: Each class in the domain specific SSDs represents a concept. The name is given in the upper row, the middle row is empty or states if the concept is mandatory or optional, and the lower row states of which generic concept the domain specific concept is.
4.1 Generic Concepts

As already earlier stated, the generic concepts have been derived from the analysis in chapter 2 and have been applied to both example domains. However the concepts are also suitable for various other domains. During the whole process of modeling the generic concepts and their relations effort was made to ensure the generic character of the framework. Not only the two example domains have been considered but also domains like configuration of telephone switching systems or travel planning. In the following section a brief description of each concept as given:

**User**

A Person that has requirements that should be satisfied by the configuration he wants. He may have Properties or restrictions that can result in additional Constraints.

**User Requirement**

Generalization of User Goal and User Restriction.

**User Goal**

A UserGoal contains desired, ‘positive’, aspects of a configuration. A User Goal is satisfied by a Component Goal or a combination of Component Goals.

**User Property**

Additional information about the User, may add Constraints to the Configuration.

**User Restriction**

Explicitly specified information about unwanted aspects in a Configuration

**User Resource Restriction**

Explicit restriction on a resource of a component, e.g. money or time.

**Constraint**

A constraint is a relation over one or several variables. A constraint restricts the valid values of a variable.

**Domain Constraints**

Domain Constraints represent knowledge provided by experts of the corresponding domain, e.g. vehicle configuration or training planning.

**Constraint Variable**

An element that is subject to constraints.

**Component Type**

Generalization of a component. Defines the general characteristics of a component.
A component is an entity that can have functions. It can be combined with others components in a specified way to satisfy a desired UserGoal. Components may add Constraints to the configuration.

Aggregate Component
A component that consists of several components. Aggregate Components have a Structure that describes how the contained components are connected.

Component Function
Component Functions satisfy the User Goals. They might be conjunctions of Component Goals that satisfy the User Goals.

Component Attribute
Components may have additional attributes that describe their characteristics.

Component Resource
A resource a component consumes. E.g. money or time.

Structure
Aggregate Components have a Structure that describes how the contained components are connected.

Structural Relation
Contains the exact relation how two components are connected.
Figure 1 - SSD of the Generic Concepts
4.2 Vehicle Concepts

The following conceptualization shows all concepts necessary for configuration of vehicles in the context of this study. Note that this is an abstract view on vehicle configuration with emphasis on general goals. Some details are omitted; other concepts like goals for safety and the purpose of the car have been added.

Figure 2 - Conceptualization Vehicle Configuration
4.3 Training Concepts

The following conceptualization shows all concepts necessary for configuration of training plans in gyms.

Figure 3 - Conceptualization Training Planning
4.4 General Approach for Applying CSPs to Configuration

This section introduces the approach of this thesis to apply constraint satisfaction on configuration. At first there is a definition of important concepts and afterwards the relation between them is explained.

4.4.1 Components

Every configuration consists of at least one or more likely of a set of components. A component is an entity that fulfills some kind of function. There exist elementary components, which can also be organized in a hierarchy using the composite pattern. We then speak of aggregate components. For example in vehicle configuration the aggregate component exterior consists of the components wheels and color.

![Figure 4 - Aggregate Component](image)

4.4.2 Goals

Goals characterize the desired functionality or features of a configuration. Technically, there is no difference between the Goals a User has and the Goals a Component fulfills, but for a better understandability the terms User Goal and Component Function are used to clarify where the Goal is coming from. Component Functions are the goals that are directly satisfied by Components.

Formally, a User Goal $G$ can be implied by conjunctions of sub goals $G_i$:

$$\bigwedge_i G_i \Rightarrow G$$

Based on the closed world assumption, a constraint can be modeled:

$$\forall K \bigwedge_i G_{ki} \Leftrightarrow G$$

A User Goal could be that a vehicle is family friendly, very safe or very fast. These abstract terms are then organized in a more hierarchical structure. The User Goal “Family-Friendly” for instance could imply the User Goal “Safe Car” and exclude the User Goal “Comfortable Car” like depicted in figure 5.
Components have Functions, as described before we call them Component Functions. A Component Function $F$ can be implied by conjunctions of Components $C_i$:

$$ \land_i C_i \Rightarrow F $$

Again, based on the closed world assumption, a constraint can be modeled:

$$ \lor_k \land_i C_{ki} \Leftrightarrow F $$

### 4.4.3 Choices

*Choices* represent points in the configuration process where a decision is necessary because more than one option is available for a *Goal* or *Component*. For instance, there are many cars that fulfill the *Goal* “Safe Car”. Therefore, when a configuration shall be created a decision needs to be made about which option shall be selected. Formally, a choice $C$ implies a disjunction of *Goals* or *Components*.

$$ C \Rightarrow \lor_i G_i $$

Based on the closed world assumption a constraint can be modeled:

$$ \lor_k \lor_i C_{ki} \Leftrightarrow C $$

### 4.4.4 Validity

A valid Configuration is characterized by the following requirements. In general, it is assumed that the *User* has *User Goals* and wants a *Configuration* that satisfies the *User Goals*. Therefore, a valid *Configuration* has to entail the *User Goals*:

1. *Configuration* $\vdash$ *User Goals*

Since the *User* may have *User Restrictions*, it needs to be checked if the *Configuration* is consistent with them:

2. *Configuration* $\cup$ *User Restrictions* $\not\vdash \bot$
It also has to be checked if the Domain Constraints are consistent with the Configuration:

3. \( Configuration \cup User\ Properties \cup Domain\ Constraints \not\models \bot \)

### 4.4.5 Satisfaction

- A Configuration is a Component \( C \) or a set of components \( C_i \) that satisfy a User Goal \( G \) or a set of User Goals \( G_i \).
- A configuration \( C \cup Structure \) satisfies a goal \( G \), if:
  
  \[
  C \cup Domain\ Constraints \not\models \bot \\
  C \cup Domain\ Constraints \models G \\
  \]

  \( C \) satisfies a set of goals \( G_i \) if \( C \) satisfies every \( G_i \)

  \( C \) is a minimal Configuration for \( G_i \) if there is no proper subset of \( C \) that satisfies \( G_i \)

### 4.4.6 Quantifications

As defined before Component Functions are equal to User Goals, if the User Goal is implied directly by a Component Function. However, some domains also need a fine-grained degree of satisfaction. Therefore a quantification of a User Goal is necessary. The quantification is realized by a weight restriction function that values the User Goal or Component Function. Then a User Goal \( G \) can be implied by a conjunction of sub goals \( G_i \) combined with the weight restriction function for the specified goals.

\[
\bigwedge_i G_i \land \text{weightRestriction}(G_i) \Rightarrow G
\]

The weight function might contain arbitrary operations like minimum, greater-than, equality or even complex operations.

Example 1: A User Goal can have a minimum number of Component functions that satisfy it. Then the conjunctions are replaced by the weight function \( \text{min}(G_i) \). A car needs to have a minimum safety level of four NCAP Stars, so there is a minimum of four stars defined in \( \text{min4Stars}(G_i) \) for the UserGoal safety:

\[
\bigwedge_i G_i \land \text{min4Stars}(G_i) \Rightarrow User\ Goal\ safety
\]

Example 2: The User Goal \( G \) says that the goal triceps muscle has to be trained for 80%. There are exercises with component functions \( G_i \) available for training the triceps, but they only train the muscle e.g. for 40% each. Obviously two of these exercises are necessary in order to fulfill the goal. Therefore a \( \text{min80}(G_i) \) restriction function is defined as a greater-than operation for the triceps goal \( G \).
4.4.7 Configuration

According to (Junker, 2006) a configuration problem is characterized by two constituents:

a) A catalog which describes the generic components in terms of their functional and technical properties and the relationship between both.

b) User requirements and user preferences about the functional characteristics of the desired configuration.

In general this definition fits quite well, but may be very abstract and too general. In order to get a concise description of the concepts used in the approach of this thesis the definition needs to be extended. The static structure diagram figure 1 depicts an overview of all concepts and their relation. Remark that in consideration of future work this approach has been developed more detailed than needed in the basic version that shall be implemented within this study. In chapter 6 a simplified approach is introduced that also fulfills the requirements stated in chapter 2.7 and has been implemented in the prototypes.

c) A Configuration is a set of components. The Component Type defines of what kind the Component is. A Component has a Component Function that it fulfills. Component Attributes define the characteristics of Components. Component Resources are a special class of Component Attributes. An Aggregate Component has a Structure how it is build. The Structure contains the Structural Relations of the contained Components. Structural Relations are pair wise connections of ports of components.

d) The User has further User Requirements and User Properties associated. User Requirements are divided in User Goals and User Restrictions. User Properties restrict via Domain Constraint(s) the solution of the configuration. The User has User Goals which have to be satisfied by the Component Goal (s). User Goals contain desired aspects of the Configuration. User Restrictions define the unwanted aspects of the configuration and have to be consistent with the Configuration. Resource Restrictions are a special class of User Restrictions and constrain the Component Resources.

e) Constraints consist of one or more Constraint Variable. Component Attribute, Structure, User Restriction, User Goal, User Property, User Goal Quantification are Constraint Variables, which may be subject to Domain Constraints. Domain Constraints represent
knowledge provided by experts of the corresponding domain. Domain Constraints have to be consistent with the Configuration.

5 Software Specification

5.1 Design Goals

The main goal of this study is to create a generic configuration framework that is adaptable to arbitrary application domains. To make decisions in the system design easier, the requirements have been analyzed before developing the system in order to derive design goals. Those design goals represent qualities enabling the prioritization of aspects of the system’s development. The major design goals identified are:

• Generic Character of the Framework
• Extendibility
• Encapsulation

In the following sections the design goals are discussed in detail.

5.1.1 Generic Character of the framework

This design goal is directly derived by the main purpose of this thesis, the development of a generic framework for configuration. As stated in the requirements analysis in chapter 2.7 the configurator must be able to support the creation of a specific application system by specialization of the generic classes and populating of these specialized classes. The configurator shall be able to support various application domains without limiting workflow, components, constraints or user requirements.

![Generic Configuration Framework](image)

Figure 6 - Components of the Generic Configuration Framework
The configurator should not be seen as a standalone application. Three major tasks can be identified. The communication with the user via a graphical user interface (GUI), the control of the configuration process (configurator) and the solving of the constraint satisfaction problem (constraint system). Every task is combined in a package as depicted in figure 6. Since for most application domains there is already a GUI available this shall not be in the scope of the prototype. It is assumed that there is an interface that allows communication with the user. The design of the constraint system is not part of this study either, but a suitable constraint system shall be selected in order to provide a functioning system. The focus of this study is the design of the generic configuration framework and the design and implementation of the configurator subsystem. The subsystem GUI is omitted; only for evaluation purposes a student will create a simple demonstrator. The constraint system shall be acquired as third party software and therefore a precise definition on requirements and necessary functions needs to be done for choosing a constraint system.

5.1.2 Extendibility

Since it is not known to what application domains the configurator will be applied it is essential to provide possibilities of extension. New Components as well as new component types need to be easy addable. New constraint types need to be available. The control of the configurator must be able to support a flexible adaption of the workflow. The automatic decision making module needs to be open for new methods. Also the constraint system should be chosen with respect to future functions. This design goal is derived by non functional requirement extendibility.

5.1.3 Encapsulation

In order to provide the generic character of the framework and ensure the extendibility of the tool the architecture should have low coupling and a clear encapsulation strategy should be applied. This design goal is derived by the non functional requirement modularity.

5.2 Proposed Architecture

At first, a conceptual overview is given in the following section. Using a UML package diagram the conceptual subsystems are described on a high level. Afterwards, the boundaries and responsibilities of the subsystems are explained in detail. The development of the configurator subsystem is a major part of this study. Therefore in chapter 5.3 use cases are identified, a detailed conceptualization is given and the dynamic behavior of the system is described. The subsystem
constraint system is covered in chapter 5.4. At first requirements are identified followed by an overview of currently available constraint systems. The decision for a specific constraint system is contained in chapter 5.4.3. A complete interface definition including functions for future work is given in chapter 5.4.4.

5.2.1 Subsystem decomposition

This section describes how the system is decomposed into subsystems and discusses their boundaries and responsibilities. This is of special importance, since the system consists of different subsystems with completely different objectives. Three major subsystems have been identified:

- Graphical User Interface (GUI)
- Constraint System (CS)
- Configurator

At first, a conceptual overview is given in form of a UML package diagram representing the conceptual subsystem decomposition. Figure 7 shows a UML package diagram describing the conceptual subsystems on a high level.

![Figure 7 - Subsystem Decomposition](image)

5.2.2 Graphical User Interface

The Graphical User Interface is responsible for allowing the user to state his requirements. During configuration progress needs to be displayed. If user interaction is necessary the GUI must support interaction with the user. Results need to be displayed. The GUI initiates the requests of the user with the configurator. For this prototype the modeling of the GUI is omitted. It is assumed that most application domains already have existing GUIs and reuse is assumed. However, for testing purposes a simple GUI has been implemented.
5.2.3 **Constraint System**

The constraint system solves the constraint satisfaction problem. It needs to support various function necessary for the configuration. Development of a constraint system is not part of the scope of this thesis. However, since the function of the whole system is depending on the constraint system a requirement and interface definition is necessary in order to select a proper system. Both are available in chapter 5.4 in order to determine a suitable constraint system that can be integrated as third party software.

5.2.4 **Configurator**

The main focus of this study is the design of the configurator module. It needs to handle the communication requests by the user. The modeling of the domain application and domain constraints need to be implemented here. Domain knowledge is translated into a domain model via constraints and mapped into variables in the constraint system. The task is to find a solution to the problem defined by the user inputs and the domain model. The control of the generic configuration framework is located in the configurator subsystem. In the following section these tasks are further specified.

5.3 **Subsystem Configurator**

This section describes the architecture of the configurator subsystem. The modeling of the configurator subsystem is the main focus of this study. The architecture needs to implement the given functional and nonfunctional requirements according to the system architecture and design goals. At first the use case model is presented, followed by the static model and the dynamic model.

5.3.1 **Use Case Model**

Based on the functional and non functional requirements identified in the requirements definition three roles are defined: User, Developer and Domain Expert. Figure 8 shows an overview of the high-level use cases of the system. Three use cases are identified:

- **Create Configuration**: The user states his user requirements and wants to acquire a valid configuration.
- **Create Domain Specific Configurator**: The developer wants to create a new configurator for a specific domain.
• Model Domain Knowledge: The domain expert wants to model the knowledge of a specific domain.

![Diagram]

Figure 8 - High-Level Use Cases

The high-level overview is now refined by more fine-grained use cases. At first the Create Configuration use case will be refined. The newly identified use cases are depicted in figure 9 and are:

• Set General Goals: The user sets his general goals from the set of available general goals.
• Set Properties: The user sets his properties from the set of available properties with his personal values.
• Set Restrictions: The user sets his restrictions from the set of available restrictions.
• Start Configuration: The user starts the configuration.
The second high-level use case that has to be refined is the Create Domain Specific Configurator use case. The refinement in figure 10 depicts the following new use cases:

- **Edit Workflow**: The developer wants to edit the flow of steps for the user during the configuration process.
- **Edit Constraint Types**: The developer wants to edit the constraint types available to the domain expert.
- **Edit Decision Making**: The developer wants to edit the decision making mechanism.

---

Figure 9 - Refinement of Create Configuration use case
5.3.2 Dynamic Model

This section documents the system’s dynamic behavior in terms of sequence diagrams. Although some use case information is described redundantly, dynamic models enable us to specify the behavior more precisely. The sequence diagrams depict the order of interactions between participating objects and therefore give a detailed description of the system’s behavior. There is a sequence diagram for the conceptual overview how the communication between subsystems is managed and then exemplary sequence diagrams how the flow of events between user and configurator could be realized in the application domains vehicle configuration and training planning.

5.3.2.1 Communication between subsystems

The following sequence diagram shows a high-level view of the communication between the subsystems. Remark that the communication is conceptualized and that for better readability distinct communication calls are omitted. Normally, SetUserInformation, update Context, or results are performed by a series of different methods. Note that arbitrary many calls due to different choices or changed user information are possible. The figure shows only two exemplary calls for setting the user information and the selection of choices after the result of the first call is shown.
5.3.2.2 Vehicle Configuration Flow of Events

This section contains UML sequence diagrams for the use case Start Configuration presented earlier in section 5.3.1. There is a diagram given for vehicle configuration as well as for training planning. Remark that the diagrams show an exemplary workflow. At all times the user is allowed to change goals, properties and restrictions or retract choices. Following you will find a brief description of the sequence diagrams.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Start Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Domain</td>
<td>Vehicle</td>
</tr>
<tr>
<td>Short Description</td>
<td>A person wants to buy a new car that fulfils his requirements</td>
</tr>
<tr>
<td>Actors</td>
<td>User, Configurator</td>
</tr>
<tr>
<td>Trigger</td>
<td>User starts Configurator System</td>
</tr>
<tr>
<td>Preconditions</td>
<td>User knows what his requirements are</td>
</tr>
<tr>
<td>Postconditions</td>
<td>Configuration fulfils requirements</td>
</tr>
<tr>
<td>Included Use Cases</td>
<td>Set General Goals, Set Properties, Set Restrictions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow of events</th>
<th>Step</th>
<th>Actor</th>
<th>Basic Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.</td>
<td>User</td>
<td>Starts Configurator</td>
</tr>
<tr>
<td></td>
<td>2.</td>
<td>System</td>
<td>Show CarGoals</td>
</tr>
<tr>
<td></td>
<td>3.</td>
<td>User</td>
<td>Specify CarGoals</td>
</tr>
<tr>
<td></td>
<td>4.</td>
<td>System</td>
<td>Show FinancialRestrictions</td>
</tr>
<tr>
<td></td>
<td>5.</td>
<td>User</td>
<td>Specify FinancialRestrictions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>System</td>
<td>Show CarProperties</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>User</td>
<td>Specify CarProperties</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>System</td>
<td>Show possible models</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>User</td>
<td>Select model</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>System</td>
<td>Show possible engines</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>User</td>
<td>Select engines</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>System</td>
<td>Show possible EquipmentLines</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>User</td>
<td>Select EquipmentLine</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>System</td>
<td>Show possible OptionalFeatures</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>User</td>
<td>Select OptionalFeatures</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>System</td>
<td>Show possible Colors &amp; Rims</td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td>User</td>
<td>Select Color &amp; Rims</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td>System</td>
<td>Show complete Configuration</td>
<td></td>
</tr>
</tbody>
</table>

**Exceptions, Extensions**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3a, 5a, 7a, 9a, 11a, 13a, 15a, 17a</td>
<td>System</td>
<td>No possible Component available, retry previous step with different value</td>
</tr>
</tbody>
</table>

Figure 11 shows the workflow as an UML sequence diagram between the user and the GUI. The diagram shows an exemplary workflow that leads to a complete configuration.
Figure 12 - Sequence Diagram for Vehicle Configuration

5.3.2.3 Training Planning Flow of events

The following sequence diagram shows the flow of events for the training planning domain. Also for the training planning the workflow is only exemplary but shows the steps to a complete configuration.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Start Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Domain</td>
<td>Training</td>
</tr>
<tr>
<td>Short Description</td>
<td>A person wants get a new training plan that</td>
</tr>
</tbody>
</table>
fulfils his requirements

<table>
<thead>
<tr>
<th>Actors</th>
<th>User, Configurator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger</td>
<td>User starts Configurator System</td>
</tr>
<tr>
<td>Preconditions</td>
<td>User knows what his requirements are</td>
</tr>
<tr>
<td>Postconditions</td>
<td>Configuration fulfils requirements</td>
</tr>
</tbody>
</table>

| Included Use Cases | Set General Goals, Set Properties, Set Restrictions |

<table>
<thead>
<tr>
<th>Flow of events</th>
<th>Step</th>
<th>Actor</th>
<th>Basic Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.</td>
<td>User</td>
<td>Starts Configurator</td>
</tr>
<tr>
<td></td>
<td>2.</td>
<td>System</td>
<td>Show Training Goals</td>
</tr>
<tr>
<td></td>
<td>3.</td>
<td>User</td>
<td>Specify Training Goals</td>
</tr>
<tr>
<td></td>
<td>4.</td>
<td>System</td>
<td>Show Properties</td>
</tr>
<tr>
<td></td>
<td>5.</td>
<td>User</td>
<td>Specify Properties</td>
</tr>
<tr>
<td></td>
<td>6.</td>
<td>System</td>
<td>Show Restrictions</td>
</tr>
<tr>
<td></td>
<td>7.</td>
<td>User</td>
<td>Specify Restrictions</td>
</tr>
<tr>
<td></td>
<td>8.</td>
<td>System</td>
<td>Inform when inconsistency occured, advise changes</td>
</tr>
<tr>
<td></td>
<td>9.</td>
<td>User</td>
<td>Accept when changes are advised</td>
</tr>
<tr>
<td></td>
<td>10.</td>
<td>System</td>
<td>Generate Training Plan</td>
</tr>
<tr>
<td></td>
<td>11.</td>
<td>User</td>
<td>Ask for details/explanations</td>
</tr>
<tr>
<td></td>
<td>12.</td>
<td>System</td>
<td>Show details/explanations</td>
</tr>
<tr>
<td></td>
<td>13.</td>
<td>User</td>
<td>Exchange exercises</td>
</tr>
<tr>
<td></td>
<td>14.</td>
<td>System</td>
<td>Show changed training plan</td>
</tr>
</tbody>
</table>

| Exceptions, Extensions | 8a, 14a | System | No possible Component available, retry configuration with different user requirements |

Figure 12 shows an exemplary workflow as a UML sequence diagram.
The constraint system is an essential part of the system as it solves the task of finding a suitable solution for the CSP. To meet the requirements and challenges stated in chapter 2 requirements for the constraint system need to be specified. Then a market overview is given and the evaluation with the decision for a system. Afterwards the interface to the constraint system is defined.

5.4 Requirements

The requirements for the constraint system have been derived from the objectives and requirements given in section 2. The following requirements have been elated:

C.FR1: Constraints (types and instances) need to be saved persistently
C.FR2: Constraints need to be added dynamically during configuration
C.FR3: Constraints need to be easy maintainable (e.g. grouping with types)
C.FR4: Conflicts (reasons for inconsistencies) need to be displayed
C.FR5: Explanations for configurations are necessary
C.FR6: User-based interactive choices and revision of choices during configuration
C.FR7: Automatic choices during configuration (different modes like random, heuristics …)
C.FR8: A way to encode preferences would be advantageous
C.FR9: Interface to the constraint system
C.FR10: Focusing on parts of the constraint network, disabling of parts of the network (e.g. due to retraction of choices)
5.4.2 Market Overview

A research of current state of the art constraint systems has been performed. In the following table are the most promising systems. Eight attributes have been identified based on the requirements presented in the previous section.

<table>
<thead>
<tr>
<th>System Type</th>
<th>Dynamic Constraints</th>
<th>Persistent Constraints</th>
<th>Maintaianable</th>
<th>Movable</th>
<th>Conflict Display</th>
<th>GUI Interface</th>
<th>Degree of Interaction</th>
<th>Explanations (ATMS)</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minion, independent solver</td>
<td>Yes</td>
<td>No</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>Boolean, manual</td>
<td>Automatic, manual</td>
<td>n/a</td>
<td><a href="http://minion.csail.mit.edu/">http://minion.csail.mit.edu/</a></td>
</tr>
<tr>
<td>ILCO, toolkit</td>
<td>No</td>
<td>Yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>C++ extension</td>
<td>Automatic, manual</td>
<td>n/a</td>
<td><a href="http://www.cs.cmu.edu/~ltm/ILCO/">http://www.cs.cmu.edu/~ltm/ILCO/</a></td>
</tr>
<tr>
<td>Geopolis, toolkit</td>
<td>Yes</td>
<td>Yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>C++ extension</td>
<td>Automatic, manual</td>
<td>n/a</td>
<td><a href="http://www.cs.cmu.edu/~ltm/ILCO/">http://www.cs.cmu.edu/~ltm/ILCO/</a></td>
</tr>
<tr>
<td>Choice, Java library</td>
<td>Yes</td>
<td>Yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>C++ extension</td>
<td>Automatic, manual</td>
<td>n/a</td>
<td><a href="http://www.cs.cmu.edu/~ltm/ILCO/">http://www.cs.cmu.edu/~ltm/ILCO/</a></td>
</tr>
<tr>
<td>CS3, Net library</td>
<td>Yes</td>
<td>Yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>C++ extension</td>
<td>Automatic, manual</td>
<td>n/a</td>
<td><a href="http://www.cs.cmu.edu/~ltm/ILCO/">http://www.cs.cmu.edu/~ltm/ILCO/</a></td>
</tr>
</tbody>
</table>

Figure 14 - Market Overview Constraint Systems
5.4.3 Evaluation

Many systems are available and differences are often obscure. The identification of the most important attributes based on the given requirements is a helpful technique in order to make a profound decision. The ILOG system distributed by IBM is a very promising approach, especially because there is a comprehensive package of software also for managing knowledge and keeping it consistent. Other approaches like choco, free and open source, are also promising candidates. At the end the explanation capabilities via ATMS made the difference. Since CS3 is the only system supporting explanations it has been chosen as constraint system.

5.4.4 Interface Definition

CS3 is closed source software distributed as a .Net library. In order to integrate the constraint system in the application environment an interface definition was necessary. The function calls have been identified based on necessary operations needed for the configuration task based on the theoretical foundation given in section 3. At first data types have been defined followed by an overview of implemented functions. A perspective for future functions is also given in chapter 5.5.2.3.

5.4.4.1 Data Types

In order to meet the requirements necessary data types and function calls have been identified. The following section gives an overview of the data types.

5.4.4.1.1 IAssumption

IAssumption defines the name of a variable and holds the corresponding value.

Interface IAssumption

String Variable

String Value

End Interface

5.4.4.1.2 IDomain

IDomain defines the name of a domain.

Interface IDomain

String Name

End Interface
5.4.4.1.3   IVariable

IVariable defines the name of a variable.
Interface IVariable
    String Name
End Interface

5.4.4.1.4   IVariableValueAssignment

Assignment of a value to a variable.
Interface IVariableValueAssignment
    IVariable Variable
    String Value
End Interface
5.4.4.2 Functions

The following function calls have been identified for a basic version of the configurator.

<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
<th>Description</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IVariable defVariable(String name, IDomain domain)</td>
<td>Creates an instance of a variable with the variable’s domain.</td>
<td>Name of the variable as IVariable.</td>
</tr>
<tr>
<td>2</td>
<td>IDomain defDomain(String name, String values())</td>
<td>Defines the name and the valid values for a domain. Variadic argument values.</td>
<td>Name of the domain as IDomain.</td>
</tr>
<tr>
<td>3</td>
<td>IConstraintType defConstraintType(String name, IDomain signature(), String tuples())</td>
<td>Defines a constraint type with name, signature and domain. Variadic argument signature and tuples.</td>
<td>Name of the constraint type as IConstraintType.</td>
</tr>
<tr>
<td>4</td>
<td>Void defConstraint(IConstraintType constraintType, IVariable variables())</td>
<td>Creates an instance of a constraint type. Variadic argument variables.</td>
<td>Void</td>
</tr>
<tr>
<td>5</td>
<td>Void applyConstraints()</td>
<td>Activates all instantiated constraints in the constraint system.</td>
<td>Void</td>
</tr>
<tr>
<td>6</td>
<td>IAssumption assume(IVariable variable, String value)</td>
<td>Creates an assumption by assigning a value to a variable.</td>
<td>Name and value of the variable as IAssumption.</td>
</tr>
<tr>
<td>7</td>
<td>List &lt;IVariableValueAssignment&gt; getVariableValues (IVariable variable, list &lt;IAssumption&gt; context)</td>
<td>Gets the valid values of a variable in the given context.</td>
<td>List of valid values for variable that are consistent with context, possibly empty set</td>
</tr>
<tr>
<td>8</td>
<td>Boolean isValueConsistent (IVariableValueAssignment vva, list &lt;IAssumption&gt; context)</td>
<td>Determines if the value of the variable is consistent with the context.</td>
<td>true, iff value ∈ getVariableValues(variable, context)</td>
</tr>
<tr>
<td>9</td>
<td>list&lt;IVariableValueAssignment&gt; getRestrictions (list &lt;IAssumption&gt; context, variable IVariable)</td>
<td>Determines which values for a variable are consistent in the context.</td>
<td>List of variables with valid values for variable, consistent with context.(empty list could mean “ALL”)</td>
</tr>
<tr>
<td>10</td>
<td>Boolean isImplication (IVariableValueAssignment vva, list &lt;IAssumption&gt; context)</td>
<td>Determines if the value of the variable is implied by the context.</td>
<td>true, iff value ∈ getVariableValues(variable, context)</td>
</tr>
<tr>
<td>11</td>
<td>Void</td>
<td>Determines what is derivable from the Delegate function.</td>
<td></td>
</tr>
</tbody>
</table>
### Future Functions

To ensure extensibility for future features, these functions have been identified:

<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
<th>Description</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>List &lt;list&lt; IAssumption&gt;&gt; <strong>MinInconsContexts</strong> (IVariable variable, String value, list&lt;IAssumption&gt; context)</td>
<td>For explanation and revision capacity. Minimal context k’ in k, with value of variable inconsistent.</td>
<td>Set of minimal Set Context' of Context, with isValueConsistent (Variable, Value, Context') = False</td>
</tr>
<tr>
<td>2</td>
<td>List &lt;list&lt; IAssumption&gt;&gt; <strong>MinInconsContextAll</strong> (IVariable variable, list&lt;IAssumption&gt; context)</td>
<td>For explanation and revision capacity. Minimal context k’ in k, with all values of variable inconsistent.</td>
<td>Set of minimal Set Context' of Context, with isValueConsistent (Variable, Value, Context') = empty</td>
</tr>
<tr>
<td>3</td>
<td>List &lt;list&lt; IAssumption&gt;&gt; <strong>Concludes</strong> (IVariable variable, list&lt;IAssumption&gt; context)</td>
<td>Minimal context k’ in k, which determine the restrictions of the variable.</td>
<td>Set of minimal Set Context' of Context, with getVariableValues(variable, context') = getVariableValues (Variable, Context).</td>
</tr>
<tr>
<td>4</td>
<td><strong>Trace()</strong></td>
<td>Which constraints are involved</td>
<td>tbd</td>
</tr>
</tbody>
</table>
6 Implementation

The following sections describe the software design for the configurator subsystem. The software design for the subsystems GUI and Constraint Systems are omitted, only necessary interfaces are included. At first the scope of the framework that shall be built is defined. Then the approach how to model the CSP in a programming language is presented. Afterwards a conceptual overview of the architecture is presented and a description of its packages and their functions and responsibilities is presented. Then the object model is described. Finally the global software control is described.

6.1 Scope

Chapter 4.4 presents a general approach how configuration problems can be modeled as constraint satisfaction problem without limiting the solution space. The prototype that shall be built within this study does not demand to implement all possible features, since it would extend the scope of a master’s thesis. The scope is reduced to a significant subset of features that covers the requirements stated in chapter 2.7. The design of the configurator is affected by this decision. The next section describes how the constraint satisfaction problem is modeled.

6.2 Simplification of the CSP

In order to obey time limitations for the prototype development a simplification of the approach for applying CSPs on configuration, introduced in chapter 4.4, has been created. Some concepts in the generic structure have been changed and a couple of assumptions have been determined.

6.2.1 Simplified Structure

The generic structure presented in chapter 4 has been simplified since it contains various concepts that are not necessary for the prototypes. Remark that in consideration of future work the theoretical approach has been developed more detailed than needed in the basic version that shall be implemented within this study. Figure 15 depicts the classes that have been considered. A new concept ConfigObject has been introduced. The sections below explain the structure in detail.

6.2.1.1 ConfigObject

A configuration contains different objects namely Goals, Components and Choices. We call these objects ConfigObjects because of their related structure. The abstract class ConfigObject defines
Implementation

their structure. ConfigObject defines a variable in the constraint system whether the object is active or not in the configuration. Also a name is defined and the type of the object, Goal, Component or Choice. The inheritance structure of classes inheriting ConfigObject is depicted in figure 15.

6.2.1.2 Choice
The class Choice implements the abstract class ConfigObject. Additionally Choice has a field with a list of options, to set an option the method SetOption is available, GetOptions returns the list of available options. The list of options is of the type ConfigObjects and represents the disjunctive options that are available for the choice. Also Choice has an attribute “satisfied” which is true if one of the options of the choice is active. Additionally when a choice is created it is determined if the choice is of the type automatic or manual. This information is used by the DecisionMaker to decide how the choice will be satisfied.

6.2.1.1 GoalOrComponent
The class GoalOrComponent inherits ConfigObject and is abstract as well. In this class two fields are added, namely implications and exclusions. Both are a list of ConfigObjects. Also the respective getter and setter methods for implications and exclusions are contained.

6.2.1.2 Goal
The class Goal implements GoalOrComponent. The inherited field implications are ConfigObjects that are implicated by the instance of the goal. The field exclusions are ConfigObjects that are not valid when the instance of the goal is active in the configuration.

6.2.1.1 Component
The class Component implements GoalOrComponent. The inherited field implications are ConfigObjects that are implicated by the instance of the component. The field exclusions are ConfigObjects that are not valid when the instance of the component is active in the configuration.
6.2.2 Assumptions

6.2.2.1 Activity of ConfigObjects

Goals, components and choices are ConfigObjects. ConfigObjects can be active or inactive. Goals can imply activity of ConfigObjects. Formally we can say:

\[
\text{Goal.active} \Rightarrow \text{ConfigObject.active}
\]

A Goal may imply an arbitrary set of ConfigObjects. Components can imply activity of ConfigObjects as well.

6.2.2.2 Inactivity of ConfigObjects

Goals can exclude goals and components via implication of not active. Formally we say:

\[
\text{Goal.active} \Rightarrow \text{NOT(ConfigObject.active)}
\]

Components can exclude goals and components as well.
6.2.2.3 Choices

Choices are simple disjunctions; quantifications of goals as described in 3.3.1.6 are not implemented. Choices are classified into manual choices and automatic choices. Since choices are ConfigObjects they can be active or not active. An active choice implies that one of its options is active. Formally we say:

\[ \text{Choice.active} \Rightarrow \text{OR(option}_i\text{.active)} \]

6.2.2.4 Constraints

An important restriction is that only one instance of each ConfigObject is allowed. Constraints are formulated over a pre-defined universe of ConfigObjects. All constraints, namely the choice implications, are instantiated and applied in the constraint system before starting the configuration process.

6.2.2.5 Satisfaction

Reconsidering the definition of satisfaction in chapter 4.4:

- A Configuration is a Component C or a set of components \( C_i \) that satisfy a User Goal \( G \) or a set of User Goals \( G_i \).
- \( C \cup \text{Domain Constraints} \vdash G \)
- C satisfies a set of goals \( G_i \) if C satisfies every \( G_i \# \)
- \( C \cup \text{Domain Constraints} \nvdash \bot \)

When all open agenda items are satisfied, all User Goals need to be satisfied. Therefore we conclude that a configuration is satisfied, when the agenda is empty. Since all constraints are instantiated before the configuration is started consistency is guaranteed at all times. Entailment of the Goals is already proven by the empty agenda.

6.2.2.6 Creation of Constraints

There are two kinds of data that needs to be translated into constraints:

- Domain knowledge: Knowledge modeled by the domain expert.
- User Requirements: General Goals, properties and restrictions of the user.

Domain knowledge is supposed to change slowly over time. Only updates from time to time by the domain expert are assumed. User Requirements on the contrary are supposed to change from every configuration.
6.3 Technology
Since the subsystems of the generic configuration framework shall be fully encapsulated and are therefore exchangeable the technology for the implementation of the configurator subsystem could be chosen freely. The chosen constraint system CS3 is implemented with Microsoft’s .Net framework. Therefore the use of a technology that uses .Net technology is feasible due to simplification of the interfaces. For the implementation of the prototype VB.Net has been chosen, one of Microsoft’s newest programming languages. VB.Net is an object-oriented programming language supporting console application but also allowing easy creation of graphical user interfaces using windows forms. Structured error handling and access to the .Net framework is possible as well.

6.4 System Design
The architecture of the configurator separates two packages, the ConfigurationCreator package and the DataCreator package. ConfigurationCreator is responsible for the actual process of creating a configuration out of components according to the user requirements. DataCreator is responsible for translating data into constraints. Figure 16 shows the conceptual architecture of the packages; note that interfaces to GUI and Constraint System are omitted for readability reasons. A brief description from a high-level point of view of the packages and the contained sub packages is given in the next two sections. Then the static and dynamic behavior is described, for better comprehension the static and dynamic behavior is described separately.
Figure 16 - Architecture Subsystem Configurator

6.4.1 System Overview

6.4.1.1 Package ConfigurationCreator

As depicted in figure 16 the package ConfigurationCreator consists of the package AgendaProcessor which contains the package ManualSelector and AutomaticSelector. AgendaProcessor creates an agenda with items that need to be processed in order to obtain a valid configuration. Items are goals that need to be further refined or choices about components. The agenda is created by using the interface to the constraint system. The package DecisionMaker contains the two packages for decision making, ManualSelector and AutomaticSelector. The ManualSelector can be used for items in the agenda that need to be selected by the user. ManualSelector therefore initiates a request to the GUI in order to obtain a decision from the user. AutomaticSelector can be used for items in the agenda that need to be selected automatically. AutomaticSelector is able to use different methods for selection. The developer is able to edit or add methods for the AutomaticSelector.

6.4.1.2 Package DataCreator

DataCreator is responsible for mapping data into right format for the constraint system. It is responsible for the translation of domain knowledge into constraints. For creating constraints it contains the package ConstraintCreator.
6.4.1.3  **ConfigurationCreator**  
The class ConfigurationCreator contains a series of important fields. It contains the interface to the constraint system as well as the domain knowledge of the configuration. The interface to the constraint system is handled by direct access to the library file of CS3. An additional field context is added that beholds the current state of the configuration. The domain knowledge is modeled in three different lists, namely goals, components and choices. They contain all ConfigObjects that are included in the configuration.

6.4.2  **System Behavior**

6.4.2.1  **Creation of Domain Knowledge**  
Initially the domain knowledge needs to be modeled by the domain expert and translated into constraints. As described in section 6.4.1.2 the DataCreator is responsible for this task. Basically the translation consists of three steps:

1. Creation of the knowledge library with the domain knowledge
2. Integration of the knowledge library in the configurator
3. Translation of the knowledge library in constraints in the CS

The first step is normally performed by an individual of the specific domain, the domain expert in collaboration with the developer. The developer needs to map the domain concepts to the generic concepts used in the generic configuration framework. The domain expert then states the relations between the domain concepts and a format for knowledge representation is arranged. The domain expert then creates the knowledge library using this representation. The knowledge library represents a major part in modeling CSPs. In this study the knowledge library was realized in simple class containing the ConfigObjects and their relations. Future work should incorporate a system for knowledge management. The second step is the integration of the knowledge library in the configurator. When external tools are used data wrappers or interfaces need to be developed by the developer, using the classes represented as ConfigObjects. In the case of this study the knowledge library could be used directly since it is represented as a VB.Net class. The last and third step is the translation of the knowledge library into constraints. The method CreateData of the DataCreator takes all ConfigObjects and creates the corresponding constraints for the constraint system and loads them automatically in the constraint system. Features for keeping constraints persistent in the constraint system are available but have not been used in this study and are left out for future work.
6.4.2.2 Global Software Control

To give an overview of the configurators’ global control flow this section describes the key concept how the configuration is processed. There are different strategies how to obtain a valid configuration. The following approach introduces a simple strategy that guarantees to obtain a valid configuration, albeit the approach may not represent the complete set of features available by the framework.

The four key steps of the configurator are:

1. Activation of the top goal of the configuration.
2. Application of all constraints, which may include activation of sub-goals, choices, perhaps components and exclusion of goals and components.
3. Creation of the agenda.
4. While the agenda is not empty:
   - pick one choice from the agenda
   - select a option for this choice
   - update the agenda
   - present changes to the user

The following figure also illustrates the key steps.

![Figure 17 – Key Steps of the Configuration Process](image)

These key steps are implemented in the configurator. The workflow of the configurator is therefore completely dependent on the agenda. The agenda processor is explained in the next section

6.4.2.3 AgendaProcessor

The AgendaProcessor is responsible for creating the agenda and contains the DecisionMaker package. The method ProcessAgenda creates the list of open agenda items. Open agenda items are
choices which are implied as active and not implied as satisfied. Therefore every choice contained in the configuration needs to be assessed if it is active, and when it is active it needs to be assessed if it is satisfied. Every assessment results in a function call to the constraint system as depicted in figure 18. Note that the figure contains only the calls for one choice. The workflow of the configurator depends on how goals and choices are modeled by the domain expert. Therefore an arbitrary workflow can be achieved, without changing the code of the AgendaProcessor.

![Figure 18 - Communication AgendaProcessor and Constraint System](image)

6.4.2.4 DecisionMaker

The DecisionMaker is responsible for selecting a valid option for open agenda items. In order to select a valid option for an open agenda item two methods are available. The ManualSelector is for choices of the type user choices. It initiates a request to the user to pick a valid option for the choice. A list of valid options is presented to the user and the user has to select one of them in order to satisfy the choice. The AutomaticSelector is for choices of the type automatic choices. A valid option is selected automatically. Different methods for automatic selection can be implemented here. The default is to pick the first available option for the choice. Future work should consider using heuristics or random selections.

6.4.2.5 VariableCreator

The class VariableCreator is responsible for the mapping of objects of the object-oriented world of VB.Net to the logic world of the constraint system. The class contains two fields, the name of the variable, and the corresponding variable in the constraint system. For the creation of the variable in the CS it uses the interface to the constraint and provides the function CreateVariable.
6.4.2.6 ConstraintCreator

The class ConstraintCreator has two methods. The method CreateConstraints creates the constraints for the constraint system according to the domain knowledge provided by the domain expert. The method InitCS is responsible for instantiating all constraints before a configuration is started.

6.4.2.7 ConfigurationCreator

The ConfigurationCreator is the control entity for the configuration process. It is called by the configurator when a user requests a configuration. The user requirements are passed as parameter. The general goals of the user are then activated in the constraint system. The AgendaProcessor is called and creates the agenda. The DecisionMaker is then called until no more open agenda items are available. The result is determined in the ConfigurationCreator and presented to the user. Figure 20 depicts a conceptual workflow. Note that multiple iterations are normally necessary to complete the agenda.
7 Vehicle Configurator

In order to assess the functionality of the generic configuration framework two domain specific configurators have been created. Chapter 7 presents vehicle configuration and section 8 presents training planning. The characteristics of both application domains are elaborated, examples for structuring the domain knowledge are given. The global software control for each application is presented. Then the extent is discussed to which the requirements given in chapter 2.7 have been achieved.

7.1 Domain Characteristics

In chapter 6 the generic configuration framework is introduced. In order to implement a domain specific solution two steps are necessary: the creation of the domain specific knowledge library and the adaption of the global software control to the needs of the application domain. The following section describes characteristics in vehicle configuration and how the domain knowledge was modeled.

7.1.1 Domain Knowledge

The overview of current vehicle configurators in section 3.1 showed that there is no sufficient support for the user in guiding him through the process of finding the best car according to his goals. Therefore it has been decided to model goals with a refinement structure, as described in FR1. Three general goals have been selected: Weekday vehicle, Family vehicle, Joyride vehicle. For each of the goals a refinement structure with more detailed goals is given as shown in table 2.

<table>
<thead>
<tr>
<th>General Goal</th>
<th>Refinement Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekday vehicle</td>
<td>Going to Work, Going to School, Professional Car</td>
</tr>
<tr>
<td>Family Vehicle</td>
<td>Large Family, Small Family</td>
</tr>
<tr>
<td>Joyride Vehicle</td>
<td>With my Partner, With my Buddies</td>
</tr>
</tbody>
</table>

Table 2 - General Goals of Vehicle Configuration

Further refinement of goals is possible, exemplary “Professional Car” was resolved in two sub goals in order to concretize the requirements. The following table shows the sub goal structure of “Professional Car” and the resolution of the refinement goals from table 2.
User restrictions shall be available, too. Exemplary for user restrictions shall be the purchase price of the car. In order to model the price restrictions of the user a price segment has been assigned to each car. There are segments from 1 to 11 which represent a virtual monetary value. The next step is the definition of the components. In order to stay as near as possible to a real world problem it has been decided to model different cars of brands of the VW Group. Due to its 12 different brands the VW Group is very suitable for a tool as we anticipate it since there are a lot if similar models that only differ in details and the user needs to be assisted in his decisions. Note that only a small subset of cars from the brands VW and Porsche have been modeled. Future work should consider modeling the full product range. Table 4 gives an overview of cars that have been chosen. Note that in this exemplary application cars are modeled as an elementary component. Normally a car would be a collection of components. Since the configuration of components is well done in current configurators, this functionality has been omitted. In order to create constraints the elementary components need to be linked to goals and restrictions. Exemplary dependencies have been created and are also shown in table 4. Remark that the table represents a conceptual overview on how knowledge in the vehicle domain can be represented. The next step is to transform this conceptual knowledge into actual constraints in the knowledge library format of the generic configuration framework. Therefore the DataCreator class is available that provides methods for describing the knowledge. The description looks like this:

familyVehicle.SetImplication(safety)
familyVehicle.SetExclude(seats2)
familyVehicle.SetImplication(typeNotVanOrTruck)
The created constraints then look like this:

*Imp.:* familyVehicle.active => safety.active

*Imp.:* familyVehicle.active => typeNotVanOrTruck.active

*Imp.:* familyVehicle.active => NOT seats2.active

<table>
<thead>
<tr>
<th>Component</th>
<th>Component Goals</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW Polo</td>
<td>lowStorage, safety, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 1</td>
</tr>
<tr>
<td>VW Golf</td>
<td>lowStorage, safety, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 2</td>
</tr>
<tr>
<td>VW EOS</td>
<td>lowStorage, safety, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 2</td>
</tr>
<tr>
<td>VW Passat</td>
<td>Safety, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 6</td>
</tr>
<tr>
<td>VW Phaeton</td>
<td>Safety, LuxuryComfortAndStyle, HigherInnerSpace</td>
<td>Price segment 9</td>
</tr>
<tr>
<td>Porsche Panamera</td>
<td>Safety, LuxuryComfortAndStyle, HigherInnerSpace</td>
<td>Price segment 9</td>
</tr>
<tr>
<td>VW Tiguan</td>
<td>Safety, StandardComfortAndStyle, HigherInnerSpace</td>
<td>Price segment 5</td>
</tr>
<tr>
<td>VW Tuareg</td>
<td>Safety, LuxuryComfortAndStyle, HigherInnerSpace</td>
<td>Price segment 11</td>
</tr>
<tr>
<td>VW Scirocco</td>
<td>lowStorage, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 4</td>
</tr>
<tr>
<td>Porsche Cayenne</td>
<td>Safety, LuxuryComfortAndStyle, HigherInnerSpace</td>
<td>Price segment 9</td>
</tr>
<tr>
<td>Porsche 911</td>
<td>lowStorage, Seats2, LuxuryComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 10</td>
</tr>
<tr>
<td>VW Transporter</td>
<td>typeVan, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 3</td>
</tr>
<tr>
<td>VW Amarok</td>
<td>typePickUp, StandardComfortAndStyle, LowerInnerSpace</td>
<td>Price segment 7</td>
</tr>
</tbody>
</table>

Table 4 - Components for Vehicle Configuration

The last part is the definition of the choices. For choices two different modes are available, automatic decision making and manual decision making. For this application it has been defined that only manual decisions are desired. A choice is defined as follows:

```
Dim choicePolo As New Choice("choicePolo", False)
```

The second parameter of the constructor determines if automatic decision making is activated or not. The “false” implicates manual decision making.
In this example we only have one type of choices, the choice for a concrete model. Normally there would be choices for optional features, colors etc. In this prototype all components are concrete models and therefore every choice has only one option, namely the option for the respective model. The options are set like in this example:

```
choicePolo.SetOption(componentPolo)
```

Future work should consider choices for colors, design, optional features and so on. All declared goals, components, restrictions and choices are available in the DataCreator in the Appendix. A list of the constraints generated by the DataCreator can be found in the appendix as well.

### 7.1.2 Global Software Control

The software control for vehicle configuration has an emphasis on user interaction, although some processes are automatic. A simple GUI has been created in order to simulate user interaction. Figure 21 shows the main page of the configurator. The overview contains the general goals available in the configurator and a field for the maximum price, a restriction the user is able to state. The user has to select one of the general goals and may state a maximum price. After pressing the continue button a new screen is shown.

![VW Group Configurator](image)

**Figure 21 - Overview Screen GUI**

As shown in figure 22, the user gets options to refine his general goal chosen in the overview screen. In this case he may specify if his family is small or large.
After clicking the next step button the AgendaProcessor creates the agenda and an open agenda item is created. Since the manual decision making was defined in the domain knowledge, the user has to select his favorite car.

In the prototype a description is then presented for the chosen option as depicted in figure 24. Future work should extend the domain knowledge of optional features, colors etc. and then offer
choices for further configuration of the desired vehicle. The workflow is, as explained before, completely depending on how the domain knowledge is modeled.

![Image of a car configuration interface](image)

**Figure 24 - Description of the Chosen Car**

### 7.2 Evaluation

In this section, the implemented prototype for vehicle configuration will be evaluated based on the requirements specification presented in chapter 2.7. It will be shown that almost all requirements have been met by the prototypical implementation. Several tests runs have been performed with different user input. At first the validity of the configurator will be shown and then the results are evaluated for each requirement stated in chapter 2.7.

#### 7.2.1 Validity

In order to show that the prototype meets the requirements for validity defined in chapter 4.4.4. We need to show that the configuration holds the following three requirements:

1. \( Configuration \models User \text{ Goals} \)
2. \( Configuration \cup User \text{ Restrictions} \not\models \bot \)
3. \( Configuration \cup User \text{ Properties} \cup Domain \text{ Constraints} \not\models \bot \)

The evaluation is based on a test run with exemplary user input. Then it is shown which constraints have been triggered and if the solution is correct.
The goal family vehicle has been chosen and the user restriction maximum price has been set to segment 5. The test run has been performed and the following debug messages have been retrieved from the configurator:

1. Activated: familyVehicle.active
2. Activated: userRestrictionprice5.active
3. Derivable by context:
4. familyVehicle.active = T
5. safety.active = T
6. seats2.active = F
7. typeNotVanOrTruck.active = T
8. userRestrictionprice5.active = T
9. choicePolo.active = T
10. choiceGolf.active = T
11. choiceTransporter.active = F

As depicted in line 2 and 4 the user input has been set accordingly to the desired parameters. Since no inconsistencies are detected, the validity requirements 2 and 3 are met, because all constraints are instantiated now. Lines 7 and 11 show that the user input is set to “T”, the notation for true in debug messages. In order to verify if the domain constraints are obeyed we need to have a look at the domain knowledge. For the goal family vehicle the following constraints have been defined:

- familyVehicle.active \(\Rightarrow\) safety.active
- familyVehicle.active \(\Rightarrow\) typeNotVanOrTruck.active
- familyVehicle.active \(\Rightarrow\) NOT seats2.active

Line 8 shows that the goal safety has been set to true. Line 9 shows that typeNotVanOrTruck.active is also set to true. Line 10 says that seats2.active is false. This proves that all constraints have been applied correctly. Line 12, 13 and 14 say that choices for Polo, Golf are active and Transporter is not active. In order to verify if this is correct we need to have a look at the refinement goals. For safety the following constraints have been defined:

- safety.active \(\Rightarrow\) choicePolo.active
- safety.active \(\Rightarrow\) choiceGolf.active
- safety.active \(\Rightarrow\) choiceEos.active
- safety.active \(\Rightarrow\) choicePassat.active
- safety.active \(\Rightarrow\) choicePhaeton.active
- safety.active \(\Rightarrow\) choiceTiguan.active
- safety.active \(\Rightarrow\) choicePanamera.active
- safety.active \(\Rightarrow\) choiceTuareg.active
- safety.active \(\Rightarrow\) choiceCayenne.active
- safety.active \(\Rightarrow\) NOT componentScirocco.active
- safety.active \(\Rightarrow\) NOT componentPorsche911.active
- safety.active \(\Rightarrow\) NOT componentAmorok.active
- safety.active \(\Rightarrow\) NOT componentTransporter.active

For typeNotVanOrTruck the following constraints have been defined:

- typeNotVanOrTruck.active \(\Rightarrow\) choicePolo.active
- typeNotVanOrTruck.active \(\Rightarrow\) choiceGolf.active
- typeNotVanOrTruck.active \(\Rightarrow\) choiceEos.active
However, the user restriction maximum price has been set to segment 5. Therefore the following constraints have been activated:

\[
\text{Imp.: userRestrictionprice5.active} \implies \text{choicePolo.active} \\
\text{Imp.: userRestrictionprice5.active} \implies \text{choiceGolf.active} \\
\text{Imp.: userRestrictionprice5.active} \implies \text{choiceTransporter.active} \\
\text{Imp.: userRestrictionprice5.active} \implies \text{choiceScirocco.active} \\
\text{Imp.: userRestrictionprice5.active} \implies \text{choiceTiguan.active}
\]

It can be concluded, that line 12, Polo, and line 13, Golf, of the debug messages are correct, because both are implied by the sub goals and the user restriction. Since Transporter is excluded by the goal typeNotVanOrTruck it can be concluded that this is correct as well. That leads to prove that the validity requirement 1 is met. Therefore it can be concluded that for the given user input the configurator works correctly.

7.2.2 Achievements

The following section evaluates each requirement and states if and to what extent the requirement has been met.

FR1: Goals with structure

The user is able to state general goals and also a refinement structure is available. Therefore the requirement is met.

FR2: User Properties

The conceptualization presented in chapter 4.1 identified only one user property, the annual mileage. The annual mileage influences the running costs of a car depending on fuel consumption, insurance and tax costs. Since the modeling of the components has been performed on a coarse-grained basis, the annual mileage could not be considered and the requirement has been left out for future work.
**FR3: User Restrictions**
The user is able to state his maximum price and is only offered cars that are in his budget. Therefore the requirement is met.

**FR4: Automatic Decision Maker**
It has been decided that automatic decisions are not necessary for the scope of the prototypical implementation. Automatic decisions should be considered when optional features are offered.

**FR5: Manual Decision Maker**
The user is able to select a vehicle according to his stated requirements. Therefore the requirement is met.

**FR6: Component Structure**
Components have been created and linked to goals. However elementary components have been used exclusively since the scope has been reduced to a coarse-grained basis. Therefore the requirement is only met partially. Albeit it is assumed that structured components are working, too since goal structures are available.

**FR7: Representation of Domain Knowledge**
Without proper representation of domain knowledge valid configurations would not be possible. Also the test of the prototype showed that the results that are feasible. Therefore it is concluded that the way of representing knowledge makes sense and domain experts are able to use it. Therefore the requirement is met. Albeit it needs to be said that only a subset of the constraints has been modeled that apply in the real world. It can be assumed that complexity of modeling constraints for a significant higher number of dependencies requires for tool support.

**FR8: Creation of a Generic Configuration Framework**
The generic classes have been used to implement the domain specific prototype and have proven to be able to model all necessary concepts for vehicle configuration. Since all functional requirements are depending on the generic configuration framework and most of the requirements are met it can be concluded that the requirement is met as well.

**FR9: Constraint System**
A requirements analysis has been performed and an interface has been defined. A constraint system has been chosen and since major parts of the functionality depend on the constraint system it can be assumed that the constraint system meets the requirements.

**NFR1: Performance**
The performance of the configurator was not measured explicitly. It can be said that there are no significant waiting times during configuration. It is assumed that response times are below one second. Since the number of components, goals and constraints are below the numbers given for large configurations, the performance for large systems cannot be determined. However it is assumed that there are enough performance resources therefore it is concluded that the requirement is met.

**NFR2: Modularity**
The architecture clearly separates each of the proposed layers. However it has not been tested to exchange a layer. It is assumed that the requirement is met.

**NFR3: Extendibility**
The integration of new goals, components, choices and constraints is done by specializing the existing classes. For smaller configurations this presents a fast and practicable way to realize desired functionality. However it needs to be said that for larger configurations it is assumed to get complicated to keep consistency. It should be considered to add tool support for extension of the generic framework. In the case of the prototype it can be concluded that the requirement is met.

**NFR4: Usability**
The configuration follows a simple approach of reducing the number of options for the user. The user is never faced with decisions with many options. Also when it comes to the decision for a car only suitable cars are displayed. Therefore the requirement is met.
8 Training Planning Configurator

The second domain application is training planning for gyms. The goal is to extend an existing website where user can plan, track and analyze their training. Therefore the GUI part has been completely omitted.

8.1 Domain Characteristics

The following two sections describe the characteristics of training planning, separated in domain knowledge and the global software control.

8.1.1 Domain Knowledge

The domain knowledge for training planning has been created by a sports scientist. Therefore he got an introduction on how to model the knowledge library for the generic configuration framework. At first like in the vehicle configurator a goal structure has been defined. Two general goals are available: Muscle Gain and Health. For each of the general goals a refinement is given. As shown in the conceptualization in chapter 4.2 every training goal is divided in multiple unit goals.

<table>
<thead>
<tr>
<th>General Goal</th>
<th>Refinement Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Goal Muscle Gain</td>
<td>Unit Goal Upper Arms, Unit Goal Breast</td>
</tr>
<tr>
<td>Training Goal Health</td>
<td>Unit Goal Abdominal, Unit Goal Lower Back</td>
</tr>
</tbody>
</table>

Table 5 - Refinement of General Training Goals

In contrary to vehicle configuration, the refinement structure that follows for training planning is significant larger. Every unit is refined in muscle groups which contain different muscles. Details are shown in table 6. Remark that only a subset of unit goals has been modeled. A regular training plan would contain about ten to fifteen unit goals.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Component Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Goal Upper Arms</td>
<td>Component Goal Biceps, component Goal Triceps, component Goal Brachialis</td>
</tr>
<tr>
<td>Unit Goal Breast</td>
<td>Component Goal Pectoralis Major Sternal</td>
</tr>
<tr>
<td>Unit Goal Abdominal</td>
<td>Component Goal Obliques, component Goal Rectus Abdominis</td>
</tr>
<tr>
<td>Unit Goal Lower Back</td>
<td>Component Goal Rector Spinae, component Goal Quadratus</td>
</tr>
</tbody>
</table>
In contrary to vehicle configuration the training planning configurator exploits the possibility of adding a structure to components. At first the elementary components are defined. Therefore a set of exercises has been defined. In table 7 they are listed with their goals already assigned. The assignment works as described for vehicle configuration. Note that the component “Isomatte” has no goal assigned, but the component is assigned to some exercises. By this logic it is expressed that the component is necessary for the corresponding exercise but has no goal it fulfills. In this way aggregate components can be defined.

<table>
<thead>
<tr>
<th>Component</th>
<th>Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brustpresse</td>
<td>component Goal Triceps, component Hand Flexors, Component Goal Pectoralis Major Sternal, Goal Brachialis</td>
</tr>
<tr>
<td>Ruderzug</td>
<td>component Goal BRACHIALIS, component Goal LATISSIMUS</td>
</tr>
<tr>
<td>Rotator</td>
<td>component Goal OBLIQUES</td>
</tr>
<tr>
<td>Bauchtrainer</td>
<td>component Goal RECTUS_ABDOMINIS</td>
</tr>
<tr>
<td>Rueckentrainer</td>
<td>component Goal HAMSTRINGS, component Goal RECTOR_SPINAES</td>
</tr>
<tr>
<td>Rumpfbeuge seitlich</td>
<td>component Goal OBLIQUES, component Goal QUADRATUS_LUMBORUM</td>
</tr>
<tr>
<td>Butterfly eng</td>
<td>component Goal PECTORALIS_MAJOR_STERNAL</td>
</tr>
<tr>
<td>Bizeps curl</td>
<td>component Goal BICEPS</td>
</tr>
<tr>
<td>Trizeps curl</td>
<td>component Goal TRICEPS</td>
</tr>
<tr>
<td>Russian Twist sitzend</td>
<td>component Goal OBLIQUES, Isomatte</td>
</tr>
<tr>
<td>Crunches</td>
<td>component Goal RECTUS_ABDOMINIS, Isomatte</td>
</tr>
<tr>
<td>Rueckenstrecker</td>
<td>component Goal RECTUS_FEMORIS, component Goal RECTOR_SPINAES.</td>
</tr>
<tr>
<td>Seitstuetz Hueftheben</td>
<td>component Goal QUADRATUS_LUMBORUM, component Goal OBLIQUES.active, Isomatte</td>
</tr>
<tr>
<td>Liegestuetz</td>
<td>component Goal TRICEPS, component Goal HAND_FLEXORS, Isomatte</td>
</tr>
<tr>
<td>Isomatte</td>
<td></td>
</tr>
</tbody>
</table>

User properties have not been used for vehicle configuration. Properties are defined as attributes of the user which change infrequently. In the training planning domain a lot of properties have been identified in the conceptualization. An overview of them is given in chapter 4.2. In the prototype
two types have been implemented: Injury and BMI. The properties result in additional implication of goals or in exclusion of goals. The description looks like this:

- `userPropertyBMIover30.SetImplication(unitGoalLower_Back)`
- `userPropertyBMIover30.SetExclude(componentGoalIsoMatte)`
- `userPropertyHandInjury.SetExclude(componentGoalHAND_FLEXORS)`
- `userPropertyKneeInjury.SetExclude(componentGoalRECTUS_FEMORIS)`

The created constraints then look like this:

- Imp.: `userPropertyBMIover30.active => unitGoalLower_Back.active`
- Imp.: `userPropertyBMIover30.active => NOT componentGoalIsoMatte.active`
- Imp.: `userPropertyHandInjury.active => NOT componentGoalHAND_FLEXORS.active`
- Imp.: `userPropertyKneeInjury.active => NOT componentGoalRECTUS_FEMORIS.active`

For training planning choices needed to be defined since the automatic decision maker shall be used since most users do not have the necessary knowledge to select the proper exercises for their goals. A choice is always necessary when there is more than one option that satisfies the goal. In case of training planning the choices are defined on the basis of the muscle groups. Therefore choices for each muscle group have been created like the following:

- `choiceHAND_FLEXORSComponents.SetOption(componentRuderzug)`
- `choiceHAND_FLEXORSComponents.SetOption(componentBizepscurl)`
- `choiceHAND_FLEXORSComponents.SetOption(componentTrizepscurl)`
- `choiceHAND_FLEXORSComponents.SetOption(componentRussion_Twist_sitzend)`
- `choiceHAND_FLEXORSComponents.SetOption(componentLiegestuetz)`

The resulting constraint looks like this:

- Imp.: `choiceHAND_FLEXORSComponents.active => componentRuderzug.active or componentBizepscurl.active or componentTrizepscurl.active or componentRussion_Twist_sitzend.active or componentLiegestuetz.active`

The automatic decision maker is then allowed to pick one of the options. In order to optimize configurations methods like heuristics can be used.

### 8.1.2 Global Software Control

The software control for training planning has an emphasis on automatic decision making since most users do not have the necessary knowledge to select the proper exercises for their goals. Since a GUI was not part of this prototype the user input has been set by using a command line tool. Then the configuration was processed completely automatic. The configuration result for the input “Training Goal Musclegain” looks like this:

**Active Components:**

- `componentRuderzug is active`
- `componentBizepscurl is active`
If the input is “Training Goal Musclegain” and the user property “Hand Injury” is activated, the component “Liegestuetz” is removed, because the component implies the use of the “Hand Flexors”. Therefore the component is replaced by the component “Butterfly eng”.

Active Components:
-------------------
componentRuderzug is active
componentButterfly_eng is active
componentBizepscurl is active
componentTrizepscurl is active
-------------------

When using the inputs “Training Goal Musclegain” and “BMIover30” all components that use the component “Isomatte” are substituted. The result then looks like that:

Active Components:
-------------------
componentRuderzug is active
componentRotator is active
componentRumpfbeuge_seitlich is active
componentBizepscurl is active
componentTrizepscurl is active
componentRueckenstrecker is active
-------------------

8.2 Evaluation

In this section, the implemented prototype for training planning will be evaluated based on the requirements specification presented in chapter 2.7. It will be shown that almost all requirements have been met by the prototypical implementation. Several test runs have been performed with different user input. At first the validity of the configurator will be shown and then the results are evaluated for each requirement stated in chapter 2.7.

8.2.1 Validity

The evaluation for validity will be performed similar to the evaluation of the vehicle configurator in chapter 7.2.1. Albeit there are some differences, since training planning is configured in automatic mode, the result of the configuration is shown as selected by the automatic decision making mechanism. Therefore the debug messages are slightly different.
For training planning the user goal trainingGoalHealth and the user property BMIover30 have been chosen. A test run has been conducted with the following result:

(1)------------------- 
(2)Active Goals: 
(3)------------------- 
(4)trainingGoalHealth is active 
(5)unitGoalAbdominal is active 
(6)unitGoalLower_Back is active 
(7)componentGoalOBLIQUES is active 
(8)componentGoalRECTUS_ABDOMINIS is active 
(9)componentGoalRECTOR_SPINAE is active 
(10)componentGoalQUADRATUS_LUMBORUM is active 
(11)unitGoalLower_Back is active 
(12)componentGoalRECTUS_FEMORIS is active 
(13)userPropertyBMIover30 is active 
(14)------------------- 
(15)Active Choices: 
(16)------------------- 
(17)choiceOBLIQUESComponents is active 
(18)choiceRECTUS_ABDOMINISComponents is active 
(19)choiceRECTOR_SPINAEComponents is active 
(20)choiceQUADRATUS_LUMBORUMComponents is active 
(21)choiceRECTUS_FEMORISComponents is active 
(22)------------------- 
(23)Active Components: 
(24)------------------- 
(25)componentRotator is active 
(26)componentBauchtrainer is active 
(27)componentRumpfbeuge_seitlich is active 
(28)componentRueckenstrecker is active 
(29)------------------- 

Line 4 and 13 prove that the user input has been set correctly. Therefore validity requirement 2 and 3 are fulfilled. The domain knowledge defines the following constraints for trainingGoalHealth:

\[ \text{Imp.}: \text{trainingGoalHealth.active} \Rightarrow \text{unitGoalAbdominal.active} \]
\[ \text{Imp.}: \text{trainingGoalHealth.active} \Rightarrow \text{unitGoalLower_Back.active} \]

Therefore line 5 and 6 are correct. The implied unit goals have the following constraints:

\[ \text{unitGoalAbdominal:} \]
\[ \text{Imp.}: \text{unitGoalAbdominal.active} \Rightarrow \text{componentGoalOBLIQUES.active} \]
\[ \text{Imp.}: \text{unitGoalAbdominal.active} \Rightarrow \text{componentGoalRECTUS_ABDOMINIS.active} \]
\[ \text{unitGoalLower_Back:} \]
\[ \text{Imp.}: \text{unitGoalLower_Back.active} \Rightarrow \text{componentGoalRECTOR_SPINAE.active} \]
\[ \text{Imp.}: \text{unitGoalLower_Back.active} \Rightarrow \text{componentGoalQUADRATUS_LUMBORUM.active} \]

Hence, lines 7, 8, 9 and 10 are valid. Since component Goals are elementary functions training planning a choice for them is necessary. Therefore lines 17 to 21 are correct. Now the selected components need to be evaluated. Therefore the constraints of the activated components are listed:

\[ \text{componentRotator:} \]
\[ \text{Imp.}: \text{componentRotator.active} \Rightarrow \text{componentGoalOBLIQUES.active} \]
\[ \text{componentBauchtrainer:} \]
\[ \text{Imp.}: \text{componentBauchtrainer.active} \Rightarrow \text{componentGoalRECTUS_ABDOMINIS.active} \]
\[ \text{componentRumpfbeuge_seitlich:} \]
\[ \text{Imp.}: \text{componentRumpfbeuge_seitlich.active} \Rightarrow \text{componentGoalOBLIQUES.active} \]
\[ \text{Imp.}: \text{componentRumpfbeuge_seitlich.active} \Rightarrow \text{componentGoalQUADRATUS_LUMBORUM.active} \]
\[ \text{componentRueckenstrecker:} \]
\[ \text{Imp.}: \text{componentRueckenstrecker.active} \Rightarrow \text{componentGoalRECTOR_SPINAE.active} \]
\[ \text{Imp.}: \text{componentRueckenstrecker.active} \Rightarrow \text{componentGoalRECTUS_FEMORIS.active} \]
The component in line 25 satisfies the choice in line 17. The component in line 26 satisfies the choice in line 18. The component in line 27 satisfies the choice in line 17 and 20. The component in line 28 satisfies the choice in line 19 and 21. The user property BMIover30 added the following constraints to the configuration:

```
userPropertyBMIover30:
  Imp.: userPropertyBMIover30.active => unitGoalLower_Back.active
  Imp.: userPropertyBMIover30.active => NOT componentGoalIsoMatte.active
```

Since no component has a constraint on the componentGoalIsoMatte and unitGoalLower_Back is set to also true as stated in line 6 it can be concluded that all goals are met and therefore the validity requirement 1 fulfilled.

### 8.2.2 Achievements

**FR1: Goals with structure**
The user is able to state general goals and also a refinement structure is available. Therefore the requirement is met.

**FR2: User Properties**
There have been two types of user properties implemented: BMI and Injuries. The user can specify if his BMI is over 30 and if he has a hand or a knee injury. The training plan is processed accordingly to these properties then. Therefore the requirement is met.

**FR3: User Restrictions**
No user restrictions have been modeled by the domain expert. Therefore no restrictions have been used, but for example blacklisting of exercises could be very easily integrated by the setExclusion function of the DataCreator. Nevertheless the requirement has not been implemented. Future work should consider time constraints for the training or training types for the exercises.

**FR4: Automatic Decision Maker**
The automatic decision maker is used for selecting the respective exercise for the users’ goals. Therefore the requirements have been met. Although the method for the selection should be extended by heuristics in future works for optimized training plans.

**FR5: Manual Decision Maker**
The user is able to select a vehicle according to his stated requirements. Therefore the requirement is met.
FR6: Component Structure
Components have been created and linked to goals. Also some components have been attached a structure. Therefore the requirement is met.

FR7: Representation of Domain Knowledge
See chapter 7.3.

FR8: Creation of a Generic Configuration Framework
See chapter 7.3.

FR9: Constraint System
See chapter 7.3.

NFR1: Performance
See chapter 7.3.

NFR2: Modularity
See chapter 7.3.

NFR3: Extendibility
See chapter 7.3.

NFR4: Usability
Since the training planning is automatic the usability depends completely on the GUI. As no GUI was available the usability for the training planning configurator was not evaluated.

9 Conclusion and Future Work

This thesis aimed at developing a tool that supports the user in complex, extensive and time consuming configuration tasks as explained in chapter 1. Chapter 2 stated six challenges this thesis wants to meet. The challenges have been resolved into concrete requirements for a new tool in chapter 2.7. The analysis of current web-based vehicle configurators in chapter 3.1 showed that none of the currently available tools meet these requirements. This motivated this thesis to develop
Conclusion and Future Work

a framework for new tools that have an extended functionality according to the requirements that have been determined. At the beginning the theoretical foundation for applying constraint satisfaction to configuration has been examined and a general approach has been proposed in chapter 4. Chapter 5 presents architecture for configurators using three layers. The requirements for the constraint system have been determined and an interface definition was created. In chapter 6 the concrete approach how configuration can be implemented as CSP has been introduced and significant system characteristics are presented. Afterwards the created software systems have been presented and evaluated in chapter 7 and 8 with remarkable results. Two prototypes have been created for domains with different characteristics. Vehicle configuration represents a classic component-based system with lots of components that have dependencies. The prototypes showed how decision guidance can be provided by connecting the components to a goal structure and enabling the user to state his goals. Training Planning can be represented as a component-based system as well, but the emphasis is on a large refinement structure of the general goals that leads to concrete goals and eventually to components. The training planning configurator shows the possibilities of automatic configuration according to requirements of the user. Nevertheless this thesis focused on creating the foundation for future work. In order to exploit the full potential of the framework further work is necessary. An overview of future work is given in the next section.

9.1 Extension of the prototypes

In this thesis only two exemplary prototypes have been built with a reduced set of goals, components, choices and constraints. The prototypes should be extended to a comprehensive level of detail in order to prove the practical benefit of the tools. For the vehicle configurator, the GUI should be extended in order to improve usability. Also the configuration of subjective components like colors, exterior and interior design should be added.

9.2 Knowledge Management

The knowledge representation is currently performed via a class in VB.net. When creating large configurations that representation will get confusing and might be inconsistent. Therefore a tool should be developed that allows the domain expert to enter knowledge without having a background in computer science. The created knowledge library should be checked for consistency and plausibility.
9.3 Extension of the CSP Model

In chapter 4.2 the general approach for applying CSP to configuration is presented. As described in chapter 6.2 only a simplified approach has been integrated in the framework. Future work should extend the simplified approach to the full functionality provided by constraint satisfaction and the constraint system. Future functions of the constraint systems have already been defined in chapter 5.4.4.3.

9.4 Explanations

Explanations via the ATMS of the constraint system are possible. However explanations have not been included in this thesis. It can be concluded that explanations on why specific components are contained in a configuration are helpful for the user. Although a lot of information is created by the ATMS, significant information needs to be identified and a clear representation needs to be found in order to gain benefit for the user.
Appendix

Domain Knowledge Vehicle Configuration

Public Class DataCreatorVehicle
    Sub createVehicleData(ByVal goals As List(Of Goal), ByVal components As List(Of Component), ByVal choices As List(Of Choice))

    '############################################################################
    'Define Goals
    Dim familyVehicle As New Goal("familyVehicle")
    Dim safety As New Goal("safety")
    Dim seats2 As New Goal("seats2")
    Dim typeVanOrTruck As New Goal("typeVanOrTruck")
    Dim typeNotVanOrTruck As New Goal("typeNotVanOrTruck")
    Dim weekdayVehicle As New Goal("weekdayVehicle")
    Dim lowComfortAndStyle As New Goal("lowComfortAndStyle")
    Dim joyrideVehicle As New Goal("joyrideVehicle")
    Dim highComfortAndStyle As New Goal("highComfortAndStyle")
    Dim goingtoWork As New Goal("goingtoWork")
    Dim goingtoSchool As New Goal("goingtoSchool")
    Dim price40 As New Goal("price40")
    Dim LowStorage As New Goal("LowStorage")
    Dim professionalCar As New Goal("professionalCar")
    Dim onlyBuddies As New Goal("onlyBuddies")
    Dim typeVan As New Goal("typeVan")
    Dim typePickUp As New Goal("typePickUp")
    Dim LowInnerSpace As New Goal("LowInnerSpace")
    Dim highInnerSpace As New Goal("highInnerSpace")
    Dim typeSuvOrCrossover As New Goal("typeSuvOrCrossover")
    Dim cityTransportation As New Goal("cityTransportation")
    Dim offroadTransporation As New Goal("offroadTransporation")
    Dim smallFamily As New Goal("smallFamily")
    Dim LargeFamily As New Goal("LargeFamily")
    Dim onlyCrush As New Goal("onlyCrush")
    Dim offroadAdventures As New Goal("offroadAdventures")

    'User Restrictions
    Dim userRestrictionprice1 As New Goal("userRestrictionprice1")
    Dim userRestrictionprice2 As New Goal("userRestrictionprice2")
    Dim userRestrictionprice3 As New Goal("userRestrictionprice3")
    Dim userRestrictionprice4 As New Goal("userRestrictionprice4")
    Dim userRestrictionprice5 As New Goal("userRestrictionprice5")
    Dim userRestrictionprice6 As New Goal("userRestrictionprice6")
    Dim userRestrictionprice7 As New Goal("userRestrictionprice7")
    Dim userRestrictionprice8 As New Goal("userRestrictionprice8")
    Dim userRestrictionprice9 As New Goal("userRestrictionprice9")
    Dim userRestrictionprice10 As New Goal("userRestrictionprice10")
    Dim userRestrictionprice11 As New Goal("userRestrictionprice11")
    Dim userRestrictionprice12 As New Goal("userRestrictionprice12")

    'add all goals to a list
    goals.Add(familyVehicle)
    goals.Add(safety)
    goals.Add(seats2)
    goals.Add(typeVanOrTruck)
    goals.Add(typeNotVanOrTruck)
    goals.Add(weekdayVehicle)
    goals.Add(lowComfortAndStyle)
    goals.Add(joyrideVehicle)
    goals.Add(highComfortAndStyle)
goals.Add(goingtoWork)
goals.Add(goingtoSchool)
goals.Add(price40)
goals.Add(lowStorage)
goals.Add(professionalCar)
goals.Add(onlyBuddies)
goals.Add(typeVan)
goals.Add(typePickUp)
goals.Add(lowInnerSpace)
goals.Add(highInnerSpace)
goals.Add(typeSuvOrCrossover)
goals.Add(cityTransportation)
goals.Add(offroadTransportation)
goals.Add(smallFamily)
goals.Add(largeFamily)
goals.Add(onlyCrush)
goals.Add(offroadAdventures)
goals.Add(userRestrictionprice1)
goals.Add(userRestrictionprice2)
goals.Add(userRestrictionprice3)
goals.Add(userRestrictionprice4)
goals.Add(userRestrictionprice5)
goals.Add(userRestrictionprice6)
goals.Add(userRestrictionprice7)
goals.Add(userRestrictionprice8)
goals.Add(userRestrictionprice9)
goals.Add(userRestrictionprice10)
goals.Add(userRestrictionprice11)
goals.Add(userRestrictionprice12)

'############################################################################
'Define Components
Dim componentPolo As New Component("componentPolo")
Dim componentGolf As New Component("componentGolf")
Dim componentEos As New Component("componentEos")
Dim componentPassat As New Component("componentPassat")
Dim componentPhaeton As New Component("componentPhaeton")
Dim componentPanamera As New Component("componentPanamera")
Dim componentTiguan As New Component("componentTiguan")
Dim componentTuareg As New Component("componentTuareg")
Dim componentScirocco As New Component("componentScirocco")
Dim componentCayenne As New Component("componentCayenne")
Dim componentPorsche911 As New Component("componentPorsche911")
Dim componentTransporter As New Component("componentTransporter")
Dim componentAmarok As New Component("componentAmarok")
components.Add(componentPolo)
components.Add(componentGolf)
components.Add(componentEos)
components.Add(componentPassat)
components.Add(componentPhaeton)
components.Add(componentPanamera)
components.Add(componentTiguan)
components.Add(componentTuareg)
components.Add(componentScirocco)
components.Add(componentCayenne)
components.Add(componentPorsche911)
components.Add(componentTransporter)
components.Add(componentAmarok)

'############################################################################
'Define Choices
Dim choicePolo As New Choice("choicePolo", False)
Dim choiceGolf As New Choice("choiceGolf", False)
Dim choiceEos As New Choice("choiceEos", False)
Dim choicePassat As New Choice("choicePassat", False)
Dim choicePhaeton As New Choice("choicePhaeton", False)
Dim choicePanamera As New Choice("choicePanamera", False)
Dim choiceTiguan As New Choice("choiceTiguan", False)
Dim choiceTuareg As New Choice("choiceTuareg", False)
Dim choiceScirocco As New Choice("choiceScirocco", False)
Dim choiceCayenne As New Choice("choiceCayenne", False)
Dim choicePorsche911 As New Choice("choicePorsche911", False)
Dim choiceTransporter As New Choice("choiceTransporter", False)
Dim choiceAmarok As New Choice("choiceAmarok", False)
choices.Add(choicePolo)
choices.Add(choiceGolf)
choices.Add(choiceEos)
choices.Add(choicePassat)
choices.Add(choicePhaeton)
choices.Add(choicePanamera)
choices.Add(choiceTiguan)
choices.Add(choiceTuareg)
choices.Add(choiceScirocco)
choices.Add(choiceCayenne)
choices.Add(choicePorsche911)
choices.Add(choiceTransporter)
choices.Add(choiceAmarok)
'############################################################################
'Define semantics of goals/components/choices/properties/restrictions
'############################################################################
'Define goal constraints
familyVehicle.SetImplication(safety)
familyVehicle.SetImplication(typeNotVanOrTruck)
weekdayVehicle.SetImplication(lowComfortAndStyle)
joyrideVehicle.SetImplication(highComfortAndStyle)
joyrideVehicle.SetImplication(typeNotVanOrTruck)
goingtoWork.SetImplication(weekdayVehicle)
goingtoWork.SetImplication(typeNotVanOrTruck)
goingtoSchool.SetImplication(weekdayVehicle)
goingtoSchool.SetImplication(price40)
goingtoSchool.SetImplication(typeNotVanOrTruck)
goingtoSchool.SetImplication(lowStorage)
professionalCar.SetImplication(weekdayVehicle)
professionalCar.SetImplication(typeVanOrTruck)
cityTransportation.SetImplication(professionalCar)
cityTransportation.SetImplication(typeVan)
offroadTransporation.SetImplication(professionalCar)
offroadTransportation.SetImplication(typePickUp)
onlyBuddies.SetExclude(seats2)
smallFamily.SetImplication(familyVehicle)
smallFamily.SetImplication(lowInnerSpace)
largeFamily.SetImplication(familyVehicle)
largeFamily.SetImplication(highInnerSpace)
onlyCrush.SetImplication(joyrideVehicle)
onlyCrush.SetImplication(seats2)
onlyBuddies.SetImplication(joyrideVehicle)
offroadAdventures.SetImplication(onlyBuddies)
offroadAdventures.SetImplication(typeSuvOrCrossover)
safety.SetImplication(choicePolo)
safety.SetImplication(choiceGolf)
safety.SetImplication(choiceEos)
safety.SetImplication(choicePassat)
safety.SetImplication(choicePhaeton)
safety.SetImplication(choiceTiguan)
safety.SetImplication(choicePanamera)
safety.SetImplication(choiceTuareg)
safety.SetImplication(choiceCayenne)
safety.SetExclude(choiceScirocco)
safety.SetExclude(choicePorsche911)
safety.SetExclude(choiceAmarok)
safety.SetExclude(choiceTransporter)

seats2.SetImplication(choicePorsche911)
seats2.SetExclude(choicePolo)
seats2.SetExclude(choiceGolf)
seats2.SetExclude(choiceEos)
seats2.SetExclude(choicePassat)
seats2.SetExclude(choicePhaeton)
seats2.SetExclude(choicePanamera)
seats2.SetExclude(choiceTiguan)
seats2.SetExclude(choiceTuareg)
seats2.SetExclude(choiceCayenne)
seats2.SetExclude(choiceAmarok)
seats2.SetExclude(choiceScirocco)
seats2.SetExclude(choiceTransporter)

typeVanOrTruck.SetImplication(choiceTransporter)
typeVanOrTruck.SetImplication(choiceAmarok)
typeVanOrTruck.SetExclude(componentPolo)
typeVanOrTruck.SetExclude(componentGolf)
typeVanOrTruck.SetExclude(componentEos)
typeVanOrTruck.SetExclude(componentPassat)
typeVanOrTruck.SetExclude(componentPhaeton)
typeVanOrTruck.SetExclude(componentPanamera)
typeVanOrTruck.SetExclude(componentTiguan)
typeVanOrTruck.SetExclude(componentTuareg)
typeVanOrTruck.SetExclude(componentCayenne)
typeVanOrTruck.SetExclude(componentScirocco)
typeVanOrTruck.SetExclude(componentPorsche911)

typeNotVanOrTruck.SetExclude(componentAmarok)
typeNotVanOrTruck.SetExclude(componentTransporter)
typeNotVanOrTruck.SetImplication(choicePolo)
typeNotVanOrTruck.SetImplication(choiceGolf)
typeNotVanOrTruck.SetImplication(choiceEos)
typeNotVanOrTruck.SetImplication(choicePassat)
typeNotVanOrTruck.SetImplication(choicePhaeton)
typeNotVanOrTruck.SetImplication(choicePanamera)
typeNotVanOrTruck.SetImplication(choiceTiguan)
typeNotVanOrTruck.SetImplication(choiceTuareg)
typeNotVanOrTruck.SetImplication(choiceCayenne)
typeNotVanOrTruck.SetImplication(choiceScirocco)
typeNotVanOrTruck.SetImplication(choicePorsche911)

LowComfortAndStyle.SetImplication(choicePolo)
LowComfortAndStyle.SetImplication(choiceGolf)
LowComfortAndStyle.SetImplication(choiceEos)
LowComfortAndStyle.SetImplication(choicePassat)
LowComfortAndStyle.SetImplication(choiceTiguan)
LowComfortAndStyle.SetImplication(choiceScirocco)
LowComfortAndStyle.SetImplication(choiceTransporter)
LowComfortAndStyle.SetImplication(choiceAmarok)
LowComfortAndStyle.SetExclude(componentPhaeton)
LowComfortAndStyle.SetExclude(componentCayenne)
LowComfortAndStyle.SetExclude(componentTuareg)
LowComfortAndStyle.SetExclude(componentPorsche911)
LowComfortAndStyle.SetExclude(componentPanamera)

highComfortAndStyle.SetImplication(choicePhaeton)
highComfortAndStyle.SetImplication(choicePanamera)
highComfortAndStyle.SetImplication(choiceTuareg)
highComfortAndStyle.SetImplication(choiceCayenne)
highComfortAndStyle.SetImplication(choicePorsche911)
highComfortAndStyle.SetExclude(componentPolo)
highComfortAndStyle.SetExclude(componentGolf)
highComfortAndStyle.SetExclude(componentEos)
highComfortAndStyle.SetExclude(componentPassat)
highComfortAndStyle.SetExclude(componentScirocco)
highComfortAndStyle.SetExclude(componentTiguan)
highComfortAndStyle.SetExclude(componentAmarok)
highComfortAndStyle.SetExclude(componentTransporter)

price40.SetImplication(choicePolo)
price40.SetImplication(choiceGolf)
price40.SetImplication(choiceEos)
price40.SetImplication(choicePassat)
price40.SetImplication(choiceTiguan)
price40.SetImplication(choiceScirocco)
price40.SetImplication(choiceAmarok)
price40.SetImplication(choiceTransporter)
price40.SetExclude(componentPhaeton)
price40.SetExclude(componentCayenne)
price40.SetExclude(componentTuareg)
price40.SetExclude(componentPorsche911)
price40.SetExclude(componentPanamera)

LowStorage.SetImplication(choicePolo)
LowStorage.SetImplication(choiceGolf)
LowStorage.SetImplication(choiceEos)
LowStorage.SetImplication(choiceScirocco)
LowStorage.SetImplication(choicePorsche911)
LowStorage.SetImplication(choiceAmarok)
LowStorage.SetImplication(choiceTransporter)
LowStorage.SetExclude(componentPhaeton)
LowStorage.SetExclude(componentCayenne)
LowStorage.SetExclude(componentTuareg)
LowStorage.SetExclude(componentTiguan)
LowStorage.SetExclude(componentPanamera)
LowStorage.SetExclude(componentPassat)

typeVan.SetImplication(choiceTransporter)
typeVan.SetExclude(componentPolo)
typeVan.SetExclude(componentGolf)
typeVan.SetExclude(componentEos)
typeVan.SetExclude(componentPassat)
typeVan.SetExclude(componentPhaeton)
typeVan.SetExclude(componentPanamera)
typeVan.SetExclude(componentTiguan)
typeVan.SetExclude(componentTuareg)
typeVan.SetExclude(componentCayenne)
typeVan.SetExclude(componentAmarok)
typeVan.SetExclude(componentScirocco)
typeVan.SetExclude(componentPorsche911)

typePickUp.SetImplication(choiceAmarok)
typePickUp.SetExclude(componentPolo)
typePickUp.SetExclude(componentGolf)
typePickUp.SetExclude(componentEos)
typePickUp.SetExclude(componentPassat)
typePickUp.SetExclude(componentPhaeton)
typePickUp.SetExclude(componentPanamera)
typePickUp.SetExclude(componentTiguan)
typePickUp.SetExclude(componentTuareg)
typePickUp.SetExclude(componentCayenne)
typePickUp.SetExclude(componentTransporter)
typePickUp.SetExclude(componentScirocco)
typePickUp.SetExclude(componentPorsche911)

lowInnerSpace.SetImplication(choicePolo)
lowInnerSpace.SetImplication(choiceGolf)
lowInnerSpace.SetImplication(choiceEos)
lowInnerSpace.SetImplication(choicePassat)
lowInnerSpace.SetImplication(choiceScirocco)
lowInnerSpace.SetImplication(choicePorsche911)
lowInnerSpace.SetImplication(choiceTransporter)
lowInnerSpace.SetExclude(componentPanamera)
lowInnerSpace.SetExclude(componentTiguan)
lowInnerSpace.SetExclude(componentTuareg)
lowInnerSpace.SetExclude(componentCayenne)
lowInnerSpace.SetExclude(componentPhaeton)

highInnerSpace.SetImplication(choicePhaeton)
highInnerSpace.SetImplication(choicePanamera)
highInnerSpace.SetImplication(choiceCayenne)
highInnerSpace.SetImplication(choiceTiguan)
highInnerSpace.SetImplication(choiceTuareg)
highInnerSpace.SetExclude(componentPolo)
highInnerSpace.SetExclude(componentGolf)
highInnerSpace.SetExclude(componentEos)
highInnerSpace.SetExclude(componentPassat)
highInnerSpace.SetExclude(componentScirocco)
highInnerSpace.SetExclude(componentPorsche911)
highInnerSpace.SetExclude(componentAmarok)
highInnerSpace.SetExclude(componentTransporter)

typeSuvOrCrossover.SetImplication(choiceCayenne)
typeSuvOrCrossover.SetImplication(choiceTiguan)
typeSuvOrCrossover.SetImplication(choiceTuareg)
typeSuvOrCrossover.SetExclude(componentPolo)
typeSuvOrCrossover.SetExclude(componentGolf)
typeSuvOrCrossover.SetExclude(componentEos)
typeSuvOrCrossover.SetExclude(componentPassat)
typeSuvOrCrossover.SetExclude(componentScirocco)
typeSuvOrCrossover.SetExclude(componentPorsche911)
typeSuvOrCrossover.SetExclude(componentAmarok)
typeSuvOrCrossover.SetExclude(componentTransporter)
typeSuvOrCrossover.SetExclude(componentPhaeton)
typeSuvOrCrossover.SetExclude(componentPanamera)

'############################################################################
'Define Restriction constraints
userRestrictionprice1.SetImplication(choicePolo)
userRestrictionprice2.SetImplication(choicePolo)
userRestrictionprice2.SetImplication(choiceGolf)
userRestrictionprice3.SetImplication(choicePolo)
userRestrictionprice3.SetImplication(choiceGolf)
userRestrictionprice3.SetImplication(choiceTransporter)
userRestrictionprice4.SetImplication(choicePolo)
userRestrictionprice4.SetImplication(choiceGolf)
userRestrictionprice4.SetImplication(choiceTransporter)
userRestrictionprice4.SetImplication(choiceScirocco)
userRestrictionprice5.SetImplication(choicePolo)
userRestrictionprice5.SetImplication(choiceTransporter)
userRestrictionprice5.SetImplication(choiceGolf)
userRestrictionprice5.SetImplication(choiceScirocco)
userRestrictionprice5.SetImplication(choiceTiguan)
userRestrictionprice6.SetImplication(choicePolo)
userRestrictionprice6.SetImplication(choiceTransporter)
userRestrictionprice6.SetImplication(choiceGolf)
userRestrictionprice6.SetImplication(choiceScirocco)
userRestrictionprice6.SetImplication(choiceTiguan)
userRestrictionprice6.SetImplication(choicePassat)
userRestrictionprice7.SetImplication(choicePolo)
userRestrictionprice7.SetImplication(choiceTransporter)
userRestrictionprice7.SetImplication(choiceGolf)
userRestrictionprice7.SetImplication(choiceScirocco)
userRestrictionprice7.SetImplication(choiceTiguan)
userRestrictionprice7.SetImplication(choicePassat)
userRestrictionprice7.SetImplication(choiceAmarok)
userRestrictionprice8.SetImplication(choicePolo)
userRestrictionprice8.SetImplication(choiceTransporter)
userRestrictionprice8.SetImplication(choiceGolf)
userRestrictionprice8.SetImplication(choiceScirocco)
userRestrictionprice8.SetImplication(choiceTiguan)
userRestrictionprice8.SetImplication(choicePassat)
userRestrictionprice8.SetImplication(choiceAmarok)
userRestrictionprice8.SetImplication(choiceEos)
userRestrictionprice9.SetExclude(componentPhaeton)
userRestrictionprice9.SetExclude(componentPanamera)
userRestrictionprice9.SetExclude(componentCayenne)
userRestrictionprice9.SetExclude(componentPorsche911)
userRestrictionprice10.SetExclude(componentPanamera)
userRestrictionprice10.SetExclude(componentPhaeton)
userRestrictionprice10.SetExclude(componentPorsche911)
userRestrictionprice11.SetExclude(componentPanamera)
userRestrictionprice11.SetExclude(componentPorsche911)
userRestrictionprice12.SetExclude(componentPorsche911)

'PEND Define Choice constraints
choicePolo.SetOption(componentPolo)
choiceGolf.SetOption(componentGolf)
choiceEos.SetOption(componentEos)
choicePassat.SetOption(componentPassat)
choicePhaeton.SetOption(componentPhaeton)
choiceTiguan.SetOption(componentTiguan)
choiceTuareg.SetOption(componentTuareg)
choiceCayenne.SetOption(componentCayenne)
choicePanamera.SetOption(componentPanamera)
choicePorsche911.SetOption(componentPorsche911)
choiceTransporter.SetOption(componentTransporter)
choiceAmarok.SetOption(componentAmarok)
choiceScirocco.SetOption(componentScirocco)

End Sub

End Class
Constraints Generated for Vehicle Configuration

---------
Begin Constraints
---------

familyVehicle:
- Imp.: familyVehicle.active => safety.active
- Imp.: familyVehicle.active => typeNotVanOrTruck.active
- Imp.: familyVehicle.active => NOT seats2.active

safety:
- Imp.: safety.active => choicePolo.active
- Imp.: safety.active => choiceGolf.active
- Imp.: safety.active => choiceEos.active
- Imp.: safety.active => choicePassat.active
- Imp.: safety.active => choicePhaeton.active
- Imp.: safety.active => choicePanamera.active
- Imp.: safety.active => choiceCayenne.active
- Imp.: safety.active => NOT componentScirocco.active
- Imp.: safety.active => NOT componentPorsche911.active
- Imp.: safety.active => NOT componentAmarok.active
- Imp.: safety.active => NOT componentTransporter.active

seats2:
- Imp.: seats2.active => choicePorsche911.active
- Imp.: seats2.active => NOT componentPolo.active
- Imp.: seats2.active => NOT componentGolf.active
- Imp.: seats2.active => NOT componentEos.active
- Imp.: seats2.active => NOT componentPassat.active
- Imp.: seats2.active => NOT componentPhaeton.active
- Imp.: seats2.active => NOT componentPanamera.active
- Imp.: seats2.active => NOT componentTiguan.active
- Imp.: seats2.active => NOT componentTuareg.active
- Imp.: seats2.active => NOT componentCayenne.active
- Imp.: seats2.active => NOT componentAmarok.active
- Imp.: seats2.active => NOT componentScirocco.active

typeVanOrTruck:
- Imp.: typeVanOrTruck.active => choiceTransporter.active
- Imp.: typeVanOrTruck.active => choiceAmarok.active
- Imp.: typeVanOrTruck.active => NOT componentPolo.active
- Imp.: typeVanOrTruck.active => NOT componentGolf.active
- Imp.: typeVanOrTruck.active => NOT componentEos.active
- Imp.: typeVanOrTruck.active => NOT componentPassat.active
- Imp.: typeVanOrTruck.active => NOT componentPhaeton.active
- Imp.: typeVanOrTruck.active => NOT componentPanamera.active
- Imp.: typeVanOrTruck.active => NOT componentTiguan.active
- Imp.: typeVanOrTruck.active => NOT componentTuareg.active
- Imp.: typeVanOrTruck.active => NOT componentCayenne.active

typeNotVanOrTruck:
- Imp.: typeNotVanOrTruck.active => choicePolo.active
- Imp.: typeNotVanOrTruck.active => choiceGolf.active
- Imp.: typeNotVanOrTruck.active => choiceEos.active
- Imp.: typeNotVanOrTruck.active => choicePassat.active
- Imp.: typeNotVanOrTruck.active => choicePhaeton.active
- Imp.: typeNotVanOrTruck.active => choicePanamera.active
- Imp.: typeNotVanOrTruck.active => choiceTiguan.active
- Imp.: typeNotVanOrTruck.active => choiceTuareg.active
- Imp.: typeNotVanOrTruck.active => choiceCayenne.active
- Imp.: typeNotVanOrTruck.active => choiceScirocco.active
- Imp.: typeNotVanOrTruck.active => choicePorsche911.active
- Imp.: typeNotVanOrTruck.active => NOT componentAmarok.active
- Imp.: typeNotVanOrTruck.active => NOT componentTransporter.active
weekdayVehicle:
Imp.: weekdayVehicle.active => lowComfortAndStyle.active
lowComfortAndStyle:
Imp.: lowComfortAndStyle.active => choicePolo.active
Imp.: lowComfortAndStyle.active => choiceGolf.active
Imp.: lowComfortAndStyle.active => choiceEos.active
Imp.: lowComfortAndStyle.active => choicePassat.active
Imp.: lowComfortAndStyle.active => choiceTiguan.active
Imp.: lowComfortAndStyle.active => choiceScirocco.active
Imp.: lowComfortAndStyle.active => choiceTransporter.active
Imp.: lowComfortAndStyle.active => choiceAmarok.active
Imp.: lowComfortAndStyle.active => NOT componentPhaeton.active
Imp.: lowComfortAndStyle.active => NOT componentCayenne.active
Imp.: lowComfortAndStyle.active => NOT componentTuareg.active
Imp.: lowComfortAndStyle.active => NOT componentPorsche911.active
Imp.: lowComfortAndStyle.active => NOT componentPanamera.active
joyrideVehicle:
Imp.: joyrideVehicle.active => highComfortAndStyle.active
Imp.: joyrideVehicle.active => typeNotVanOrTruck.active
highComfortAndStyle:
Imp.: highComfortAndStyle.active => choicePhaeton.active
Imp.: highComfortAndStyle.active => choicePanamera.active
Imp.: highComfortAndStyle.active => choiceTuareg.active
Imp.: highComfortAndStyle.active => choiceCayenne.active
Imp.: highComfortAndStyle.active => choicePorsche911.active
Imp.: highComfortAndStyle.active => NOT componentPolo.active
Imp.: highComfortAndStyle.active => NOT componentGolf.active
Imp.: highComfortAndStyle.active => NOT componentEos.active
Imp.: highComfortAndStyle.active => NOT componentPassat.active
Imp.: highComfortAndStyle.active => NOT componentScirocco.active
Imp.: highComfortAndStyle.active => NOT componentTiguan.active
Imp.: highComfortAndStyle.active => NOT componentAmarok.active
Imp.: highComfortAndStyle.active => NOT componentTransporter.active
goingtoWork:
Imp.: goingToWork.active => weekdayVehicle.active
Imp.: goingToWork.active => typeNotVanOrTruck.active
goingtoSchool:
Imp.: goingToSchool.active => weekdayVehicle.active
Imp.: goingToSchool.active => price40.active
Imp.: goingToSchool.active => typeNotVanOrTruck.active
Imp.: goingToSchool.active => LowStorage.active
price40:
Imp.: price40.active => choicePolo.active
Imp.: price40.active => choiceGolf.active
Imp.: price40.active => choiceEos.active
Imp.: price40.active => choicePassat.active
Imp.: price40.active => choiceTiguan.active
Imp.: price40.active => choiceScirocco.active
Imp.: price40.active => choiceAmarok.active
Imp.: price40.active => choiceTransporter.active
Imp.: price40.active => NOT componentPhaeton.active
Imp.: price40.active => NOT componentCayenne.active
Imp.: price40.active => NOT componentPanamera.active
Imp.: price40.active => NOT componentTuareg.active
LowStorage:
Imp.: LowStorage.active => choicePolo.active
Imp.: LowStorage.active => choiceGolf.active
Imp.: LowStorage.active => choiceEos.active
Imp.: LowStorage.active => choiceScirocco.active
Imp.: LowStorage.active => choicePorsche911.active
Imp.: LowStorage.active => choiceAmarok.active
Imp.: LowStorage.active => choiceTransporter.active
Imp.: LowStorage.active => NOT componentPhaeton.active
Imp.: LowStorage.active => NOT componentCayenne.active
Imp.: LowStorage.active => NOT componentPanamera.active
Imp.: LowStorage.active => NOT componentTuareg.active
Appendix

Imp.: LowStorage.active => NOT componentTiguan.active
Imp.: LowStorage.active => NOT componentPanamera.active
Imp.: LowStorage.active => NOT componentPassat.active

professionalCar:
Imp.: professionalCar.active => weekdayVehicle.active
Imp.: professionalCar.active => typeVanOrTruck.active

onlyBuddies:
Imp.: onlyBuddies.active => jayrideVehicle.active
Imp.: onlyBuddies.active => NOT seats2.active

typeVan:
Imp.: typeVan.active => choiceTransporter.active
Imp.: typeVan.active => NOT componentPolo.active
Imp.: typeVan.active => NOT componentGolf.active
Imp.: typeVan.active => NOT componentEos.active
Imp.: typeVan.active => NOT componentPassat.active
Imp.: typeVan.active => NOT componentPhaeton.active
Imp.: typeVan.active => NOT componentPanamera.active
Imp.: typeVan.active => NOT componentTiguan.active
Imp.: typeVan.active => NOT componentTuareg.active
Imp.: typeVan.active => NOT componentCayenne.active
Imp.: typeVan.active => NOT componentTransporter.active
Imp.: typeVan.active => NOT componentAmarok.active
Imp.: typeVan.active => NOT componentScirocco.active
Imp.: typeVan.active => NOT componentPorsche911.active

typePickUp:
Imp.: typePickUp.active => choiceAmarok.active
Imp.: typePickUp.active => NOT componentPolo.active
Imp.: typePickUp.active => NOT componentGolf.active
Imp.: typePickUp.active => NOT componentEos.active
Imp.: typePickUp.active => NOT componentPassat.active
Imp.: typePickUp.active => NOT componentPhaeton.active
Imp.: typePickUp.active => NOT componentPanamera.active
Imp.: typePickUp.active => NOT componentTiguan.active
Imp.: typePickUp.active => NOT componentTuareg.active
Imp.: typePickUp.active => NOT componentCayenne.active
Imp.: typePickUp.active => NOT componentTransporter.active
Imp.: typePickUp.active => NOT componentAmarok.active
Imp.: typePickUp.active => NOT componentScirocco.active
Imp.: typePickUp.active => NOT componentPorsche911.active

lowInnerSpace:
Imp.: lowInnerSpace.active => choicePolo.active
Imp.: lowInnerSpace.active => choiceGolf.active
Imp.: lowInnerSpace.active => choiceEos.active
Imp.: lowInnerSpace.active => choicePassat.active
Imp.: lowInnerSpace.active => choiceScirocco.active
Imp.: lowInnerSpace.active => NOT componentPanamera.active
Imp.: lowInnerSpace.active => NOT componentTiguan.active
Imp.: lowInnerSpace.active => NOT componentTuareg.active
Imp.: lowInnerSpace.active => NOT componentCayenne.active

highInnerSpace:
Imp.: highInnerSpace.active => choicePhaeton.active
Imp.: highInnerSpace.active => choicePanamera.active
Imp.: highInnerSpace.active => choiceCayenne.active
Imp.: highInnerSpace.active => NOT componentPolo.active
Imp.: highInnerSpace.active => NOT componentGolf.active
Imp.: highInnerSpace.active => NOT componentEos.active
Imp.: highInnerSpace.active => NOT componentPassat.active
Imp.: highInnerSpace.active => NOT componentCayenne.active
Imp.: highInnerSpace.active => NOT componentTransporter.active

typeSuvOrCrossover:
Imp.: typeSuvOrCrossover.active => choiceCayenne.active
Imp.: typeSuvOrCrossover.active => choiceTiguan.active
Imp.: typeSuvOrCrossover.active => choiceTuareg.active
Imp.: typeSuvOrCrossover.active => NOT componentPolo.active
Imp.: typeSuvOrCrossover.active => NOT componentGolf.active
Imp.: typeSuvOrCrossover.active => NOT componentEos.active
Imp.: typeSuvOrCrossover.active => NOT componentPassat.active
Imp.: typeSuvOrCrossover.active => NOT componentScirocco.active
Imp.: typeSuvOrCrossover.active => NOT componentPorsche911.active
Imp.: typeSuvOrCrossover.active => NOT componentAmarok.active
Imp.: typeSuvOrCrossover.active => NOT componentTransporter.active
Imp.: typeSuvOrCrossover.active => NOT componentPhaeton.active
Imp.: cityTransportation.active => professionalCar.active
Imp.: cityTransportation.active => typeVan.active
Imp.: offroadTransportation.active => professionalCar.active
Imp.: offroadTransportation.active => typePickUp.active
Imp.: smallFamily.active => familyVehicle.active
Imp.: smallFamily.active => lowInnerSpace.active
Imp.: largeFamily.active => familyVehicle.active
Imp.: largeFamily.active => highInnerSpace.active
Imp.: onlyCrush.active => joyrideVehicle.active
Imp.: onlyCrush.active => seats2.active
Imp.: offroad Adventures.active => onlyBuddies.active
Imp.: offroad Adventures.active => typeSuvOrCrossover.active
Imp.: userRestrictionprice1.active => choicePolo.active
Imp.: userRestrictionprice1.active => choiceGolf.active
Imp.: userRestrictionprice2.active => choicePolo.active
Imp.: userRestrictionprice2.active => choiceGolf.active
Imp.: userRestrictionprice2.active => choiceTransporter.active
Imp.: userRestrictionprice3.active => choicePolo.active
Imp.: userRestrictionprice3.active => choiceGolf.active
Imp.: userRestrictionprice3.active => choiceTransporter.active
Imp.: userRestrictionprice3.active => choiceScirocco.active
Imp.: userRestrictionprice4.active => choicePolo.active
Imp.: userRestrictionprice4.active => choiceGolf.active
Imp.: userRestrictionprice4.active => choiceTransporter.active
Imp.: userRestrictionprice4.active => choiceScirocco.active
Imp.: userRestrictionprice5.active => choicePolo.active
Imp.: userRestrictionprice5.active => choiceGolf.active
Imp.: userRestrictionprice5.active => choiceTransporter.active
Imp.: userRestrictionprice5.active => choiceScirocco.active
Imp.: userRestrictionprice6.active => choicePolo.active
Imp.: userRestrictionprice6.active => choiceGolf.active
Imp.: userRestrictionprice6.active => choiceTransporter.active
Imp.: userRestrictionprice6.active => choiceScirocco.active
Imp.: userRestrictionprice6.active => choiceTiguan.active
Imp.: userRestrictionprice7.active => choicePolo.active
Imp.: userRestrictionprice7.active => choiceGolf.active
Imp.: userRestrictionprice7.active => choiceTransporter.active
Imp.: userRestrictionprice7.active => choiceScirocco.active
Imp.: userRestrictionprice7.active => choiceTiguan.active
Imp.: userRestrictionprice7.active => choicePassat.active
Imp.: userRestrictionprice7.active => choiceAmarok.active
userRestrictionprice8:
Imp.: userRestrictionprice8.active => choicePolo.active
Imp.: userRestrictionprice8.active => choiceGolf.active
Imp.: userRestrictionprice8.active => choiceTransporter.active
Imp.: userRestrictionprice8.active => choiceScirocco.active
Imp.: userRestrictionprice8.active => choiceTiguan.active
Imp.: userRestrictionprice8.active => choicePassat.active
Imp.: userRestrictionprice8.active => choiceAmarok.active
Imp.: userRestrictionprice8.active => choiceEos.active

userRestrictionprice9:
Imp.: userRestrictionprice9.active => NOT componentPhaeton.active
Imp.: userRestrictionprice9.active => NOT componentPanamera.active
Imp.: userRestrictionprice9.active => NOT componentCayenne.active
Imp.: userRestrictionprice9.active => NOT componentPorsche911.active

userRestrictionprice10:
Imp.: userRestrictionprice10.active => NOT componentPanamera.active
Imp.: userRestrictionprice10.active => NOT componentPhaeton.active

userRestrictionprice11:
Imp.: userRestrictionprice11.active => NOT componentPanamera.active
Imp.: userRestrictionprice11.active => NOT componentPorsche911.active

userRestrictionprice12:
Imp.: userRestrictionprice12.active => NOT componentPorsche911.active

---------------
componentPolo:
componentGolf:
componentEos:
componentPhaeton:
componentPanamera:
componentTiguan:
componentTuareg:
componentScirocco:
componentCayenne:
componentPorsche911:
componentTransporter:
componentAmarok:

---------------
choicePolo:
Imp.: choicePolo.active => componentPolo.active

choiceGolf:
Imp.: choiceGolf.active => componentGolf.active

choiceEos:
Imp.: choiceEos.active => componentEos.active

choicePassat:
Imp.: choicePassat.active => componentPassat.active

choicePhaeton:
Imp.: choicePhaeton.active => componentPhaeton.active

choicePanamera:
Imp.: choicePanamera.active => componentPanamera.active

choiceTiguan:
Imp.: choiceTiguan.active => componentTiguan.active

choiceTuareg:
Imp.: choiceTuareg.active => componentTuareg.active

choiceScirocco:
Imp.: choiceScirocco.active => componentScirocco.active

choiceCayenne:
Imp.: choiceCayenne.active => componentCayenne.active

choicePorsche911:
Imp.: choicePorsche911.active => componentPorsche911.active

choiceTransporter:
Imp.: choiceTransporter.active => componentTransporter.active

choiceAmarok:
Imp.: choiceAmarok.active => componentAmarok.active

---------------
End Constraints
Domain Knowledge Training Planning

Imports CS3.eGym

Public Class DataCreatorTraining

    Sub createTrainingData(ByVal goals As List(Of Goal), ByVal components As List(Of Component), ByVal choices As List(Of Choice))

        '############################################################################
        'Define Goals(TrainingGoals)
        Dim trainingGoalMuscleGain As New Goal("trainingGoalMuscleGain")
        Dim trainingGoalHealth As New Goal("trainingGoalHealth")

        'Define UnitGoals)
        Dim unitGoalUpper_Arms As New Goal("unitGoalUpper_Arms")
        Dim unitGoalBreast As New Goal("unitGoalBreast")
        Dim unitGoalAbdominal As New Goal("unitGoalAbdominal")
        Dim unitGoalLower_Back As New Goal("unitGoalLower_Back")

        'Define Component Goals(Muscles of Exercises)
        Dim componentGoalBICEPS As New Goal("componentGoalBICEPS")
        Dim componentGoalBRACHIALIS As New Goal("componentGoalBRACHIALIS")
        Dim componentGoalTRICEPS As New Goal("componentGoalTRICEPS")
        Dim componentGoalLATISSIMUS As New Goal("componentGoalLATISSIMUS")
        Dim componentGoalHAND_FLEXORS As New Goal("componentGoalHAND_FLEXORS")
        Dim componentGoalPECTORALIS_MAJOR_STERNAL As New Goal("componentGoalPECTORALIS_MAJOR_STERNAL")
        Dim componentGoalOBLIQUES As New Goal("componentGoalOBLIQUES")
        Dim componentGoalRECTUS_ABDOMINIS As New Goal("componentGoalRECTUS_ABDOMINIS")
        Dim componentGoalRECTOR_SPINAE As New Goal("componentGoalRECTOR_SPINAE")
        Dim componentGoalQUADRATUS_LUMBORUM As New Goal("componentGoalQUADRATUS_LUMBORUM")
        Dim componentGoalHAMSTRINGS As New Goal("componentGoalHAMSTRINGS")
        Dim componentGoalRECTUS_FEMORIS As New Goal("componentGoalRECTUS_FEMORIS")
        Dim componentGoalIsoMatte As New Goal("componentGoalIsoMatte")

        'Define User Properties
        Dim userPropertyBMIover30 As New Goal("userPropertyBMIover30")
        Dim userPropertyHandInjury As New Goal("userPropertyHandInjury")
        Dim userPropertyKneeInjury As New Goal("userPropertyKneeInjury")

        goals.Add(trainingGoalMuscleGain)
        goals.Add(trainingGoalHealth)
        goals.Add(unitGoalUpper_Arms)
        goals.Add(unitGoalBreast)
        goals.Add(unitGoalAbdominal)
        goals.Add(unitGoalLower_Back)
        goals.Add(componentGoalBICEPS)
        goals.Add(componentGoalTRICEPS)
        goals.Add(componentGoalLATISSIMUS)
        goals.Add(componentGoalBRACHIALIS)
        goals.Add(componentGoalHAND_FLEXORS)
        goals.Add(componentGoalPECTORALIS_MAJOR_STERNAL)
        goals.Add(componentGoalOBLIQUES)
        goals.Add(componentGoalRECTUS_FEMORIS)
        goals.Add(componentGoalRECTUS_ABDOMINIS)
        goals.Add(componentGoalRECTOR_SPINAE)
        goals.Add(componentGoalQUADRATUS_LUMBORUM)
goals.Add(componentGoalHAMSTRINGS)
goals.Add(componentGoalRECTUS_FEMORIS)
goals.Add(componentGoalISO_Matte)
goals.Add(userPropertyBMIOver30)
goals.Add(userPropertyHandInjury)
goals.Add(userPropertyKneeInjury)

'############################################################################
'Define Choices
Dim choiceBICEPSComponents As New Choice("choiceBICEPSComponents", True)
Dim choiceBRACHIALISComponents As New Choice("choiceBRACHIALISComponents", True)
Dim choiceTRICEPSComponents As New Choice("choiceTRICEPSComponents", True)
Dim choiceHAND_FLEXORSComponents As New Choice("choiceHAND_FLEXORSComponents", True)
Dim choicePECTORALIS_MAJOR_STERNALComponents As New Choice("choicePECTORALIS_MAJOR_STERNALComponents", True)
Dim choiceOBLIQUESComponents As New Choice("choiceOBLIQUESComponents", True)
Dim choiceRECTUS_ABDOMINISComponents As New Choice("choiceRECTUS_ABDOMINISComponents", True)
Dim choiceRECTOR_SPINAECOMPONENTS As New Choice("choiceRECTOR_SPINAECOMPONENTS", True)
Dim choiceHAMSTRINGSComponents As New Choice("choiceHAMSTRINGSComponents", True)
Dim choiceRECTUS_FEMORISComponents As New Choice("choiceRECTUS_FEMORISComponents", True)
Dim choiceLATISSIMUSComponents As New Choice("choiceLATISSIMUSComponents", True)

choices.Add(choiceBICEPSComponents)
choices.Add(choiceBRACHIALISComponents)
choices.Add(choiceTRICEPSComponents)
choices.Add(choiceHAND_FLEXORSComponents)
choices.Add(choicePECTORALIS_MAJOR_STERNALComponents)
choices.Add(choiceOBLIQUESComponents)
choices.Add(choiceRECTUS_ABDOMINISComponents)
choices.Add(choiceRECTOR_SPINAECOMPONENTS)
choices.Add(choiceHAMSTRINGSComponents)
choices.Add(choiceRECTUS_FEMORISComponents)
choices.Add(choiceLATISSIMUSComponents)

'############################################################################
'Define Components(Exercises)
Dim componentBrustpresse As New Component("componentBrustpresse")
Dim componentRuderzug As New Component("componentRuderzug")
Dim componentRotator As New Component("componentRotator")
Dim componentBauchtrainer As New Component("componentBauchtrainer")
Dim componentRueckentrainer As New Component("componentRueckentrainer")
Dim componentRumpfbeuge_seitlich As New Component("componentRumpfbeuge_seitlich")
Dim componentButterfly_eng As New Component("componentButterfly_eng")
Dim componentBizepscurl As New Component("componentBizepscurl")
Dim componentTrizepscurl As New Component("componentTrizepscurl")
Dim componentRussia_Twist_sitzend As New Component("componentRussia_Twist_sitzend")
Dim componentCrunches As New Component("componentCrunches")
Dim componentRueckenstrecker As New Component("componentRueckenstrecker")
Dim componentSeitstuetz_Hueftheben As New Component("componentSeitstuetz_Hueftheben")
Dim componentLiegestuetz As New Component("componentLiegestuetz")
Dim componentISO_Matte As New Component("componentISO_Matte")
components.Add(componentBrustpresse)
components.Add(componentRuderzug)
components.Add(componentRotator)
components.Add(componentBauchtrainer)
components.Add(componentRueckentrainer)
components.Add(componentRumpfbeuge_seitlich)
components.Add(componentButterfly_eng)
components.Add(componentBizepscurl)
components.Add(componentTrizepscurl)
components.Add(componentRussian_Twist_sitzend)
components.Add(componentCrunches)
components.Add(componentRueckenstrecker)
components.Add(componentSeitstuetz_Hueftheben)
components.Add(componentLiegestuetz)
components.Add(componentIsoMatte)

############################################################################
'Define semantics of goals/components/choices/properties/restrictions
'############################################################################

'Define goal constraints
trainingGoalMuscleGain.SetImplication(unitGoalUpper_Arms)
trainingGoalMuscleGain.SetImplication(unitGoalBreast)
trainingGoalHealth.SetImplication(unitGoalAbdominal)
trainingGoalHealth.SetImplication(unitGoalLower_Back)
unitGoalUpper_Arms.SetImplication(componentGoalBICEPS)
unitGoalUpper_Arms.SetImplication(componentGoalTRICEPS)
unitGoalUpper_Arms.SetImplication(componentGoalBRACHIALIS)
unitGoalBreast.SetImplication(componentGoalPECTORALIS_MAJOR_STERNAL)
unitGoalAbdominal.SetImplication(componentGoalOBLIQUES)
unitGoalAbdominal.SetImplication(componentGoalRECTUS_ABDOMINIS)
unitGoalLower_Back.SetImplication(componentGoalRECTOR_SPINAE)
unitGoalLower_Back.SetImplication(componentGoalQUADRATUS_LUMBORUM)

componentGoalRECTUS_FEMORIS.SetImplication(choiceRECTUS_FEMORISComponents)
componentGoalRECTUS_ABDOMINIS.SetImplication(choiceRECTUS_ABDOMINISComponents)
componentGoalOBLIQUES.SetImplication(choiceOBLIQUESComponents)
componentGoalPECTORALIS_MAJOR_STERNAL.SetImplication(choicePECTORALIS_MAJOR_STERNALComponents)
componentGoalLATISSIMUS.SetImplication(choiceLATISSIMUSComponents)
componentGoalBICEPS.SetImplication(choiceBICEPSComponents)
componentGoalBRACHIALIS.SetImplication(choiceBRACHIALISComponents)
componentGoalTRICEPS.SetImplication(choiceTRICEPSComponents)
componentGoalHAND_FLEXORS.SetImplication(choiceHAND_FLEXORSComponents)

'Define Property constraints
userPropertyBMIover30.SetImplication(unitGoalLower_Back)
userPropertyBMIover30.SetExclude(componentGoalIsoMatte)
userPropertyHandInjury.SetExclude(componentGoalHAND_FLEXORS)
userPropertyKneeInjury.SetExclude(componentGoalRECTUS_FEMORIS)

'Define Choice constraints
choiceBICEPSComponents.SetOption(componentBizepscurl)
choiceBRACHIALISComponents.SetOption(componentRuderzug)
choiceLATISSIMUSComponents.SetOption(componentBrustpresse)
choiceTRICEPSComponents.SetOption(componentTrizepscurl)
choiceTRICEPSComponents.SetOption(componentBrustpresse)
choiceTRICEPSComponents.SetOption(componentLiegestuetz)
choiceHAND_FLEXORSComponents.SetOption(componentRuderzug)
choiceHAND_FLEXORSComponents.SetOption(componentBizepscurl)
choiceHAND_FLEXORSComponents.SetOption(componentTrizepscurl)
choiceHAND_FLEXORSComponents.SetOption(componentRussion_Twist_sitzend)
choicePECTORALIS_MAJOR_STERNALComponents.SetOption(componentLiegestuetz)
choicePECTORALIS_MAJOR_STERNALComponents.SetOption(componentBrustpresse)
choicePECTORALIS_MAJOR_STERNALComponents.SetOption(componentButterfly_eng)
choiceOBLIQUESComponents.SetOption(componentRotator)
choiceOBLIQUESComponents.SetOption(componentRumpfbeuge_seitlich)
choiceOBLIQUESComponents.SetOption(componentRussion_Twist_sitzend)
choiceOBLIQUESComponents.SetOption(componentSeitstuetz_Hueftheben)
choicePECTORALIS_MAJOR_STERNALComponents.SetOption(componentBauchtrainer)
choicePECTORALIS_MAJOR_STERNALComponents.SetOption(componentCrunches)
choiceRECTUS_ABDOMINISComponents.SetOption(componentRueckenstrecker)
choiceRECTUS_ABDOMINISComponents.SetOption(componentLiegestuetz)
choiceRECTOR_SPINAEComponents.SetOption(componentRueckenstrecker)
choiceRECTOR_SPINAEComponents.SetOption(componentRueckentrainer)
choiceQUADRATUS_LUMBORUMComponents.SetOption(componentRumpfbeuge_seitlich)
choiceQUADRATUS_LUMBORUMComponents.SetOption(componentSeitstuetz_Hueftheben)
choiceHAMSTRINGSComponents.SetOption(componentRueckenstrecker)
choiceHAMSTRINGSComponents.SetOption(componentRueckentrainer)
choiceHAMSTRINGSComponents.SetOption(componentButterfly_eng)
choiceHAMSTRINGSComponents.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentTrizepscurl)
choiceLiegestuetz.SetOption(componentRussion_Twist_sitzend)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentSeitstuetz_Hueftheben)
choiceLiegestuetz.SetOption(componentQuadratus_Lumborum)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
choiceLiegestuetz.SetOption(componentRueckenstrecker)
choiceLiegestuetz.SetOption(componentRueckentrainer)
choiceLiegestuetz.SetOption(componentButterfly_eng)
choiceLiegestuetz.SetOption(componentIsoMatte)
End Sub
End Class
Constraints Generated for Training Planning

--------------
Begin Constraints
--------------
trainingGoalMuscleGain:
  Imp.: trainingGoalMuscleGain.active => unitGoalUpper_Arms.active
  Imp.: trainingGoalMuscleGain.active => unitGoalBreast.active
trainingGoalHealth:
  Imp.: trainingGoalHealth.active => unitGoalAbdominal.active
  Imp.: trainingGoalHealth.active => unitGoalLower_Back.active
unitGoalUpper_Arms:
  Imp.: unitGoalUpper_Arms.active => componentGoalBICEPS.active
  Imp.: unitGoalUpper_Arms.active => componentGoalTRICEPS.active
  Imp.: unitGoalUpper_Arms.active => componentGoalBRACHIALIS.active
unitGoalBreast:
  Imp.: unitGoalBreast.active => componentGoalPECTORALIS_MAJOR_STERNAL.active
  Imp.: unitGoalAbdominal.active => componentGoalRECTUS_ABDOMINIS.active
unitGoalLower_Back:
  Imp.: unitGoalLower_Back.active => componentGoalRECTOR_SPINAE.active
  Imp.: unitGoalLower_Back.active => componentGoalQUADRATUS_LUMBORUM.active
componentGoalBICEPS:
  Imp.: componentGoalBICEPS.active => choiceBICEPSComponents.active
componentGoalTRICEPS:
  Imp.: componentGoalTRICEPS.active => choiceTRICEPSComponents.active
componentGoalLATISSIMUS:
  Imp.: componentGoalLATISSIMUS.active => choiceLATISSIMUSComponents.active
componentGoalBRACHIALIS:
  Imp.: componentGoalBRACHIALIS.active => choiceBRACHIALISComponents.active
componentGoalHAND_FLEXORS:
  Imp.: componentGoalHAND_FLEXORS.active => choiceHAND_FLEXORSComponents.active
componentGoalPECTORALIS_MAJOR_STERNAL:
  Imp.: componentGoalPECTORALIS_MAJOR_STERNAL.active => choicePECTORALIS_MAJOR_STERNALComponents.active
componentGoalRECTUS_ABDOMINIS:
  Imp.: componentGoalRECTUS_ABDOMINIS.active => choiceRECTUS_ABDOMINISComponents.active
componentGoalRECTOR_SPINAE:
  Imp.: componentGoalRECTOR_SPINAE.active => choiceRECTOR_SPINAEComponents.active
componentGoalQUADRATUS_LUMBORUM:
  Imp.: componentGoalQUADRATUS_LUMBORUM.active => choiceQUADRATUS_LUMBORUMComponents.active
componentGoalHAMSTRINGS:
  Imp.: componentGoalHAMSTRINGS.active => choiceHAMSTRINGSComponents.active
componentGoalRECTUS_FEMORIS:
  Imp.: componentGoalRECTUS_FEMORIS.active => choiceRECTUS_FEMORISComponents.active
componentGoalIsomatte:
  Imp.: userPropertyBMIover30.active => unitGoalLower_Back.active
  Imp.: userPropertyBMIover30.active => NOT componentGoalIsomatte.active
userPropertyHandInjury:
  Imp.: userPropertyHandInjury.active => NOT componentGoalHAND_FLEXORS.active
userPropertyKneeInjury:
  Imp.: userPropertyKneeInjury.active => NOT componentGoalRECTUS_FEMORIS.active
--------------
componentBrustresse:
  Imp.: componentBrustresse.active => choiceTRICEPSComponents.active
  Imp.: componentBrustresse.active => choiceHAND_FLEXORSComponents.active
  Imp.: componentBrustresse.active => choicePECTORALIS_MAJOR_STERNALComponents.active

componentRuderzug:
Imp.: componentRuderzug.active => componentGoalBRACHIALIS.active
Imp.: componentRuderzug.active => componentGoalLATISSIMUS.active

componentRotator:
Imp.: componentRotator.active => componentGoalOBLIQUES.active

componentBauchtrainer:
Imp.: componentBauchtrainer.active => componentGoalRECTUS_ABDOMINIS.active

componentRueckentrainer:
Imp.: componentRueckentrainer.active => componentGoalRECTOR_SPINAE.active
Imp.: componentRueckentrainer.active => componentGoalHAMSTRINGS.active

componentRumpfbeuge_seitlich:
Imp.: componentRumpfbeuge_seitlich.active => componentGoalOBLIQUES.active
Imp.: componentRumpfbeuge_seitlich.active => componentGoalQUADRATUS_LUMBORUM.active

componentButterfly_eng:
Imp.: componentButterfly_eng.active => componentGoalPECTORALIS_MAJOR_STERNAL.active

componentBizepscurl:
Imp.: componentBizepscurl.active => componentGoalBICEPS.active

componentTrizepscurl:
Imp.: componentTrizepscurl.active => componentGoalTRICEPS.active

componentRussion_Twist_sitzend:
Imp.: componentRussion_Twist_sitzend.active => componentGoalHAND_FLEXORS.active
Imp.: componentRussion_Twist_sitzend.active => componentGoalOBLIQUES.active
Imp.: componentRussion_Twist_sitzend.active => componentGoalIsoMatte.active

componentCrunches:
Imp.: componentCrunches.active => componentGoalRECTUS_ABDOMINIS.active
Imp.: componentCrunches.active => componentGoalIsoMatte.active

componentRueckenstrecker:
Imp.: componentRueckenstrecker.active => componentGoalRECTOR_SPINAE.active

componentSeitstuetz_Hueftheben:
Imp.: componentSeitstuetz_Hueftheben.active => componentGoalOBLIQUES.active
Imp.: componentSeitstuetz_Hueftheben.active => componentGoalQUADRATUS_LUMBORUM.active

componentLiegestuetz:
Imp.: componentLiegestuetz.active => componentGoalTRICEPS.active
Imp.: componentLiegestuetz.active => componentGoalHAND_FLEXORS.active
Imp.: componentLiegestuetz.active => componentGoalPECTORALIS_MAJOR_STERNAL.active
Imp.: componentLiegestuetz.active => componentGoalRECTUS_ABDOMINIS.active
Imp.: componentLiegestuetz.active => componentGoalIsoMatte.active

ComponentIsomatte:
--------------

choiceBICEPSComponents:
Imp.: choiceBICEPSComponents.active => componentBizepscurl.active

choiceBRACHIALISComponents:
Imp.: choiceBRACHIALISComponents.active => componentRuderzug.active or componentBrustpresse.active

choiceTRICEPSComponents:
Imp.: choiceTRICEPSComponents.active => componentTrizepscurl.active or componentBrustpresse.active or componentLiegestuetz.active

choiceHAND_FLEXORSComponents:
Imp.: choiceHAND_FLEXORSComponents.active => componentRuderzug.active or componentTrizepscurl.active or componentBrustpresse.active or componentLiegestuetz.active

choicePECTORALIS_MAJOR_STERNALComponents:
Imp.: choicePECTORALIS_MAJOR_STERNALComponents.active => componentLiegestuetz.active or componentBrustpresse.active or componentButterfly_eng.active

choiceOBLIQUESComponents:
Imp.: choiceOBLIQUESComponents.active => componentRotator.active or componentRumpfbeuge_seitlich.active or componentRussion_Twist_sitzend.active or componentSeitstuetz_Hueftheben.active

choiceRECTUS_ABDOMINISComponents:
Imp.: choiceRECTUS_ABDOMINISComponents.active => componentBauchtrainer.active or componentCrunches.active or componentLiegestuetz.active

choiceRECTOR_SPINAEComponents:
Imp.: choiceRECTOR_SPINAEComponents.active => componentRueckenstrecker.active or componentRueckentrainer.active
choiceQUADRATUS_LUMBORUMComponents:
Imp.: choiceQUADRATUS_LUMBORUMComponents.active => componentRumpfbeuge_seitlich.active or componentSeitstuetz_Hueftheben.active
choiceHAMSTRINGSComponents:
Imp.: choiceHAMSTRINGSComponents.active => componentRueckentrainer.active
choiceRECTUS_FEMORISComponents:
Imp.: choiceRECTUS_FEMORISComponents.active => componentRueckenstrecker.active
choiceLATISSIMUSComponents:
Imp.: choiceLATISSIMUSComponents.active => componentRuderzug.active
---------------
End Constraints
---------------

Training Planning Test Run Results

Activated: trainingGoalHealth.active
Activated: userPropertyBMIover30.active
Derivable by context:
trainingGoalHealth.active = T
unitGoalAbdominal.active = T
unitGoalLower_Back.active = T
componentGoalOBLIQUES.active = T
componentGoalRECTUS_ABDOMINIS.active = T
componentGoalRECTOR_SPINAE.active = T
componentGoalQUADRATUS_LUMBORUM.active = T
componentGoalIsoMatte.active = F
userPropertyBMIover30.active = T
componentBauchtrainer.active = T
componentRumpfbeuge_seitlich.active = T
componentRussion_Twist_sitzend.active = F
componentCrunches.active = F
componentSeitstuetz_Hueftheben.active = F
componentLiegestuetz.active = F
choiceOBLIQUESComponents.active = T
choiceRECTUS_ABDOMINISComponents.active = T
choiceRECTOR_SPINAEComponents.active = T
choiceQUADRATUS_LUMBORUMComponents.active = T
Open Agenda Item: choiceOBLIQUESComponents
Open Agenda Item: choiceRECTUS_ABDOMINISComponents
Open Agenda Item: choiceRECTOR_SPINAEComponents
Open Agenda Item: choiceQUADRATUS_LUMBORUMComponents
Selected Option: componentRotator  for Choice: choiceOBLIQUESComponents
Selected Option: componentBauchtrainer  for Choice: choiceRECTUS_ABDOMINISComponents
Selected Option: componentRueckenstrecker  for Choice: choiceRECTOR_SPINAEComponents
Selected Option: componentRumpfbeuge_seitlich  for Choice: choiceQUADRATUS_LUMBORUMComponents
Open Agenda Item: choiceRECTUS_FEMORISComponents
Selected Option: componentRueckenstrecker  for Choice: choiceRECTUS_FEMORISComponents
Configuration Complete:
Active Goals:
trainingGoalHealth is active
unitGoalAbdominal is active
unitGoalLower_Back is active
componentGoalOBLIQUES is active
componentGoalRECTUS_ABDOMINIS is active
componentGoalRECTOR_SPINAE is active
componentGoalQUADRATUS_LUMBORUM is active
unitGoalLower_Back is active
componentGoalRECTUS_FEMORIS is active
userPropertyBMIover30 is active

-------------
Active Choices:
-------------
choiceOBLIQUESComponents is active
choiceRECTUS_ABDOMINISComponents is active
choiceRECTOR_SPINAEComponents is active
choiceQUADRATUS_LUMBORUMComponents is active
choiceRECTUS_FEMORISComponents is active

-------------
Active Components:
-------------
componentRotator is active
componentBauchtrainer is active
componentRumpfbeuge_seitlich is active
componentRueckenstrecker is active

-------------

Glossary

ATMS
An assumption based truth maintenance system (ATMS) is a knowledge representation method for representing both beliefs and their dependencies.

Body-Mass-Index (BMI)
The Body-Mass-Index is a heuristic proxy for human body fat based on an individual's weight and height.

Component Goal
Component Goals satisfy the User Goals. They might be conjunctions of Component Goals that satisfy the User Goals.

Component
A component is an entity that can have Goals. It can be combined with others components in a specified way to satisfy a desired UserGoal. Components may add Constraints to the configuration.

Configuration
The task of composing a system out of components according to requirements of a user.

Choice
Choices represent points in the configuration process where a decision is necessary because more than one option is available for a Goal or a Component.

Constraint
A constraint is a logical relation over one or several variables, each having a value over a finite domain. A constraint restricts the valid values of a variable.

Composite Pattern
A design pattern which represents a hierarchy of different objects.

*Constraint Satisfaction problem (CSP)*

Constraint satisfaction problems are mathematical problems defined as a set of objects whose state must satisfy a number of constraints.

*Constraint System (CS)*

A software system that solves Constraint Satisfaction Problems using a problem solver.

*Constraint Programming (CP)*

A computer-science paradigm that can be used for configuration.

*Developer*

Author of a software system. A developer has an education in computer science.

*Domain Expert*

A Person with background in a specific domain. For example a automotive engineer or sports scientist.

*Graphical User Interface (GUI)*

The visual interface of an application. It serves as a bridge between the user and an application and allows the user to input information into the application in a visual way.

*Goals*

Goals can be stated by the user, they represent his requirements. Goals can be organized in a refinement structure. Components also have goals that fulfill specific function. These goals are called Component Goals.

*Miles per gallon* (MPG)

Miles per gallon, a motor fuel consumption measurement.

*Static Structure Diagram (SSD)*

SSDs show the static structure of a model, that is, elements that exists, their internal structure and how the elements are related. The conceptual diagrams represent concepts from the real world and the relationships between them.

*Unified Modeling Language (UML)*

An Object Management Group (OMG) standard for modeling software artifacts. It is a widely-used industry-standard language for the specification, visualization, construction, and documentation of the software system components or software systems. It contains a set of symbols for creating diagrams to model software systems.

*User*

A Person that has requirements that should be satisfied by the configuration he wants. He may have Properties or restrictions that can result in additional Constraints.
Appendix

User Goal
A UserGoal contains desired, ‘positive’, aspects of a configuration. A User Goal is satisfied by a Component Goal or a combination of Component Goals.

User Property
Additional information about the User, may add Constraints to the Configuration.

User Restriction
Explicitly specified information about unwanted aspects in a Configuration

VB.Net
An object-oriented programming language with portable binaries. It can be used to program applications and forms and is portable to a large number of operating systems.

Bibliography


Mittal (90) Mittal S. & Frayman F., Dynamic Constraint Satisfaction Problems, Proceedings AAAI-90
Mittal (89) Mittal S. & Frayman F., Towards a generic model of configuration Tasks, Proceedings 11th Int. Joint Conf. on AI, 1395–140

Soininen (98) Soininen T., Tiittonen J., Männistö T. & Sulonen R., Towards a general ontology of configuration, Helsinki University of Technology, TAI Research Centre and Laboratory of Information Processing Science, Helsinki, Finland


Citroen (2012), Citroen website, http://www.citroen.de/home/#/home/, Last visit 09/15/12.

Toyota (2012), Toyota website, http://www.toyota.de/, Last visit 09/15/12.