

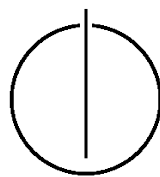
FAKULTÄT FÜR INFORMATIK

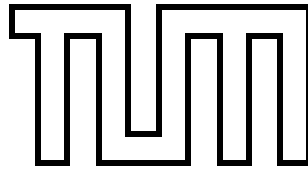
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Wirtschaftsinformatik

Implementing a Web Client for Integrated Data, Role, Function, and Task Modelling

Tobias Schrade





FAKULTÄT FÜR INFORMATIK

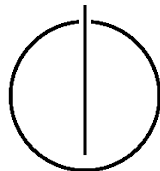
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Wirtschaftsinformatik

Implementing a Web Client for Integrated Data, Role,
Function, and Task Modelling

Umsetzung eines Webclients für integrierte Daten-,
Rollen-, Funktions- und Aufgabenmodellierung

Author: Tobias Schrade
Supervisor: Prof. Dr. Florian Matthes
Advisor: Thomas Reschenhofer
Date: August 15, 2016



Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. August 2016

Tobias Schrade

Abstract

Hybrid Wikis combine the flexibility of normal wikis and the rich functionality of enterprise modelling systems to support collaborative information and model management. To achieve this goal a user interface for the modelling of agile data and processes is necessary. The users must not only be able to collaboratively define attribute and task models but analyze the metadata. Due to the emergent nature of this data small changes happen on a regular basis and such a tool should support users in keeping data models, task models, and analysis models consistent.

At the current state of the art hybrid wikis do not offer any integrated user interface to handle agile data, role, function and task modelling and analysis within one application.

In order to solve this problem, a prototype interface will be implemented into a hybrid wiki platform following the design science approach. Through the process of evaluation by industry partners and applying the results to the implementation, the prototype will be improved.

This thesis will describe the prototypical implementation of a modelling user interface for a hybrid wiki with the focus on usability for the industrial application.

Contents

Abstract	vii
I. Introduction	1
1. Introduction	3
1.1. Important Concepts	4
1.1.1. Hybrid Wiki	4
1.1.2. MxL	4
1.1.3. Adaptive Case Management and Tasks	4
1.2. Problem Statement	5
1.3. Research Questions	5
1.4. Research Methodology and Outline	5
II. Foundations	7
2. Foundations	9
2.1. Hybrid Wiki Meta Model	9
2.2. Access Control Meta Model	10
2.3. MxL Meta Model	11
2.4. ACM Meta Model	13
III. Integrated Hybrid Wiki Model	15
3. Integrated Hybrid Wiki Model	17
3.1. Explanation of the Model	17
3.2. Behavioural Model	19
IV. Prototype Implementation	25
4. Prototype Implementation	27
4.1. Technical Foundation for the Prototype	27
4.1.1. SocioCortex	27
4.1.2. sc-angular	28
4.1.3. mxl-angular	29
4.1.4. Angular Material	29

4.2. Overall Architecture	30
4.3. Core Features	31
4.3.1. Workspace Dashboard	32
4.3.2. Attribute Definitions	33
4.3.3. Derived Attribute Definitions	34
4.3.4. Tasks and Stages	35
V. Evaluation	39
5. Evaluation	41
5.1. Evaluation Approach	41
5.1.1. Evaluation Group	42
5.1.2. Questionnaire	42
5.2. Results	43
5.3. Possible Improvements of the Prototype	45
VI. Potential further Enhancement of the Model	49
6. Potential further Enhancement of the Model	51
6.1. MxL for (default) access rights	51
6.1.1. Entity	51
6.1.2. Attribute	51
6.1.3. Task	52
6.2. MxL for default Values	52
6.3. MxL for Constraints of Attribute Definitions	53
6.4. MxL for Tasks and Sentries	53
6.4.1. MxL for Completion of a Task	53
6.4.2. MxL for Enabling of a Task	53
6.5. User roles for Tasks	54
6.6. Extended Meta Model	54
VII. Conclusion	57
7. Conclusion and Critical Reflection	59
7.1. Summary	59
7.2. Conclusion	59
7.3. Critical Reflection	61
VIII. Outlook + Future Work	63
8. Outlook and Future Work	65

Appendix	69
A. Evaluation Scenario	69
B. Evaluation Questionnaire	73
Bibliography	77

Part I.

Introduction

1. Introduction

The amount of digital information, which has to be handled by companies, is rapidly growing. To manage it, enterprises need to adopt new applications and methods [7]. Furthermore force increasingly turbulent business environments, technological innovations and legal regulations continuous changes in enterprise information systems [1]. Another main problem is, that the amount of support for the company, an enterprise information system provides rapidly decreases, if the system is not able to adapt itself to the changing environment fast enough [25].

To tackle these non-trivial problems, different solutions arose during the last years. Matthes et al., for instance, suggested a hybrid wiki approach in 2011 [14] (c.f. Section 1.1.1). Thereby a wiki is enriched with an underlying model to provide additional structure elements for the wiki-pages like attributes and types. The main goal thereby is to empower non-expert users to gather the information in the wiki collaboratively [14]. As using a hybrid wiki approach for an enterprise information system can cause challenges of different kinds, Reschenhofer et al. improved the model based on five years of testing and evaluations [17].

Not only Reschenhofer et al. worked on the hybrid wiki approach, but also Hauder et al. [10]. They took a more process centred approach and applied it to knowledge work in general. The main motivation thereby was, that on the one hand that according to Davenport "the most important processes for organizations today involve knowledge work" [5] and on the other hand existing work-flow management tools are not well suited to support these processes, due to a large amount of exceptions in knowledge intensive processes [24, 22]. The introduction of tasks in their implementation called Darwin Wiki is the main contribution of their work.

Another extension to the hybrid wiki model was the creation of a domain specific language called model-based expression language (MxL), it is specially tailored to the underlying meta model. With the help of MxL quantitative KPIs can be defined and computed [19, 15].

As shown above different concepts around the hybrid wiki meta model were created in the past years. Although the basis of all presented works is the same model, they evolved it into different directions resulting in various applications. This master's thesis is about integrating the different meta models and therefore providing one combined model. Besides that, a prototypical user interface to work with the model will be implemented and evaluated.

1.1. Important Concepts

This section further explains the important concepts, which were shortly introduced above.

1.1.1. Hybrid Wiki

The hybrid wiki concept was created by Matthes et al. in 2011 [14]. It combines the loose wiki pages and an evolving page structure behind them. This can be compared to classes and objects in object oriented programming. Every page in the wiki then is an object of a certain class, which is called the pages type. Besides the textual content, the page then also has attributes, which are defined by its type. The attributes can have different constraints on multiplicity and their type. Available types are ranging from primitive ones like text, number, date or boolean to more complex types like references to other pages (of a certain type) or files within the system. Thanks to this construct, the pages are structured and there is an evolving model behind them.

Matthes et al. also suggested using hybrid wikis for enterprise architecture management, due to the fact that its a lightweight solution which supports collaborative work in an ever-changing environment like the architecture of an enterprise [13, 12].

The latest version of the hybrid wiki meta model is presented and explained in detail in Section 2.1.

1.1.2. MxL

Model-based expression language is the domain specific language for the hybrid wiki model. It is mainly developed by Thomas Reschenhofer and Ivan Monahov since 2013 [15, 16]. The main purpose of the language is to enable the definition and calculation of key performance indicators within the hybrid wiki model. With the help of MxL expressions it is possible to define additional attributes, which are always calculated during life-time, while it is possible to refer to other elements within the system. One simple example therefore is the calculation of the age of an person based on the date of birth.

Another important concept of MxL is type safety. Thanks to this it is possible to calculate the type of the outcome of an expression and therefore use them as functions for other expressions [19].

A more detailed explanation of MxL can be found in Section 2.3.

1.1.3. Adaptive Case Management and Tasks

[9] Adaptive case management (ACM) as defined by Swenson et. al in 2010 [23] is the basis for the task concept of Hauder [9], which itself is part of the foundations for this thesis. Coupled with hybrid wikis, tasks enable a controlled work-flow to fill out the attributes of wiki pages. A tasks always includes a set of attributes, which need to be filled out to complete the task. The absence of complex behaviour of tasks enables the capturing of implicit knowledge of unskilled workers [9]. To further structure the work-flow, stages enable hierarchical elements and preconditions define an order, in which the tasks need to be completed.

1.2. Problem Statement

All models introduced above have the hybrid wiki model as mutual basis. The fact that they were developed by different persons simultaneously led to models, which do not have many interconnections between them. This is a missed opportunity, because these links could generate additional merit for the individual models. Furthermore there is no integrated user interface, which is able to model the data for all concepts resulting in an inhomogeneous user-experience. This problem will be addressed by this thesis.

1.3. Research Questions

The following research objective and is deducted from the problem statement:

Facilitate data, role, function and task modelling within a hybrid wiki platform with the help of a web based user interface to provide a way to generate and administrate the above mentioned models.

To full-fill this objective it is further divided into the two research questions:

Question 1: "How does an integrated meta-model for data, role, function and task models look like?"

Question 2: "How to design an integrated user interface for the management of data, role, function and task models?"

1.4. Research Methodology and Outline

The basis of the work is Hevner et. al's design science approach [11], but due to time constraints there will only be one evaluation. The knowledge base for the prototype is consisting of the mock-ups, which are already provided by Sirma Gjorgievska and the foundations presented in Chapter 2. Combining all foundation models into one integrated model is done in Chapter 3. Furthermore the behaviour of the important elements of the models is elaborated and explained there. Based on this model, Chapter 4 then introduces the implemented prototype. The important functions, which are directly connected to the research question 2, are highlighted and explained in detail in the second part of the chapter. To finish the first circle of Hevners approach, the implementation is evaluated in Chapter 5. Based on the impressions of the evaluation and own theories, potential further additions to the integrated model are brought up in Chapter 4. Concluding the thesis are Chapters 7 and 8 containing a conclusion of my work and an outlook with future research possibilities based on it.

Part II.

Foundations

2. Foundations

In this chapter the Foundations of this work are explained. Namely these are the Hybrid Wiki Meta Model, the MxL Meta Model and the ACM Meta Model, all introduced in Section 1.1.

2.1. Hybrid Wiki Meta Model

The hybrid wiki meta model is based on the model presented in the Paper “Lessons Learned in Aligning Data and Model Evolution in Collaborative Information Systems” by Thomas Reschenhofer et. al [17]. Figure 2.1 shows their model, which will be further explained in this chapter.

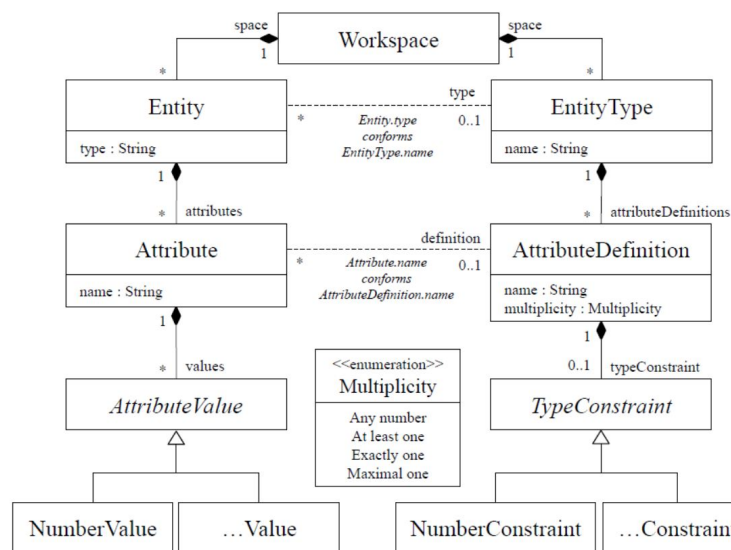


Figure 2.1.: Hybrid Wiki Meta Model [17]

The right side of the model is responsible for the structure of the wiki whereas the left side represents the content. Workspace is located in the middle, due to the fact that it is a mix of content and structure and therefore cannot be assigned to one side.

A workspace can have multiple entitytypes and entities. It is the main unit to structure the whole wiki. One workspace is one unit and it is only rarely the case that there are references between workspaces. A workspace always has at least one entity, which is its home page. This entity cannot be deleted.

Entities can be compared to normal pages of an wiki. They have to belong to a workspace

and they can be of a certain entitytype. The entitytype with its attributedefinitions predefines the structure of an entity.

Attributedefinitions define the type(constraints) of an attribute, the multiplicity and its name. There are many options for the attribute typeconstraint. It can be a simple type like "number" or "text" but also a reference to entitytypes, users or files is possible. This enhances the possibilities to structure the wiki and model complex systems like an enterprise architectures the university life. Latter will be used for the evaluation in Chapter 5. To make the system more flexible, attributes do not have to meet the requirements given by the corresponding definition. Furthermore there is also the option to attributes without a set attributedefinition to an entity. The multiplicity is restricted by the enumeration multiplicity. Therefore only the multiplicities "Any number" (*), "At least one" (1..*), "Exactly one" (1) and "Maximal one" (0..1) are possible multiplicities. This helps to lower the complexity and is beneficial for the understanding of the resulting models.

2.2. Access Control Meta Model

The access control meta model as presented by Thomas Reschenhofer [18] is shown in figure 2.2.

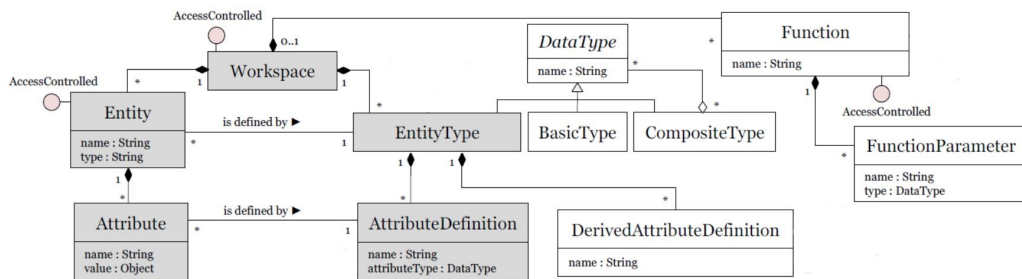


Figure 2.2.: Access Control Meta Model [18]

The interface "AccessControlled" with Users and Groups are added to the model (see Figure 2.3). A class implementing the "AccessControlled" interface like entity can have explicit access rights set for every user or group. These rights have 3 permission levels. The lowest access level is read only, where the user can see the element, but can not edit it, followed by editor where he or she can also edit the resource. At the top administration is the highest level of access, which can be granted to a principal.

The organization of users can be done with groups and the use of the abstract superclass "Principal" once more utilizing a variant of the composite pattern. Worth mentioning is, that a principal can be in various groups which results in the possibility for an user to be part of many groups on the one hand and a group to be a sub group of various super groups on the other.

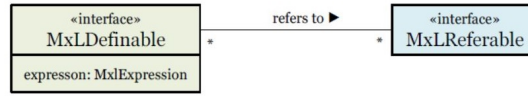


Figure 2.5.: MxLReferable and MxLDefineable Definition [18]

Introduction to MxL Expressions

As stated in 1.1.2, MxL is a domain specific language, which grants type safety and supports sub typing[19]. The type hierarchy of MxL is shown in Figure 2.6. The part marked in green is adapted on the current state of the hybrid wiki.

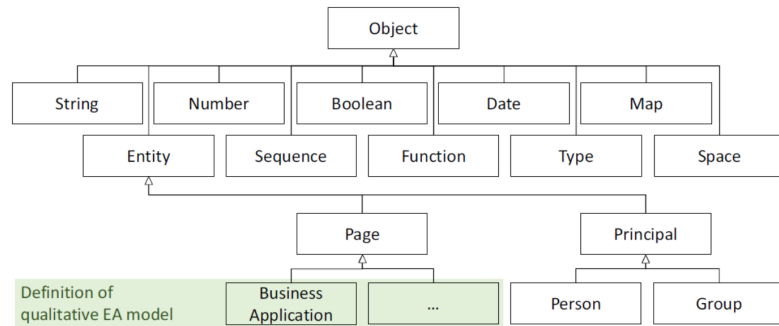


Figure 2.6.: MxL Type Hierarchy [19]

Another feature of the language is the built in type checker. This checker can validate any given MxL expression and calculate the outcomes inferred type based on the, to the current state of the wiki adapted, hierarchy. This also enables simplified MxL expressions. The expression

```

find "Business Application"
  .select(ba =>
    ba ["Function points"].first())
  .sum()
  
```

implemented with the the untyped core expression language of the hybrid wiki can be simplified to:

```

find 'Business Application'
  .sum('Function points')
  
```

Example taken from Reschenhofer et. al [19]. Thanks to these functions the user is able to access data and perform calculations on the basis of the hybrid wikis current state.

2.4. ACM Meta Model

The ACM Meta Model as introduced in Section 1.1.3 is all about Tasks and their management. The model is taken from Matheus Hauders dissertation "Empowering End-Users to Collaboratively Structure Knowledge-Intensive Processes" [9]. Figure 2.7 shows the whole model, which will be explained in detail afterwards.

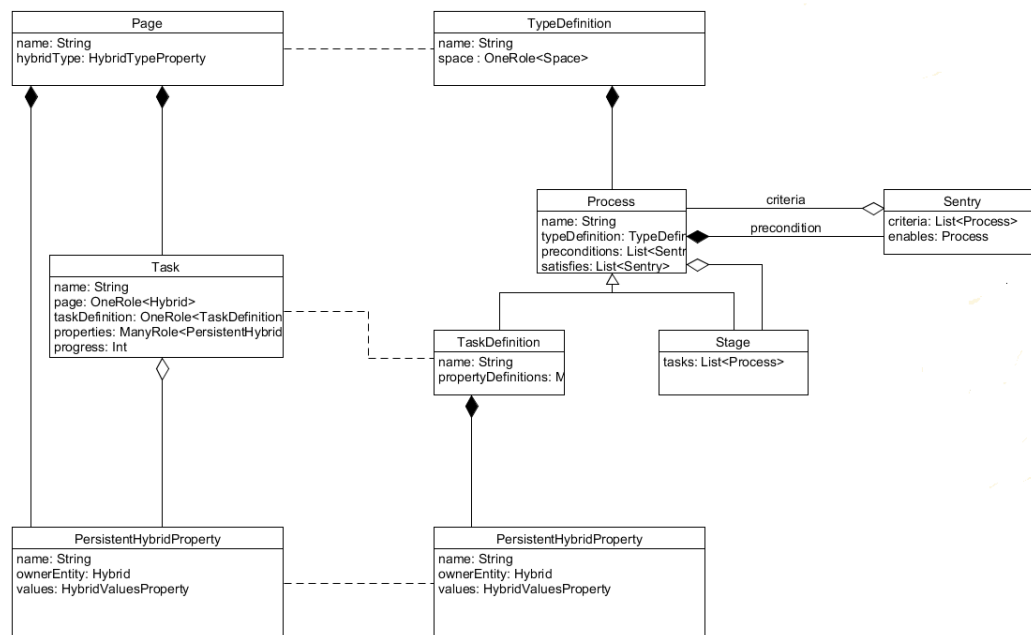


Figure 2.7.: ACM Meta Model [9]

The four most important classes of the model are "Process", "TaskDefinition", "Stage" and "Sentry". TypeDefinition is mapped to the class "EntityType" and "Page" is the same as "Entity" from Section 2.1, the ACM meta model just uses an older version of the hybrid wiki model as its basis. The name changes are further explained in [17].

Process is the super-class of "TaskDefinition" and "Stage". The composite pattern is used to implement a hierarchical structure within the processes. Stage is the composite and therefore can have multiple children of the class process (component) attached to it. Objects of the class "TaskDefinition" are the leaves in the resulting tree.

A process is always directly attached to a typedefinition and only visible within this context. To enhance the processes even further sentries are introduced. A sentry can be connected to a process via two different attributes, either preconditions or satisfies. The sentries stored in preconditions are linked with the logical or, therefore one of the sentries requirements needs to be met to enable the process. The sentries stored in satisfies on the other hand are all sentries, which use the process as criteria to be fulfilled.

Sentry itself has two attributes, criteria and enables. Enables is a reference to a process, which is enabled when all criteria of the sentry are fulfilled and criteria is a list of processes, which are linked with the logical and.

Within the attribute "propertyDefinitions" of the class "TaskDefinition" PersistentHybridProperties are stored. Referring to the hybrid wiki model presented in Section 2.1 PersistentHybridProperties are attributedefinitions. To complete a task on instance level all attributes of the corresponding page, which are listed in propertydefinitions need to have a value set.

Part III.

Integrated Hybrid Wiki Model

3. Integrated Hybrid Wiki Model

In this chapter, the integrated hybrid wiki model is presented and the behaviour of important classes is explained in detail.

3.1. Explanation of the Model

Integrating the three models presented in chapter 2 resulted in the model shown in figure 3.1. The hybrid wiki model thereby is the baseline, as the other models are connected to it. The model describes the current back-end of SocioCortex. Classes filled in green are of high importance for data structure and therefore also for the prototype implementation. Their behaviour will be explained in the following section 3.2.

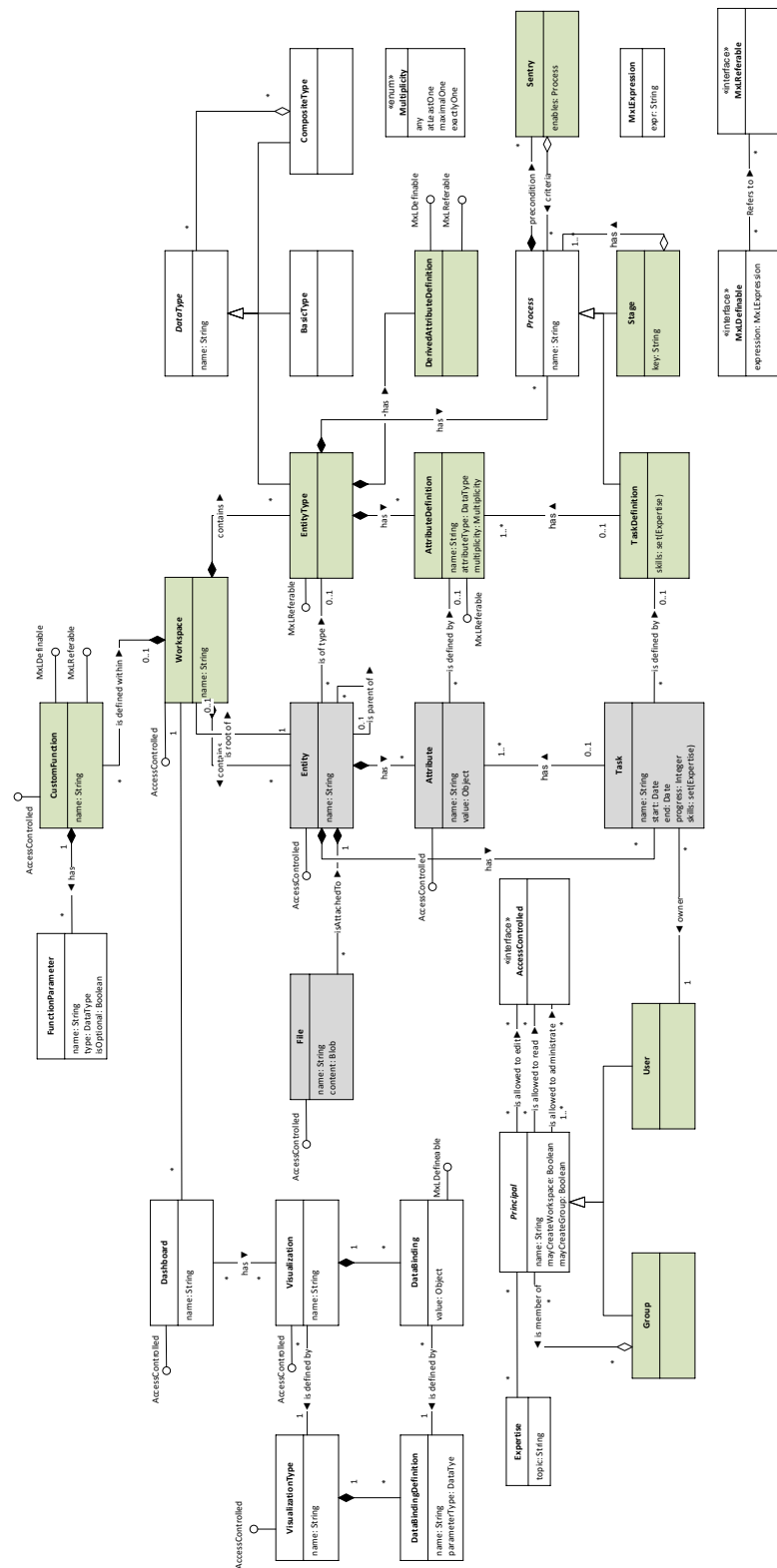


Figure 3.1.: Integrated Model

3.2. Behavioural Model

In this section the behavioural model is explained class by class in detail with a special emphasis on the CRUD operations and their effect on other elements. The classes discussed are "Workspace", "EntityType", "AttributeDefinition", "CustomFunction" and "TaskDefinition", in this exact order. At the end of each part a table shown a summary of the behavioural model of the explained class.

The access hierarchy is defined as follows: Administrator > Writer > Reader. Having administrator rights also includes write and read access and being able to write also includes read access.

Workspace

To be able to create a workspace, the attribute "mayCreateWorkspace" of the currently logged in user must be set to true. Creating a workspace has several consequences, namely the following: First of all the new workspace is created. Secondly a new entity of the type "Text Page" is created and set as the home-page of the new workspace. Lastly the creator of the workspace is also set as administrator for it.

To be able to see and read a workspace, the user needs at least explicit read access to the workspace. Reading a workspace has no consequence for it.

To update a workspace the user needs explicit administrator rights for this exact workspace. The consequences are depending on the element being updated. If the name of the workspace is changed, there are no consequences for other elements. Changing the home-page has two consequences. On the one hand is the new home-page the new landing-page when accessing the workspace and therefore not deletable and on the other hand losses the old home-page its status and therefore becomes deletable. Updating the access rights for a workspace can have different outcomes. If all settings are on default, the user gets administration/write/read access to the workspace and all its elements like entities, functions or files. But if one of the elements does not inherit the workspace settings, then the user will not get access to this particular element, resulting in a possible scenario, where the user only gets access to the workspace alone but not to any of its elements, if no element inherits the access rights. However this can be overruled by resetting the rights for all Entities within the workspace.

To delete a workspace, the user has to have administrator rights to the workspace. A deletion of a workspace also includes deleting all its entities, files and functions.

Operation	Precondition	Consequence
Create	mayCreateWorkspace = true	workspace created Entity home-page created
Read	reader of workspace	none
Update	admin of workspace	name: Namechange access rights: Change applied to workspace and all its elements with default settings
Delete	admin of workspace	workspace and all elements within are deleted

Table 3.1.: Behaviour Workspace Summary

EntityType

To create an entitytype within a workspace, the user needs administrator rights to that particular workspace. Creating an entitytype has the consequence, that a new entitytype with the given name is generated.

The visibility of entitytypes is depending on the users rights. If the user is a reader of the workspace, he or she can see the entitytypes within this workspace as well, but restricted to a list of instances and versions of the type. To be able to see the settings of a type, the user needs administrator rights for the workspace.

When the user is an administrator of the workspace, he can also update all entitytypes within that workspace. Renaming the entitytype has the consequence, that the type gets its new name and all entities of this particular type keep the entitytype with the new name. The same behaviour is valid for changing the plural name. Updating the setting, whether an entitytype allows free attributes or not has the following consequences: If the function was disabled and gets enabled, the users are from now on able to create free attributes for all entities of this type. If the function was enabled and gets disabled, the users can not longer add new free attributes, but the old ones will not get deleted and therefore stay and may cause unwanted inconsistencies within the different entities. A similar behaviour occurs for changing the name generation pattern for the title of a page. If the pattern is changed, the title of already existing entities will remain the same, but a warning will be displayed to point out the inconsistency and after changing the pattern the user can apply it to all pages via the settings of the type.

To delete an entitytype, the user again needs administration rights for the workspace. Deleting an entitytype however does not delete all its entities as well. They only loose their Type and become entities of the type "Text Page". While an entity has the type Text Page, none of their attributes will be displayed, even-though they still exist and will reappear when the entity is assigned to another type. For SocioCortex a second variant, where the entities are also deleted, when their type is deleted is also possible, as they might be more tightly connected to their type and entities without a type loose their relevance for the system. Furthermore all attribute definitions, derived attribute definitions, task definitions and stages of the type are deleted as-well. Attributes referring to the deleted type loose their constraint and only keep the limitation to link to another entity. MxL expressions using this entitytype might become invalid.

Operation	Precondition	Consequence
Create	admin of workspace	Entity Type created
Read	reader of workspace (limited) admin of workspace (full)	none
Update	admin of workspace	name: Entities get type with new Name enable free attributes: Only for new attributes, already existing attributes stay
Delete	admin of workspace	Entities of the Type loose their Type and become a "Text page". Their attributes are no longer visible all Attribute Definitions, Tasks etc. of the type are deleted Attribute Definitions referring to the type loose their constraint

Table 3.2.: Behaviour Entity Type Summary

Attribute Definition

To create an attribute definition for an entity, the user needs administrator rights for the entities workspace. Creating a new attribute definition only needs a name to be valid. After creating the definition, the attribute is added to every entity of the definitions entity type, but with no value set. The name of the attribute definition has to be exclusive, as an entitytype cannot posses multiple attribute definitions of the same name.

To be able to see the attribute definition, the user needs to be an administrator of the entitytypes workspace. As a writer or reader of the workspace or an entity using the attribute definition, the user can only see the attributes name and value, but not the definition.

To perform updates on an attribute definition, the user again has to have administration rights for the workspace. The different update actions explained in this section are: Changing the name of the attribute definition, changing its multiplicity and setting a new attribute type. When updating the name of an attribute definition, the name is changed on an entity level for all attributes as-well, but they keep the value of the attribute. Changing the multiplicity can have different outcomes. When changing from a strict multiplicity, like "exactly one value" to a more general multiplicity as for instance "at least one value" no problems occur on an entity level, as all attributes following the old definition will still be compliant with the new definition as-well. Problematic can be a change the other way round, where the new multiplicity is more restrictive or has other restrictions than the old one. The change of the definition will not affect the attributes already having values. For example, when an attribute with multiplicity "at least one" has two values and the multiplicity of the definition is changed to "exactly one" then the attribute will keep both values, but display a warning message, that it is not compliant to the definition. Updating the type of an attribute definition also does not change already instantiated values of that attribute, but displays a warning message.

To delete an attribute definition, the user again needs administrator rights for the workspace

of the definitions entitytype. Deleting an attribute definition removes the definition and the user can choose to delete the values of the attributes or keep them. MxL expressions using that attribute definition might become invalid.

Operation	Precondition	Consequence
Create	admin of workspace	Attribute Definition created Attribute added to Entitites without value
Read	admin of workspace	none
Update	admin of workspace	name: attributes also change multiplicity: violating values stay (with warning) type: violating values stay (with warning)
Delete	admin of workspace	Attributes become free attributes even-though the Type might not allow free attributes

Table 3.3.: Behaviour Attribute Definition Summary

Custom Function

As custom functions are not connected to many other classes in the meta model (see Figure 3.1), their behavioural model is also not that complex.

To create a custom function, the user has to be an administrator of the workspace, where he intends to create the function. A name and the method sub with executable MxL code are required to create a custom function.

To be able to see the custom functions of a workspace, the user needs to be an administrator of the workspace. Reading a custom function has, of course, no consequence.

As an administrator of a workspace, the user is also able to update all its functions. Updating the method sub has the consequence, that on the one hand the implicit return type can change and therefore MxL expressions, where the function is used can become invalid on the other hand the outcome can change and also invalidate expressions using the function. Therefore the user should be fully aware, where the function is used before updating its expression. For recursive functions the explicit return type is of high importance, as the system cannot compute an implicit one. Changing it can have the same consequences as changing the method expression itself and because of that is also a task, which has to be handled with care.

To delete a custom function, administration rights for the workspace are required. As deleting has the consequence, that the function cannot be used in other expressions any more, the user has to make sure to correct the expressions, which had used the function afterwards.

Operation	Precondition	Consequence
Create	admin of workspace	Custom Function created
Read	admin of workspace	none
Update	admin of workspace	expression: may cause new inferred type + invalid code expecting the old outcome explicit type: may cause invalid code expecting the old type
Delete	admin of workspace	other code using the function may become invalid

Table 3.4.: Behaviour Custom Function Summary

Task Definitions

As for custom functions the user needs administration rights for a workspace to generate task definitions. Adding a new task definition to the system will automatically add its task to newly instantiated entities of the definitions type but not to already existing entities of the given Type.

To see task definitions, the user also needs administration rights. But to see a task, the user only needs read access to the entity using it. Reading a task definition has no consequences.

There are various things that can be updated within a task definition, if the user is an administrator of the workspace. The three possible changes discussed in this section are: The task definitions name, its attributes and its preconditions. Changing the name, of course, changes the name of the definition and of all future tasks of this definition, but not of the already instantiated tasks, which can lead to major inconsistencies depending on the frequency of name changes. The same behaviour is also true for adding or deleting attribute definitions to the task, they will not update for already existing entities. The corner case of deleting the last attribute definition of a task does not delete the task definition, but results in empty tasks, when a new entity is created. The third possible change is adding or removing preconditions to the task. This can be done in two different ways, adding an "or" precondition or an "and" precondition. The first resulting in creating a new sentry with the task definition as "enables" and the new precondition as "criteria" and the second resulting in the precondition being added to the existing sentry. Both updates take no effect on already instantiated tasks whatsoever. As this behaviour differs strongly from update operations on all the other elements described in this section it should be revised and maybe adjusted.

To delete an task definition, the user also needs to be an administrator of the workspace. Deleting a task definition can have multiple consequences. Firstly, unlike the update operations, all instantiated tasks of this definition are deleted. Secondly the definition is removed from all preconditions, which can result in the deletion of a sentry, if it was the only element in criteria of the sentry. Thirdly the task definition is removed from all stages and lastly all sentries, which enabled this task definition are deleted as-well.

Operation	Precondition	Consequence
Create	admin of workspace	Task Definition created Tasks not added to existing Entities
Read	admin of workspace	none
Update	admin of workspace	attributes: no changes for existing Tasks precondition: adding Sentry or modifying Sentry
Delete	admin of workspace	Tasks are deleted aswell, Task Defintion removed from Sentries and Stages

Table 3.5.: Behaviour Task Definition Summary

Part IV.

Prototype Implementation

4. Prototype Implementation

In the following chapter the prototypical implementation is presented. Therefore the first section explains the technical foundation. The second part is about the overall architecture of the prototype and the third part exemplifies the core features, which are part of the answers to the second research question stated in Section 1.3.

4.1. Technical Foundation for the Prototype

SocioCortex, sc-angular and the Angular Material framework are all tightly connected to the prototype and therefore explained in the following.

4.1.1. SocioCortex

SocioCortex is a software project of the chair "Software Engineering for Business Information Systems (Sebis)" at the Technical University of Munich. They describe SocioCortex as "The Social Information Modelling Platform for Collaborative, Evolutionary Data and Process Management" ¹. It is the successor of Tricia, which was also developed by the Sebis chair. SocioCortex therefore also uses the Hybrid Wiki approach introduced in Section 1.1.1. The whole software ecosystem around SocioCortex is visualized in Figure 4.1. On the application side, there is the default client suite and the vertical applications. In addition to these there are also content sources and identity providers connected to SocioCortex. Within the default client suite the applications modeler, content manager and visualizer are located. Vertical applications among others Spread-sheet 2.0 and Lexalyze. The easy accessible interface via REST API and available tutorials also encourages people to implement their own applications on top of SocioCortex. The prototype of this thesis is the modelling application within the default client suite. The REST API of SocioCortex is accessed via sc-angular, which is explained in the following subsection.

¹Source: <http://www.sociocortex.com/>



Figure 4.1.: SocioCortex Architecture [21]

4.1.2. sc-angular

Sc-Angular is tightly connected to the REST API of SocioCortex and enables web applications to access all API commands via angular-js calls. The documentation of the API can be found on the SocioCortex-homepage: <http://www.sociocortex.com/documentation/> and sc-angular itself can be downloaded via github (<https://github.com/sebischair/sc-angular/> Open Source). It is mainly developed by the sebis chair and therefore updated on a regular basis. It consists of seven modules all addressing a different part of the API. The first module is called scAuth and responsible for the authentication process with the server. Via scData the data side of SocioCortex can be loaded, namely Workspaces, Entities and Attributes. scModel provides read and write functions for the data-model side. This includes EntityTypes, AttributeDefinitions, DerivedAttributeDefinitions and Functions. A sub-part of scModel (scModel.processes) enables access to the process part of SocioCortex, described in Section 2.4. The forth module is called scMxL and provides various functions for MxL. scPrincipal is responsible for user and group management, scSearch accesses the SocioCortex search engine and scUtil provides some Utility functions. The most important parts of sc-angular for the prototype are highlighted in Figure 4.2.

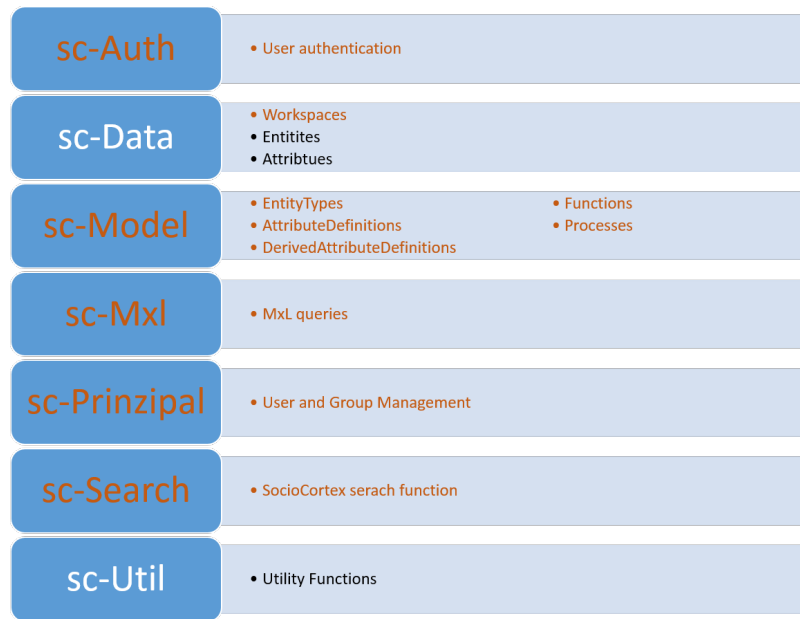


Figure 4.2.: sc-Angular Architecture based on [21]

4.1.3. mxl-angular

The mxl-angular framework, developed by the sebis chair, provides functions to work with MxL code and to generate class diagrams with the use of MxL. Furthermore it also includes some utility functions. Mxl-angular is open source and can be accessed via git hub: (<https://github.com/sebischair/mxl-angular>).

The two main elements used in this prototype are the mxl-editor and the mxl-model view. The first provides an editor view for MxL-code, with auto completion hints, code validation and code execution and is therefore a useful tool to write MxL-code. It is used for the definition of Functions on a workspace level and Derived Attribute Definitions as shown in Section 4.3.3. The mxl-model view is used to display the UML class-diagram of a workspace (Section 4.3.1) and to enhance the coding of MxL for Functions and Derived Attribute Definitions (Section 4.3.3).

4.1.4. Angular Material

“For developers using AngularJS, Angular Material is both a UI Component framework and a reference implementation of Google’s Material Design Specification. This project provides a set of reusable, well-tested, and accessible UI components based on Material Design.” [8] The prototype uses Angular Material as basis for its design and almost all of the 36 style components find use within the implementation. The most important of these are: List, Card, Input, Button and Dialog. Implementation demonstrations can be found on the angular material homepage (<https://material.angularjs.org/>) and within the next pages of this chapter, where the core features of the prototype and implementation details are explained (Section 4.3).

4.2. Overall Architecture

The prototype is divided up into 29 components all listed in Figure 4.3. Each component is easily reusable within the prototype. The components each consist of a html file, responsible for the layout and a javascript directive, managing the data, functions and API calls. As stated above, all html files follow the Angular Material guidelines, with some adoptions to provide all needed functionalities.

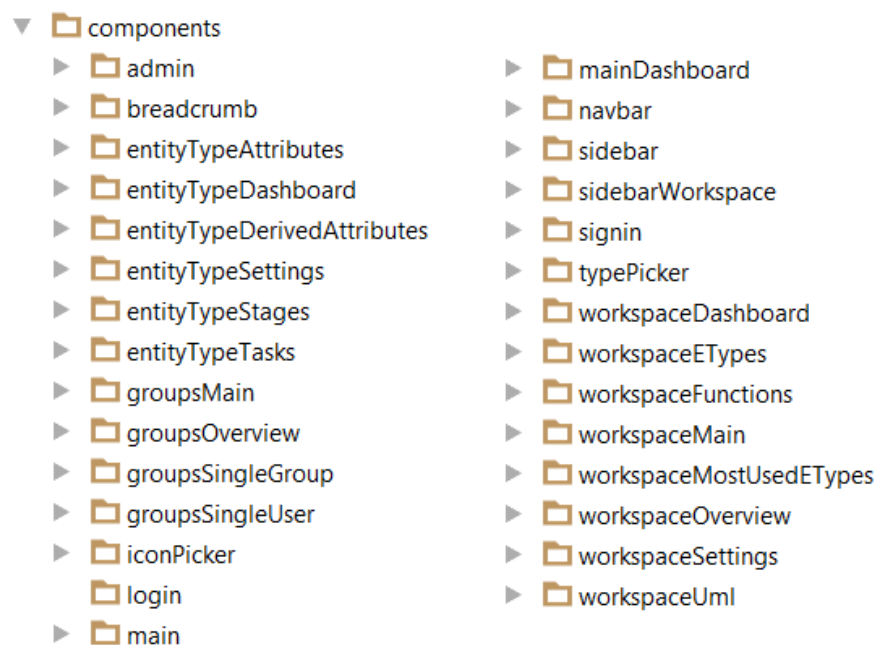


Figure 4.3.: Components of the prototype

To navigate through the application, ui-router with different states is used. A total of 11 states are implemented in the prototype and visualized in Figure 4.4. "Home" and "WorkspaceMain" only provide the skeleton for their sub-states and are never addressed without a sub-state. Therefore they are marked in grey. The layout of the states "Home" and "WorkspaceMain" are shown in Figure 4.5. The area ui-view marked in grey is the area where the sub-states are displayed. The default sub-state for "home" is "MainDashboard" and for "WorkspaceMain" it is "WorkspaceDashboard".

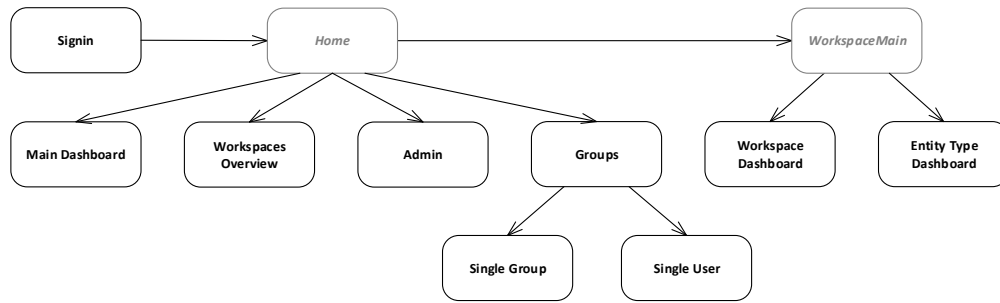


Figure 4.4.: State-Hierarchy of the prototype

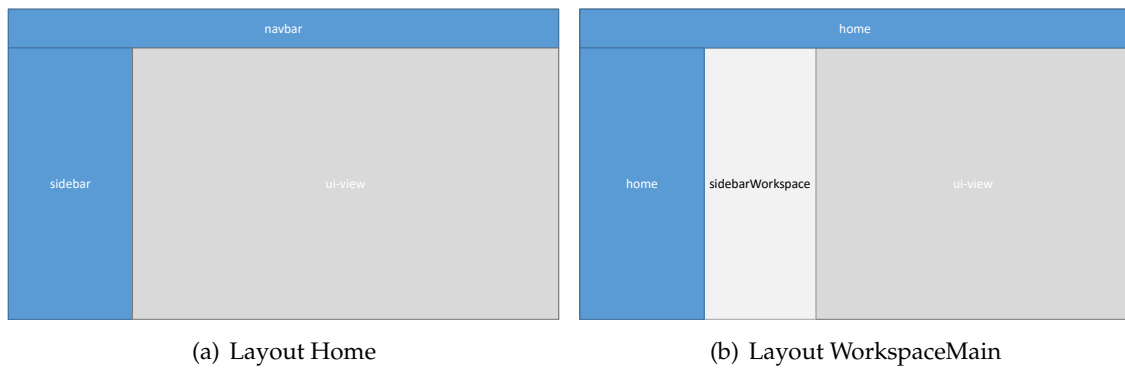


Figure 4.5.: Layout Home and WorkspaceMain

4.3. Core Features

In this section, the features and functionalities of the prototype, which refer to the research question in Section 1.3 are explained in detail. The Workspace Dashboard and Attribute Definitions explain the data side, functions are covered by the Derived Attribute Definitions and task by Tasks. The role model is not mentioned, as it is integrated into the other parts.

4.3.1. Workspace Dashboard

The Workspace Dashboard includes several components to analyse the selected workspace on different levels of detail. A screen-shot of an example workspace is shown in Figure 4.6

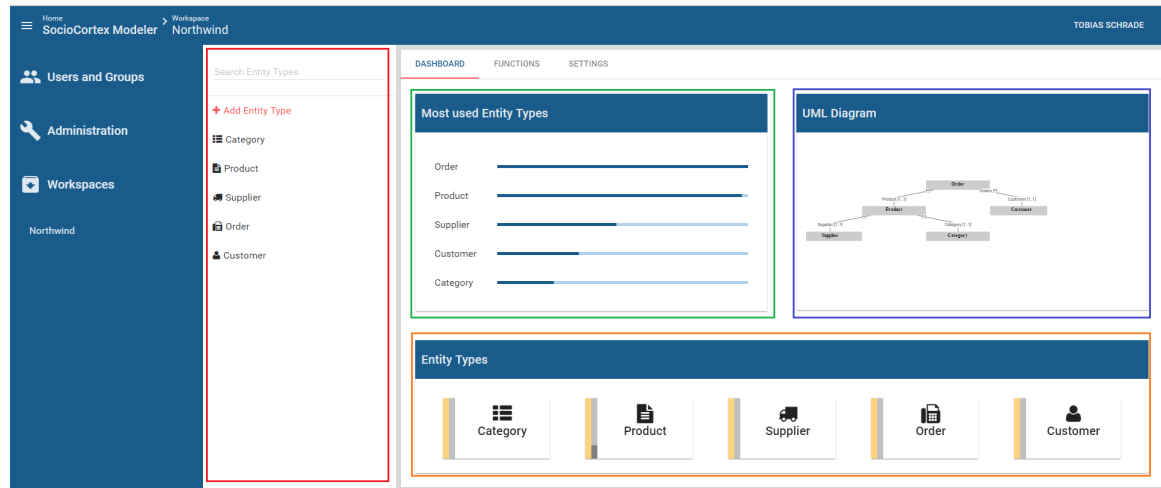


Figure 4.6.: Screen-shot Workspace Dashboard

The Workspace Dashboard can be divided up into 4 main parts, marked by different frames in the figure.

The white sidebar (red frame on the left) facilitates navigating through the various EntityTypes of the selected workspace. A input field at the top thereby filters the Types based on their names, to facilitate working with bigger workspaces using more then 20 different Entity Types.

Framed in green are the 5 most used Entity Types of the workspace. The bars are always adjusted in a way, that the most used Entity Type has a full bar. The user can get a quick overview of the importance of a Type by looking at this area.

The third part uses the package mxl-angular, explained in Section 4.1.3 to display a UML class-diagram of the workspace. Within the diagram the Entity Types of the workspace are shown as classes. The references between the classes are build out of their Attribute Definitions. In this workspace an Order always has exactly one Customer attached to it via the Attribute Definition Customer with multiplicity "Exactly one value".

Lastly the orange area again shows the EntityTypes of a workspace. This time the data is enriched with 2 values shown in the bar diagrams. On the one hand, the orange bar visualises the consistency of all instances of the given EntityType. A full bar resembles low and an empty bar high consistency. The meaning of consistency in this case is, that the instances follow all the rules given by the definitions of the Entity Type. On the other hand, the grey bar indicates the structuredness. Empty bar symbolises high and a full bar low structuredness within all instances of the Type. Structuredness is a metric to show, whether the instances implement a lot of free attributes or not and is an indicator, where re-factoring might be needed.

4.3.2. Attribute Definitions

The view "Attribute Definitions" provides all functionality needed to add, delete and modify attribute definitions of an entitytype. A screen-shot is displayed in Figure 4.7.

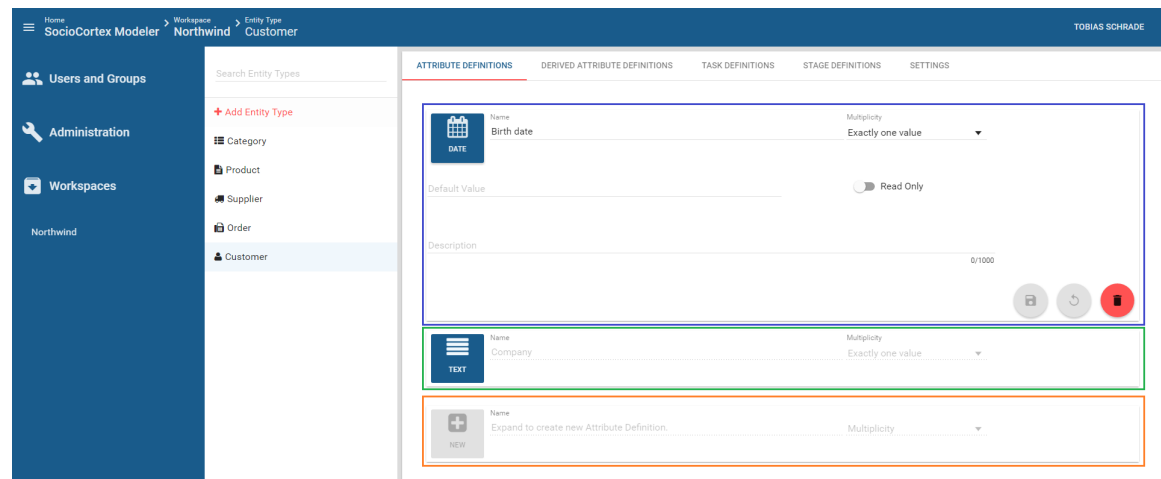


Figure 4.7.: Screen-shot Attribute Definitions

To modify the attribute definitions of an entitytype they are all displayed within a list of angular material cards. The content of the cards is adjusted depending on the Type of the attribute definition.

Within the blue frame, an opened attribute definition is shown. The various input fields are activated, so the user can change the parameters of the definition. When the definition differs from the one stored on the server, the save and reverse buttons in the bottom right corner are enabled and the user can either store or cancel his changes. Depending on the attribute type, different input fields are shown. For enumerations, for example, a new field named enumeration values is displayed to define the different values the instantiated attribute can have.

To change the Type of the attribute definition, clicking the blue button in the top left corner opens the type picker, shown in Figure 4.8 as a pop-up. All available types are displayed in 4 columns. The first column contains the entity types defined for the current workspace and a search bar to look for types in different workspaces utilizing sc-search. The second column, containing the generic link types, is for general references to other objects of all kinds. The third for basic types, like "Text", "Number" or "Date". The last column only contains the Type "Any Type", only used if no restriction is applicable to the Attribute. Due to the fact, that the Type Picker has its own template and controller, it can easily be adjusted in the future, if new Types are introduced to SocioCortex.

The green frame surrounds a minimized attribute definition, which is its normal representation. In this state, the definition can not be changed. The minimized view is needed to grant the possibility to see several definitions without scrolling the page. This is important, because the order of the attribute definitions can be changed via drag and drop of the cards and scrolling makes this harder to use. To enlarge the card a button appears when hovering it.

4. Prototype Implementation

The last card, framed in orange, represents a new attribute definition, which can be added to the entity type. To save a new definition on the server, the card needs to be enlarged and a type and name have to be set, otherwise the save button remains disabled as these two values are required by SocioCortex.

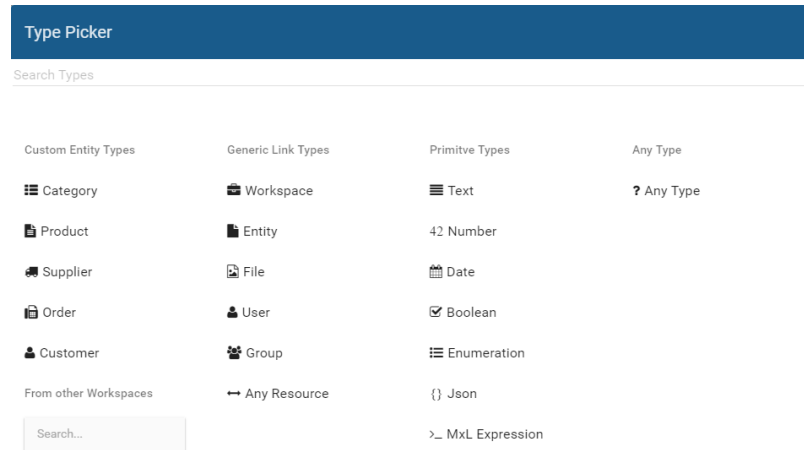


Figure 4.8.: Screen-shot Type Picker

4.3.3. Derived Attribute Definitions

This section describes the page to create, edit and delete derived attribute definitions. A screen-shot of the view is shown in Figure 4.9.

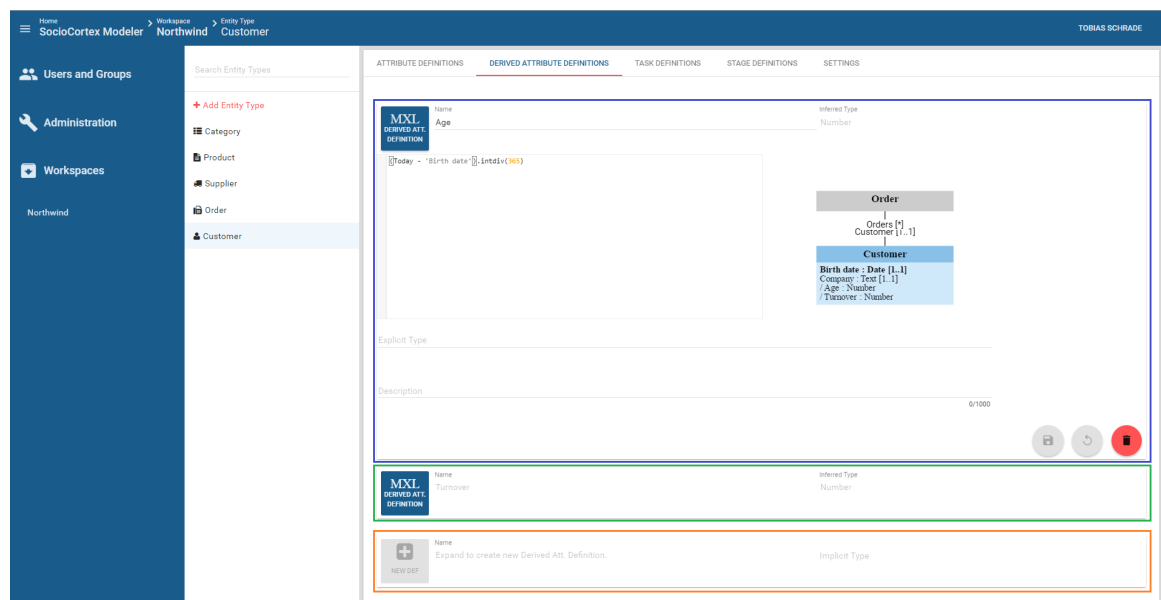


Figure 4.9.: Screen-shot Derived Attribute Definitions

The structure of the tab is similar to the "normal" attribute definitions. They are ar-

ranged in a list of cards and can be expanded to enable editing (blue framed part). The special part about derived attribute definitions are the mxl-editor and the mxl-model view. As described in Section 4.1.3 both are part of the mxl-angular framework. The important part hereby is, that the editor and the model view are linked together. That means, depending of the MxL-expression in the editor, the used entitytypes and attributes are highlighted in the model. In this case, the expression calculates the age of a customer based on his birth date via the expression `"(Today - 'Birth date').intdiv(365)"`. Thus the attribute definition birth date and the entity type customer are highlighted. The adjacent classes in the model are still visible to offer an overview over the other attributes, which can be used within the expression.

Another important singularity of derived attribute definitions is the fact, that they have an inferred type and an explicit type. The first one is calculated based on the result of the MxL-expression, but due to the fact, that this calculation is not possible for recursive expressions, the user is also able to define an explicit type, in case he uses recursion.

The green part is again the minimized representation of an derived attribute definition, which is the standard for all definitions when navigating to the tab for overview reasons.

The orange cards function is to create a new derived attribute definition. The expanded card also features the mxl-editor and mxl-model view as described above.

4.3.4. Tasks and Stages

Tasks and stages are the last two pages described in this section. The two views also incorporate sentries, but they are handled in the background. A screen-shot of the view to edit task definitions is shown in Figure 4.10.

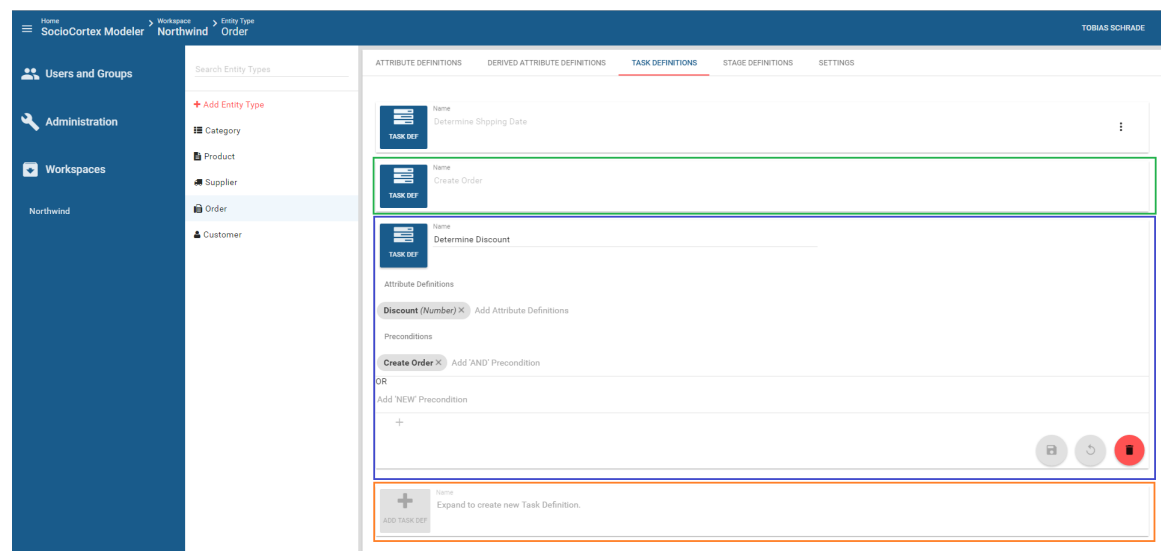


Figure 4.10.: Screen-shot Tasks Definitions

The blue frame surrounds a maximized task with its 3 main attributes: The tasks name, the attribute definitions included in the task and its preconditions. The name is just the name of the task and has no further meaning. The attribute definitions are responsible for

4. Prototype Implementation

the completion of a task on the instance-level. All attributes listed within the task definition need to be filled out in the entity to complete a task. Furthermore is it only possible for a definition to appear in one task within the entity, otherwise the system does not work properly. To avoid this, the attribute definitions are filtered for each task definition and can only be added via auto-completion of the filtered list. The preconditions are a simplification for the user, as they are represented by sentries in the back-end. Essentially each line of preconditions (which can be tasks or stages) is saved as one sentry. All preconditions of one sentry are connected by the logical and. Adding preconditions to the second line results in another sentry being added to the task, which is connected by the logical or to the other Sentries. As the tasks can be arranged in a hierarchy, the preconditions also use auto-completion and task specific lists to avoid illegal inputs (for more details about this, see below).

Framed in green is a minimized task definition and marked in orange is again the card to add a new task definition to the entitytype. Adding a new task definition is also following the same rules for attribute definitions and therefore has its own list of available attribute definitions.

To structure and group task definitions the view stages is used, which is shown in Figure 4.11.

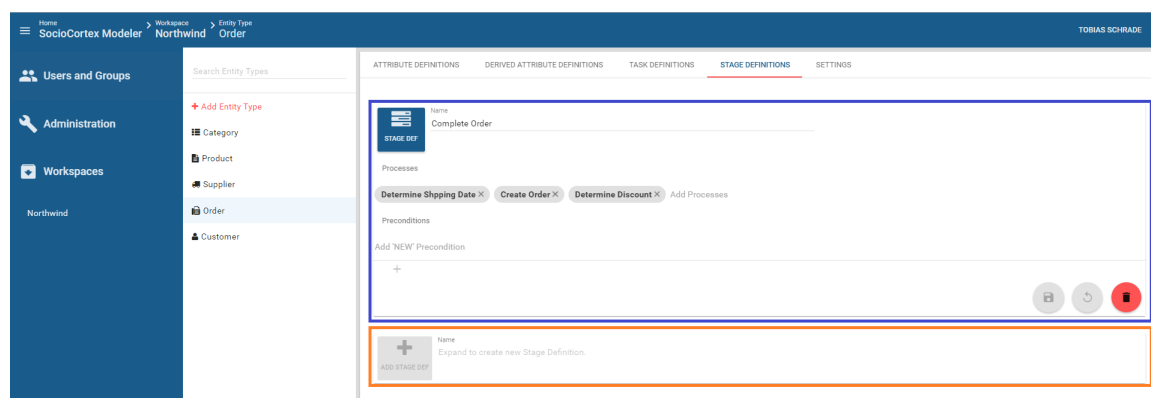


Figure 4.11.: Screen-shot Stages

The orange box is again the field to create a new stage and a minimized stage is not visible in the screen-shot. The maximized card of a stage is framed in blue. Besides the name a stage has processes and preconditions. Processes are either stages themselves or tasks. In the example the stage "Complete Order" consists of the processes "Determine Shipping Date", "Create Order" and "Determine Discount". When all these processes are completed, the stage itself is completed. The structure resulting of the assigned processes to stages is determining which process can be a precondition of which task or stage. To explain the rules behind it, we look at the hierarchy shown in Figure 4.12. Stages are not allowed to have any process below or above them as a precondition. In this case, stage 2 can only have task 1 or task 2 as preconditions and stage 1 none. For task definitions the rule is not as strict - only processes above them are not allowed as preconditions. Task 3 therefore can have task 1, 2 or 4 as precondition but not stage 1 or 2. It is possible to get circles of preconditions. They are not (yet) checked by the system and the user has to take

care of that.

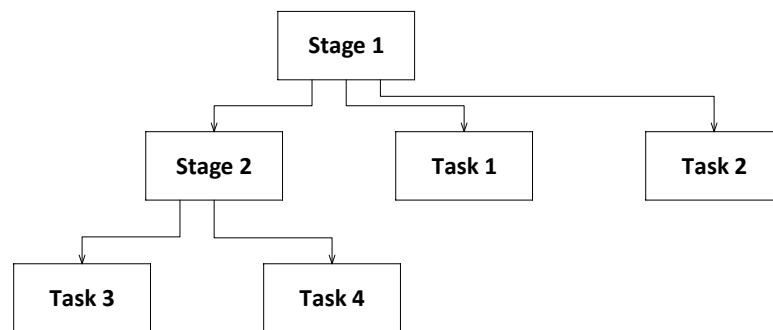


Figure 4.12.: Example Stage Hierarchy

Part V.

Evaluation

5. Evaluation

To find weaknesses and identify further possibilities for the prototype an evaluation was conducted. The methodology, evaluation group, questionnaire and results are described in the following section.

5.1. Evaluation Approach

In one evaluation session both, the implemented prototype presented in this thesis (modeler) and another prototype, a generic content manager for SocioCortex, are evaluated. One session is divided up into 6 major steps and takes about one hour time. The steps are: The introduction, explanation of the scenario, evaluation of the modeler, questionnaire for the modeler, evaluation of the generic client and questionnaire for the generic client.

To explain the purpose of the evaluation the participants are first introduced into the general structure of the evaluation with the two testing phases of the two prototypes and the two separate questionnaires. This is of particular importance to avoid a mix up in the answers of the questionnaires. Furthermore the think aloud idea was introduced to the participants. The method, based on the article "The think aloud method" by van Someren et. al. [26] advises the interviewee to always express his thought-process verbally and therefore grant a better understanding for the interviewer of what the interviewee is currently doing and what outcome he or she is expecting. This is particularly important to identify, which behaviour of the prototype is not intuitive for the participant. Another point of the introduction is, to make sure that the interviewee understands, that the system is tested and not he or she and critical feedback is welcome. Lastly the interviewee is asked, if it is ok for him to record audio and the screen of the testing laptop.

In the second part of the evaluation, the scenario is explained. The participant gets the scenario in written form and has time to read it. Afterwards open questions concerning the scenario are answered. The whole scenario is attached to Appendix A.

After getting familiar with the scenario, the participant gets the 5 tasks, which should be done within the system. They are again written down and if the instructions are unclear, questions are answered. While the participant is fulfilling the tasks, the interviewer does not interrupt him, to avoid distraction and falsification of the users experience of the prototype. Due to time constraints the given tasks do not cover all functionalities of the system, but the most important ones for later all day use. The written form of the tasks are again in Appendix A.

To be able to collect further information a questionnaire is handed out to the participant after he or she did all the tasks. It is further explained in Section 5.1.2.

Part five and six of the evaluation are concerning the other evaluated prototype. The participant again works on five tasks based on the same scenario and afterwards fills out the questionnaire.

To round up the evaluation there is an open exchange of thoughts about the two prototypes in the end.

5.1.1. Evaluation Group

The evaluation group consists of 6 persons, all familiar with the hybrid wiki model. Three people are PhD students of the sebis chair at TUM and 3 are practical partners of the chair. This distribution was chosen, to cover both, the industrial and research sector equally as both sectors can provide different inputs for the prototype. Industrial partners to check the practical viability of the prototype and research assistants to also double-check the theoretical background and utilized methodologies. The practical partners work as enterprise architects for three different companies in Munich. One for a big bank (> 7.000 employees), one for an association in the healthcare sector and the last one for a IT-consulting company with over 70.000 employees worldwide.

5.1.2. Questionnaire

The questionnaire used to collect the evaluation results consists of two parts. The first part is the standard usability scale [3]. It incorporates 10 standardised questions concerning the usability of web and other applications. The ten questions are:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The questions have to be answered on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). The statements alternate between a positive and a negative statement, which results in five positive and 5 negative statements. "This was done in order to prevent response biases caused by respondents not having to think about each statement." [3].

The second part of the questionnaire consists of 3 qualitative questions concerning the prototype. Their goal is it to get explicit points where the prototype can be improved and what functionalities are missing. The whole questionnaire is shown in Appendix B.

5.2. Results

The results of the evaluation are divided up into the results of the standard usability scale questions and the qualitative results from verbal and written feedback from the participants.

SuS Results

To calculate the final SuS score, the points for each question need to be normalized using the functions $X-1$ for the odd questions and $5-x$ for the even questions. The sum needs to be multiplied by 2.5 to get a score between zero and 100 [20].

The mean SuS score per question and the average final score are shown in Table 5.1. The final score of 72.5 points can be considered as above average regarding Sauro (2011) and Bangor et. al.(2009) [2, 20]. Sauro considered 500 evaluations, which resulted in an average score of 68 and Bangor et. al. had a look at 1433 web application evaluations with a mean score of 68.2.

Remarkably positive is, that the users did not need to learn a lot of things to get going with the system and also felt very confident using the system. The question which got the worst results is "I would imagine that most people would learn to use this system very quickly". Talking with the interviewees about this particular question revealed, that its not only the systems design fault but also the fact, that a new user needs to learn a lot about hybrid wikis and the underlying meta model to get going in the system.

5. Evaluation












Question	Average SuS (0-4)	Graph
I think that I would like to use this system frequently.	2.83	
I found the system unnecessarily complex.	3.00	
I thought the system was easy to use.	2.66	
I think that I would need the support of a technical person to be able to use this system.	3.00	
I found the various functions in this system were well integrated.	2.83	
I thought there was too much inconsistency in this system.	2.83	
I would imagine that most people would learn to use this system very quickly.	2.16	
I found the system very cumbersome to use.	3.00	
I felt very confident using the system.	3.33	
I needed to learn a lot of things before I could get going with this system.	3.33	
Average SuS score (0-100)	72.50	

Table 5.1.: Average Results from SuS Questions

Qualitative Results

The qualitative results can be summed up into three mayor findings: Some minor design changes could improve the usability of the prototype, as most testers complained about the same problems, comparing to Tricia the separation of modeling and generic client is perceived as a good step and introducing Tasks and MxL is considered useful, especially for practical application. All in all the results were very positive and everyone agreed, that the implementation is going the right way.

Design-wise the following points got criticised the most. Adding a new Attribute Definition, Derived Attribute Definition or Task Definition is not intuitive enough. This was remarked by both groups of participants equally. Five out of the six participants tried to enlarge the card for the new entry by just clicking it and not the three dots at the right end of the card. The next problem was, that it was not clearly indicated where to click to change the type of the new Attribute Definition and it was not displayed good enough, which input fields need to be filled out to be able to save the new Definition.

Selecting the Type was not very intuitive for the interviewees as-well. The first hurdle thereby was finding the correct button to open the type picker and the second one finding the right type. This was especially true for a generic type. One participant stated, that the

left column (where the generic types are listed) looked different and therefore he did not search the type there.

The order of the input fields got some critics, because the name is more important than the type for the participants, but in the cards the type is on the left, followed by the name. Another problem with the cards for two people was the colour of the text. They said, that the text should be darker to be more easy to read.

For the Task Definition view the remarks where, that the plus to add a new Sentry is misleading. Another finding is, that not all users are used to the material "chips" element, used to add Attribute Definitions to a Task. Four participants needed a longer time to find out where to add them and even needed help from the interviewer.

More general remarks where, that inputs should always be able to be submitted by pressing "Enter", that the material design guidelines are not always followed and therefore there are minor inconsistencies within the design. Also the user interface could be optimized by adding more hints and toasts to the buttons and actions performed, as-well as minimizing the mouse path between buttons, which are usually used after another.

5.3. Possible Improvements of the Prototype

Based on the findings above there are different possible improvements of the prototype. Besides the already mentioned changes, three new mock-ups are therefore presented in the following sections.

Introduction Floating Action Button

The floating action button (FAB) is a design element of material design, which is used by many web applications like the "gmail" client and "whatsapp web". It is a button, which is always displayed once per view, mostly in the bottom right corner and is the central button to create something new.

This button can be used for all views within the prototype, where the user is able to add a new element and should help to providing a consistent user experience. In order to not confuse the users, no other button should look like this FAB and therefore some buttons need to be redesigned. An example of the prototype with the new button is shown in Figure 5.1.

5. Evaluation

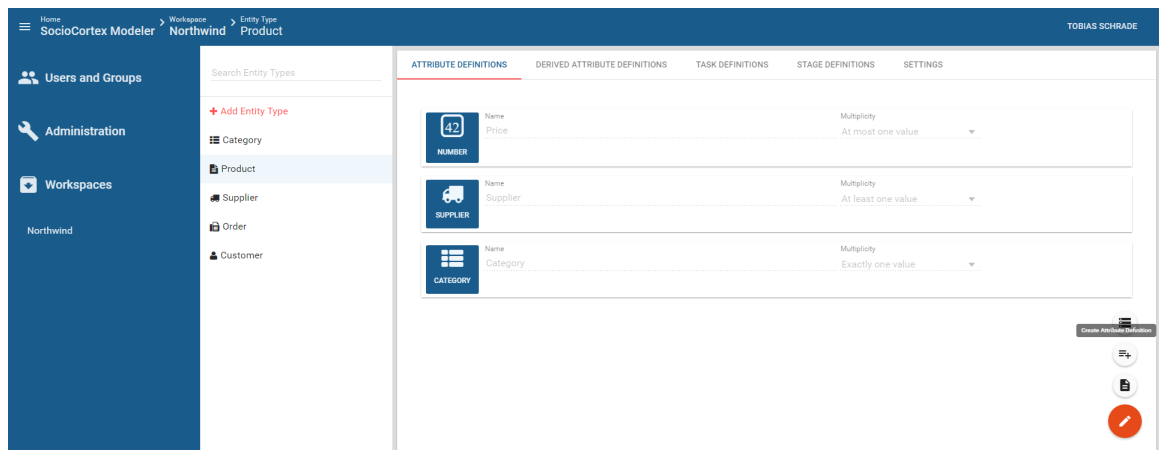


Figure 5.1.: Floating Action Button Mock-up

New Card Layout

As the cards displaying the Attribute Definitions were criticised, Figure 5.2 shows a possible new card layout featuring a new order and a new text colour.

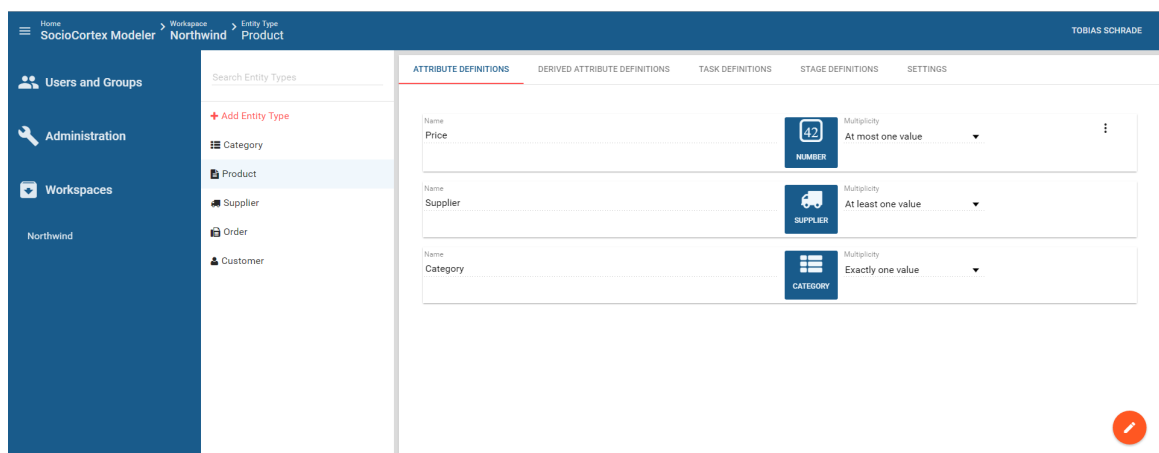


Figure 5.2.: New Card Layout Mock-up

New Task Definition Layout

To improve the Task Definition Layout hints should be added to the important input fields. Furthermore the Preconditions for a Task should be aligned differently. A possible new layout is shown in Figure 5.3. To further improve the layout, the auto-complete field could be replaced with select fields, but this is currently not supported by the angular material version used for the prototype.

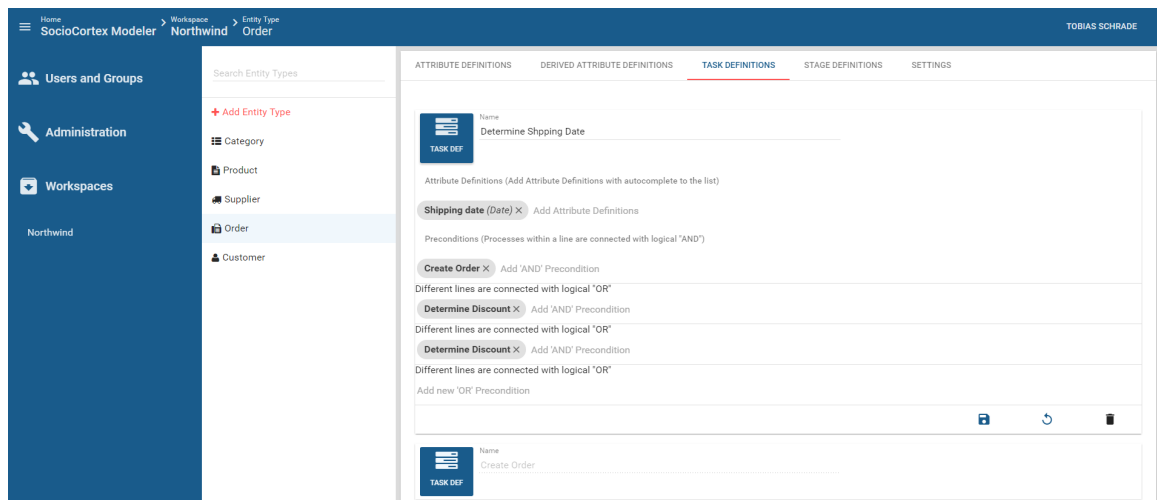


Figure 5.3.: New Task Definitions Layout Mock-up

Part VI.

Potential further Enhancement of the Model

6. Potential further Enhancement of the Model

This section highlights and describes four new ways, how the use of MxL could further improve the integrated meta model of Chapter 3 using the university environment introduced in the evaluation (Part V).

6.1. MxL for (default) access rights

Default access rights to Entities, Attributes or Tasks might be defined via MxL.

6.1.1. Entity

On an Entity-level this a possible example is: The EntityType "Masterarbeit" gets the additional information saved as MxL statement, that the Users saved as: "Supervisor", "Advisor" and "Student" automatically get administration rights for the entity. This simplifies and accelerates the process of adding new Entities of this EntityType by a lot, due to the fact that access control rights do not have to be set in an additional work-flow. In Figure 6.1 the users "Prof. Dr. Florian Matthes", "Thomas Reschenhofer" and "Tobias Schrade" would get administration rights for the Entity "Master's thesis Tobias Schrade" automatically.

Entity: „Master's thesis Tobias Schrade“
supervisor: „Prof. Dr. Florian Matthes“ advisor: „Thomas Reschenhofer“ student: „Tobias Schrade“

Figure 6.1.: MxL for default access rights on Entity-level

6.1.2. Attribute

Similarly as for the Entities this could also apply to attributes to provide certain users automatically access to a specific attribute of an Entity. One example could be a seminar with group projects. The students are assigned to different groups, which are stored as Attributes in an Entity that represents the whole group project. To hand in the results of the different groups, the members of a group get write access to a dedicated Attribute,

where they can upload their final results. To prevent cheating, they are not able to see the upload Attribute of the other groups. For example in Figure 6.2 the users "Thomas" and "Felix" only get automatically read and write access to the attribute "resultsGroup1" but cannot see the "resultsGroup2,3 and 4".

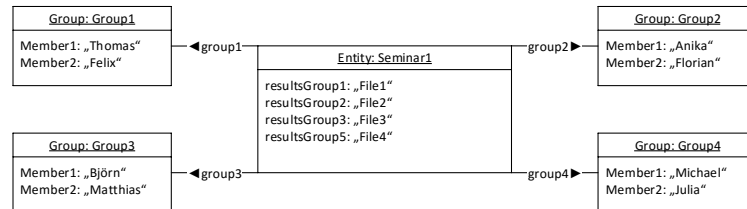


Figure 6.2.: MxL for default access rights on Attribute-level

6.1.3. Task

On a Task-level the functionality is basically the same. Referring to the seminar example from above (Figure 6.2), the Entity "Seminar1" can have multiple Tasks for each group. Again only the members of a specific group should be granted access to their task so they cannot see the progress of the other groups. In this example a task could be to upload the final results and attach them to the wiki page. In this case every group would have its own task.

6.2. MxL for default Values

Another possibility to enhance the meta model is by using MxL for default values of Attributes. Figure 6.3 displays the Entity "Master's thesis Tobias Schrade" but this time with the two new Attributes "start" and "end". As a master's thesis usually has a duration of half a year an MxL-rule automatically setting the end date to start + 6 months would speed up the process of generating the Entity with all its attributes. To set the end date as DerivedAttribute does not make sense, since it is possible to extend the time frame due to sickness or other reasons and DerivedAttributes cannot handle this by now.

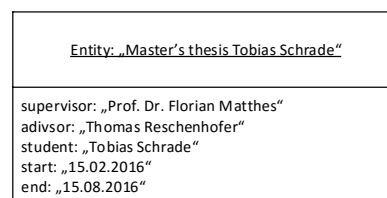


Figure 6.3.: MxL for default values

6.3. MxL for Constraints of Attribute Definitions

Besides defining default values for Attributes, MxL could also be used to define constraints for Attribute Definitions. In the current model is it not possible to formulate complex constraints for Attributes. Besides for enumerations, where the enumeration values can be set, all other types for an Attribute do not have any further constraints. A few examples, where MxL could help in this regard are: To define a number range for a specific attribute, with the university background in mind this could be the range of grades a student project can get (only 1-5). Another example is, that attributes which point to a user can only be filled with members of a specific group. An advisor of a master's thesis for example always needs to be a phd student of the chair and therefore a member of the group phd students and a supervisor has to be a professor and member of the according group.

These constraints function as a security mechanism to prevent false values. But keeping the thoughts of evolving data within hybrid wikis in mind, they should only throw a warning, if a value outside the defined range is entered, or at least need an option to either allow or forbid values outside the range.

6.4. MxL for Tasks and Sentries

The last area, where MxL has a wide spectrum of possible use-cases are Tasks and Sentries. At the current state Tasks are tightly connected to Attributes and when the users sets all Attribute tied to a Task, it is considered as completed. A similar thing is true for the enabling of a Task. A Task only can have other Processes and when they are completed, the Task gets enabled.

6.4.1. MxL for Completion of a Task

To increase the possibilities for Tasks, a first step would be to enable other goals then setting a predefined Attribute, because at the moment this is one of the most limiting factors of Tasks. With the help of MxL queries various other ways of completing a Task are possible. A simple example therefore is, that the free text of an entity needs to exceed a certain number of words to complete a task. The task "write abstract" for example could be completed, when the text is long enough. Another alternative for the completion could be when different attributes from varying entities exceed a certain value. This is more interesting for the industrial sector, where the sales person might get a bonus when he makes high profit for the company and therefore has customers with big orders.

6.4.2. MxL for Enabling of a Task

Besides Task completion, the enabling of a Task is also important, as stated above the options here are limited at the moment. One example to extend them could be, that a value for an DerivedAttribute needs to be bigger then a defined threshold. For example the Tasks "issuing a masters's degree" should start when the student has collected 120 credits or more. Tasks could also be linked to dates, for example the task "Buy Christmas present" should always start at the 1st December of a year.

6.5. User roles for Tasks

Besides the MxL functionalities mentioned above, User roles could be a nice addition to tasks as-well. The basic concept here is, that certain Tasks are only visible to a set amount users and need to be done by them. In practice this could have many different use-cases. One example would be within a review or approval process. The review should only be done by a certain group of users and therefore only be visible for them.

Another possible use-case for access control for Tasks is the simplification of Entities. By presenting all possible Tasks to all users, the amount of Tasks could get high and therefore the users might loose the overview over all tasks. This can be avoided by only showing relevant Tasks for a user. An practical example therefore is the purchase process in a big company. From creating an order till buying the product usually many people are involved and look at the order. The many different Tasks then could be confusing. Especially when the user should not complete one of the Tasks.

6.6. Extended Meta Model

Figure 6.4 shows how the additions within the integrated model would look like. It is important to note, that only the small extensions in the model already enable the functionalities mentioned above. The code of the interface `AccessControlled` has the biggest changes, due to the addition of the interface `MxLDefinable`.

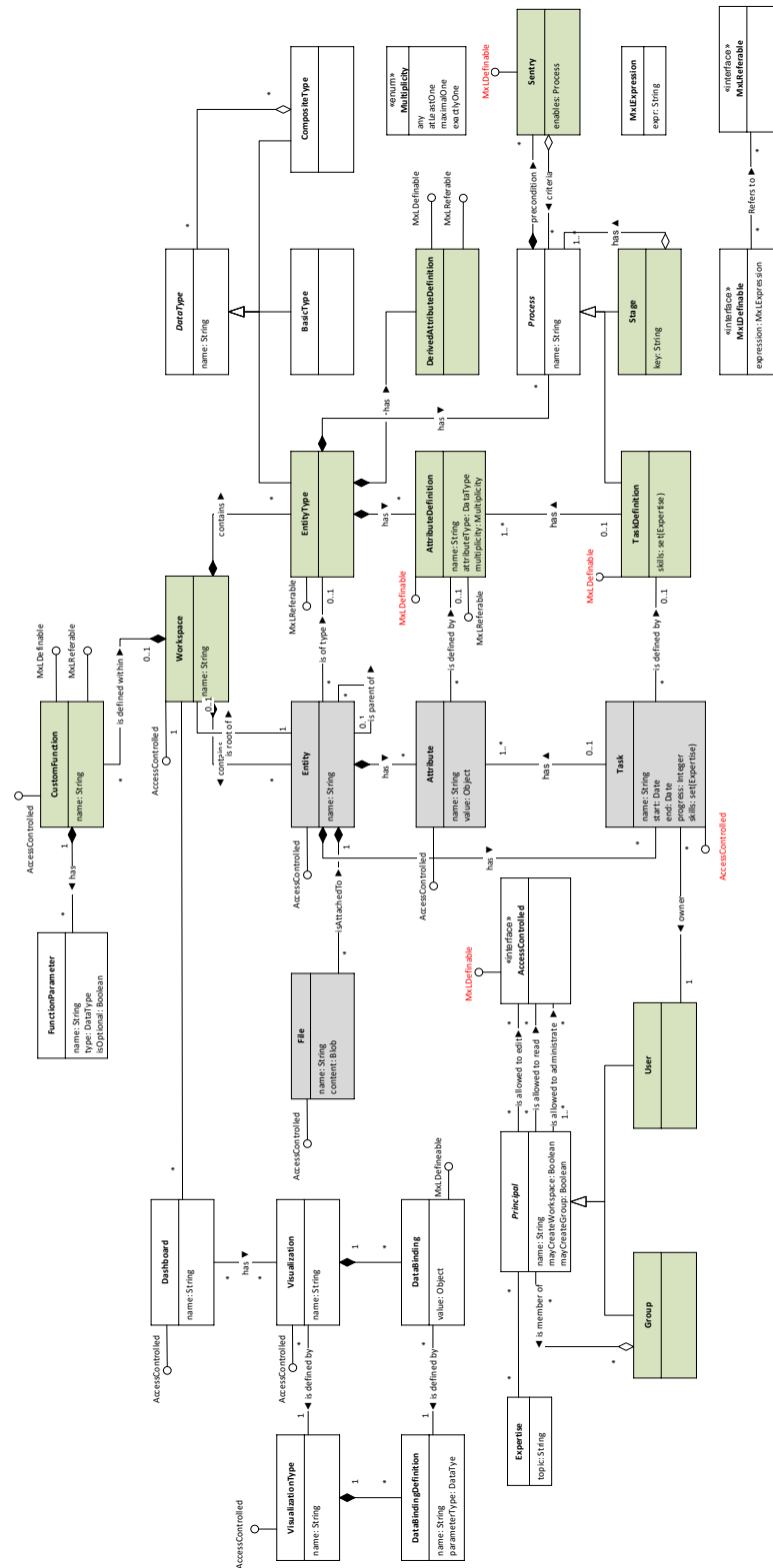


Figure 6.4.: Extended Integrated Meta Model

Part VII.

Conclusion

7. Conclusion and Critical Reflection

In this section, the whole thesis is summarized and the main findings are pointed out. Furthermore it is critically reflected to point out aspects, which could have been handled differently.

7.1. Summary

This thesis integrates four different models, which arouse around the hybrid wiki approach. In the first section, the work is motivated and the different models are introduced. Beyond that, the research questions are stated and the outline of the thesis is introduced. Chapter 2 further explained the underlying models. The third chapter presents the integrated model. In addition to the static model, the behavioural model is presented as-well. Afterwards the prototypical implementation of a modeller client, with its technical foundations, overall architecture and core features is explained. The evaluation of the prototype is then shown in Chapter 5. Potential further enhancements of the Model are displayed in Chapter 6. They further integrate the model presented in Section 3.1 and suggest additional links between the models. Chapters 7 and 8 conclude the thesis by summarizing it and also show an outlook and possible future work.

7.2. Conclusion

The main artefacts of this thesis are concluded in this section.

Integrated Model

The integrated model represents the basis for this work. It combines four different approaches to define a big meta model for hybrid wikis. This model now features all functionalities provided by the sub models together and can be used for a hybrid wiki platform with integrated data, role, function and task modelling.

Behavioural Model

The behavioural model takes a deeper look into consequences and preconditions for CRUD operations within the model. The focus thereby is on the classes, which are important for the implemented prototype. Knowing about possible consequences when working in a productive system is important for every user and this model provides exactly that.

Possible Model Extensions

To show up further possible model extensions, a second integrated model with additional details was created. This model further combines the underlying concepts by displaying additional links between them. This model is not yet implemented, but shows potential options for future work.

Prototypical Implementation of Modeller

A prototype to work with the integrated model was also implemented, explained and evaluated in this thesis. It provides an integrated user interface for the management of data, role, function and task models. The evaluation showed, that the implementation is already in a usable status with a SuS score of 72.5, but also revealed some weaknesses, which should be enhanced in the future to provide an better and more intuitive user-experience.

7.3. Critical Reflection

Even though the evaluation results were positive, there are some aspects, which could have been done differently and may have resulted in a superior outcome.

At first there are the time constraints under which this thesis was done. The short time frame did not allow for more than one evaluation. This means, that the circle proposed by Hevner et al. [11] was only done once. This led to a prototype, where the first set of weaknesses is known, but they are not changed in the implementation.

Another problem is, that the evaluation took place late in the process of implementation. Rapid prototyping as suggested by Dey et al. [6] combined with an early evaluation could have led to an implementation, which would present the, for the end user important, functionalities in a different and maybe more intuitive way. Another approach could have been to not only do rapid prototyping but also evaluate the mock-ups or click-able mock-ups with the practical partners. This would have shown the weaknesses of the user interface even earlier in the design process and would have made changes to the layout easier. However these steps were not taken due to the time constraints of this thesis.

Another question arose during the evaluation of the prototype. Was material design as basis the right decision or are there better alternatives? Two findings led to this question, the first one being, that material design was not always suited to provide the needed functionalities and therefore there are different points where the prototype now differs from the material design guidelines. This led to a somewhat inconsistent user experience according to the interviewees. Another negative point about material design are the "Chips", a design element to represent lists, which was used in the implementation. Two out of the three interview partners from the practical side did not understand their usage from the beginning and needed explanation. In this thesis the design was handled as pre set due to the mock ups, but maybe alternatives like Twitter Bootstrap [4] would have been superior.

The last point, which could have been improved is the questionnaire for the evaluation. We used the initial version of the ten SuS questions, even though there is a improved version published by Bangor et al. [2]. We encountered exactly the problems described in his paper, for instance the word cumbersome had to be explained to several interviewees. Another problem were the english questions in general. Two interviewees did not answer the questions correctly in the first place. This only attracted attention, because the answers given in the questionnaire were not matching the impressions they expressed verbally.

Part VIII.

Outlook and Future Work

8. Outlook and Future Work

This thesis can be basis for various future works. Implementing the possible improvements presented in Section 5.3 being the first open point. The implementation can be done based on the presented mock-ups and afterwards should be re-evaluated to complete another cycle based on the design science approach [11].

Another important step for the future is the implementation of a graphical solution for modelling task definitions and stages, as current view is not suitable to manage big hierarchies of tasks and stages. The focus there should not only be set on the hierarchies but also on the precondition dependencies between processes.

Furthermore the potential further enhancements of the model presented in Chapter 6 can be implemented into the current version of SocioCortex. Before doing so, an evaluation with participants from the industry is recommended to verify the usefulness of the functions with the needs in the industry.

To round up the prototypical implementation of this thesis, an administration area and further user and group functionalities should be added to it. At the current state there are no administration functions implemented and only basic user and group operations.

The last point worth mentioning for the future is the link between the prototype of this thesis and the generic SocioCortex client implemented by Bj rn Michelsen. There are several connection points between the two clients and a well rounded link between both applications would improve the user experience.

Appendix

A. Evaluation Scenario

Scenario Introduction

In the scenario for this evaluation you are a PhD. Student at the SEBIS chair. You have access to a new knowledge management system at the chair, SocioCortex.

You are in charge of modeling data for the system as well as using it to support you in your activities such as overseeing students doing their thesis at the chair.

The following task will cover activities regarding the modeling of data as well as entering data for a specific use case.

Scenario 1 - Modeler

Task 1: Extension of the Entity Type Master's Thesis (Part 1)

As a PhD. Student at the SEBIS chair you noticed, that the Entity Type "Master's Thesis" in the workspace "Evaluation" is not fully modelled and there are Attribute Definitions missing. You want to add the Attribute Definition "End", which is a date and should be set exactly once and the Derived Attribute Definition "Duration", which should calculate the duration of the thesis automatically.

Task 2: Creation of the Entity Type Person

To complete the Entity Type Master's Thesis you need to add the Entity Type "Person" to the workspace. A person needs to have a Attribute Definition "Name" and an "E-Mail"

Task 3: Extension of the Entity Type Master's Thesis (Part 2)

Add the Attribute Definition "Advisor" to the Entity Type Master's Thesis. "Advisor" should be of the Type "Person". Also make sure that every Thesis has at least one Advisor.

Task 4: Extension of the Task "Define Thesis"

To guarantee a controlled workflow, you now need to align the Task "Define Thesis" of the Entity Type "Master's Thesis" to also include the Attribute "End".

Task 5: Creation of the Task "Conduct Evaluation"

To finish up your work on the model side, you now want to create the new Task "Conduct Evaluation". The Attribute of the Task should be Evaluation and its precondition is, that the abstract is written.

B. Evaluation Questionnaire

Questionnaire

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5

10. I needed to learn a lot of things before I could get going with this system

1	2	3	4	5

Overall, how does this system compare to Tricia?

--

Overall, what would you like to change in the system?

--

Is there any other feedback?

--

Bibliography

- [1] Frederik Ahlemann, Eric Stettiner, Marcus Messerschmidt, and Christine Legner. *Strategic enterprise architecture management: challenges, best practices, and future developments*. Springer Science & Business Media, 2012.
- [2] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.
- [3] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.
- [4] David Cochran. *Twitter Bootstrap Web Development How-To*. Packt Publishing Ltd, 2012.
- [5] Thomas H Davenport. *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press, 2013.
- [6] Anind K Dey, Gregory D Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction*, 16(2):97–166, 2001.
- [7] Angela Edmunds and Anne Morris. The problem of information overload in business organisations: a review of the literature. *International journal of information management*, 20(1):17–28, 2000.
- [8] Google. Angular material. <https://material.angularjs.org/>, 2016. Accessed: 18.07.2016.
- [9] Matheus Hauder. *Empowering End-Users to Collaboratively Structure Knowledge-Intensive Processes*. Dissertation, Technische Universitaet Muenchen, Muenchen, 2016.
- [10] Matheus Hauder, Rick Kazman, and Florian Matthes. Empowering end-users to collaboratively structure processes for knowledge work. *18th International Conference on Business Information Systems (BIS), Poznan, Poland*, 2015.
- [11] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in informationsystems research. *MIS Quarterly Vol. 28*, pages 75–105, 2004.
- [12] Florian Matthes and Christian Neubert. Enabling knowledge workers to collaboratively add structure to enterprise wikis. In *12th European Conference on Knowledge Management-ECKM*, volume 2011, 2011.

- [13] Florian Matthes and Christian Neubert. Wiki4eam: Using hybrid wikis for enterprise architecture management. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 226–226. ACM, 2011.
- [14] Florian Matthes, Christian Neubert, and Alexander Steinhoff. Hybrid wikis: Empowering users to collaboratively structure information. *ICSOF (1)*, 11:250–259, 2011.
- [15] Ivan Monahov, Thomas Reschenhofer, and Florian Matthes. Design and prototypical implementation of a language empowering business users to define key performance indicators for enterprise architecture management. In *2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 337–346. IEEE, 2013.
- [16] Thomas Reschenhofer. *Design and prototypical implementation of a model-based structure for the definition and calculation of Enterprise Architecture Key Performance Indicators*. PhD thesis, Master’s Thesis. Technische Universität München, 2013.
- [17] Thomas Reschenhofer, Manoj Bhat, Adrian Hernandez-Mendez, and Florian Matthes. Lessons learned in aligning data and model evolution incollaborative information systems. *Proceedings of the International Conference on Software Engineering (ICSE), Austin, Texas USA*, 2016.
- [18] Thomas Reschenhofer and Florian Matthes. Empowering end-users to collaboratively manage and analyze evolving data models. *Proceedings of the American Conference on Information Systems (AMCIS), San Diego, USA*, 2016.
- [19] Thomas Reschenhofer, Ivan Monahov, and Florian Matthes. Type-safety in ea model analysis. In *EDOC Workshops*, pages 87–94, 2014.
- [20] Jeff Sauro. Measuring usability with the system usability scale (sus). 2011.
- [21] sebis. Sociocortex: A social information hub. http://sebischair.github.io/sociocortex_web/files/160209%20Michel%20SocioCortex%20Eco-System.pdf, 2016. Accessed: 15.07.2016.
- [22] Diane M Strong and Steven M Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems (TOIS)*, 13(2):206–233, 1995.
- [23] Keith D Swenson, Nathaniel Palmer, et al. *Mastering the unpredictable: how adaptive case management will revolutionize the way that knowledge workers get things done*, volume 1. Meghan-Kiffer Press Tampa, 2010.
- [24] WMP Van der Aalst, Moniek Stoffele, and JWF Wamelink. Case handling in construction. *Automation in Construction*, 12(3):303–320, 2003.
- [25] Marcel van Oosterhout, Eric Waarts, and Jos van Hillegersberg. Change factors requiring agility and implications for it. *European Journal of Information Systems*, 15(2):132–145, 2006.
- [26] Maarten W. van Someren, Yvonne F. Barnard, and Jacobijn A.C. Sandberg. The think aloud method. *Academic Press*, 1994.