iteraplan

Enterprise Architecture Management (EAM) Made Easy

Table of Contents

User Guide

This manual describes how to work with iteraplan 3.3 Enterprise Edition. iteraplan is a simple, flexible IT landscape management tool provided by iteratec GmbH .

Audience

The intended audience of this manual are both frequent and occasional users of iteraplan.

Content

iteraplan uses the iteratec IT landscape management method, which is based on the iteratec best-practice enterprise architecture.

iteraplan provides a range of functions for working on the elements of the best-practice enterprise architecture and for generating reports. The manual illustrates the use of these functions and explains them on practical examples.

With its extensive analysis capabilities and diagram reports, iteraplan provides configurable views of the landscape data which can be used by the various stakeholders for documenting, analysing and planning the IT landscape.

Reports can be generated in spreadsheet format and beyond as diagrams in various formats. Users can also run consistency checks to test the completeness, soundness, and quality of the data entered. This manual explains how to configure and generate the various reports and analyses.

Conventions

This manual uses the following typographical conventions:

Italic: Italic type is used to emphasise terms and indicates references to other publications.

Bold: Bold type designates elements of the user interface.

Monospace: URL addresses are shown in non-proportional (monospace) type.

Furthermore, different boxes are used, which are explained as follows:

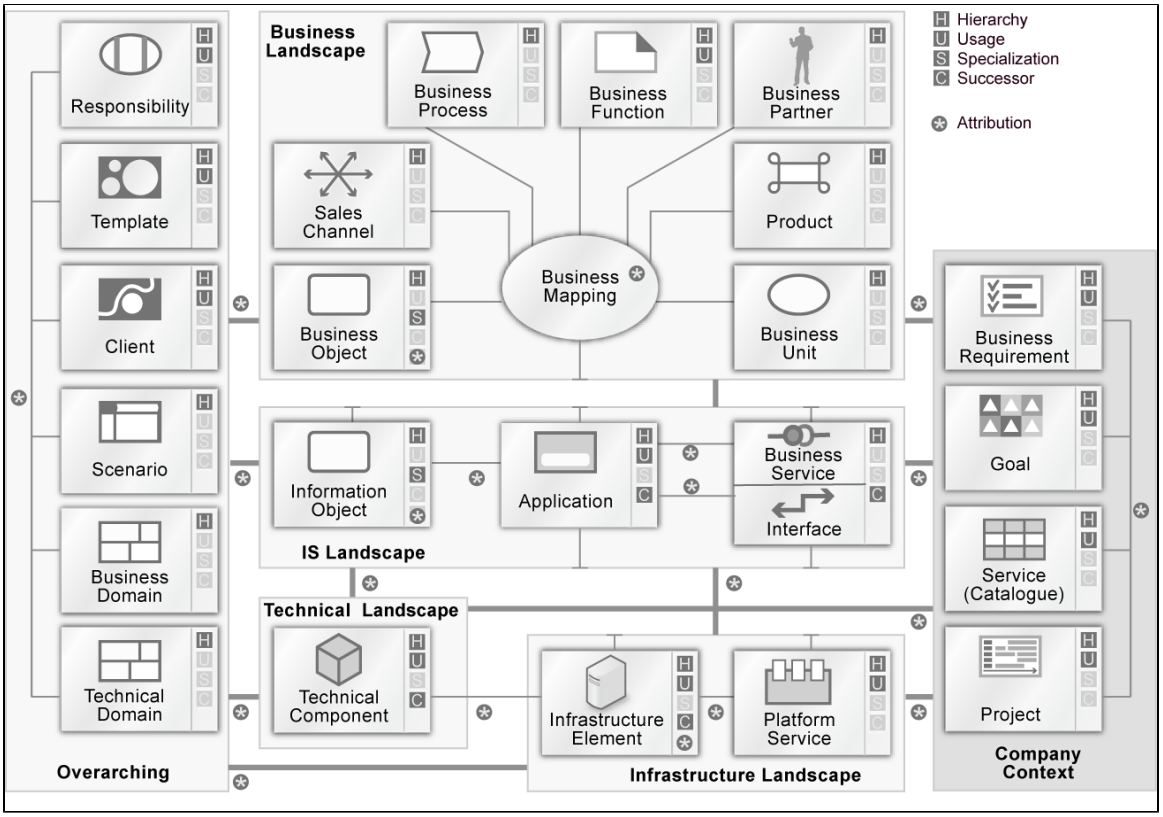
This box refers to improvements of iteraplan due to new releases.

This box provides further information and hints.

This box describes possible pitfalls where you have to pay attention.

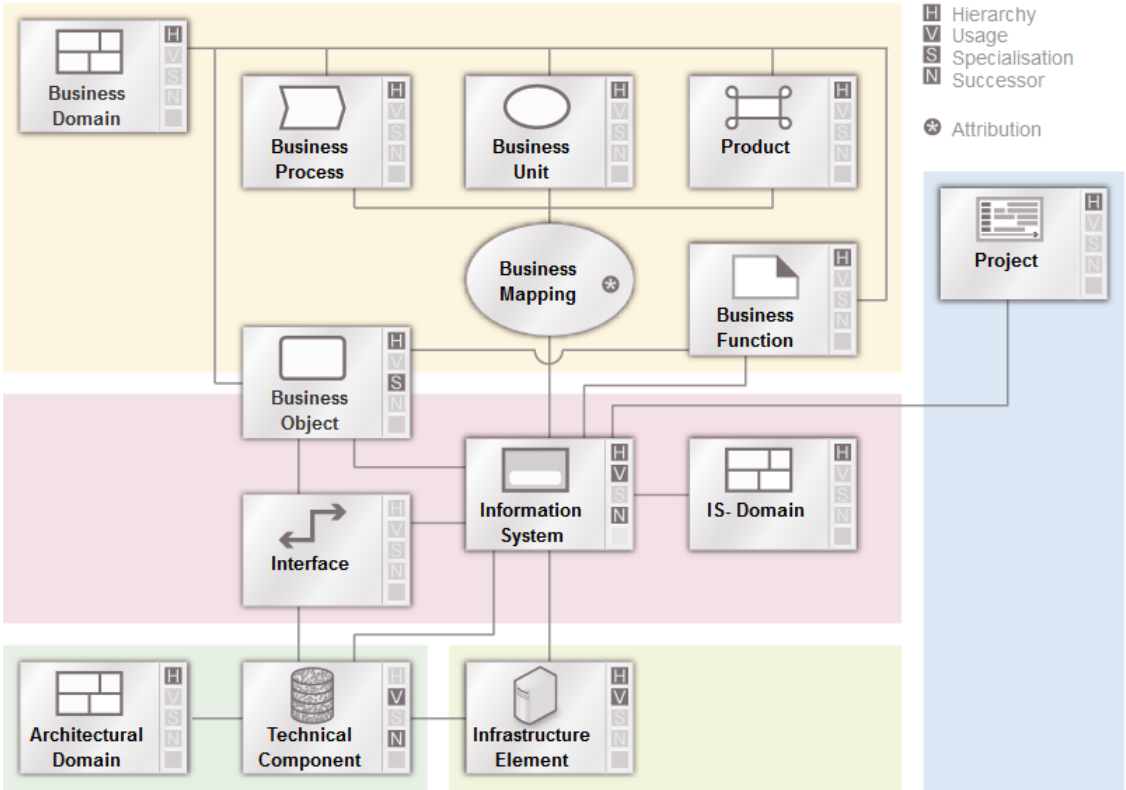
iteratec Best-Practice-EAM

iteraplan is based on the best-practice- EAM developed by iteratec for modelling business and technical landscaping in enterprises. The best-practice-EAM has been distilled from experience gained in a multitude of projects.



Best-practice-EAM

The best-practice-EAM presents the target (to-be) model for current development of iteraplan. Version 3.3 of iteraplan implements most parts of this model, but not all.



iteraplan meta model

This section surveys the elements of the iteraplan meta model – termed "building blocks" in the following. For more detailed information about the concepts and models, you may want to refer to the book "Strategic IT Management" by Inge Hanschke, published in 2009 by Springer Verlag. There is also a German edition of that book available, "Strategisches Management der IT-Landschaft" (Inge Hanschke, Hanser Verlag, 2009).

The building blocks of the iteraplan meta model are:

Business Domain
A business domain is a structural element that serves to group associated building blocks in the business landscape.

Business Processes
A sequence of logically connected activities or sub-processes that contributes in some way to the enterprise's value added. Each process has a defined start and end, is as a rule recurring and is expressed in terms of performing some action for customers.

Business Function
A distinct, cohesive set of business functionality such as "customer relationship management". The enterprise's capabilities are expressed in terms of the business functions it carries out. Business functions can exist independently of their use in business processes – i.e. they can be used in multiple business processes.

Product
The outcome or deliverable of an enterprise's service or delivery process. Products can be either material (e.g. goods such as cars or computers) or immaterial (services).

Business Unit
Logical or structural units of the enterprise, such as departments, sites and plants; also logical user groups such as "field sales team" or "internal administration".

Business Object
A business object represents a real-world entity – abstract or concrete – which encapsulates some part of the business activity of an enterprise (customers, for example, products or orders). Business objects can be associated with one another by relationships. Business objects are used by business processes and business functions.

Information System Domain
An IS domain groups a number of information systems with common criteria. IS domains are commonly used to organise the IS landscape – and the responsibilities for landscape planning – into related units.

Information System
Software or software package for associated functionalities which are logically and technically distinct from other areas of functionality, and which can be supported entirely or to a large extent by IT.

Interface
An interface defines a dependency between two information systems. Some interfaces are one-way, others permit two-way communication. They can take the form of information flows or control flows. In the context of IT landscape management, the term interface is taken to mean an information flow between information systems.

Architectural Domain
Like domains for information systems, architectural domains are structural elements. They can be used to group technical components.

Technical Component
Technical components provide information pertaining to the technical realisation of information systems or interfaces. The standardisation is part of the IT architecture management. This results in a catalogue of standardised technical components, also called technical blueprint.

Infrastructure Elements
An infrastructure element is one logical unit of the IT infrastructure required to run information system releases.

Project
A project is an activity or undertaking with the declared aim of implementing, rolling out or iteratively developing information system releases. As such, a project has the effect of modifying the enterprise IT landscape.

Introducing iteraplan

This section describes the fundamental concepts of iteraplan, including the user transactions and role model concepts which underpin the application's simple and effective approach to manage landscape data.

Logging on

You open iteraplan in a standard web browser. The application is optimised for Microsoft Internet Explorer 8 (or later) and for Mozilla Firefox 3.5.x

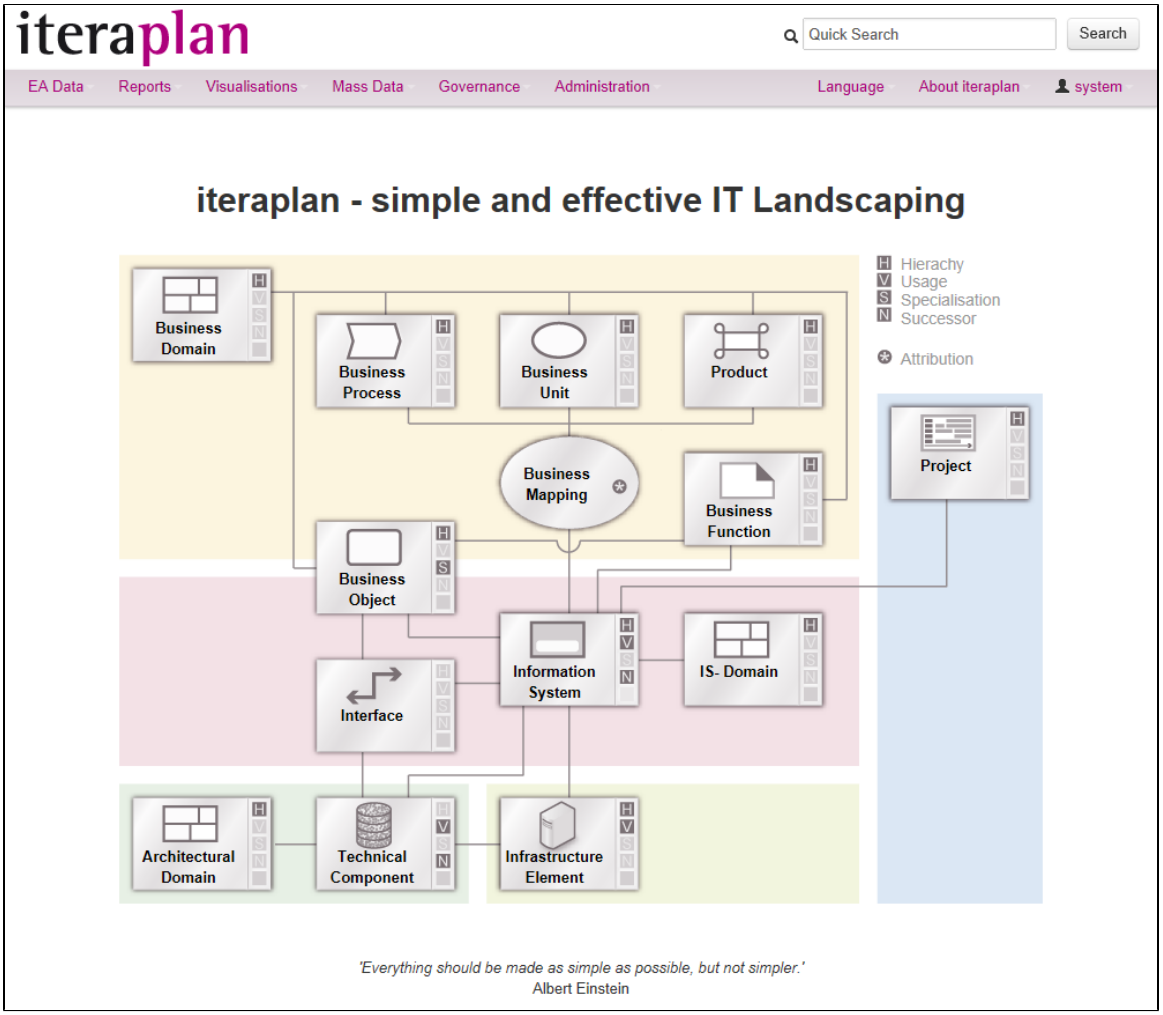
(or later). Your screen resolution must be set to at least 1280 x 1024 pixels to display the user interface correctly. The application is launched by entering a URL, which you can obtain from your system administrator or whoever is responsible for iteraplan in your company.

After entering the URL, you are prompted to log on with your user name and password. Please bear in mind that the password is case-sensitive (unlike the user name, which you can enter without regard to lowercase or uppercase).

iteraplan login dialog box

Home Page and Menus

After logging in, the iteraplan home page appears. At the top right you can see a menu of link buttons and a list box (**Language**), where you can choose your preferred interface language. **About iteraplan** provides the iteraplan documentation, a glossary (definitions) and more information about iteraplan. Next to this button the user (in this case **system**) is placed, where you can choose your profile, clear the session or logout. This menu is always visible.



iteraplan home page

The workspace on the home page shows the simplified iteraplan data model. This presents an overview of the landscape building blocks and the associations that exist between them. You can click any of the blocks in this model to open the page for viewing and editing its data, and can return to this overview at any time by clicking **Home** in the top menu or the iteraplan logo in the top left corner of your screen.

iteraplan main menu

On the top of each page you can find the main menu, which is always visible. It enables you to navigate through the application. It is possible to move between pages without losing the data you have already entered – but do always bear in mind that your changes are not permanently saved until you actually execute the **Save** command by clicking the **Save** button (at the top left of the page in question).

The menu in iteraplan main menu shows the main menu of iteraplan. Through **EA Data** you can select an overview of the most used elements, the full text search and access all building blocks. Furthermore, the main menu contains **reports** and **visualizations** to analyze and display your data. **Mass data** contains bulk updates, as well as import and export possibilities. Choose **governance** to manage your application, whereby you can define user, roles and assign permissions to them, as well as run consistency checks. Under administration you can find configuration possibilities of iteraplan.

Each iteraplan user has at least one role. The set of commands you see in the menu depends on the nature of the roles which you have been assigned, i.e. it reflects the functional permissions which your role is assigned.

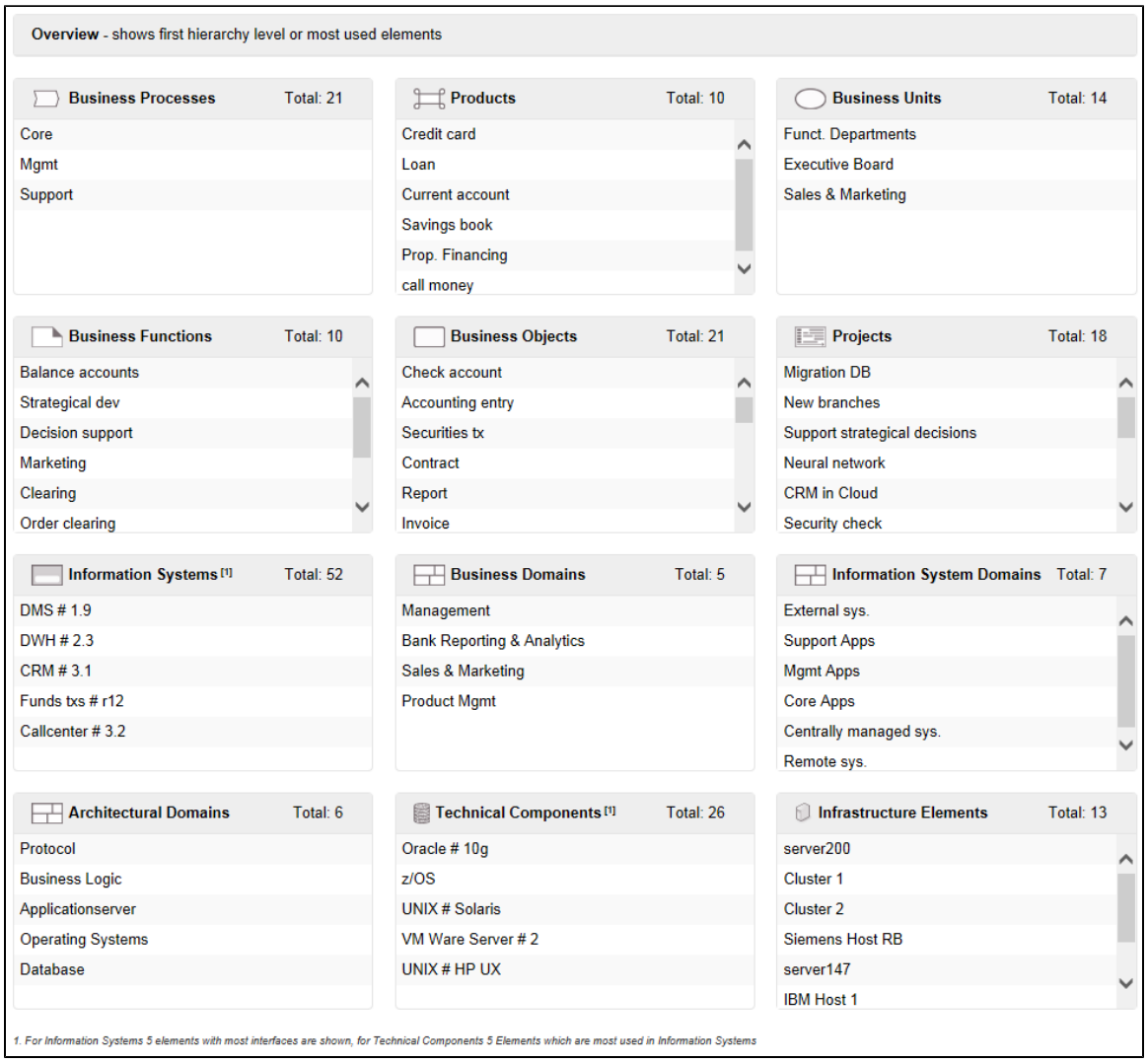
EA Data - Overview

When opening the overview page (in EA Data), lists for all Building Block Types will appear on one page. Each content is a snippet of all all building blocks, for the specific type.

- For **Hierarchical Building Blocks** the elements on the first level in hierarchy is shown. This means, all building blocks that have the

virtual element as parent.

- For **Information Systems** and **Technical Components** the most frequently connected resp. used elements are shown. These will be top five Information Systems which have the highest number of connected interfaces and the top five Technical Components which are most frequently used in Information Systems. If there is no clear cut between the top five elements and the other elements, the system makes a stable selection.



All the Elements are linked. So you can get the important information's quickly (Building Block Details). Also the headings are linked, they will bring you to the Building Block site, where all elements are listed (Building Block Lists and Search).

Permissions

There is one Permission for the menu point of this page, called "View overview of all Building Blocks'". To check the lists better, the permissions for each Building Block Type will also be important. For more information on permissions go to Permission to view and edit EA data.

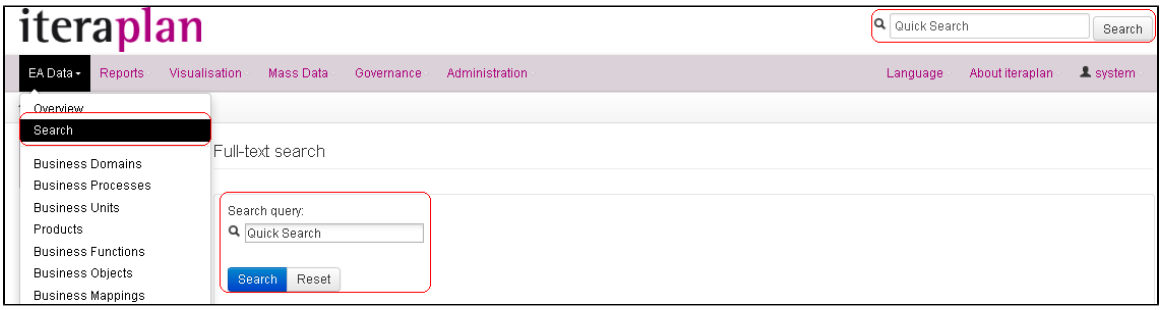
Global search

iteraplan's full-text search retrieves data from names, descriptions and attributes values. The results are ranked in order of importance. Its search technique is similar to search engines like Google, i.e. words are not searched literally and character by character. Instead, similar words are taken into consideration as well. The search index is basically a list of reduced words with references to occurrences in the actual data base, and it does not contain exact punctuation etc. The search results are ordered by computed relevance, which may appear counter-intuitive in some cases.

> ⚠ Note that common "small" English words, such as *the*, *a*, *these*, *is, he*, *she*, *it*, *and*, *but*, *to*, ... are always ignored in searches.
>
> Searches are case-insensitive, i.e. IT is equivalent to it (which is an ignored word).

You can either use the search box in the top right corner of each page or the explicit search dialog in the left navigation bar in section EA Data. Submit your search by clicking the **Search** button or by pressing the return key.

> The search index needs to be recreated if you import data into the database directly (e.g. by using direct DB access).



Global search

You can use the following expressions and wild cards:

- ***** - the asterisk represents a character string of any length
- ? - the question mark '?' represents zero or exactly one character
- Quotation marks ("abc xyz") allow to search for word groups. Word groups without quotation are interpreted as OR expression, e.g. abc OR xyz and will yield more result then you might expect
- AND - allows to query for elements which contain both strings on either side of the AND operator. Alternatively, you can put a + (plus sign) directly in front of each string
- OR - allows to query for elements which contain at least of the two strings. This is the default operation for two strings separated by a space
- NOT - allows to exclude any elements with contain the subsequent string. Alternatively, you can put a - (minus sign) or ?! (exclamation mark) directly in front of the string to be excluded
- (and) - use brackets to form groups for boolean expressions

There are several special characters which can further influence the meaning of a search query. iteraplan uses the Lucene project internally for its full text search functionality. The following paragraph summarizes Lucene's full query syntax documentation.

+	Can appear in front of a search term, and indicates that the term is mandatory. Example: A query **+iteraplan Lucene** means that a building block *must* contain the word iteraplan and *may* contain the word Lucene.
-	Can appear in front of a search term, and indicates that this term must not appear in matching building blocks. Example: A query **itera plan -iteratec** finds all building blocks with the word iteraplan, but which do not have the word iteratec.
&	Can only be used as && and expresses a boolean AND. Example: **ite raplan && Lucene** is equivalent to **iteraplan AND Lucene**
|	Can only be used as || and expresses a boolean OR. Example: **itera plan || Archimate** is equivalent to **iteraplan OR Archimate**
!	Can appear in front of a search term, and excludes results which contain the search term. It is equivalent to the NOT operator. Example: A query **iteraplan !iteratec** finds all building blocks with the word iteraplan, but which do not have the word iteratec.
(	Can be used to group several search terms together and build complex boolean expressions. It must always be matched by a closing bracket.

)	Can be used to group several search terms together and build complex boolean expressions. It must always be matched by a preceding opening bracket. Example: **(iteratec OR iteraplan) && "EA Management"** finds building blocks which contain the phrase EA Management and at least one of the two words iteratec and iteraplan.
{ and }	Lucene uses curly brackets and square brackets to express range queries. iteraplan does not use this feature. These characters are illegal.
[iteraplanDocumentation303: and]	Lucene uses curly brackets and square brackets to express range queries. iteraplan does not use this feature. These characters are illegal.
^	Can appear at the end of a search term to boost that term in result ranking. It must be followed by a positive number. Example: **iterapla n^3 Java** find all building blocks containing both terms. The number of occurrences of the word iteraplan is three times more important for result ordering than the occurrence of Java.
"	Can be used to form a phrase, i.e. a search term consisting of more than one word which must be found as is. For each opening quotation mark there must be a closing quotation mark. Example: **"Apache Tomcat"** finds all building block containing this word group. Building blocks which contain both words, but not next to each other, are ignored.
~	Can appear either at the end of a single word search term, or after a phrase. After a word search term, it enable fuzzy searching, e.g. **inter aplan~** would also match building blocks which contain the (properly spelled) word iteraplan. After a phrase, the tilde enable proximity searching. Example: **"Apache Tomcat"~10** means that matching building blocks can have up to 10 words appear between Apache and Tomcat.
*	Can appear in front of a words, at the end or in the middle, or any combination of those. It is a wildcard character and can stands for any number (including zero) of characters. Example: **te*t** matches building blocks with *test* or *text* or *testament*, while **test*** only matches *test* and *testament.*
?	Can appear in front of a words, at the end or in the middle, or any combination of those. It is a wildcard character and stands for a single character. Example: **te?t** matches building blocks with *test* or *t ext.*
:	Can be used to separate index field names from search terms. iteraplan does not use this feature. The character is illegal.
\	Can be used to escape any of the above characters and prevents interpretation of that character's special meaning. Example: **Hello\&B ye** is a valid search query, while Hello&Bye is not. (Note that the ampersand is not included in the search index, and therefore cannot be searched. It is a legal construct, but will return incomplete or misleading results.)
§ $ % / ` ´ ' # . , ; < >	All these characters have no special meaning, but do not become part of the search index. Queries containing these characters will always return misleading results!

Query Console & iteraQL

The query console & iteraQL

The query console provides a further mechanism for extracting landscape information from iteraplan by using the iteraQL query language. iteraQL provides you therefore with a powerful tool making it possible to investigate your model in iteraplan in a fast and comfortable way.

This section of iteraplan's documentation offers you both a comprehensive introduction how to use the iteraplan query console as well as a reference you may use frequently in your later usage of the query console.

The sections of this documentation

In the first part you will get to know the conceptual aspects of iteraQL in order to equip you with the knowledge and ideas that help you to understand the functioning and characteristics of iteraQL. If you've never had any experience with iteraQL, this is the perfect way to start.

For the advanced user, the howTo page gives further examples and applications of iteraQL.

The reference pages give you the possibility to explore the tools and methods in iteraQL in every detail, as well as they provide you a reference text you may use during the use of iteraQL later on.

Concept

In this section of iteraQL's documentation you will get to know the basic concepts of iteraQL in order to get you ready to work with iteraQL on your own.

The very first thing to do here is reading how to get started with iteraQL which will present you the basic ideas behind iteraQL and an overview of the conceptual framework it is built in.

Getting started with iteraQL

Getting started with iteraQL may consist of the following stations:
1) The elastic model's parts

The environment in iteraQL builds up on the elastic model behind iteraplan. Therefore the objects appearing in iteraQL live in the setting of *Buildin g Blocks*, *Properties* and *Relations*. While *Building Blocks* are the key elements, the meta model in iteraplan is made of, *Properties* enhance them with further characterization possibilities and *Relations* make it possible to set all the *Building Blocks* into relation to each other. These are the objects iteraQL understands and deals with, so it is helpful to understand the meaning and way of their appearances in iteraQL. You will find all the facts about *Building Blocks*, *Properties* and *Relations* in iteraQL on the Building Blocks, Relations and Properties page in the concept section.
2) The meta-model in iteraplan

At this point you have understood what a Building Block or Property or Relation *is*, but you might still wonder what the Building Blocks and their Properties and Relations in the iteraplan meta-model actually are. Although not necessary to understand the iteraQL language, you might have a glimpse on the naming conventions for Building Blocks and Relations page to get to know the landscape of the meta-model in iteraplan.
3) What is an iteraQL query

IteraQL is used by setting up queries and sending them via the query console. A query is a written text in the language of iteraQL. This text tells iteraplan what part of your model you would like to investigate. The list of operators defined for iteraQL will give you the possibility to develop queries returning exactly the information you want. To understand the structure of an iteraQL query, read the Queries, Operators and Predicates p age.
4) Build your own queries

Now you should be able to define iteraQL queries on your own. You might first try to reconstruct the examples seen on the linked pages from above until feeling familiar enough with iteraQL to develop your own queries. The how to page may as well present some inspiration in order to use the query console efficiently.
5) The reference section

Use the iteraQL's reference whenever having trouble to create an iteraQL statement that returns the desired results. Among others, detailed explanations and examples for every operator are given.

Building Blocks, Relations and Properties

An iteraQl query uses the the parts of the meta-model, that is building blocks, their properties and relations between building blocks, and combines them with filters and operators. This section defines the different kinds of building blocks, properties and relations in detail. Based on these definitions, the next section defines the input and results of applying filters and operators.

- Building Blocks (BB)
    - Stand-alone Building Block (sBB)
    - Associative Building Block (aBB)
- Relations (R)

Building Blocks (BB)

Building Blocks are the basic elements to describe an EA model. First of all, all elements accessible in iteraplan's EA Data menu, are building blocks.

Example
Information System and Interface are Building Blocks.

In this documentation, Building Block is abbreviated "BB".

A Building Blocks has Properties and Relations which are described below.

A Building Block is either stand-alone or associative.

Stand-alone Building Block (sBB)

A stand-alone Building Blocks (short: sBB) can exist on its own. It does not require the existence of another Building Block. They can be seen as the more basic kind of Building Blocks.

Example
Information System is a stand-alone Building Block.

Associative Building Block (aBB)

An associative Building Block (short: aBB) describes a relation with attributes that connect two or more stand-alone Building Blocks.

Compared to a simple relation between to Building Blocks (defined below), an associative Building Block is more expressive in two ways: First, it can associate more than two Building Blocks at the same time. Second, it can can have additional Properties and (simple) Relations. These Properties and Relations describe the connection of the Building Blocks, not the Building Blocks themselves.

Example
Interface is an associative Building Block. Basically, an Interface associates two Information Systems and other Building Blocks.

Example
Business Mapping is an associative Building Block. A Business Mapping associates an Information System with Business Process, Business Unit and Product.

A Business Mapping may not exist without an Information System and at least one building block of the three remaining types Business Process, Business Unit and Product.
An Interface describes the relationship between Business Objects, Information Systems, their releases and Technical Components and hence connects up to four Building Blocks. Furthermore it has attributes, such as their degree of automation.

Note that there are associative Building Blocks immediately visible to the user in the EA Data menu and others that are only accessible in an indirect way, e.g. when editing the connected Building Blocks. Nonetheless, since iteraQl queries enable the user to traverse the entire meta-model of iteraplan, those hidden associative Building Blocks need to be taken into account when defining queries.

Example
Interfaces are accessible directly using the EA Data menu. By contrast, there is no direct connection between an Interface and an Business Object but an associative Building Block called *Information Flow* set in-between. However, the associative Building Block Information Flow is not accessible through the EA Data menu, but e.g. by editing an Interface's relations.

A list of all associative Building Blocks in iteraplan is at the page naming conventions for Building Blocks and Relations.

Relations (R)

A Building Block can be connected to another Building Block (or itself) by a *Relation* (R). A Relation do not possess any further properties and always connects exactly two Building Blocks, but not three or more. Generally, the Relations of a Building Block are displayed relations panel of the Building Block in iteraplan.

In iteraQL, a Relation is denoted by the name of the destination Building Block beginning with an lowercase letter and using the plural expression to indicate that a single Building Block element may be connected to several elements of the destination Building Block. The exact naming conventions for the Relations in iteraplan are given at naming conventions for Building Blocks and Relations. How to use Relations in iteraQL will be described on the Queries, Operators and Predicates#relations

Example
The Building Block *Information System*, in iteraQL denoted by InformationSystem has (among others) a Relation to the Building Block *Interface*. This Relation has the name *interfaces*.

Self-referencing Relations (sR)

A special type of Relations are those that are *self-referencing* (abbreviated by sR), which means they connect elements of a Building Block to elements of the same Building Block. This distinction is necessary as some operations in iteraQL may only work using self-referencing Relations. Typical self-referencing Relations that occur frequently are child and parent relations.

Example
An Information System has the self-referencing Relations *children* and *uses* to other Information Systems.

Properties (P)

A Building Block has properties that describe its characteristics. Generally, different Building Blocks have different sets of Properties.

Some Properties are used for all or most Building Blocks: In particular, each Building Block has the Property *id* and each stand-alone Building Block the Property *name*.

Example
An Information System has (among others) the Properties *id*, *name* and *manufacturer*.

Naming of iteraplan-pre-defined and user-Defined Properties

A list including their naming conventions of the pre-defined properties of each Building Block in iteraplan may be found in the naming conventions for Building Blocks and Relations.

User-defined properties are not subject to special naming. This means that if, for example, a user-defined attribute 'Number of Users' is defined in iteraplan, then 'Number of Users'' is also the name to be used for this attribute when denoting it in iteraQl.

Types of Properties

Obviously, there are very different types of nature of Properties. Properties may be numbers, text-strings, dates and others. Although this is not going to be made more precisely at this point, the user should keep this in mind, as certain operations on Properties may require certain types or the same operation may have a different meaning when applied to different types.

Example
Information Systems possess (among others) The Property *id* is a number, hence two elements of a Building Block may be compared by a ">" operation that decides whether the first element's id is the larger one. The Property *manufacturer* is a text-string. However, the ">" operation does not work on this Property.

Multi-valued Properties

A degeneration of Properties that an advanced user may focus at a certain point using iteraQL may be *multi-valued* Properties, i.e. Properties that can consist of a whole list of entries.
Although this is not the general case, it may occur, for example, when certain synthetic Properties (see below) are being considered. Again, in those cases the user should be aware of the consequences when dealing with operations in iteraQL.

Example

The Building Block *Information System* has the multi-value Property *Accountability* meaning an Information System can have more than one person (or other "accountability unit") being accountable for it.

Synthetic Building Blocks/Properties/Relations

A more sophisticated but nevertheless for a trouble-free handling of iteraQL essential aspect are *synthetic* Building Blocks, Properties and Relations.
Informally spoken, iteraQL provides the user with a certain toolbox that may be used to work on the given entities of iteraplan to generate new ones, which may be called synthetic and which are deduced from the given ones in a certain manner. Depending on the executed operation, these new entities may or may not inherit the characteristics of the entities they were deduced from.

Example

In iteraQL, there are the so-called filter and power operations (that are both described in detail at the operator's reference page). Using the filter operator, the user may receive all releases of Information Systems with costs are at most 1000. As well as the Information System Release is a certain Building Block, the result of this filter operation is as well a Building Block, whose elements are all Information Systems Releases with costs at most 1000, but a synthetic one and obviously different from the general Information System Release Building Block. However, the Building Block "Information Systems Release with costs at most 1000" does possess the same Relations and Properties as the "Information System Release" one does.

On the other hand, the power operator applied to the Information System Release Building Block returns a very special synthetic one by "summarizing" all Information System Releases into one single comprehensive element having nothing than a Relation to all the elements of the "Information System Release"-Building Block.

An awareness of the occurrence of synthetic Building Blocks, Properties and Relations in iteraQL is necessary, because usually one does not only want to use a single operator in a query, but a list of operators executed in a certain order. Each operator in iteraQL requires a certain input, such as a stand-alone Building Block, and produces a certain output, that is always synthetic but can be used as input of another operator assuming it meets the operator's input requirements. Therefore, when concatenating several operators, the user should be aware of the intermediate outputs that are used as inputs for the consecutive operators and hence have to meet their requirements.

Example

Consider the same example operations above. However, this time the user first applies the filter an and then the power operator on the "Information System Release"-Building Block. The result is synthetic Building Block having exactly one element that has a Relation to all elements of the synthetic Building Block "Information Systems Releases with costs at most 1000" (and not to the Building Block "Information Systems Releases").

If the user had applied these two operations with switched order, the first power-operator would have produced the synthetic "summarizing" Building Block as described in the example above and the secondly applied filter operation would have failed, as this intermediate Building Block would not have had the Property "costs" and thus not met the operation's requirements.

Instances of Building Blocks/Properties/Relations

Having understood the concept of Building Blocks, Properties and Relation is certainly the most crucial requirement for the usage of iteraQL. However, at some point it might be necessary not to talk about a Building Block, Property or Relation in general, but to go a little deeper into detail and look at particular *instances* of these. Theoretically spoken, instances are not a part of iteraplan's meta model any more, but belong to its practical implementation meaning that the instances of a Building Block, Property or Relation respectively are their specific realization. However, a simple example should make things a lot easier to understand:

Example

Information System is a Building Block and part of the standard iteraplan meta model. In an (fictional) iteraplan implementation for the IT landscape of a financial institute, there are (among others) the realized Information Systems *CRM* (the customer realtionship management), *electronic Banking*, *Callcenter*, which are therefore all instances of the Building Block Information System.
Each of these instances has then its own instances of the Building Block's Properties, for example *CRM* has the costs "1000", while *Callcenter* has the costs "800".

Proceed to the Queries, Operators and Predicates.

Queries, Operators and Predicates

Having understood the concept of Building Blocks and their properties we now want to get to know the actual usage of iteraQL, which is defining

queries that use operators and predicates.

Queries

A iteraQL query is a written statement that the user wants to be executed by iteraplan in order to recieve particular parts of the model in iteraplan as return.

General Structure

First of all a iteraQL query consists of two parts: a statement that defines the desired output of the query and a semicolon ";" that signalizes iteraplan the query's end. Everything written after this end-defining semicolon will be ignored by iteraQL.

Example

```
InformationSystem; IgnoredStatement
```

The desired output of this statement is the Information System Building Block. The "IgnoredStatement" string appears after the semicolon and will not be executed by iteraplan.

The output-defining statement is a composition of Building Blocks, Relations and Properties by using operators and predicates to set them in context. Before describing this in more detail, we first want to distinguish between two general types of queries: Building Block and Relation queries.

Building Block queries

In a Building Block query, the output-defining statement determines a (mostly synthetic) Building Block.
In return the user will recieve exactly this Building Block out of the model.

Example

```
InformationSystem;
```

The desired output of this statement is the Information System Building Block.

Relation queries

In the second query type, the output-defining statement determines (again mostly synthetic) Relation. As a Relation connects exactly two Building Blocks, iteraplan will return each of the first Building Blocks together with the according Building Blocks of the second one. If a Building Block is not related to any of the targeted one, it will not be part of the output, as well as the other way round.

Example

Assume the Building Block *Technical Domain* having the three instances "TD A", "TD B" and "TD C". Technical Domains have a Relation "technicalComponents" to the Building Block *Technical Component*. Let "TD A" be related to the Technical Components "TC 1" and "TC 2", further "TD B" be related to "TC 1", whereas the Technical Domain "TD C" is assumed to be not related to any. The query

```
TechnicalDomains/technicalComponents;
```

will therefore produce the output

Technical Domain	**/technicalComponents**
TD A	TC 1
TD A	TC 2
TD B	TC 1

The result of the query is a Relation between the Bulding Blocks *Technical Domain* and *Technical Component*, whereby the output in the iteraplan query console will be presented by every related *Technical Domain* and *Technical Component* instance couple as in the table above.

Working with Building Blocks

Referring to a Building Block's Relation

References to a Building Block's relations are made by a "/" followed by the iterQL's name of the desired Relation.

Example

The Building Block *Information System* has the Relation *informationSystemDomains*.
This Relation can be addressed via

```
InformationSystem/informationSystemDomains;
```

The use of "/" followed by a Relation does not necessarily follow a Building Block defining statement immediately. In particular if the Relation of interest is used as an operator's input, it can follow later as isolated expression (see as well later in the Operators section)

Example

The Building Block *Information System* has the Relation *informationSystemDomains*. Assume a imaginary Operator *.EXAMPLEOPERATOR()* that needs as input a Building Block as well as one of its Relations. Then a query may be:

```
InformationSystem.EXAMPLEOPERATOR(/informationSystemDomains);
```

Referring to a Building Block's Property

A Building Block's Property can be reached by the us of a "@" followed by the iteraQL's name of the desired Property.

Example

The Building Block *Information System* has the Property_costs_.
In an example using the filter Operator (the use of operators in general is described below), this Property is addressed via "@costs":

```
InformationSystem[ @costs > 1000 ];
```

Working on related Building Blocks

After addressing a Building Block's Relation, the next statement will not work on the original Building Block any more, but on the one the Relation is targeting at. If a Building Block *A* is connected to another Building Block *B* via the Relation *ab*, every statement after *A/ab* will work on the Building Block *B*. This includes any statement referring to a Relation or Property. This means if *.EXAMPLEOPERATOR*( )_ is some Operator needing further input such as a Relation or a Property, statements will have the form of *A/ab.EXAMPLEOPERATOR(@PROPERTY OF B)* or *A/ab.EXAMPLEOPERATOR(/RELATION OF B)*, respectively.

Example

As seen above, the Building Block *Information System* may be filtered via

```
InformationSystem[ @costs > 1000 ];
```

However, after first addressing the *Information System* Building Block's Relation *informationSystemDomains* and then applying the filter operator, not Information Systems, but their related Domains will be filtered:

```
InformationSystem/informationSystemDomains[ @costs > 1000 ];
```

While in the first statement "@costs" had referred to the costs of InformationSystems, in the latter the same statement refers to the costs of *Information System Domains*.

In fact, every statement following the "InformationSystem/informationSystemDomains" will work on the Building Block *Information System Domain*.

Operators

Operators are the tools in iteraQL that make it as powerful as it is. They receive as input one or more Building Blocks, Relations and/or Properties and return exactly one new Building Block, Relation or Property. Any operator's usage is in context with a Building Block and then works with this Building Block's Relations, Properties or the Building Block itself.

General usage of Operators

The use of operators may vary regarding their required in- and output.

The syntax for every operator in iteraQL is therefore given individually in a general way on the [Reference of Operators page |Operators] and illustrated by a short example. Such a syntax may have the form

```
OPERATOR(  %REQUIRED INPUT% )
```

Everything written in "% ... %" must thereby be replaced with another valid iteraQL statement. The operator statement may then be written directly after a Building Block statement or a Relation statement with the consequence that the targeted Building Block is the context-defining one.

Example

The objectify -Operator (whose usage and purpose is described on the reference page but not of any interest for this example) has the syntax

```
.objectify( %P% )
```

whereby "%P%" is a placeholder for a Property statement. This means it needs as input some Property (of the context Building Block).

For an application example consider the *Information System* Building Block having the Property *costs*.
If one wants to apply the objectify()-Operator on a Property of the Building Block *Information System* meaning this is the context Building Block, one may write this operator immediately after the statement referring to *Information System*, i.e.

```
InformationSystem . objectify( ...
```

As described above, this Property is referred to via "@costs". Placing now this "@costs" statement in the brackets of the objectify operator, iteraQL understands that the Property "costs" of the Building Block *Information System* is meant, as this is the current context Building Block. Hence a complete iteraQL statement would be:

```
InformationSystem.objectify(@costs);
```

(At this point it is not necessary to understand what the outcome of this particular query may be.)

Another possible application could be to apply the objectify operator not on *Information Systems*, but on their related *Projects*. Via the relation *projects* the Building Block *Information System* is related to the Building Block *Project*. Hence another valid statement would be:

```
InformationSystem/projects.objectify(@costs);
```

In this case the "objectify(...)" expression follows a Relation statement "InformationSystem/projects". As consequence, the context Building Block is the one targeted by the Relation, which is *Project* in this example. As the the context Building Block is *Project*, the "@costs" statement does *not* refer to the costs of *Information System* as in the example above, but to the cost Property of the *Proje ct* Building Block.

When operators are not written immediately after a Building Block or Relation statement, but inside an input argument of another operator, the operator inherits the context Building Block from the operator it is embedded in.

Example

In a statement like

```
InformationSystem[ count( /projects)  > 0];
```

two operators, namely the filter (expressed by the square brackets) and the count are applied. In particular the count()-operator is embedded in the filter-operator and therefore inherits the context Building Block *Information System*. So iteraQL understands the "/projects" statement inside the count()-operator as referring to the Relation *projects* of the Building Block *Information System*.

Concatenate multiple operators

In iteraQL it is indeed possible to use one operator to define the input of another operator and this way to concatenate several operators. If an operator requires as input a Building Block,Property or Relation, it is just necessary to place any valid iteraQL statement in position of this input that provides a Building Block, Property or Relation, respectively. This statement may indeed include one or more further operators.

So the main issue one should pay regard to is to make sure that the output of the input statement really meets the input requirements of the outer operator.

One example for concatenating two operators has already been given above, when the count() operator was embedded into an comparison statement (" > 0") which then produced a valid Predicate (see below), which is required as input of the filter operator.
Another example might be:

Example

```
InformationSystem.objectify( view(/projects @count ) );
```

As seen in the earlier example, the objectify-operator requires some Property as input. The view-Operator (not explained in any detail at this point) has as output a Property, hence this statement is valid.

Predicates

When working with the filter-Operator, the use of so-called *Predicates* is needed. The filter operator is used to make a selection of certain instances of a Building Block according to a particular selection rule, which is given by the Predicate. A *Predicate* for a Building Block attaches every of its instances with a "yes" or "no" label, such that the filter operator can then choose exactly those instances being attached with a "yes". The decision, whether an instance gets the label "yes" or "no", is made by another operator (with a Predicate as output). Usually, such operators are comparisons or equality checks meaning "<",>" and "=". Another possibility may be a check for some other requirement an instance has to fulfill to obtain a "yes"-label. For example, one may check the instances names for containing a certain word to be labelled with "yes".

The predicate-defining operators and the combining of several predicates is described in the reference section under Predicates and Predicate-Defining Operators.

Example

Consider the Building Block *Information System* having the Property *costs* and assume the following scenario:

Information System	**costs**
CRM	700
Callcenter	300
EAM	99
BI	2000

A Predicate may be defined using the comparison operator ">" to select exactly those instances having costs less then 500. This Predicate may be achieved via

```
@costs < 500
```

The resulting Predicate's labeling would then be:

Information System	CRM	Callcenter	EAM	BI
Predicate *@costs < 500*	no	yes	yes	no

Consequently in connection the filter-Operator (look at the according reference page for more details), would provide a result as follows:

```
InformationSystems[ @costs < 500];
```

InformationSystems[@costs < 500]
Callcenter
EAM

iteraQl How To

This page provides a step-by-step introduction to the syntax of the iteraQl query language. In the forthcoming sections the syntax is presented with examples, covering the different operators and language features. Before you continue, it is recommended to also have a look at the naming conventions, since these are made use of in the example queries.

Selection Queries

Selection queries are the simplest ones available in iteraQl. A selection query does not employ operators and only uses the elements of the iteraplan meta-model. There are two kinds of selection queries, for building blocks and relationships, respectively. A simple building block selection query is

```
InformationSystem ;
```

which simply retrieves the list of all instances of the Information System building block. Instead of 'InformationSystem' the name of any building block or association can be used. A selection query can also select a relationship, in which case a binding set is returned. A simple relationship section query is

```
InformationSystem /infrastructureElements ;
```

which retrieves all pairs of related instances of the Information System and Infrastructure Element building blocks. In general, a query always specifies an initial building block or association, optionally followed by operators. A query may further contain one or more relationships or relationship operators and terminates with a semicolon (';').

Querying with Operators

Having presented the simplest kind of queries, the selection queries, we can now continue with the introduction of a first operator.

The Join Operator

The join operator is the simplest and one of the most relevant operators available in iteraQl. Furthermore, what distinguishes the join from the other operators is the fact that it requires no keyword. This operator takes two relationships and produces a new relationship by 'gluing' them to each other. For example

```
InformationSystem /businessMappings /businessProcess ;
```

is a query which retrieves all pairs of instances of Information System and Business Process, which are connected over a Business Mapping instance. Also, the join operator is recursive, which means that it can be applied to an arbitrary number of relationships. For instance, the query

```
InformationSystem /businessMappings /businessProcess /businessDomains ;
```

evaluates just as the previous one does, but further extends the result to all Business Domain instances reachable from any Business Process instance which was in the result of the last query.

Filters

The second most relevant operator is the filter operator. This operator can be applied to any building block or association, as well as after each relationship in a join operator. The filter operator allows the definition of some criteria, which is used to reduce the set of selected instances to only those entities, which satisfy the criteria. When applied after a relationship, the operator filters the set of elements reachable over this relationship. Syntactically, filters are given denoted with square brackets ('[' and ']'), which enclose the filter criteria. The condition itself can contain any property of the current meta-model element, or any property obtained through a property operator, or a complex combination of several properties (see below). An example for a simple filter is

```
InformationSystem [ @Costs > 100 ] ;
```

which reduces the set of Information System instances to only those, whose value of the 'Costs' attribute exceeds one hundred. The '@' character denotes the name of an attribute of the context building block or association (here Information System). There are several more comparison operators for properties, all of which are listed in the Predicates and Predicate-Defining Operators. Furthermore, a simple filter can not only specify the dependency between a property and a value, but also between two properties. With the last example in mind, let us assume that the Information System building block also has a numeric attribute 'Revenue'. Then, to select all instances of Information System which cost more than they return, one would write:

```
InformationSystem [ @Costs > @Revenue ] ;
```

Complex criteria

Except for the simple criteria based on the value of one attribute, iteraQl also supports the definition of complex criteria through the three boolean operators **AND**, **OR** and **NOT**. The **AND** boolean operator is given through the ampersand character '&' given between the two sub-criteria. For example

```
InformationSystem [ @Costs > 100 & @Accountability = "tom" ] ;
```

will select all instances of Information System with costs greater than one hundred which also are accounted for by tom. Further, one might also want to select all Information Systems with costs over hundred which are accounted for by either tom or alice. This can be formulated as

```
InformationSystem [ @Costs > 100 & (@Accountability = "tom" |
@Accountability = "alice" ) ] ;
```

where the '|' character denotes the boolean **OR**. Note also that the two alternatives for the Accountability attribute are enclosed in brackets. This guarantees the unambiguous order of application of the sub-criteria. The third boolean operator in iteraQl is the negation (**NOT**), which is denoted with an exclamation mark ('!'). For example, the query

```
InformationSystem [ (!@Costs > 100) & (@Accountability = "tom" |
@Accountability = "alice" ) ] ;
```

will select all instances of Information System with costs no more than hundred and which are also accounted for by tom or alice. Note that while it is possible to include an arbitrary number of sub-criteria by combining them with the **AND** and **OR** operators, one should take the possible ambiguousness of the resulting expression into account and enclose in brackets accordingly.
Property Operators

Property operators extend the set of attributes available for a given building block or association by allowing the user to construct further, synthetic, attributes. The following property operators of iteraQl are covered: **view**, **count** and **foldLevel**.

The **view** operator creates a new property in the context building block or association type by 'copying' the property from a related building block or association. For example the query

```
InformationSystem [view(/infrastructureElements @name).contains("server")]
;
```

will create a synthetic attribute for the Information System building block and will obtain all instances, for which this property has a value containing the string 'server'. Note that the resulting instances are exactly the instances of Information System connected to an Infrastructure Element instance whose name satisfies the given condition.

The **count** operator can be applied to an attribute or to a relationship and retrieves the number of values or related instances, respectively. Consider the query

```
InformationSystem [count(/businessMappings/businessUnit) > 2];
```

Here, the **count** operator creates a new attribute in the Information System building block and for each instance the value of the new attribute is the number of Business Unit instances, related over any instance of the Business Mapping association.

The third property operator is the **foldLevel** operator. This operator requires either a self-referencing relationship (like the 'parent' direction of a hierarchy relationship), or a cyclic relationship, obtained for example through the join of two or more relationships. For each instance of the context building block or association, the value of the new **foldLevel** attribute will be the number of steps over the given relationship, until no further instances can be reached. To illustrate this, let us consider the query

```
BusinessProcess [foldLevel(/parent) = 1];
```

The result of this query will contain all instances of Business Process whose parent instance is the root of the hierarchy of Business Processes,

i.e. all first level Business Processes.

Further Operators

There are several further operators available in the iteraQl query language.
Objectify

The **objectify** operator can be used both for building blocks and associations, and for relationships. The operator is provided with an attribute of the context building block or association (in the case of a relationship this is the destination type of building block or association) and derives a new building block from the selected attribute. The values of this new building block are exactly the values of the attribute for the context type. Furthermore, every instance of the context building block or association is provided with a relationship to the instance(s) of the new building block which represent the value(s) of the attribute of the original instance. An example for an objectified building block is the following:

```
InformationSystem .objectify(@Accountability) /isValueOf;
```

The query will retrieve the relationship between the new building block and Information Systems. Every instance of the new building block will be related (over the **isValueOf** relationship) to exactly those instances of Information System which have the value represented by the corresponding instance of the new building block set for their objectified attribute.
Expand

The **expand** operator is also available for both kinds of query - building blocks or associations, and relationships. This operator requires a self-referencing relationship or a cyclic path and produces a new building block or association which enriches the base one with all instances reachable through the provided self-referencing relationship or cyclic path. When applied in the context of a relationship, the context type of the expand operator is the one the relationship leads to. The expand operator can, for example, be used to answer the question 'Which Information Systems depend on the same Technical Components as the CRM Information Systems?'

```
InformationSystem[@name.contains("CRM")] .expand(
/technicalComponentReleases /informationSystemReleases);
```

First, a new building block type which represents all Infomration System instances with 'CRM' in their name is created. Then this new type becomes part of a further building block type - the expanded type - which also includes all Information System instances which share a Technical Component instance with any instance from the first type. Thus, the instances of the expanded type are both the Information Systems with 'CRM' in their name and all instances related over a Technical Component instance.
Nullify

The **nullify** operator can be used both for building blocks and associations and for relationships, although there is a slight difference in its application. When applied to a building block, the nullify operator requires no argument. For example, the query

```
InformationSystem .nullify();
```

will create a new building block type, the Nullified Information System type, which extends the Information System type with one feature - the new type has one further instance, the **null instance**. The null instance has the property that it relates everything which is not related to any other instance of the Information System building block type. This can be useful, for example, when querying for Information Systems together with their related Technical Components, while also including those instances of Technical Component which are not related to any instance of Information System. The result set of the query

```
InformationSystem .nullify() /technicalComponentReleases;
```

will contain bindings from the null instance to all instances of Technical Component which are related to no Information System instance.

When applied in a relationship context, the nullify operator requires a further relationship, which is given as argument. In this case, the relationships are not from, but rather to the null instance. For example, the query

```
InformationSystem [@name.contains("Mgmt") | @name.contains("SAP")]
/children .nullify(/technicalComponentReleases);
```

will have bindings to the Technical Component null instance whenever an Information System instance from the children of the 'SAP' and 'Mgmt' Information Systems has no other Technical Component instance related.
Power

The **power** operator can only be applied to building blocks and creates a new building block type, the power type. The type has only one instance, which is related to all instance of the original building block type over the '**isContainer**' relationship. An example query is

```
InformationSystem[@name.contains("CRM")] .power() /isContainer;
```

which maps the power instance to all instance of Information System whose name contains 'CRM'.
Unfold

The **unfold** operator can be used only in a relationship context. The operator requires a self-referencing relationship or a cyclic composition of relationships, and produces a new relationship itself. The instances of the new relationship are all instances reachable over the argument relationship, after an arbitrary number of hops, until no new elements can be added. For example the query

```
BusinessProcess[@name.contains("Mgmt")] .unfold(/children);
```

will return the 'Mgmt' Business Process with bindings to all its hierarchical children, direct and indirect.

Reference

This part of the iteraQl user manual gives a reference of the elements of iteraQL:

- Predicates and Predicate-Defining Operators
- Naming conventions for Building Blocks and Relations
- Operators

Predicates and Predicate-Defining Operators

As described on the Queries, Operators and Predicates page, predicates are used to make selections of a Building Block's instances. This page gives a comprehensive overview on the use of predicates and predicate-defining operators.

- Working with Predicates
  - Negation of a Predicate
  - Combining several Predicates
    - "&"-combination
    - "|" (or)-combination
    - Combining more than two predicates
- Comparison Operators
  - Numeric Comparison
  - String Comparison
  - Date Comparison
  - Comparison Operators on Multi-Valued Properties
    - Multi-valued Left-Side
    - Multi-valued Right-Side
- Other predicate-defining Operators
  - Contains
  - Begins With
  - Ends With

Working with Predicates

A single predicate may be used by simply placing it in the square brackets of the filter-operator's syntax.

Example

Consider the Building Block *Information System* having the Property *costs* and assume the following scenario:

Information System	**costs**
CRM	700
Callcenter	300
EAM	99
BI	2000

The predicate *@costs < 500* may then be used via

```
InformationSystem[ @costs < 500];
```

which then returns the result:

InformationSystems[@costs < 500]
Callcenter
EAM

Negation of a Predicate

A predicate may be negated by placing a "!" in front of it. Taking the original predicate this will turn every "yes" into a "no" label and vice versa.

Example

Consider the basic example on this page.

The predicate *@costs < 500* gives the labeling

Information System	CRM	Callcenter	EAM	BI
Predicate *@costs < 500*	no	yes	yes	no

It may be negated by putting a "!" in front of it

```
! @costs < 500
```

This gives a new predicate:

Information System	CRM	Callcenter	EAM	BI
Predicate *!@costs < 500*	yes	no	no	yes

Therefore the query

```
InformationSystem[ ! @costs < 500 ];
```

consequently returns:

InformationSystems[@costs < 500]
CRM
BI

Combining several Predicates

Several predicates may be combined using "&" or "|" (meaning or).
"&"-combination

Two predicates may be combined via "&", meaning an instance of the Building Block must be labelled by *both* predicates with a "yes" to obtain a "yes" of the resulting predicate. Note that therefore the result of the use of "&" between two predicates provides itself a single predicate (that may be used further just as any other predicate). As well "&" is symmetric, meaning that the order of the two predicates to be combined does not matter.

Example

Consider the basic example on this page.

Combining the predicates _ @costs < 500 _ and _ @costs > 100 _ via "&" is achieved via

```
@costs < 500  & @costs > 100
```

which gives the predicate labeling exactly those instances of the Building Block *Information System* with "yes" that have costs greater than 100 and less than 500.

The labeling of all occurring predicates is:

Information System	CRM	Callcenter	EAM	BI
@costs < 500	no	yes	yes	no
@costs > 100	yes	yes	no	yes
@costs < 500 & @costs > 100	no	yes	no	no

"|" (or)-combination

Two predicates may be combined via "|", meaning an instance of the Building Block must be labelled by *at least one of the two* predicates with a "yes" to obtain a "yes" of the resulting predicate.Just as seen for "&", the result of the use of "|" between two predicates provides itself a single predicate and "|" is symmetric meaning that the order of the two predicates to be combined does not matter.

Example

Consider the basic example on this page.

Combining the predicates _ @costs < 500 _ and _ @costs > 100 _ via "|" is achieved via

```
@costs < 500  |  @costs > 100
```

which gives the predicate labeling exactly those instances of the Building Block *Information System* with "yes" that have costs greater than 100 or less than 500.

The labeling of all occurring predicates is:

Information System	CRM	Callcenter	EAM	BI
@costs < 500	no	yes	yes	no
@costs > 100	yes	yes	no	yes
@costs < 500 | @costs > 100	yes	yes	yes	yes

Combining more than two predicates

As seen above, the result of combining two predicates via "&" or "|" provides again a predicate that may be combined itself with another predicate. However, when using "&" and "|" combinations at the same time, it might be important to state the wanted order these combinations are intended to be made. This can be achieved using brackets "(...)". Any predicate may be placed into brackets without changing its meaning or validity. By placing a predicate resulting from a combination of two other predicates into brackets one forces iteraQL two first achieve the predicate inside the brackets and then execute any further operation with this result. Whenever "&" and "|" are used simultaneously, the use of brackets is highly recommended. If no brackets are used, iteraQL will execute the combinations in the order they are written in the query.
As well when negating combined predicates, the use of brackets is necessary.

Example

Consider the basic example on this page.

Compare the resulting predicates of

```
( @costs < 500 & @costs > 100 ) | @costs > 1000
```

and

```
@costs < 500 & ( @costs > 100  | @costs > 1000  )
```

In the former, the "&" combination is executed first, in the latter the "|" one.
Therefore the results differ as follows:

Information System	CRM	Callcenter	EAM	BI
(@costs < 500 & @costs > 100) | @costs > 1000	no	yes	no	yes
@costs < 500 & (@costs > 100 | @costs > 1000)	no	yes	no	no

Comparison Operators

This section resents the different comparison operators available for the iteraQl filter criteria for each supported data type. Comparisons may be made with numeric, string or date attributes. For obtaining a valid predicate, make sure both sides that are to be compared with each other are of the same nature.

Numeric Comparison

Syntax	**Meaning**
arg1 = arg2	Numeric equality between two arguments.
arg1 != arg2	Numeric inequality between two arguments.
arg1 < arg2	First argument is lass than second argument.
arg1 <= arg2	First argument is lass than or equal to second argument.
arg1 > arg2	First argument is greater than second argument.
arg1 >= arg2	First argument is greater than or equal to second argument.

String Comparison

Note: All string comparison operators are case-insensitive.

Syntax	**Meaning**
arg1 = arg2	String equality between the arguments.
arg1 != arg2	String inequality between the arguments.
arg1.contains(arg2)	The value of the property represented by arg1 contains the string or value of property represented by arg2.

arg1.beginsWith(arg2)	The value of the property represented by arg1 begins with the string or value of property represented by arg2.
arg1.endsWith(arg2)	The value of the property represented by arg1 ends with the string or value of property represented by arg2.

Date Comparison

Note: Date values should be given in quotes, to ensure correct recognition, e.g. "09/01/12" instead of just 09/01/12.

Synstax	Meaning
arg1 = arg2	The two arguments represent the same date with day accuracy.
arg1 != arg2	The two arguments represents different dates with day accuracy.
arg1 < arg2	The date represented by arg1 is before the date represented by arg2.
arg1 <= arg2	The date represented by arg1 is before, or the same as, the date represented by arg2.
arg1 > arg2	The date represented by arg1 is after the date represented by arg2.
arg1 >= arg2	The date represented by arg1 is after, or the same as, the date represented by arg2.

Comparison Operators on Multi-Valued Properties

It may happen that one of the two sides happens to be a multi-valued Property. In this case, the statement remains valid if the single values of both sides are still of the same kind, meaning both numeric, a string or a date.

There are two cases:
Multi-valued Left-Side

If the left side in the use of a comparison operator is a multi-valued Property, the predicate returns a "yes" label if *at least one* value in the list of the left side obtains a "yes" from the comparison.
Multi-valued Right-Side

If the right side in the use of a comparison operator is a multi-valued Property, the predicate returns a "yes" label only if *all* values in the list of the right side obtain a "yes" from the comparison.

Other predicate-defining Operators

There are further predicate-defining operators as well. Those will be presented as the operators on the operators' reference page .

Contains

Summary

Effect:	States whether a text contains a given text-string.
Input:	Text Property
Output:	Predicate

Syntax

```
%P%.contains( "%TEXTSTRING%" )
```

Explanation

The contains operator returns a predicate that selects instances of the affected Building Block by checking whether or not the given Property does

contain the text-string of interest.
Example

Example

Assume there are the Information Systems "CRM #1", "cc CRM #2", "t CRM" and "BI #1".

Then _@name.contains("CRM") gives the predicate

Information System	"CRM #1"	"cc CRM #2"	"t CRM"	"BI #1"
@name.contains("CRM")	yes	yes	yes	no

and the query

```
InformationSystem[ @name.contains("CRM") ];
```

would then return the Information Systems "CRM #1", "cc CRM #2" and "t CRM" .

Begins With

Summary

Effect:	States whether a text begins with a given text-string.
Input:	Text Property
Output:	Predicate

Syntax

```
%P%.beginsWith( "%TEXTSTRING%" )
```

Explanation

The contains operator returns a predicate that selects instances of the affected Building Block by checking whether or not the given Property begins with the text-string of interest.
Example

Example

Assume there are the Information Systems "CRM #1", "cc CRM #2", "t CRM" and "BI #1".

Then _@name.beginsWith("CRM") gives the predicate

Information System	"CRM #1"	"cc CRM #2"	"t CRM"	"BI #1"
@name.beginsWith("CRM")	yes	no	no	no

Ends With

Summary

Effect:	States whether a text begins with a given text-string.
Input:	Text Property
Output:	Predicate

Syntax

```
%P%.endsWith( "%TEXTSTRING%" )
```

Explanation

The contains operator returns a predicate that selects instances of the affected Building Block by checking whether or not the given Property begins with the text-string of interest.
Example

Example

Assume there are the Information Systems "CRM #1", "cc CRM #2", "t CRM" and "BI #1".

Then _@name.endsWith("1") gives the predicate

Information System	"CRM #1"	"cc CRM #2"	"t CRM"	"BI #1"
@name.endsWith("1")	yes	no	no	yes

Naming conventions for Building Blocks and Relations

In the iteraplan meta-model there are certain building block and relationship types. The IteraQL name of each building block or relationship type is almost identical with the name used in iteraplan, in English, the only difference being the omission of all spaces. For example, the Information System Domain building block of iteraplan is depicted as 'InformationSystemDomain'.
The following list provides an IteraQL reference of all building block and relationship types including their properties and relationship naming conventions.

Building Blocks

Building Block (iteraQL Name)	**Properties***	**Available Relations (iteraQL Name)**
All Building Blocks	• id • name • [description] • [lastModificationTime] • [lastModificationUser]	
BusinessDomain	• position • [Accountability]	• businessFunctions • businessProcesses • businessObjects • businessUnits • products
BusinessProcess	• position • [Accountability] Strategic Measurement • [Strategic value]	• businessDomains • businessMappings

BusinessUnit	• position • [Accountability]	• businessDomains • businessMappings
BusinessFunction	• position • [Accountability] Strategic Measurement • [Strategic value]	• businessDomains • informationSystems • businessObjects
Product	• position • [Rollout date] • [Accountability] Strategic Measurement • [Strategic value] LifeCycle • [Development (Start)] • [Development (End)] • [Live (Start)] • [Live (End)] • [Replacement (Start)] • [Replacement (End)]	• businessDomains • businessMappings
BusinessObject	• position • [Accountability]	• businessDomains • businessFunctions • informationSystemReleaseAssociations • informationSystemReleaseAssociations /informationSystemRelease • informationFlows

InformationSystem	• position • typeOfStatus • [System size] • [Complexity] • [Maintenance activity] • [Accountability] Strategic Measurement • [State of health] • [Costs] • [Strategic drivers] • [Value added] • [Strategic value] • [Operating expenses] LifeCycle • [Development (Start)] • [Development (End)] • [Live (Start)] • [Live (End)] • [Replacement (Start)] • [Replacement (End)]	• businessObject Associations • businessObject Associations /businessObject • businessMappings • businessFunctions • informationSystemDomains • informationFlows1 • informationFlows1 /informationSystemInterface • informationFlows2 • informationFlows2 /informationSystemInterface • Projects • technicalComponentReleases • infrastructureElements
InformationSystemInterface	• [Complexity] • [Data exchange] • [Degree of automation]	• informationFlows • informationFlows /informationSystemRelease1 • informationFlows /informationSystemRelease2 • technicalComponentReleases
InformationSystemDomain	• position • [Accountability]	• informationSystemReleases
ArchitecturalDomain	• position • [Accountability]	• technicalComponentReleases
TechnicalComponent	• availableForInterfaces • typeOfStatus • [Manufacturer] • [Accountability] Strategic Measurement • [Technical state of health] • [Compliance to guidelines]	• informationSystemReleases • informationSystemInterfaces • infrastructureElementAssociations • architecturalDomains

InfrastructureElement	• position • [Accountability]	• informationSystemReleases • technicalComponentReleaseAssociations
Project	• position • [Accountability] Strategic Measurement • [Costs] • [Strategic drivers] • [Value added] • [Strategic value]	• informationSystemReleases

*Square brackets imply their values being optional (i.e. they might be left empty).

Relationships

Relationship (iteraQL Name)	**Properties***	**Available Relations (iteraQL Name)**
All Relationships	• id • lastModificationTime • lastModificationUser	
BusinessMapping		• businessProcess • businessUnit • product • informationSystemRelease
Isr2BoAssociation	• CRUD	• businessObject • informationSystemRelease
InformationFlow		• informationSystemInterface • informationSystemRelease1 • informationSystemRelease2 • businessObject
Tcr2IeAssociation		• technicalComponentRelease • infrastructureElement

Self-Referencing Relationships

The following table describes the naming convention for each self-referencing relationship in both directions.

Relationship	**Names in iteraQl**

Hierarchy	parent and children
Usage	baseComponents and parentComponents
Specialization	generalisation and specialisations
Successor	successors and predecessors

iteraQL Example:

```
BusinessProcess /parent
[@name="Support"];
```

Operators

This page gives a complete reference of the operators available in iteraQL.

For each operator, there will be a short summary including its purpose, input and output, a specification of its syntax, a detailed explanation and a short example illustrating its usage.

An operator's syntax will be described using placeholders. Those are intended to be replaced by any valid iteraQL expression that meets their specification, for an explanation of the concept of Building Blocks, Relations, etc. see here. The placeholders are:

Placeholder	Meaning
%BB%	a Building Block, i.e. any iteraQL statement that produces a (synthetic) Building Block
%sBB%	a stand-alone Building Block
%aBB%	an associative Building Block
%R%	a Relation, i.e. any iteraQL statement that produces a (synthetic) Relation
%sR%	a self-referencing Relation
%P%	a Property, i.e. any iteraQL statement that produces a (synthetic) Property
%PREDICATE%	a Predicate

or any combination of placeholders indicated by "|", for example %BB | R% standing for either a Building Block or a Relation.

The available operators in iteraQL are:

- Join or "/"-Operator
- Filter or "[]"-operator
- Objectify
- Nullify
    - Nullify for Building Blocks
    - Nullify for Relations
- foldLevel
- Expand
- Power
- Count
- View

Join or "/"-Operator

Summary

Effect:	Combines two Relations to a single one.
Input:	Relation, Relation
Output:	Relation

Syntax

```
%R% / %R%
```

Explanation

The join operator is the simplest and one of the most frequent operators available in iteraQl. This operator takes two relationships and produces a new relationship by 'gluing' them together. If Building Block *A* has a Relation to Building Block *B* called *ab* and Building Block *B* has a further Relation *bc* to Building Block *C*, then the result of using the join operator by *ab/ac* will be a Relation between Building Block *A* and Building Block *C*.

What distinguishes the join operator from the others is the fact that it requires no keyword while it its grammar consists of a single "/". However, the user might be aware that not every use of a "/" corresponds to the usage of the join operator, as the "/" character is as well used to reference to a Building Block's relation and in fact the abstract example above would be "A/ab/bc", where the first "/" is not a join operator but just the reference to a Relation of *A*, whereas the second one is. (In fact this sophisticated distinction of the us of "/" should never produce any irritations as its intuitive use should produce the wanted results)

Also, the join operator is transitive, which means that it can be applied to an arbitrary number of relationships. If there was a fourth Building Block *D* and a Relation *cd* from *C* to *D*, "A/ab/bc/cd" would return the according Relation from *A* to *D*. (This transitive usage follows as well from the ability to concatenate operators as described in Queries, Operators and Predicates, as the output is a Relation and hence can be used as input for a further join operator.)
Example

Example

```
InformationSystem/businessMappings/businessProcess;
```

The Building Block *Information System* maintains a Relation *businessMappings* to the Building Block *Business Mapping* which furthermore has a Relation *businessProcess* to the Building Block *Business Process*. The result of the query above is therefore the Relation connecting all Information Systems to the Business Processes that belong to their according Business Mappings.

Note that this is in fact a single use of the join operator, as the first "InformationSystem/businessMappings" statement simply returns the businessMappings Relation of the Information System Building Block.

Filter or "[]"-operator

Summary

Effect:	Reduce a Building Block's instances according to a filter.
Input:	Building Block, Predicate
Output:	Building Block Relations and Properties as the input Building Block

Syntax

```
%BB% [ %PREDICATE% ]
```

Explanation

Besides the join operator one of the most frequently used ones is the filter operator.

Given a Building Block and a Predicate, it produces a filtered Building Block having the same features (i.e. Relations and Properties) as the original one, but does only possess those instances of the original one that satisfy the Predicate. Therefore it creates a new Building Block by taking a Building Block and throwing away selected instances of it.
Example

Example

```
InformationSystem[ @costs < 1000 ];
```

Given Building Block *Information System* and the Predicate "@costs < 1000", the filter operator will return a filtered Building Block of all Information Systems having costs less than 1000.

Objectify

Summary

Effect:	Transform a Property into a Building Block.
Input:	Property
Output:	Building Block only Relation "\isValueOf"

Syntax

```
.objectify( %P% )
```

Explanation

The objectify operator is a rather sophisticated one, as it takes a Property and transforms it to a synthetic Building Block with a Relation "*isValueOf" and no further Properties nor Relations. For each different value of the Property that has been given to at least one of the input Building Block's instances, an instance of the new Building Block is created that is related via _/isValueOf* to all instances of the input Building Block with this value.
Example

Example

Consider the Building Block *Information System* having the two instances "CRM and "BI" and its property *Accountability* and assume the following setting:

Information System	**Accounttability**
CRM	Mueller, Huber
BI	Mueller, Mayer

Given the ".objectify(@Accountability)" statement will then create a synthetic Building Block with one instance for each different values *Accountability* , which are "Huber","Mueller" and "Mayer".The "/isValueOf" statement then refers this new Building Block to the according Information Systems having these values:

.objectify(@Accountability)	**/isValueOf**
Huber	CRM
Mayer	BI
Mueller	CRM, BI

A complete iteraQL could be

```
InformationSystem .objectify(@Accountability) /isValueOf;
```

In the latter case the output of this statement would then be:

.objectify(@Accountability)	**/isValueOf: Information System**
Huber	CRM
Mayer	BI
Mueller	CRM
Mueller	BI

Nullify

Nullify for Building Blocks Summary

Effect:	Enhances a Building Block with an artificial "null" instance such that formerly unrelated instances are related to it.
Input:	Building Block
Output:	Building Block same Properties and Relations as input one

Syntax

```
%BB%.nullify()
```

Explanation

The nullify operator for Building Blocks enhances a Building Block by an artificial "null" instance that extends all of the Building Block's Relations in a certain way: Consider *A* and *B* to be two different Building Blocks where *A* has a Relation *b* to Building Block *B*. Now *A*.nullify() is the Building

Block generated by the nullify operator and having as well the Relation *b* to Building Block *B*. However, all instances of *B* that were not related to any instance of *A* by *A/b* are now related to the *null* instance of *A.nullify()*.

In comparison to the nullify Operator for Relations one might notice that the nullify operator for Building Blocks extends the destination instance (meaning itself) with respect to all its Relations., whereas the nullify Operator for Relations extends the target Building Block of this particular Relation by a null instance.
Example

```
InformationSystem/informationSystemDomains;
```

creates the output:

InformationSystem	**Information System Domain**
CRM	Application Server
BI	Application Server, Business Logic

Now, apply the nullify() operator

```
InformationSystem.nullify()/informationSystemDomains;
```

and obtain

nullified InformationSystem	**Information System Domain**
CRM	Application Server
BI	Application Server, Business Logic
null	Database

As the Information System Domain "Database" was not connected to any Information System, it is now related to the null-instance of the "nullified" Building Block *Information System*.

Nullify for Relations Summary

Effect:	Enhances a Relation by relating all instances of the destination Building Block to an artificial "null" instance of the target Building Block.
Input:	Relation
Output:	Relation to the same Building Block as the input one

Syntax

```
.nullify( %R% )
```

Explanation

The nullify operator for Relations extends a given Relation by relating all instances of the destination Building Block to an artificial "null" instance of the target Building Block.Consider *A* and *B* to be two different Building Blocks where *A* has a Relation *b* to Building Block *B*. Now *A.nullify(/b)* is the Relation generated by the nullify operator and establishes the same Relation from *A* to *B* as _ /b_ did, but furthermore relates all instances of *A* that were not related to any instance of *B* by *A/b* to an artificial null instance of *B*.

In comparison to the nullify Operator for Building Blocks one might notice that the nullify operator for Relations extends the target Building Block

by a null instance, whereas the nullify Operator for Building Block extends the destination instance (meaning itself) with respect to all its Relations.
Unfold Summary

Effect:	Summarizes all levels of a self-referencing Relation to a single one.
Input:	self-referencing Relation
Output:	self-referencing Relation to the same Building Block as the input one

Syntax

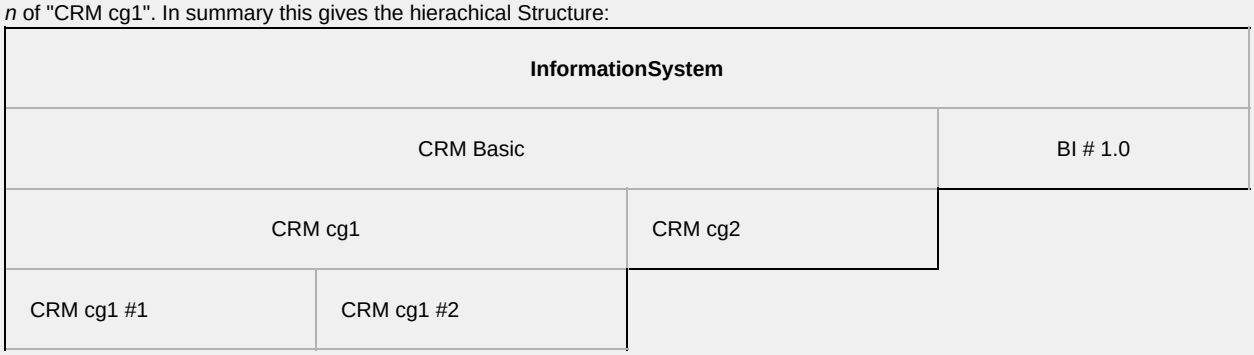
```
.unfold( %sR% )
```

Explanation

The unfold operator takes a self-referencing Relation and summarizes all of its levels by "unfolding" them to a single one. As the input Relation is self-referencing, it relates the a Building Block to itself and therefore the target Building Block of this self-referencing Relation must have itself the identical self-referencing Relation - by repeating this procedure one obtains the different *levels* of this self-referencing relation. Putting now all the different levels of this self-referencing Relation to a single one together gives the output of the unfold operator. To be more illustrative: The most commonly self-referencing relationship might be the */children* one. Considering the children of a Building Block they have their own children themselves, whereby this *children* Relation might be considered to be of 2nd level. The *.unfold(/children)* operation establishes now a single Relation from a Building Block not only to the children of first, but of all possible levels.
Note that this works as well with "circle" Relations. For Example if Building Block *A* has a Relation */ab* to Building Block *B* having a Relation *bc* to Building Block *C* having a Relation *ca* to Building Block *A*, then *ab/bc/ca* is a self-referencing Relation of A and therefore .unfold(/ab/bc/ca) a valid iteraQL statement.
Example

Example

Assume for this example the *InformationSystem* Building Block having the instances "CRM Basic", "CRM cg1", "CRM cg2", "CRM cg1 #1", "CRM cg1 #2", as well as "BI # 1.0". Furthermore assume that "CRM cg1" and "CRM cg2" are both systems derived from "CRM Basic", meaning they are *children* of "CRM Basic" in the model, and as well that "CRM cg1 #1" and "CRM cg1 #2" are *children* of "CRM cg1". In summary this gives the hierachical Structure:



Applying the ".unfold"-Operator by

```
InformationSystem.unfold(/children);
```

to the /children Relation then produces a new Relation:

InformationSystem	**.unfold(/children)**
CRM Basic	CRM cg1, CRM cg2, CRM cg1 #1, CRM cg1 #2
CRM cg1	CRM cg1 #1, CRM cg1 #2

Note that the Information Systems "CRM cg2", "CRM cg1 #1" and "CRM cg1 #2" do not appear as rows in this result, as they do not possess any children (if you would indeed want them to be included, look at the nullify() operator).

foldLevel

Effect:	Returns the foldLevel with respect to a self-referencing Relation, i.e. the Building Blocks position in the hierarchical structure induced by the Relation.
Input:	self-referencing Relation
Output:	Number Property

Syntax

```
foldLevel( %sR% )
```

Explanation

The foldLevel operator returns the level of a Building Block with respect to a self-referencing Relation's hierarchical structure. Given a self-referencing relation, the foldLevel of each instance of the Building Block is given by counting the number of times this Relation can be repeatedly applied until an instance is reached not connected to any further one via the given Relation. As an abstract Example, consider the Building Block *A* having the self-referencing Relation */a* instances "inst A start" , "inst A middle" and "inst A end" where */a* connects "inst a start" with "inst a middle" and the latter with "inst a end". The foldLevels with respect to */a* of "inst A basic" , "inst A middle" and "inst A end" are therefore 2, 1 and 0 respectively. A more precise example can be found just below.
Note that counting starts at "0", meaning an instance not related to any further one is considered having foldLevel 0.

Example

Consider the same example as above for the unfold operator. The following table presents the foldLevels of all instances of Information Systems:

foldLevel(/children)	InformationSystem
0	CRM cg2, CRM cg1 #1, CRM cg1 #2 , BI # 1.0
1	CRM cg1
2	CRM Basic

```
InformationSystem[ foldLevel(/children) = 1];
```

would return those Information Systems whose foldLevel is extactly one, meaning those Information Systems that indeed have at least one child, but whose children do not possess any further children. In the given example, this means

Information System
CRM cg1

Expand

Summary

Effect:	Expands the a (most of the times filtered) Building Block with instances gained via a self-referencing relation.
Input:	self-referencing Relation

Output:	Building Block expanded by instances reachable by the self-referencing Relation.

Syntax

```
.expand( %sR% )
```

Explanation

The expand operator extends a Building Block by all instances that can be reached via a self-referencing Relation. Obviously, this only seems reasonable if the Building Block has been filtered before, as otherwise all it would already possess all instances in iteraplan. Consider a Building Block *A* having the property *name* and a self-referencing Relation */a*. Assuming Bulding Block a having the instances "inst B #1","inst B #2" and "inst C #1" as well as "inst D #1" which is a child of "inst B #1". Then *A[@name.constains("B")].expand(/children)* gives the Building Block with instances "inst B #1" "inst B #2" (as both have a "B" in their names) and "inst D #1" (added by the expand operator).

Power

Summary

Effect:	Creates a new Building Block having a single instances that summarizes all instances of the input Building Block.
Input:	Building Block
Output:	Building Block having no Properties and exactly one Relation "/isContainer" and exactly one instance that is related via "/isContainer" to all instances of the input Building Block.

Syntax

```
%BB%.power()
```

Explanation

The power operator gives a Building Block that has a single instance that is connected to all instances of the given one. It therefore produces from a given Building Block, a new (very synthetic) Building Block, that does not inherit any Relations or Properties from the input Building Block, but does possess a single Relation */isContainer* and single instance that is then related to all instances of the input Building Block via /isContainer. This operator might be helpful whenever one needs to take all Building Block's instances into consideration at once.
Example

Example

Consider the Building Block *Information System* that has the property *Accountability* and assume the following setting

Information System	Accountability
CRM	Mayer, Mueller
BI	Mayer
CC	Schmidt

If the question was, whether Mr Mayer and Mr Huber are responsible each for at least one Information System, the two queries

```
InformationSystem.power()[ view( /isContainer @Accountability) = "Mayer"] ;
```

```
InformationSystem.power()[ view( /isContainer @Accountability) = "Huber"] ;
```

would give the answer by returning for the first a non-empty and for the latter an empty result.
Review the according references of the hereby used filter and view operator for more detailed information about their usage.

Count

Summary

Effect:	Counts the number of related Building Blocks or set values of a Property, respectively.
Input:	Relation OR Property
Output:	Number

Syntax

```
count( %R | P%)
```

Explanation

The count operator maybe applied to a Relation or a Property and counts the number of related instances via a Relation or the number of set values of a Property, respectively.
Example

Example

Consider the Building Blocks *Information System* and *Information System Domain, whereby the former is connected to the latter via the Relation "/informationSystemDomains". Furthermore the Building Block _Information System* possesses the Property *Accountability*. Assume the following scenario:

Information System	**Accountability**	**/informationSystemDomains**
CRM	Mueller, Mayer	Business Logic, Database
BI	Mueller	-

Applying the *count* operator to the Property *Accountabiliy* and the Relation */informationSystemDomains*, respectively, does then give:

Information System	**count(@Accountability)**	**count(/informationSystemDomains)**
CRM	2	1
BI	1	0

In a more practical example, a possible iteraQL query answering the question "Which Information System does not have / belong to any IS Domain?" via

```
InformationSystem[ count( /informationSystemDomains ) = 0];
```

This would return the Information System "BI".

View

Summary

Effect:	Projects a Property of a related Building Block to the Building Block of interest.
Input:	Relation and Property
Output:	Property

Syntax

```
view( %R% %P% )
```

Explanation

Given a Relation and a Property of the Building Block targeted by the Relation, the *view* operator projects this Property to the original Building Block. If the Relation may connect one instance of the original Building Block to several instances of the destination Building Block, all Property values those instances' are projected as a multi-valued Property. In an abstract example, consider the Building Block *A* connected via the Relation */ab* to the Building Block *B* that has the Property *@p*. Then *view(/ab @p)* returns a Property of *A* where each instance of *A* possesses all values of *@p* the related instances of *B* have.
Example

Consider the Building Blocks *Information System Domain* and *Information System* whereby the former is connected to the latter via the Relation */informationSystemReleases*. Now assume the following scenario:

Information System Domain	/informationSystemReleaseess
ISD 1	CRM, BI
ISD 2	BI

Information System	Accountability
CRM	Mayer, Mueller
BI	Huber

The view Operator applied via *view(/informationSystemReleases @Accountability)* therefore returns a Property of the Building Block *Technical Domain* with the instantiation

Information System Domain	Accountability
ISD 1	Mayer, Mueller, Huber
ISD 2	Huber

A possible iteraQL query is

```
InformationSystemDomain[ view( /informationSystemReleases @Accountability ) =
"Mayer" ];
```

requesting all IS Domains with at least one Information System belonging to the Accountability of Mr Mayer - in this case this query would return the IS Domain "ISD 1".

Building Block Lists and Search

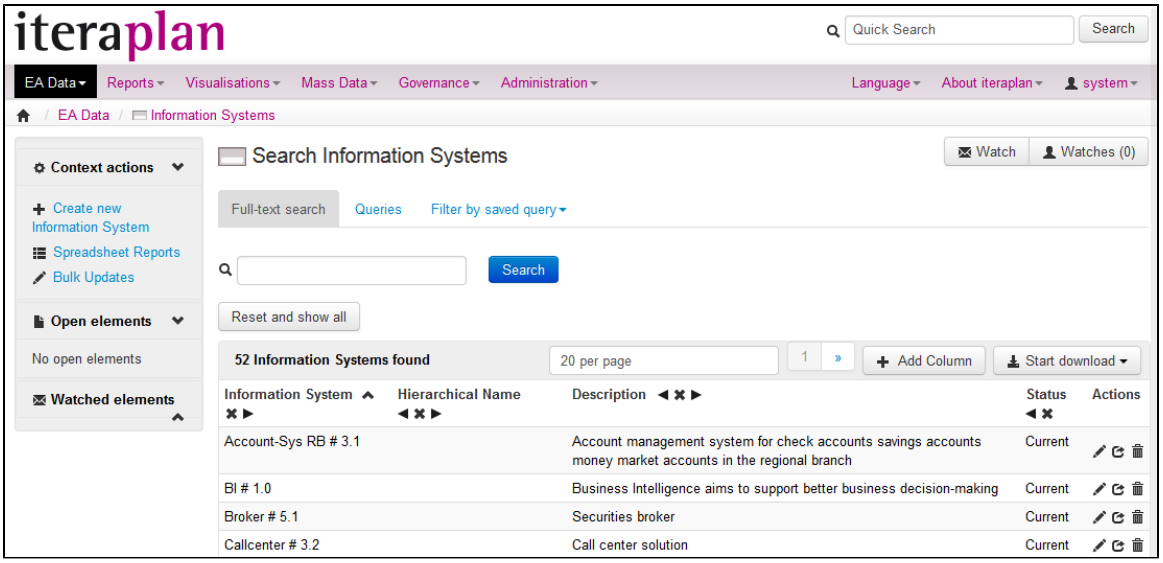
Once you select a building block from the menu on the top or from the home screen you will be taken to an overview page that displays all elements you created for this type of building block. An overview of all elements for all building block types can be found here: EAData - Overview

New since iteraplan 3.0: Context actions and column actions

You find most context specific actions at the left side of each page. In this case you can create a new information system, create spreadsheet reports and define bulk updates. Moreover you can subscribe to receive a notification, when changes of this element occur ("Watch" button at the top right corner) and see who watches this element.

You can add, remove, reorder and sort by columns.

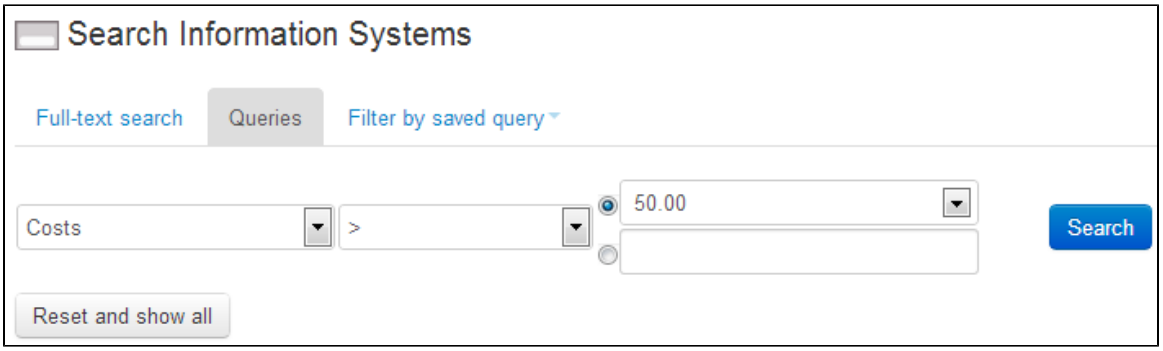
- Add a new column: Click on the "+ Add Column" button, choose the new column in the appearing dialog, and confirm with "Update". You can doubleclick as well.
- Remove a column: Click on the x sign below or next to the column header.
- Reorder columns: Click on the right/left arrow icon below or next to the column header.
- Sorting: Click on the column header. If the table wasn't sorted, it will be sorted ascending by this column. Otherwise it will toggle the sort order. The table(s) are initially sorted ascending by the first column.



Overview page of the building Block "Information Systems"

New since iteraplan 2.7: Full text search

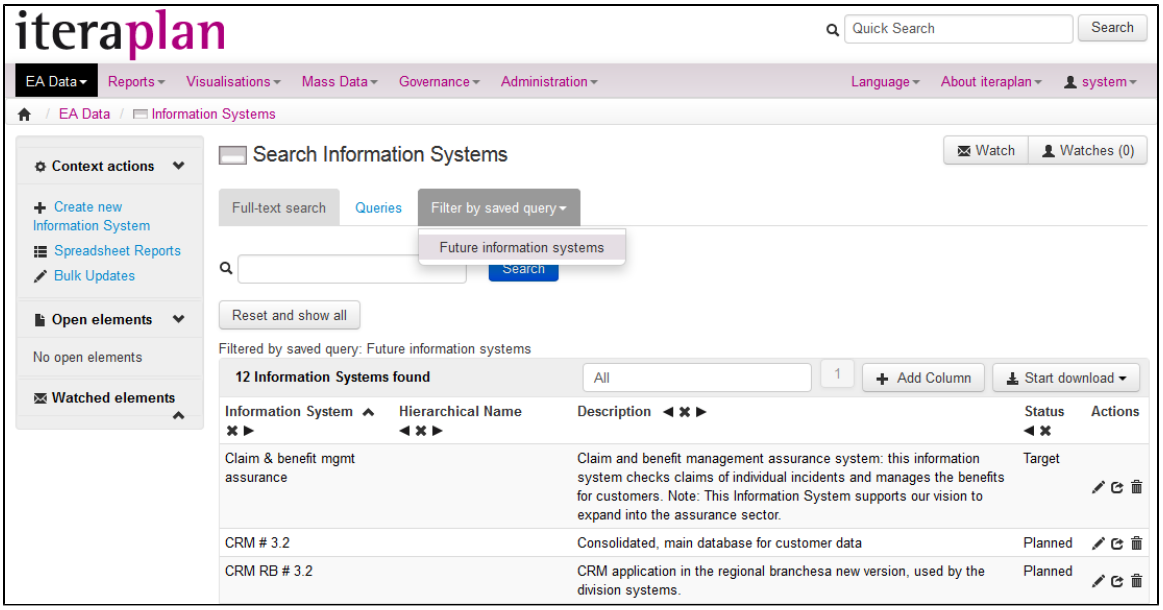
You can also use the search box on this page to limit the results. This means not only the name of the building block is searched for, but all assigned attributes and description are taken into account. When you execute a search with a blank search box or press the "Reset" button, all items will be returned.



Searching Information Systems that have Costs over 50

You can also use the attribute filter that is displayed in figure "Search Information Systems". It allows you to quickly formulate queries for building blocks of the current type without using the more complex Spreadsheet Reports. You may herein only choose one attribute for the search. You can either pick one of the available values or type your own text below. If you want to carry out more advanced queries, you have to use the Spreadsheet Reporting capabilities.

Once you've saved a Spreadsheet Report query ("saved query"), you can use it in the overview pages to filter the result list. To do so, click on the **filter by saved query** drop down button and choose the name of the query you want to execute. The result list will be displayed underneath. This allows you to run powerful queries against your data from this overview screen. One example to use this feature is to display all Information Systems that use a specific Technical Component. You can also use the address (url) of this filtered view and bookmark it or send it via mail.

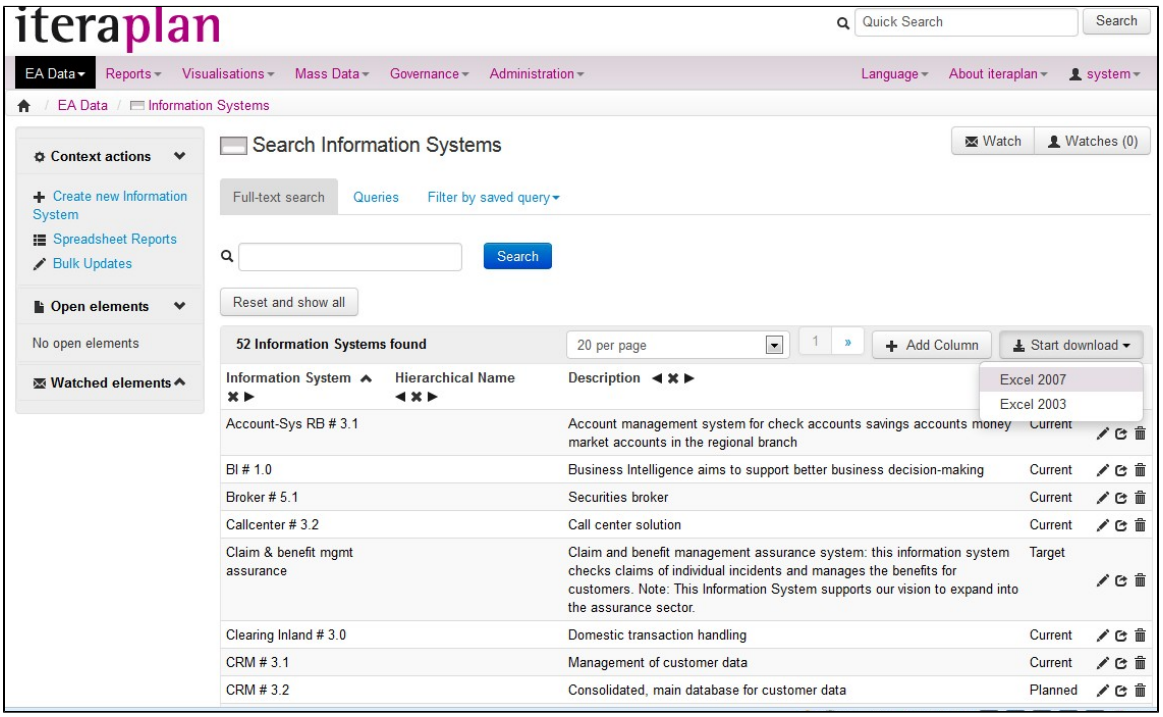


Information systems filtered by a saved spreadsheet report query

List-Export as Excel

New since iteraplan 3.3: List-Export

By clicking on the button "Start download" on the right top of the list and selecting a Excel format, you can download the current list as Excel workbook.



The downloaded workbook will contain exactly one sheet with the current Building Block type and the same number of found elements as in the current view (all elements in one sheet).

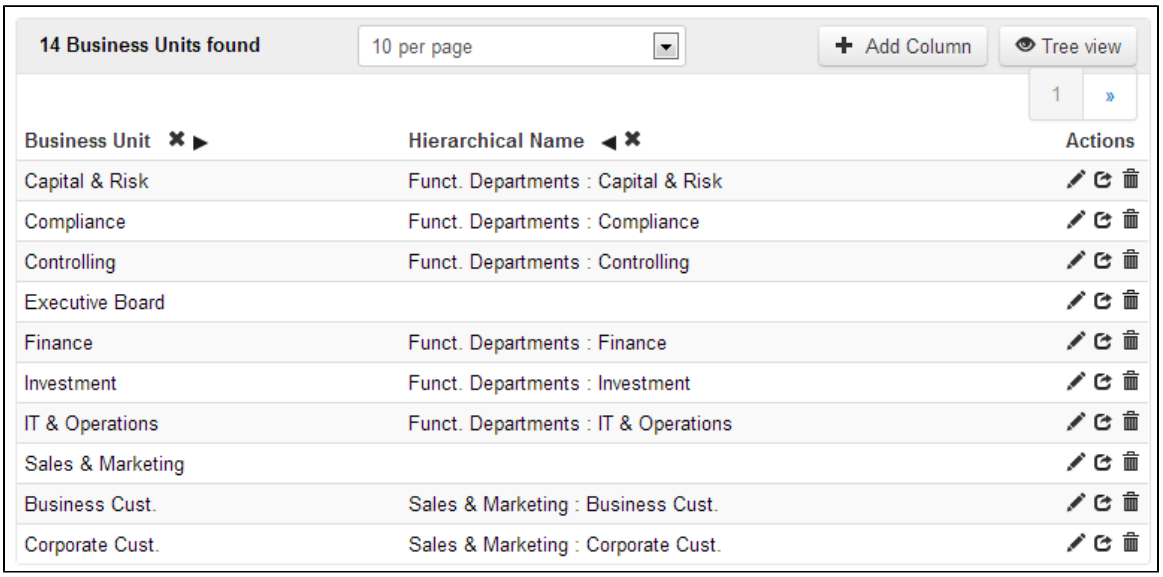
⚠ The Excel sheet can't be re-imported again. It is just a snapshot of the current page view.

Building Block Views

Once the user selects a building block type from the main menu or from the home screen, an overview page is displayed listing all building blocks of this type. The user can view them as a list with elements spread over one or several pages or as a tree representing the hierarchical structure of the elements.

List View

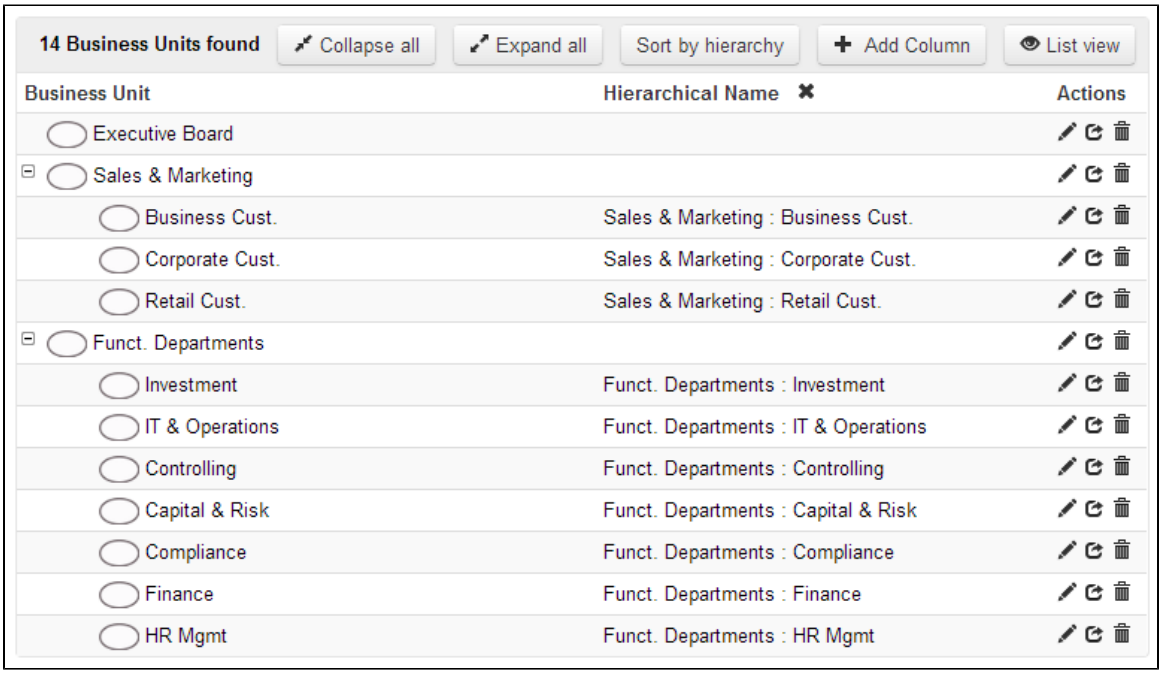
A default view for all elements. Please see Building Block Lists and Search for details.



Tree View

This view provides a comprehensive overview of the elements and the hierarchical child-parent relations among them. This feature is available for all the building block types which contain a hierarchical structure.

The tiers can be collapsed by clicking the "-" icon near the name of the element, and expanded by clicking the "+". You can expand or collapse all tiers by clicking the respective buttons as well. You can open an element by clicking its name.



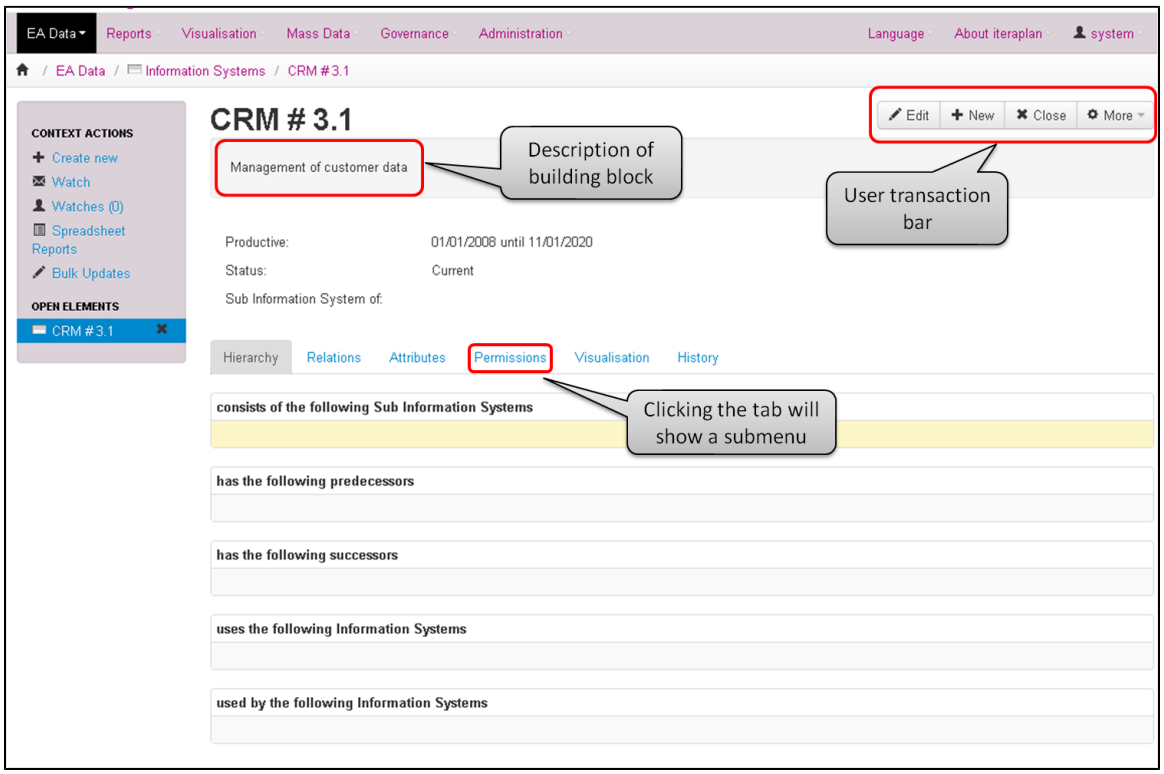
Drag & Drop

Once the tree is sorted by hierarchy (see respective button) it is possible to enable reordering. Switching to that mode allows for a simple way to alter parent-child relationships as well as the ordering of building blocks on the same hierarchical level. To move a building block or a whole sub tree drag it with the mouse and drop it at the desired position in the tree. Building blocks can only be placed above, between or below existing building blocks. After a short automatic refresh the building block is repositioned.

> **instant persistance**
> Moving a building block cannot be undone automatically as the reordered tree is saved instantly after each drag & drop action! Disabling reordering should not be confused with saving the reordering.

Building Block Details

When opening a Building Block, its details are categorised in different tabs (i.e. Hierarchy, Relations, Attributes, Permissions, Visualisation, or History). The user transaction bar offers various actions like edit, print or delete.



Building Block Details

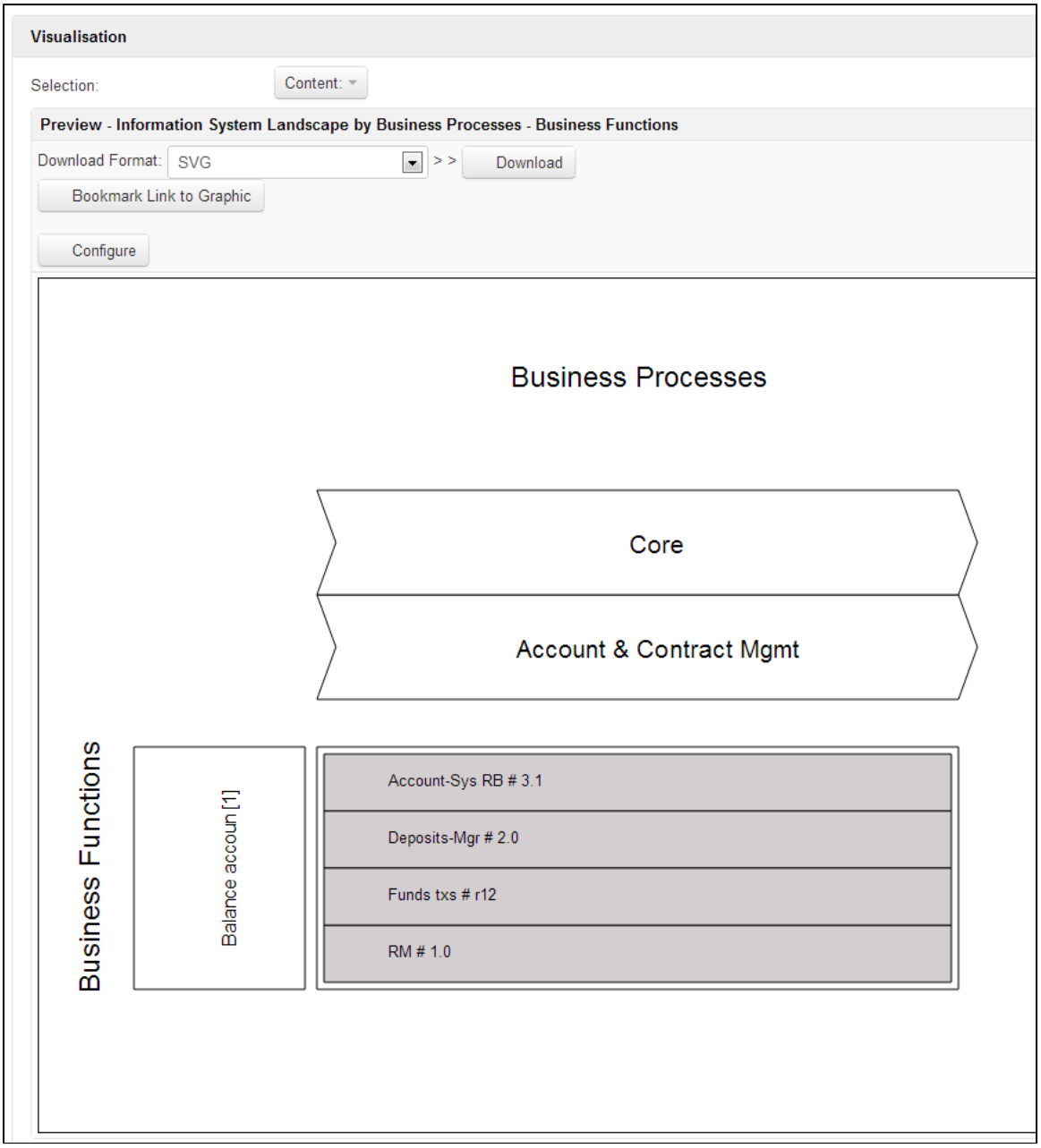
Visualisation Tab – Selecting a Graphical Report

For several types of Building Blocks such as Information Systems, Business Processes, Projects, or Technical Components, there is a special tab called 'Visualisation'. Clicking this tab opens an interface which gives you easy access to graphical reports (see Diagram Reports for details). These reports are specific to the current Building Block; they show the relation of the current Building Block to its neighbouring Building Blocks.

Let's have a closer look at the screenshot below:

Under 'Content' you can pick a diagram type. A visualisation for an Information System, for instance, consists of the set of related Technical Components and their respective Architectural Domains in a Landscape Diagram, or all adjacent Information Systems in an Information Flow Diagram. Alternatively, the set of related Projects can be seen in a Masterplan Diagram.

The screenshot illustrates an exemplary preview of a business landscape for an Information System. You can download the displayed graphical report by selecting your format of choice, and then pressing the 'Download' button. Alternatively, you can jump to the graphic's configuration page (button 'Configure') where more options are available, e.g. choosing colours for different attributes, assigning various line styles etc.

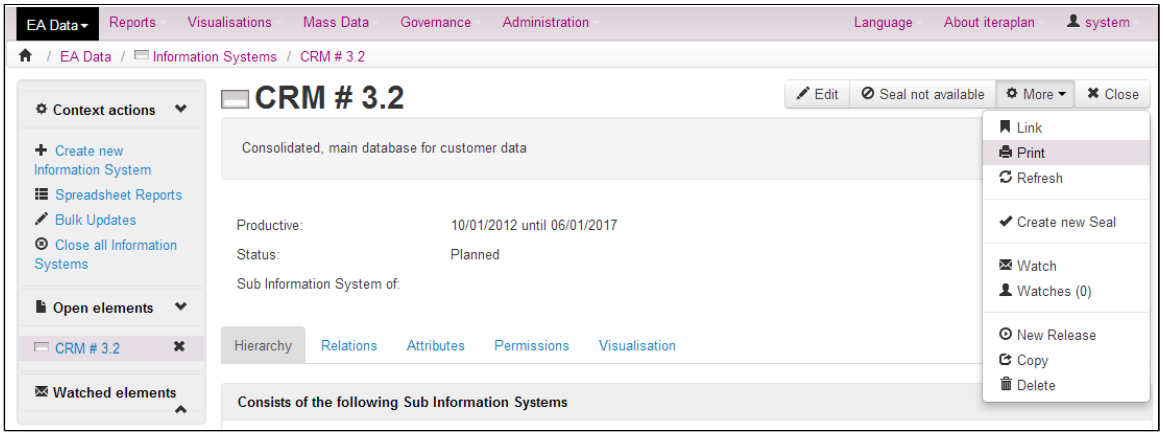


Visualisation of an Information System's Business Landscape by Business Processes and Business Functions

More Functions

Printing

Usually, all data of a Building Block is separated into different tabs. When printing a page, however, all data belonging to the current Building Block is displayed at once. To start printing you can either use the **Print** icon in the upper right of the transaction bar or use the printing command from your browser (usually Ctrl+P / File -> Print). You may also use the print preview function of your browser before printing, in order to check the

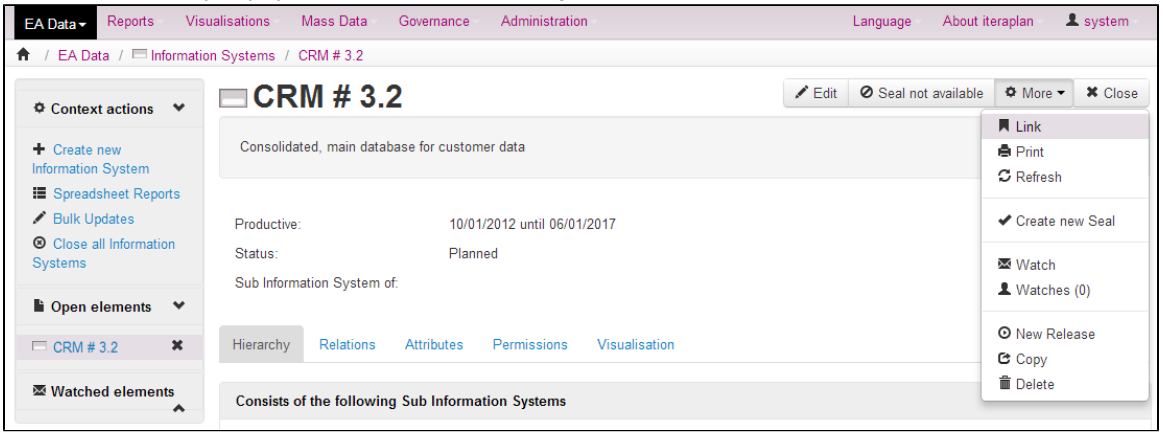


results.

Print button to start the print dialog

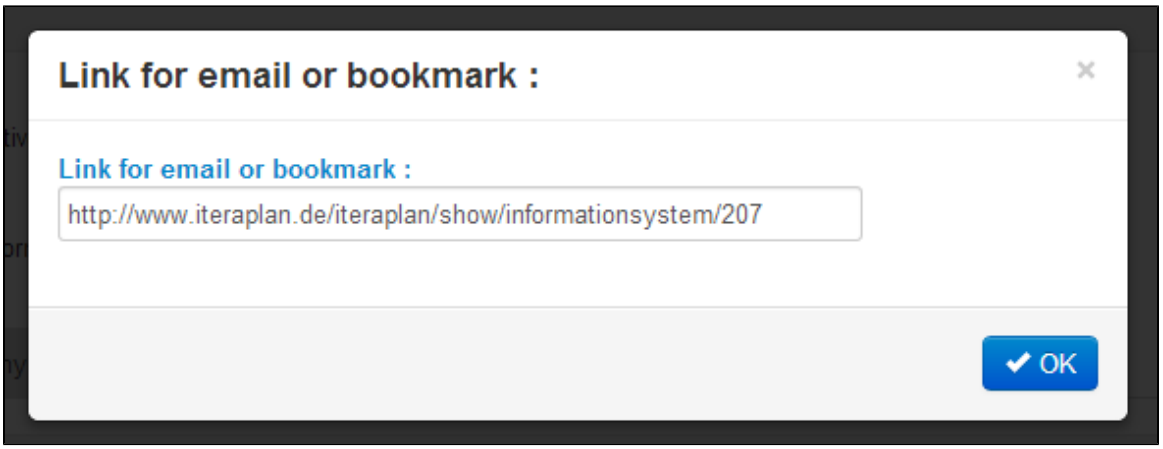
Bookmark Building Blocks

In iteraplan you are able to access building blocks directly via URL. In order to get the exact address, you have to click on the **Link** icon. Analog to the **Print** icon it is only displayed in View mode of a Building Block.



Bookmark Icon for Building Blocks

Clicking on the icon opens a little popup window where the address of this building block is linked in the title and displayed in an input field, so you can copy the URL to your email or to add it to your browser bookmarks.



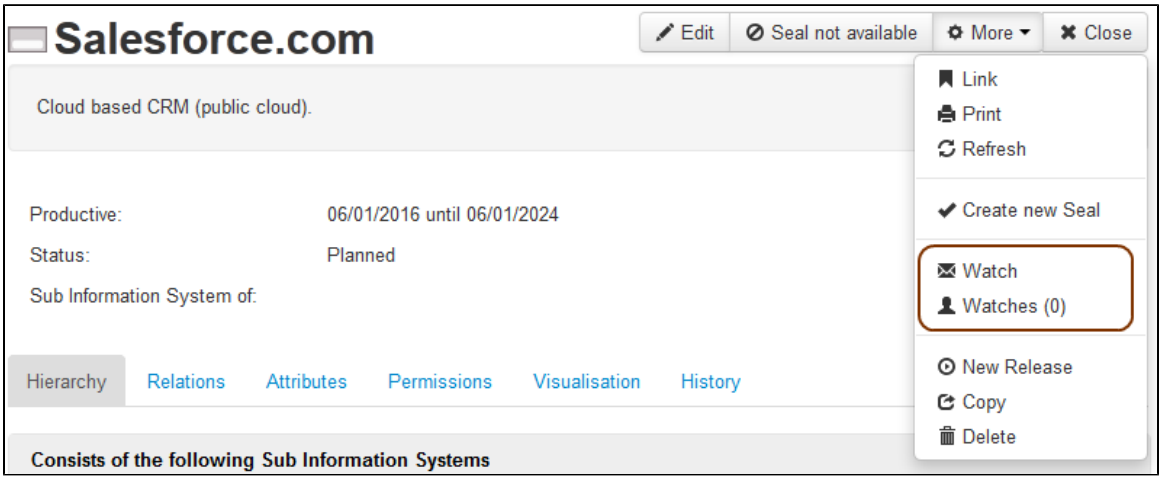
Bookmark of a Building Block

Watch Element Changes with Email

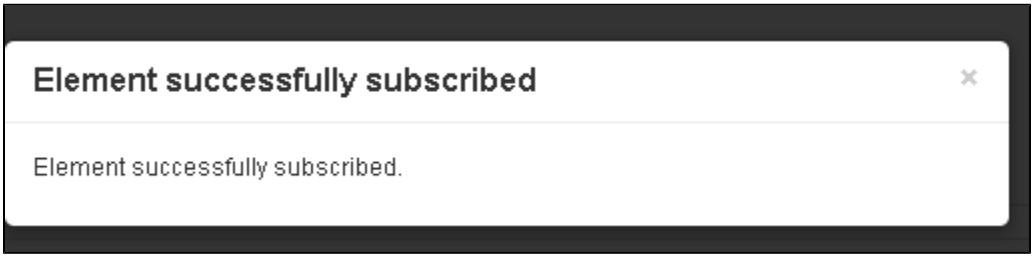
Users can watch building blocks, so that they receive a notification email every time the watched element is updated or when the element is deleted. A user can also unsubscribe from watched elements in the same way. Every user can see the list of all users that are subscribed to a certain element.

Note that only direct edits on a watched element trigger a notifications. If the list of associated elements of a watched element changes because the element on the other side of the association is edited, no notification is triggered.

When a user watches an element, this relates to this particular element only, but not to any related elements (like sub-elements in a hierarchy). If another element is edited and a new relation to a watched element is created, the watchers will not get notified. Only if the relation is created from the watched element, its watchers will receive an email. This is a known limitation.



The user doesn't watch this element yet. in order to subscribe he clicks on Watch.



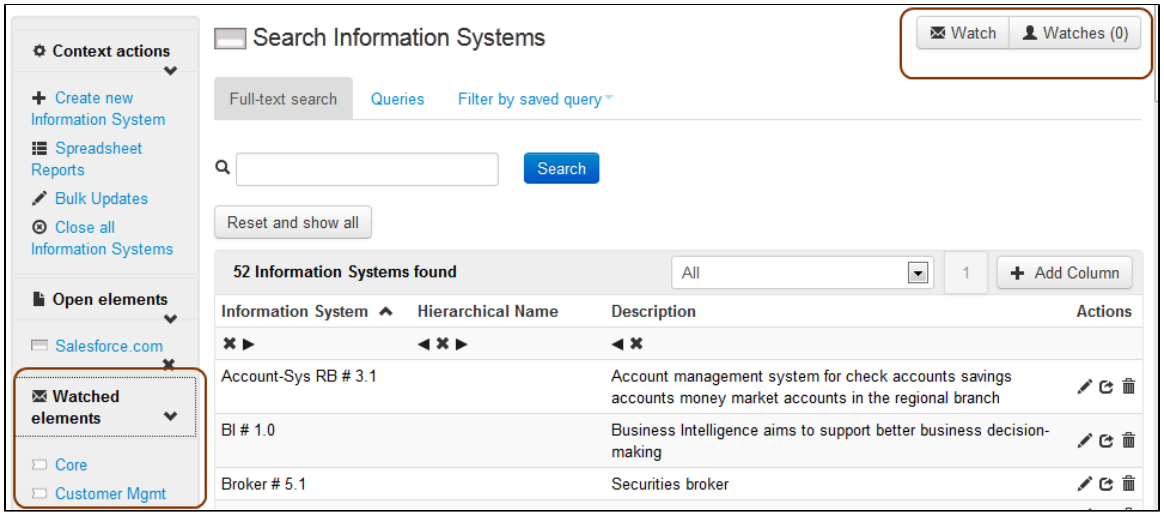
Confirmation that the subscription was successful.

An additional use case is to watch the list of elements of one building block type. If you choose to watch the list of Business Processes, for example, you will receive a notification as soon as a new Business Process is created or an existing Business Process is deleted. Note however, that you will not be notified about simple changes to individual Business Processes. You still need to watch each process individually.

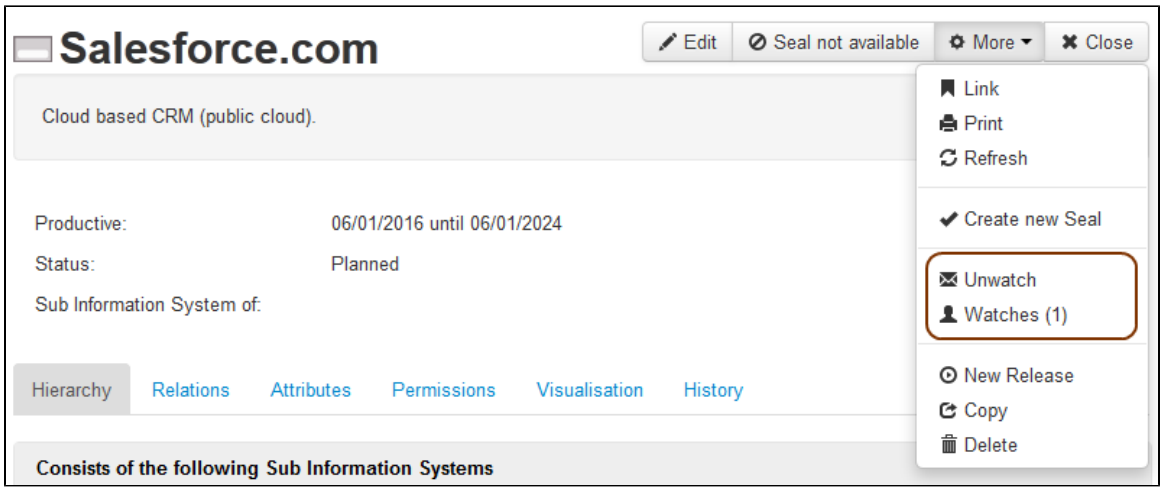
Bulk updates are handy for subscribing to many elements at once. Select the desired elements from the result list and click *More options* and *Bulk subscribe to the selected elements*.

Before you can watch elements you have to enter a valid e-mail address for your user in iteraplan. To do so, click your username in the upper right corner of the page, and click *Profile*. Click Edit to change your email address. Log off and on again to make the change become effective.

Notification emails will only be sent to you if the server administrator has activated the functionality and set up the email submission parameters.



Users can watch the list of all elements of a type (here: Information Systems). When an element is deleted or a new one is created, he will be notified. On the left, an expandable area lists the currently watched elements.



The user now watches this element for changes. He can choose to unsubscribe again.

iteraplan stores the email address of each user in his profile. These addresses are used to send change notifications to. The sender address (in the *From* field) of notification emails is configured once in the `iteraplan.properties` file and can only be modified by the server administrator. The server administrator also has to make all email-related settings in that configuration file, including SMTP server, port, encryption and authentication settings. All required configuration properties are explained in table below.

Parameter Name	Description	Example
notification.activated	Used to enable the notifications. Possible values are **true** and **false**	true
notification.smtpserver	The SMTP server address	smtp.company.com
notification.email.from	The address used for sending the emails	iteraplan@company.com
notification.port	The SMTP server port	25 or 465 (SSL)
notification.ssl.enable	Used to enable SSL-secured communication with the SMTP server. Possible values are **tr ue** and **false**	true
notification.starttls.enable	Used to enable use of the STARTTLS mechsnism for securing communication with the SMTP server. Possible values are **true** a nd **false**	false

notification.username	The username for authenticating with SMTP server. Should normally only be used only if the SSL is enabled	user@company.com
notification.password	The password for authenticating with SMTP server. Should normally only be used if the SSL is enabled	userpassword

Email texts and subject lines are created from template files on the server and are not localised. User profiles don't contain a preferred language setting, so all emails are sent out in English.

> ⚠ The notification service does not consider any permissions on Attribute Groups. Therefore an email might contain more information than the user is permitted to see on the web page. Be aware of this fact when setting up Roles and Permissions.

Creating and Editing Landscape Data

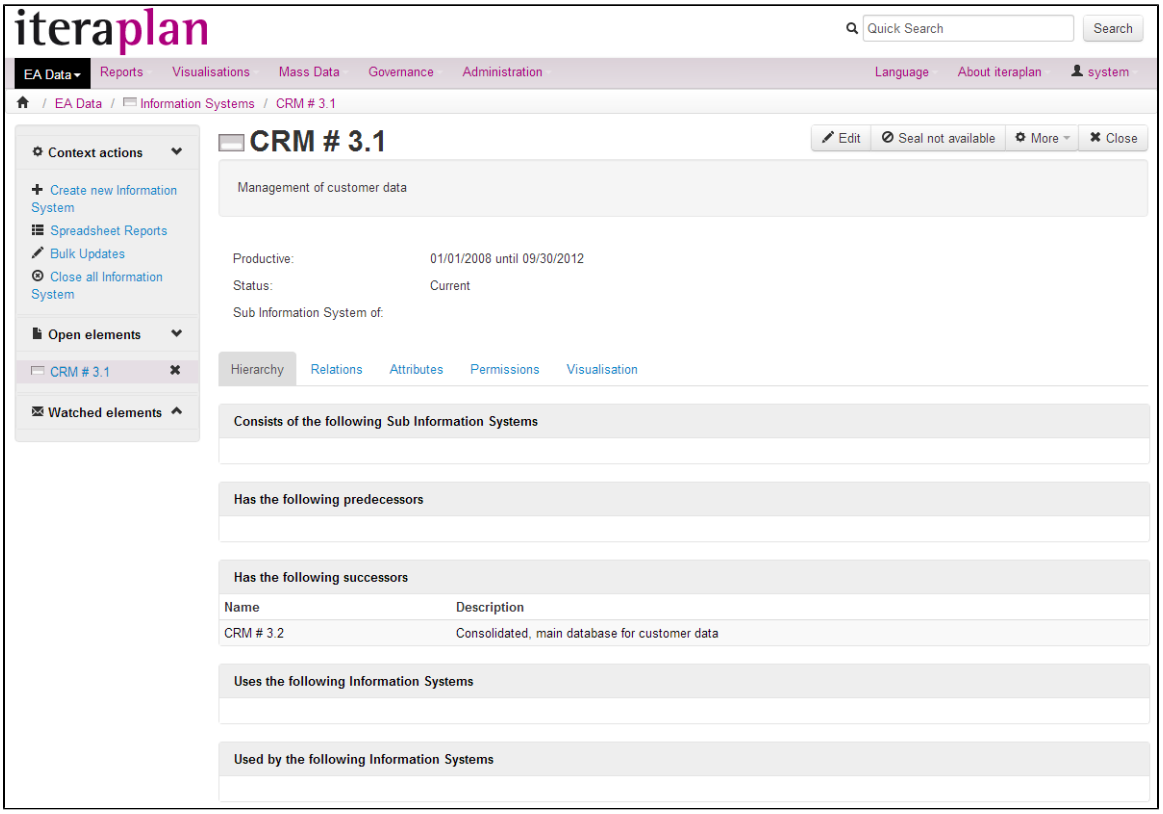
This section surveys the iteraplan pages with which you enter and edit the landscape building blocks and their attributes.

General

Screen layout

Each application page for entering and modifying data has a similar layout. Along the top of the iteraplan window is the toolbar with which you can toggle between Edit and View mode (and create new releases).

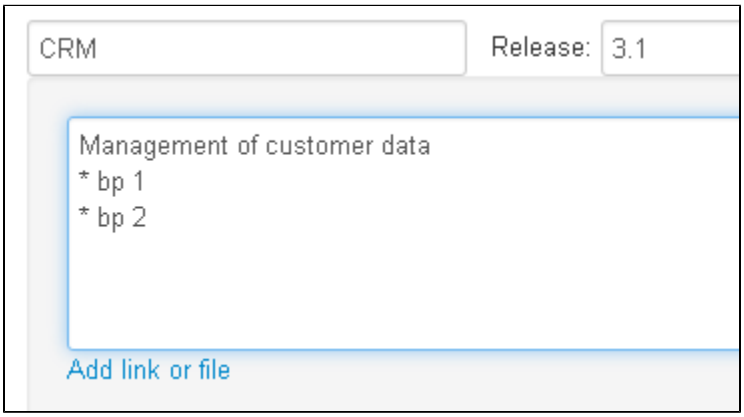
In View mode, the area beneath the toolbar displays the properties of the selected Building Block (name, description, links to external documents and, if available, additional data such as the productive timespan). Below this is a tabbed area: each of the tabs opens a set of information pertaining to the block. The choice of tabs differs depending on what type of block you are working on. With default settings, this area includes tabs for entering and modifying user-defined attribute values and for setting permissions. On the left side, you can find an overview of all available context actions, like create new, watch and watches, spreadsheet reports and bulk updates.



Layout of a page for editing a Building Block

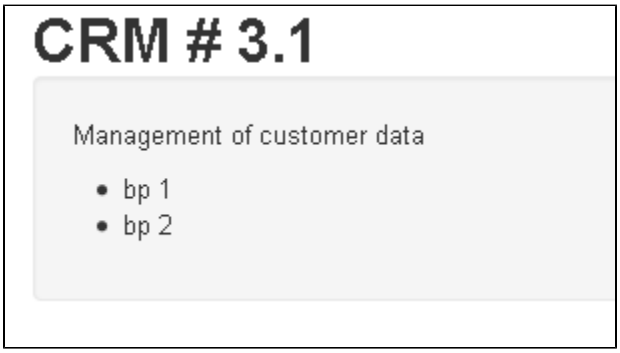
Using wiki syntax

You can use Wiki syntax inside of description and Text-Attribute fields. This is also symbolized by a small icon that is displayed in the top right corner of the text input box. By default the XWiki 2.0 syntax is used. A reference of the Xwiki notation can be found here.



Description input field that supports wiki syntax

When the description is displayed, the wiki syntax will be interpreted and displayed as styled html.



Interpreted wiki syntax

Some of the most common markup you might want to use:

What you need to type	What you will get
****bold****	**bold**
//italics//	*italics*
__*underline*__	underline
--~~strike~~--	~~strike~~
##monospace##	`monospace`
^^superscript^^ Normal	superscript Normal
,,subscript,, Normal	subscript Normal
> blockquoted text	*blockquoted text*
=== Heading 3 === ==== Heading 4 ====	***Heading 3*** **Heading 4**

* un-numbered ** and	• un-numbered • and
1. numbered 11. lists	1. numbered a. lists
image:[http://someAbsoluteAdressOfAnImage Example image:http://www.iteratec.de/sites/default/files/pictures/iteratec-logo-180.jpg	
(% style="text-align:center;color:blue" %) Text	Text

You can also use tables:

|=Title 1|=Title 2
|Word 1|Word 2

will become:

Title 1	Title 2
Word 1	Word 2

Please note that when exporting to excel, the mark-up is retained to ensure round trip capabilities.

The description that is displayed in search result lists is not styled, because this would lead to very unappealing result lists, instead the plain text without mark-up is displayed.

Links to other resources

In order to help you to use links to resources within the description of elements, iteraplan provides you an appropriate form. It is accessible in Edit mode via the link **Add link or file** located direct underneath the description text field.

The form provides three input fields. The first one is for the **Link title** (which will be displayed in View mode), the second for the **Link address**, which references a resource, and third one enables referencing a file. Link address can be either a URL or address of a file. To specify a file, click **Browse** to open the standard file dialog box and locate the file. Be sure that other computers in your network will also be able to access this path. To add the link to the description of the building block, click the **Add** button. This will add the wiki-syntax for a link to the resource.

In case of a link to a file this will only generate the wiki-markup for the link and the filename. You now need to manually add the full path to the file. This cannot be done automatically, because browsers refuse to gather this information using javascript for security reasons. Example:

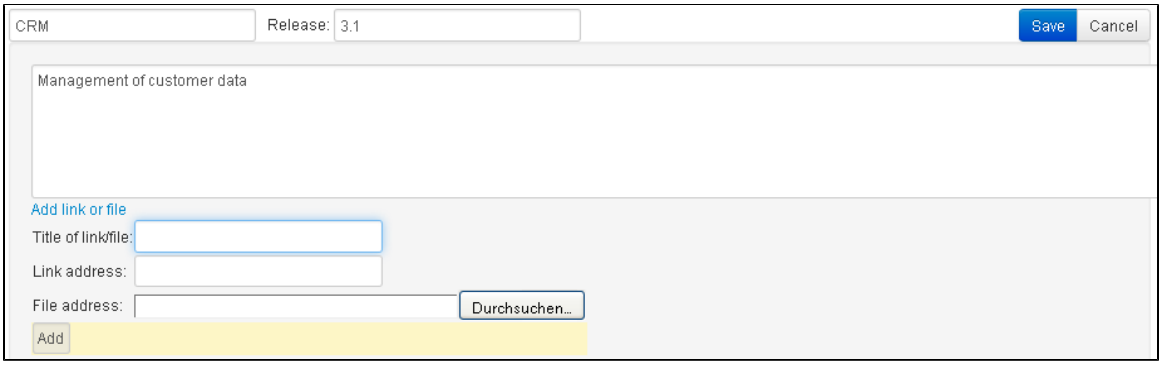
```
[[LinkTitle>>file:///Filesrv1/path/to/file.pdf]]
```

or, if you use network drives:

```
[[LinkTitle>>file:///G:/path/to/file.pdf]]
```

After saving the element, the resource will be linked within the description field if the building block.

Standard Web-Links will open up in a new Browser-Tab on click. When linking a file, browsers refuse to open the link directly by clicking onto it. This is also for security reasons. If you want to open such a link, you have to right-click the link to open the shortcut menu, then select **Copy link** t o copy the address, and open it in a separate browser window or directly in a Windows Explorer.
For more information about linking files, also see Using wiki syntax.

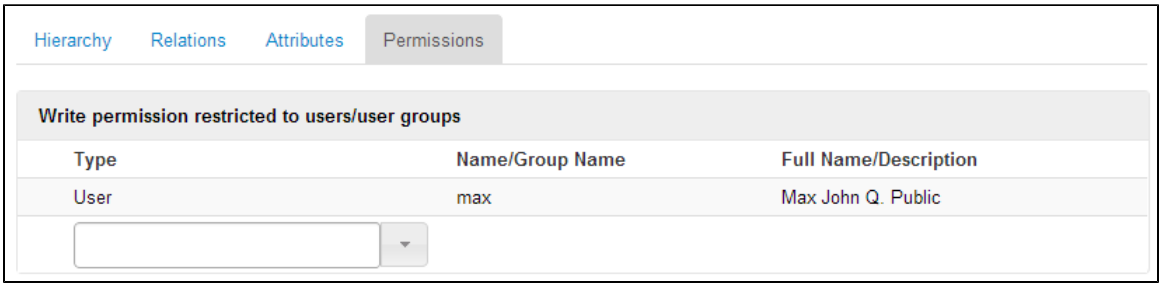


Adding a link to the Description field of a Building Block

Some browsers may allow both, gathering the complete path to a file using javascript and opening pseudo-urls to files directly by clicking onto them. Some other browsers may allow only one of them.

Permissions for individual building blocks

Every Building Block contains a **Permissions** tab, which can be used to assign permissions for particular instances of the block. The users and user groups listed on the **Permissions** tab are the only ones authorised to modify the element. If there are no permissions entered, write permissions are determined solely by the user's own assigned role (see 'Object related permissions').

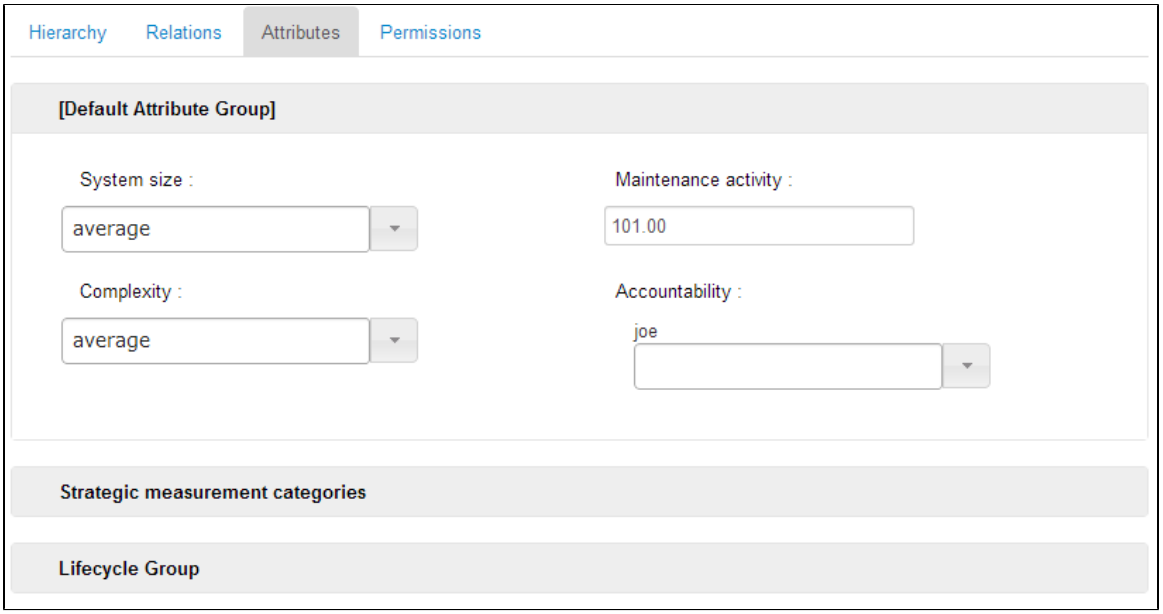


Assigning explicit Building Block permissions

superuser permissions
A superuser, most notably the user named "system" in the demo instances, does have write access to an object, even if the access is restricted via object related permissions.
The corresponding role in iTURM is *iteraplan_Supervisor*.

Assigning attribute values for individual Building Blocks

You can assign attribute values to the Building Block you are currently working on, by opening the **Attributes** tab and entering the values there. The procedure for assigning an attribute value to a Building Block is equal for all types of Building Blocks. These steps are described in section Attribute Groups and Attributes.



Attributes Tab in Edit Mode

Additional tabs for core Building Blocks

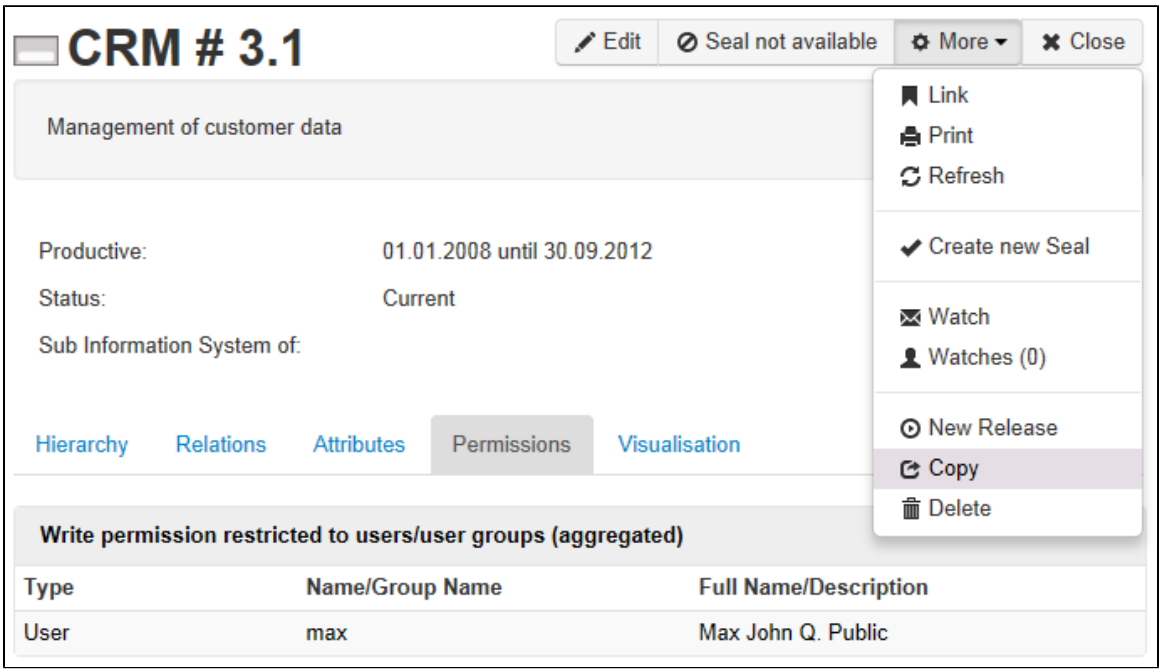
Additional tabs are provided for the core Building Blocks, which deliver greater structural clarity to the large volume of properties and relations you can define for these core blocks. These tabs are described separately for the related blocks.

Copying a Building Block

To copy a Building Block click on **More -> Copy** button in the Transaction Bar. In the new Building Block the following will be copied:

- superordinate Building Block
- Relations
- Attributes
- Permissions

Furthermore before saving the new copy of the Building Block you also can remove some of the copied elements or add new.

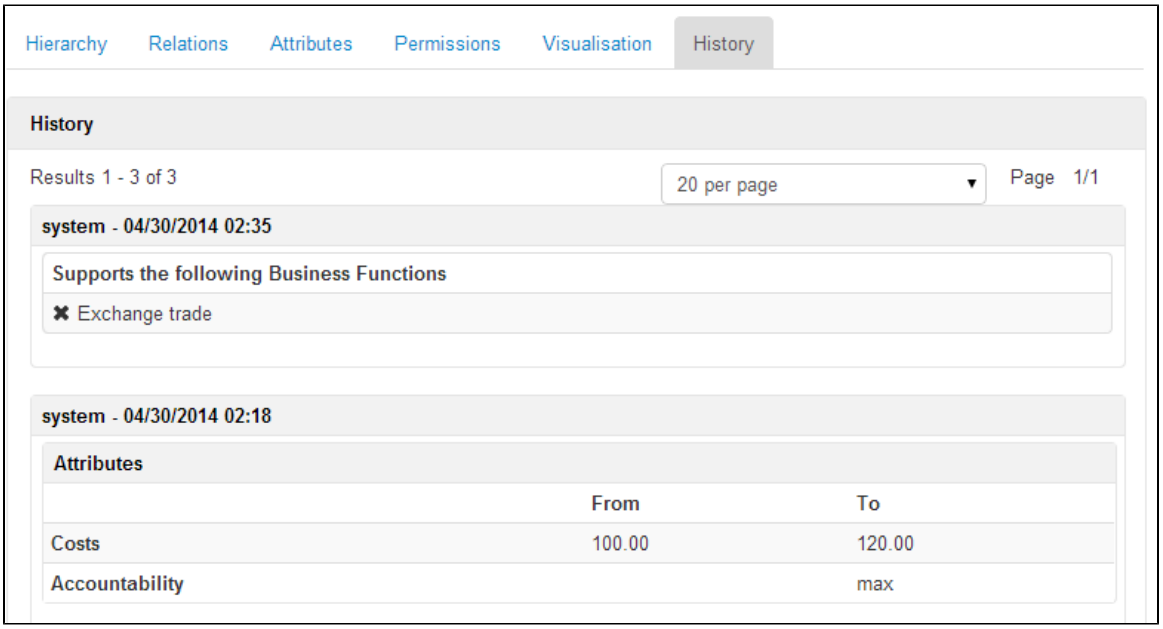


Transaction Bar of a Building Block

History

Every building block has a **History** tab. Inside this tab, a list of local changes (changes to this particular building block) are shown.

History is only saved if history.enabled is set to true in iteraplan.properties



Scope of History Entries

The following information is tracked and displayed in the History tab:

- At what time changes were made, and by whom
- Changes in name, description, and other core attributes
- Changes in attributes
- Changes in hierarchy (superordinate and subordinate elements)
- Changes in direct, binary relations

The following is **not** currently shown:

- Changes in attributes of associations (relationship types) with attributes
- Changes in relations via relationship types
- Changes in object-specific permissions

If a user opens a building block for edit, does not edit a property or a relationship, and saves the building block, this action is recorded as an edit, with user and timestamp. This history entry shows no edit, and correctly so. A user can avoid this entries if he cancels the edit instead of saving.

Editing Building Blocks

This section describes how to edit the different tabs of the Building Blocks.

You can access each tab individually, by pressing any key from **1** (leftmost) to **5** (rightmost).

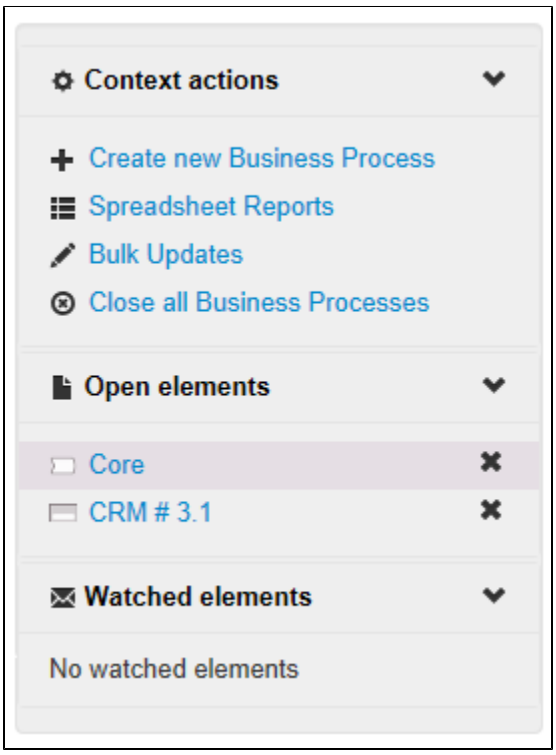
User Transactions

iteraplan works with a transaction concept that ensures maximum ease of use in editing landscape data. Each of the editing pages – these are the pages you open from the *Base Data* and *Core Building Blocks* menus – and the pages for *User Management, Roles and Permissions* and for *Attributes* and *Attribute Groups* (see Home Page and Menus) are assigned to a toolbar by which you can toggle between *View* mode and *Edit* mode and thus also modify the status of the user transaction (see screenshots below).

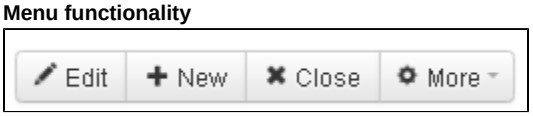
New since release 2.5
You can now open several instances of your building blocks at the same time. They will be displayed in the context menu under Open Elements, each one below the other. Your transaction won't be interrupted when you open a new element. For closing your element just click the **X** next to the corresponding name in the navigation bar. You can close all elements of a Building Block category by clicking the

close all within the context actions.

New since release 2.5
Each transaction button can be directly accessed by a shortcut key. When hovering over a transaction button, in some cases a tooltip will appear which indicates the shortcut keys for the respective button. To give an example, the shortcut key for edit is **Shift + E**. A comprehensive list of all shortcut keys for the transaction bar can be found in the Table below. There are also a couple of other buttons outside the transaction bar which have shortcut keys. Consulting their tooltips will always provide you the right key combination.



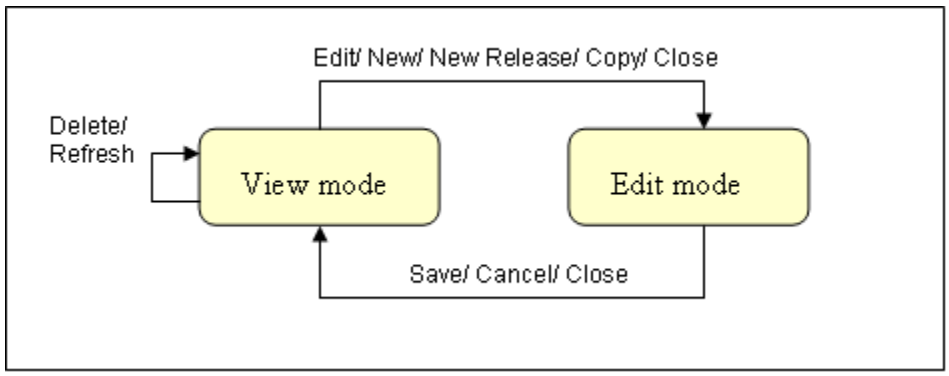
Menu functionality



Toolbar for directing a user transaction (View mode)

Toolbar in Edit mode

Short explanation of the interrelation of the two modes:



Explanation of interrelation of View and Edit mode

The following Table summarises the buttons and their functionality:

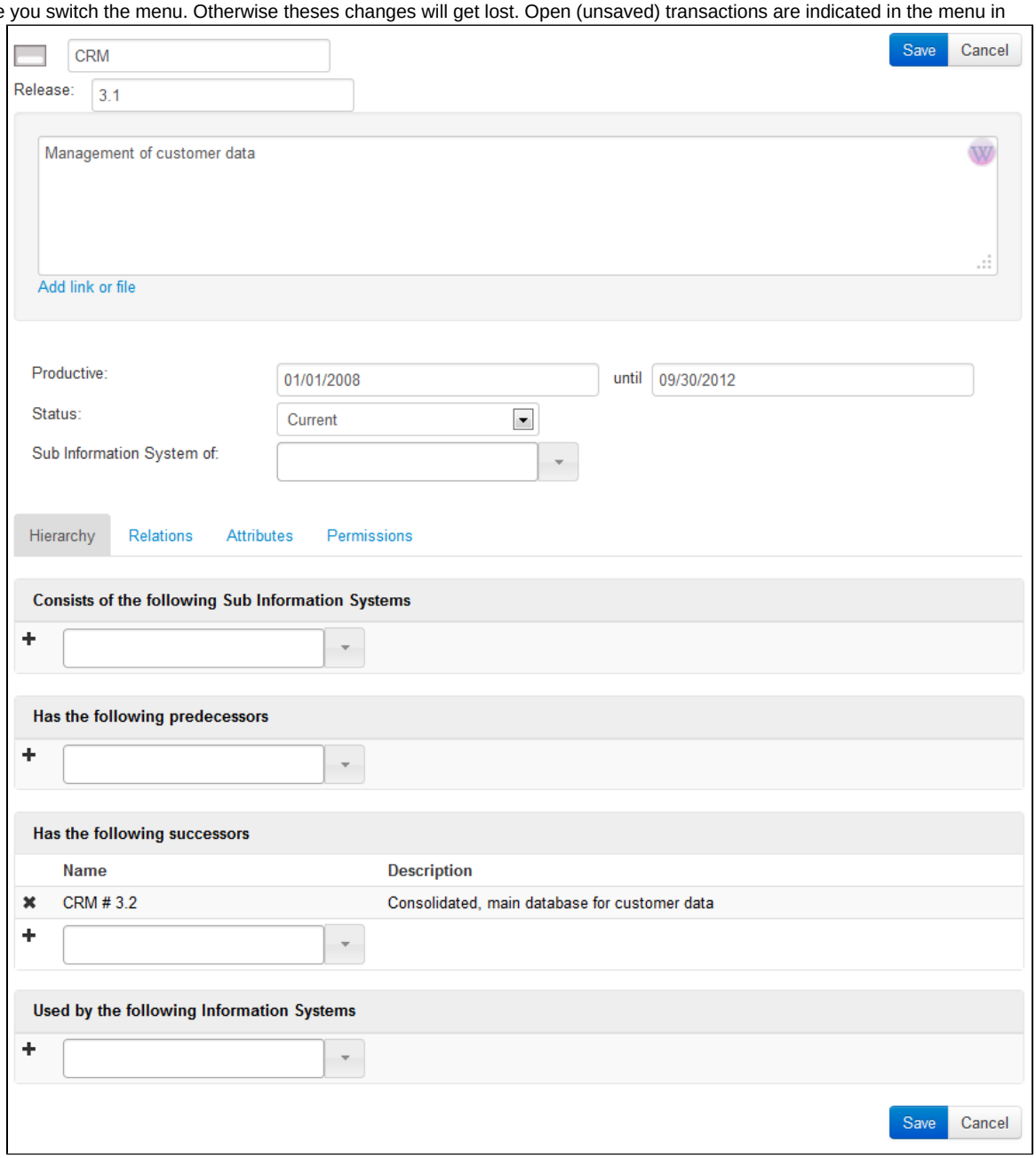
> Please note that you are only shown the toolbar for a landscape Building Block if you actually have the access privileges that permit you to edit Building Blocks of the type in question.

Table: Functions for directing user transactions

Button	Functionality	Visible in...
Edit Shift + E	Switches to Edit mode, permitting you to modify all properties, attributes and relationships of the *selected* instance of the Building Block.	View mode
Delete Shift + D	Depending what page you are working on, this function deletes the *selected* instance of the Building Block, Attribute or Attribute group, or the selected User, User Group or Role.	View mode
New Shift + N	Creates a new instance of a Building Block. Initially you can enter only default properties for the new element – relationships, for instance, but no attributes. You can add the remaining items afterwards, using Edit mode.	View mode
New Release Shift + X	Creates a new release on the basis of the *sel ected* Building Block. You can enter default properties for the new release, and also have the release inherit the relationships and attributes from the predecessor release. You can add the remaining attributes afterwards, using Edit mode.	View mode Visible only on pages for **Information Systems** and **Technical Components**.
Copy Shift + C	Unlike the **New Release** button, which creates a new release on the basis of an existing one, this function creates a completely new Building Block. The selected block serves merely as the template and has no further relation to the new one.	View mode Visible only on pages for **Information Systems** and **Technical Components**.
Cancel Shift + A	Discards the changes you have entered and switches to View mode.	Edit mode
Save Shift + S	Saves the data you have entered and switches to View mode.	Edit mode
Refresh Shift + R	Refreshs the data of the current Building Block.	View mode
Close Shift + W	Close the dialog of the current element without saving changes.	View mode

A user transaction encapsulates a set of changes which are all saved permanently to the database when you execute the **Save** command, or which are all discarded if you click **Cancel**. A single user transaction encompasses an entire menu command. In other words, if you make

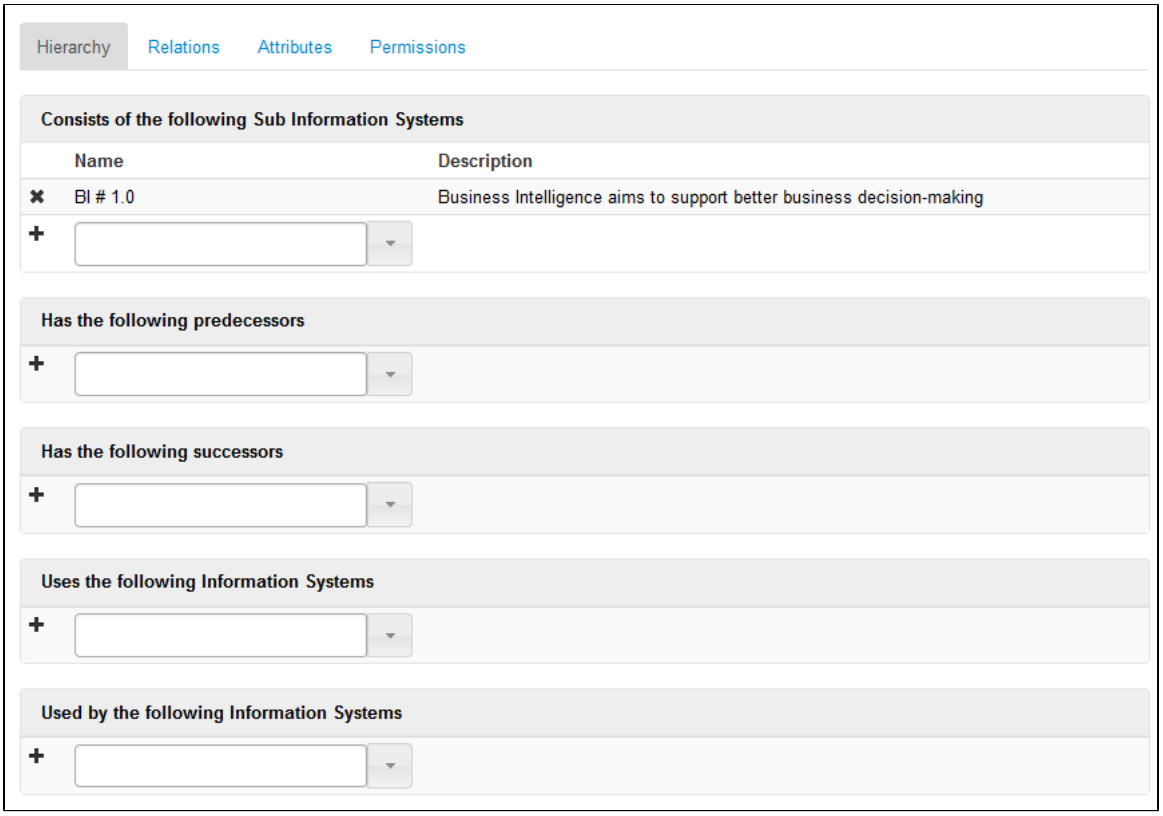
changes in several subsections of the same menu (by altering data on multiple tabs), these changes will all be either saved or discarded together. While you are working on a building block in Edit mode, i.e. within a user transaction, it is possible to change to a different menu and return to the original one afterwards to complete the transaction you originally started. If some of the data fields you changed are highlighted in blue, you must save them before you switch the menu. Otherwise theses changes will get lost. Open (unsaved) transactions are indicated in the menu in magenta as well.



Unsaved changes of an open transaction

Hierarchy

Use the **Hierarchy** tab to specify subordinated and superordinated Building Blocks. For information systems, also predecessors and successors, as well as use-relationships can be defined.



Editing building block hierarchy

You can specify superordinated Building Blocks in the same way as other relations, mentioned above.

The drop-down list shows the so called *hierarchical name* of all available elements, using the following format:

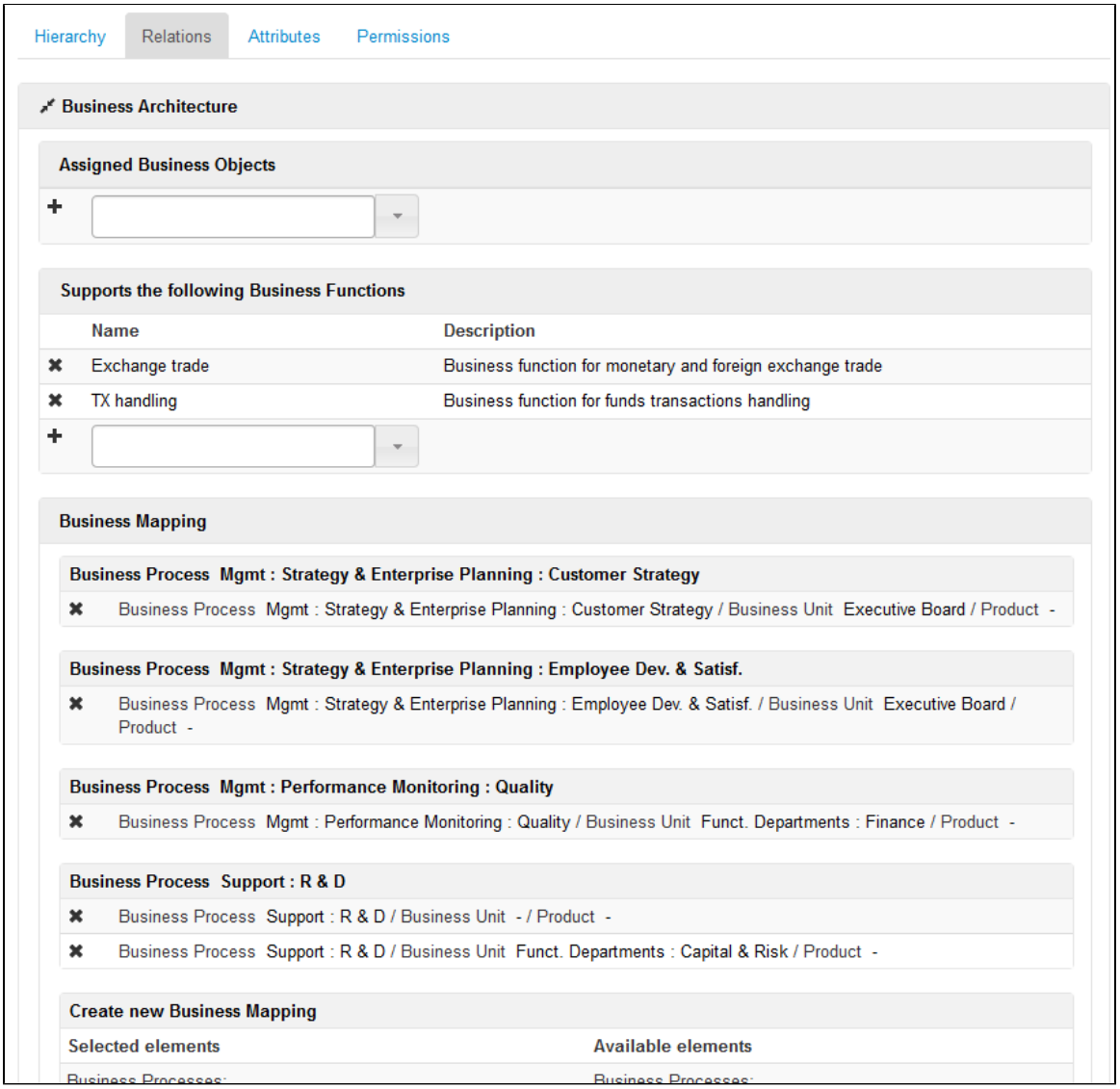
[<...> : <name of the superordinate building block> : <name of the building block >]

The order of subordinate Elements determines how elements will show up in diagrams.

Relations

Use the **Relations** tab to assign relations between particular instances of Building Blocks.

To define a relation one selects a Building Block from the drop-down list and adds it by clicking the **plus**. You can delete the added elements by clicking the corresponding **cross**.



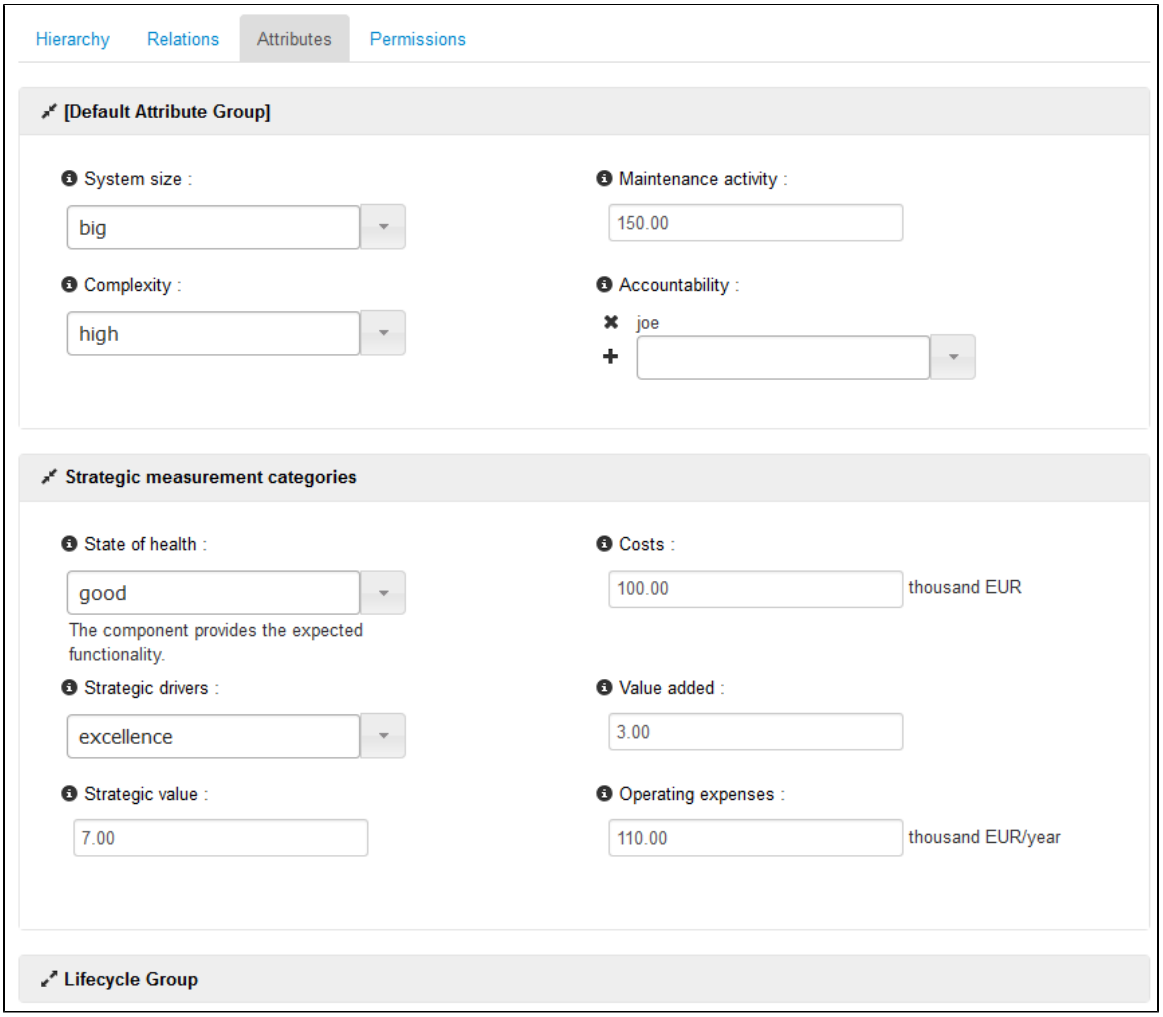
Editing building block relationships

Attributes

You can define various properties for Building Blocks. Each instance of a Building Block must have at least the properties **Name** and **Description**. If you wish to specify additional properties, iteraplan enables you to do this by defining attributes. Enterprise-specific attributes can be created by a user with the appropriate role and can be assigned to any type of Building Block (e.g. to Information Systems). To give a practical example:

You could define an Attribute called *Costs* for Information Systems; A specific value for the *Costs* attribute can then be assigned for each instance of an Information System. For more information about Attributes, please refer to Section Building Block Attributes.

You are able to assign self defined **Attributes** to your Building Blocks at the **Attributes** tab.

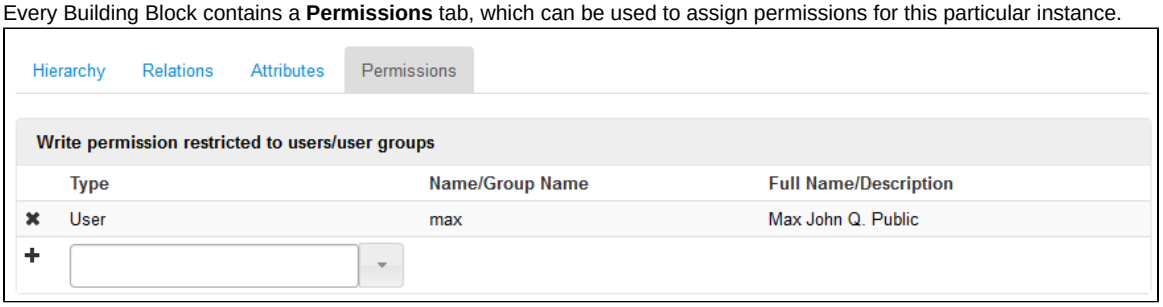


Editing building block attributes

Enumeration Attributes are edited in the same way as other relations, mentioned above. For other fields simply add the text, numbers or dates as required.

Permissions

Every Building Block contains a **Permissions** tab, which can be used to assign permissions for this particular instance.



Editing building block permissions

Permission – which are relations to Users or User Groups – are edited in the same way as other Relations .

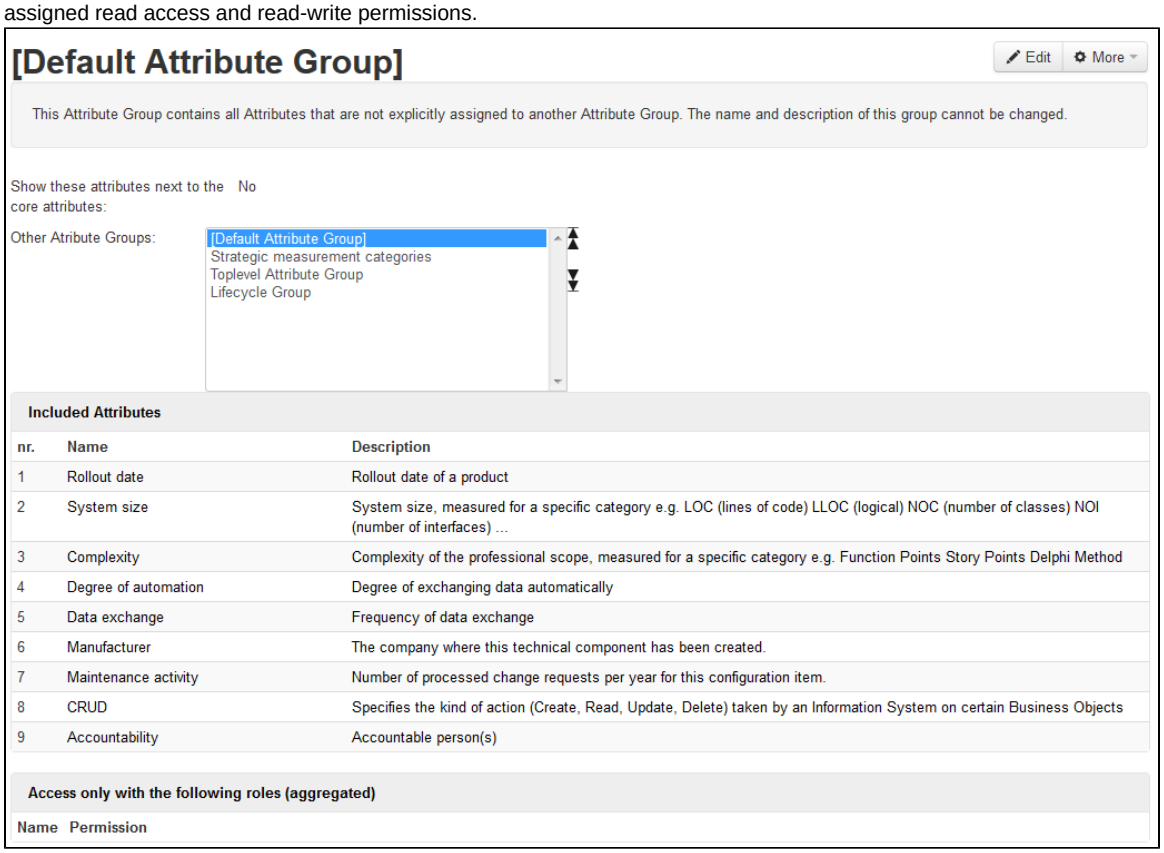
Attribute Groups and Attributes

Attributes enable you to enrich the information associated with Building Blocks flexibly and simply by adding enterprise-specific features. You can configure attributes for any Building Block in iteraplan, enabling you to react swiftly to any changes in modelling requirements.

Attribute Groups

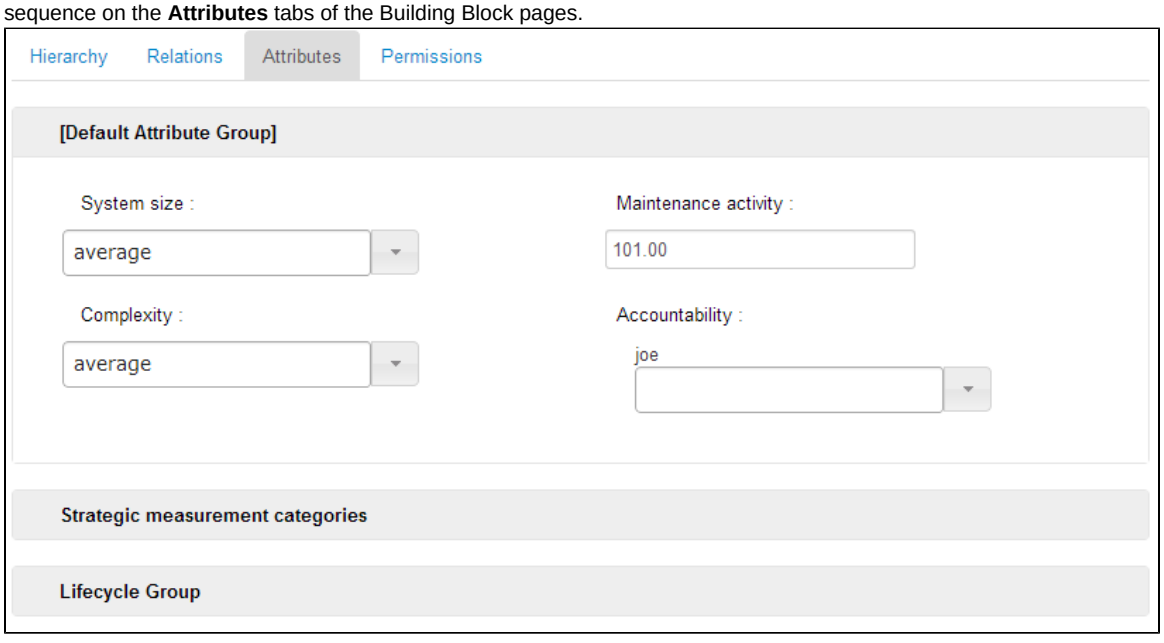
Attributes in iteraplan are always organised into groups. An Attribute Group has a name, description and list of the Attributes it is assigned. The *D efault attribute group*, as shown in the screenshot below, comprises different attributes such as *System size, Complexity, ...*. Each group can be

assigned read access and read-write permissions.



Overview and sequence of Attribute Groups

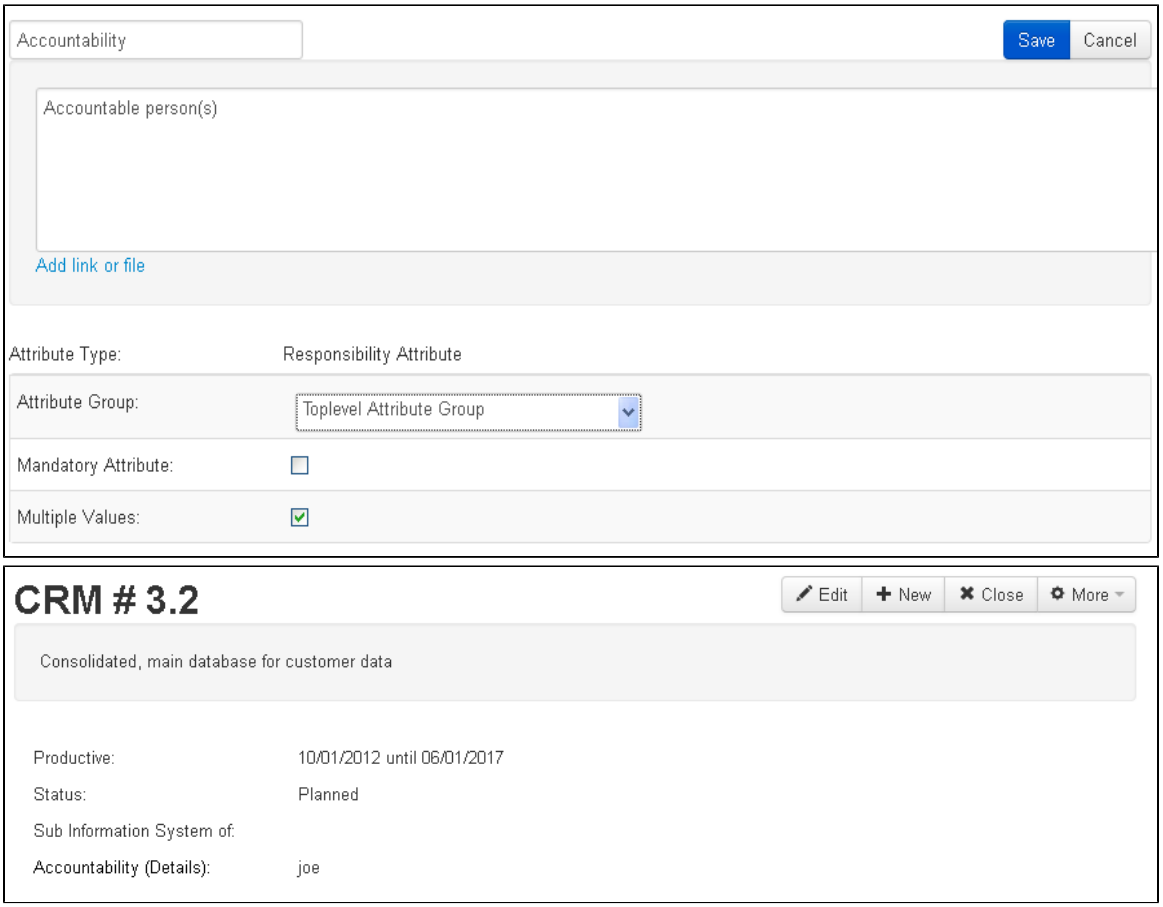
You can also define the order in which groups are displayed, and the order of attributes within the groups. To change the order of groups, move list entries up and down with the arrows to the right of the list. The Attribute Groups, and the Attributes they contain, are presented in this sequence on the **Attributes** tabs of the Building Block pages.



Page for editing Attribute Groups

Each Attribute belongs to exactly one Attribute Group. If no particular group is specified, the Attribute is assigned to the *Default Attribute Group*. If you delete an Attribute Group, the Attributes it contains are automatically assigned to the default group.

Read and write permissions for Attribute Groups can also be restricted to particular roles. If there are no permissions assigned for an Attribute Group, this means access is unrestricted. By setting the *Toplevel Attribute Group* option to enabled, the Attributes within this group are displayed at the top of the Building Block instead of within the attribute tab.



Attribute is shown at the top of the Building Block instead of within the attribute tab

Read (view) access is assigned to the roles *CEO/CIO/Strategist, Enterprise architect - information systems, Enterprise architect – processes, IT Architect and Person accountable for the information system.* Users can only modify attribute values in this group if they have the roles *Enterprise architect- information systems* or *Enterprise architect – processes*. Users who are not assigned any of these roles will not be able to view the Attribute Group on the Building Block pages and are therefore unable to modify the Attributes in the group.

Building Block Attributes

iteraplan provides five different types of Attributes:

- *Enumeration Attribute*: these Attributes have a specific number of possible values defined for them. When assigning an attribute value to a Building Block, users must select one of these predefined values;
- *Text Attribute*: unlike enumeration-type attributes, Text Attributes permit users to enter text strings of their choice as attribute values. These texts are then assigned as attribute values to instances of Building Blocks;
- *Numeric Attribute*: Numeric Attributes require users to enter numbers as attribute values. Up to two places following the decimal point are permitted. The numeric value can be entered when the attribute value is assigned to a particular Building Block;
- *Date Attribute:* a Date Attribute requires users to enter a date as the attribute value (format depends on the currently active user interface language). The date can be entered when the attribute value is assigned to a particular Building Block;
- *Accountability Attribute*: these Attributes require the entry of a user or user group (see Users, Roles and Permissions) as attribute value. The user or group can be entered when the attribute value is assigned to a particular Building Block.

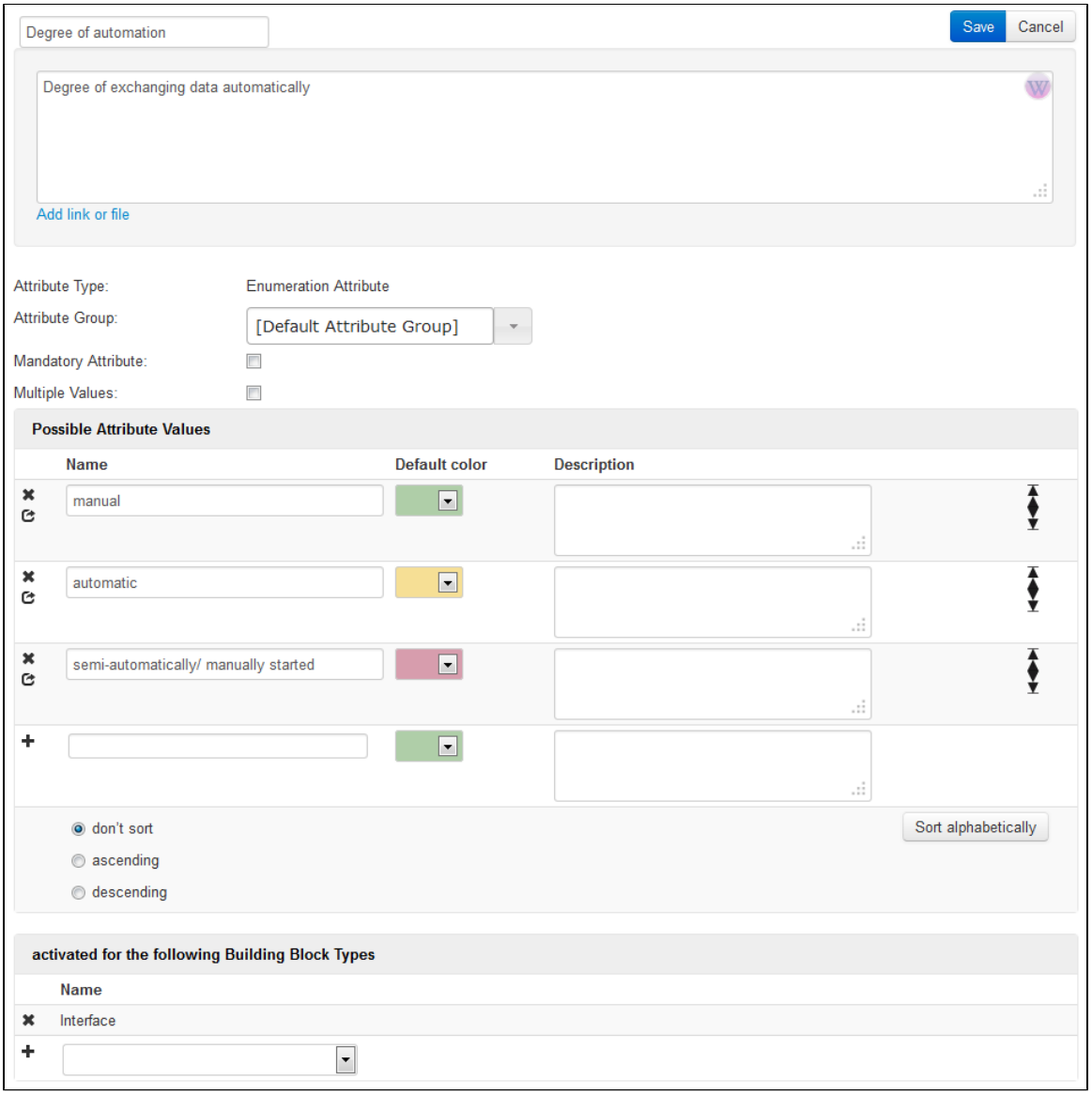
The following properties are the same for all types of attribute:

- *Name: The usage of "." and "@" in attribute names is not recommended, because currently (release 3.3) it may cause problems when such attributes are used in iteraQl, for example in the Query console or in the filter for the partial data export. Furthermore attributes must not have the same name as a Building Block Type in any locale, as that leads to issues with, for example, the Import/Export functionality.*
- *Description;*

- *Attribute Type*: enumeration, text, numeric, date or accountability. You cannot change this property once the attribute has been created;
- *Attribute Group*: one of the groups already defined. The affiliation of an attribute to a particular group can be entered either on the form for editing attributes or the form for editing attribute groups;
- *Mandatory Attribute*: this attribute serves as a notice field. If the user omits an entry from a mandatory field when defining or modifying Building Blocks, no error message is issued. However, a Consistency Check can be performed to flag this error (see Consistency Checks).

Certain of the attribute types have additional properties:

- Enumeration Attributes: a predefined set of values have to be defined for attributes of this type, since users must assign one of these values when setting attributes for a particular instance of a Building Block. You can also enable an attribute for multiple-value selection. Users can then select multiple attribute values from the list. If this option is not set, only one value can be selected from the set of predefined options. For Enumeration Attributes it is also possible to set a standard colour to be loaded on default for representation in diagrams. Enumeration Attributes with just two values (and for which no multiple choice is permitted) can be used to represent either-or choices;
- Text Attribute: herein you have the option of selecting *multi-line attribute values*. A multi-line field instead of the single-line field is then presented for users to enter their text;
- Numeric attributes: it is possible to specify a lower and upper limit for attribute values, and a unit. Attribute values are still accepted even if they are outside the permitted range, but are shown in red font in the view mode of the Building Block. The use of colour enables iteraplan to indicate a possible inconsistency in attribute values without getting in users' way unnecessarily. Out-of-range attribute values can also be revealed by a Consistency Check (see Consistency Checks). Limit values can be defined for each Numeric Attribute. These are relevant in diagram reports, in which colours can be assigned to defined ranges. With the default values, the ranges are selected such that the instances of the attribute are evenly distributed. However these limits are user-definable (see Defining Ranges for Numeric Attributes);
- Date Attribute: no additional specific properties;
- Accountability Attribute: you have to define a set of values (users or user groups) for attributes of this type, since users must select from a predefined set when they assign an attribute value to an instance of a Building Block. You can also enable an attribute for multiple-value selection. For Accountability Attributes it is also possible to set a standard colour to be loaded on default for representation in diagrams. Users can then select multiple attribute values from the list. If this option is not set, only one value can be selected from the set of predefined options.



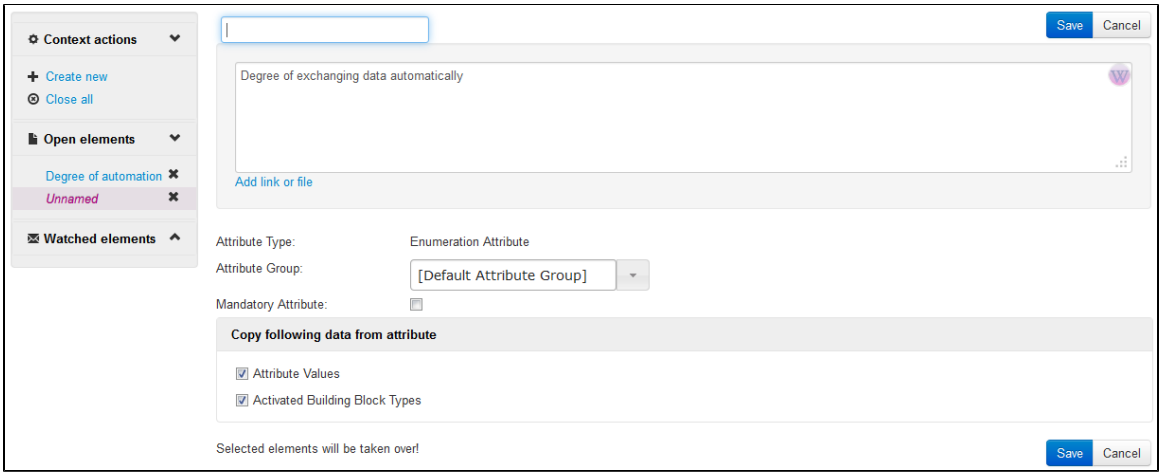
Page for editing Enumeration Attributes

Before an attribute value can be assigned to a specific instance of a Building Block, the Attribute must be enabled for the type of block in question. You make this assignment with the list of activated Building Block Types. For example, the attribute "Data exchange" in the figure above is activated only for interfaces. Accordingly, values of this attribute can be defined only for Interfaces, but not for other types of Building Blocks.

Furthermore, attributes can be activated for selected associations between building block types, like relations between Information Systems and Business Objects or between Technical Components and Infrastructure Elements. Look here for more details.

Copying Attributes

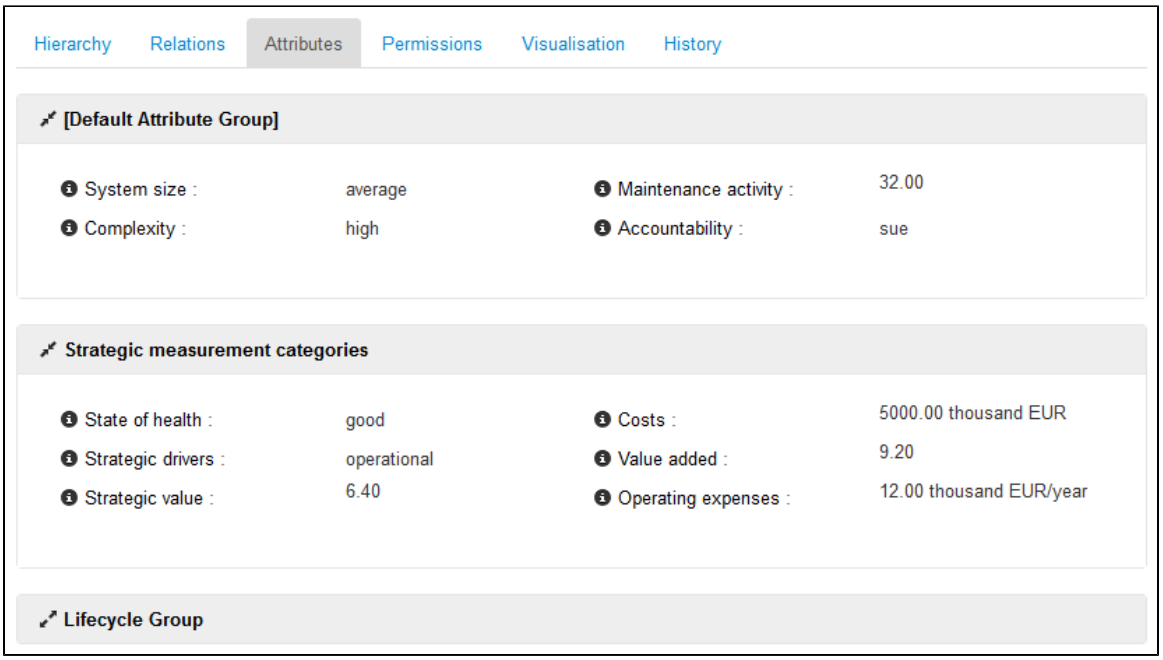
Besides creating new attributes, it is also possible to create a copy of an existing Attribute by clicking on the **Copy Attribute** button. The copied Attribute has the same property values like the existing one, except for the Attribute Name, which must be assigned with a new unique value. Further more, you are also able to adopt existing attribute values and activated Building Block Types of the existing attribute by checking the related checkboxes in the screen.



Copying Attribute page

Assigning attribute values to building blocks

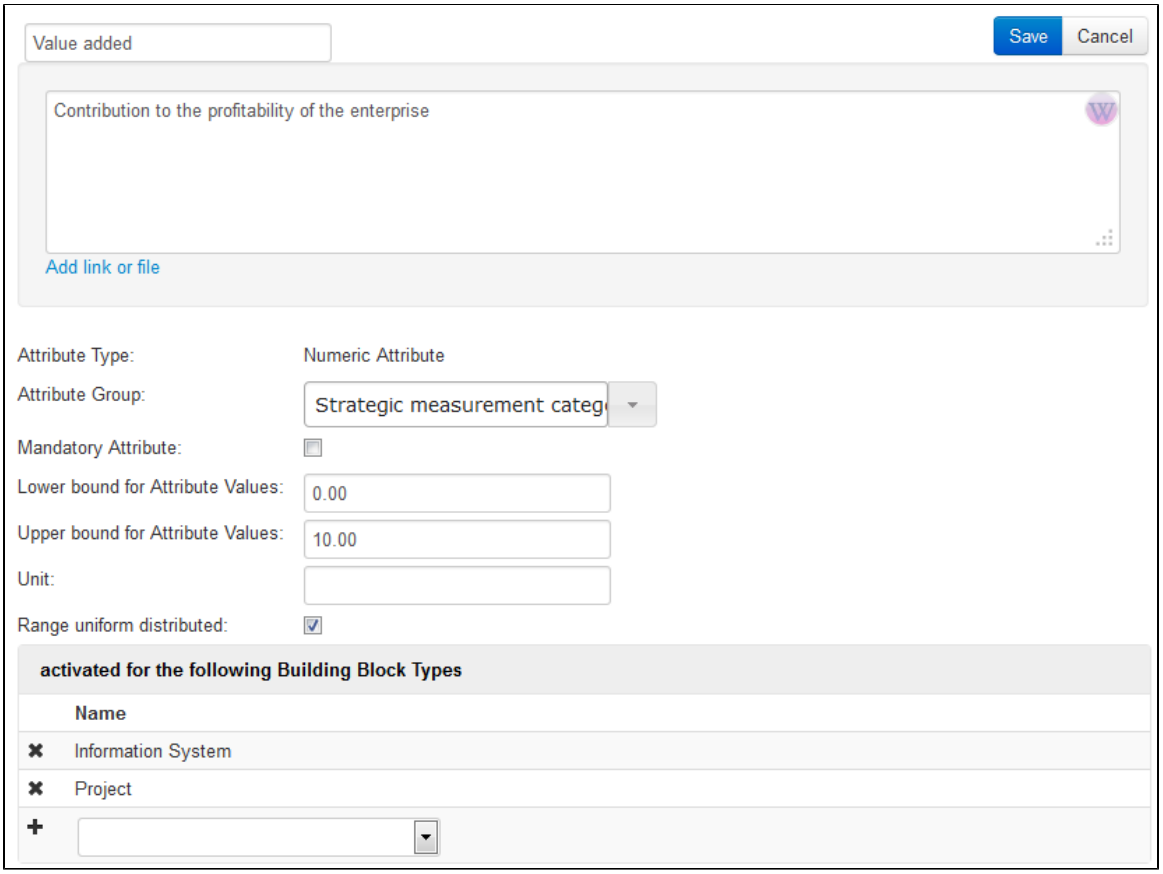
As stated above, the procedure for assigning attribute values to a Building Block is the same for every type of block. When you are modifying Building Blocks, the Edit mode pages show you a list of all the Attributes organised into groups. Enumeration and Accountability Attributes provide drop-down lists from which you choose the value you wish to assign. You can assign multiple values if the attribute in question permits multiple choice: simply click the green plus '+' symbol to display the drop-down list again. Text and Numeric Attributes permit you to enter exactly one value. Date Attributes can be keyed in (in the format dd.mm.yyyy) or selected from the calendar.



Assigning attribute values to Building Blocks

Defining Ranges for Numeric Attributes

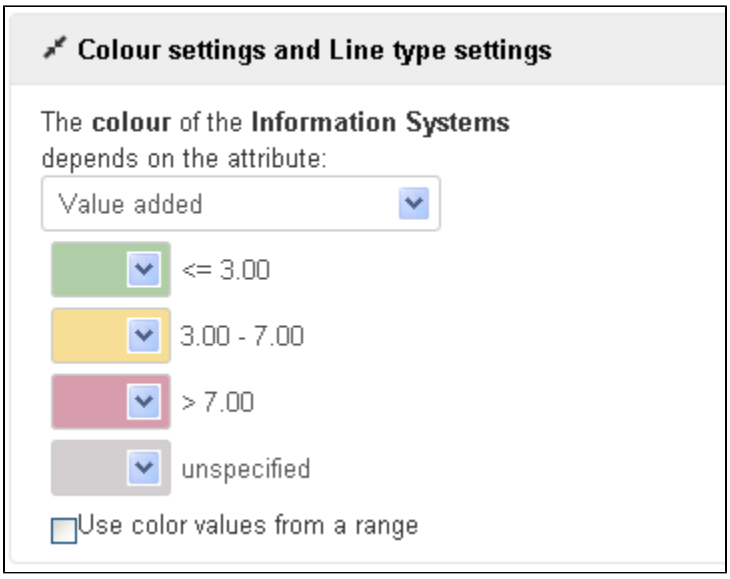
For better visual clarity in Diagram Reports with Numeric Attributes, the Attributes are grouped into ranges. If you check the "Range uniform distributed" box when editing the Attribute, iteraplan automatically calculates the ranges such that the values are distributed uniformly between them. If you leave the box unchecked, you can define the ranges manually.



Page for editing Numeric Attributes

Ranges are defined as follows: the smallest value you enter, combined with minus infinity, serves as the lowest range. The largest value, with plus infinity, is the highest range. Other values between the two form ranges with the notation as shown below. For example, the values 10 and 50 would result in the following ranges:
(-? - 10 <=) (> 10 – 50 <=) (> 50 - +?)

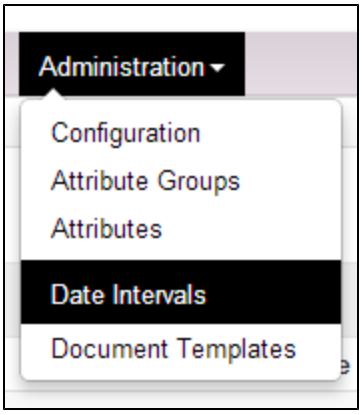
The ranges are used for a number of dimensions, in colour definitions for example.



Color definition for a Numeric Attribute with defined ranges

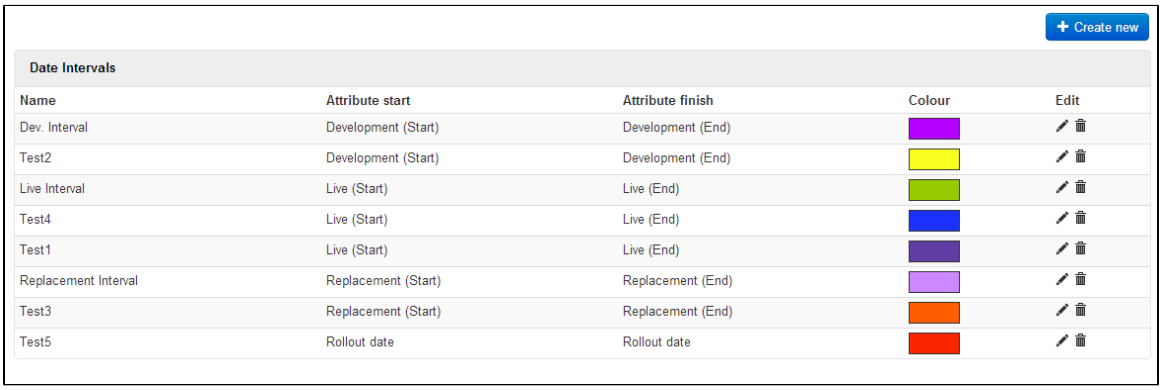
Date Intervals

In the Administration Menu we have added an element to manage Date Intervals.



On the Date Intervals main page you have a list of the existing Date Intervals you can quickly view, edit, or delete, through direct links on each list element.

You can also create new Date Intervals with a simple form page (used also for editing them).

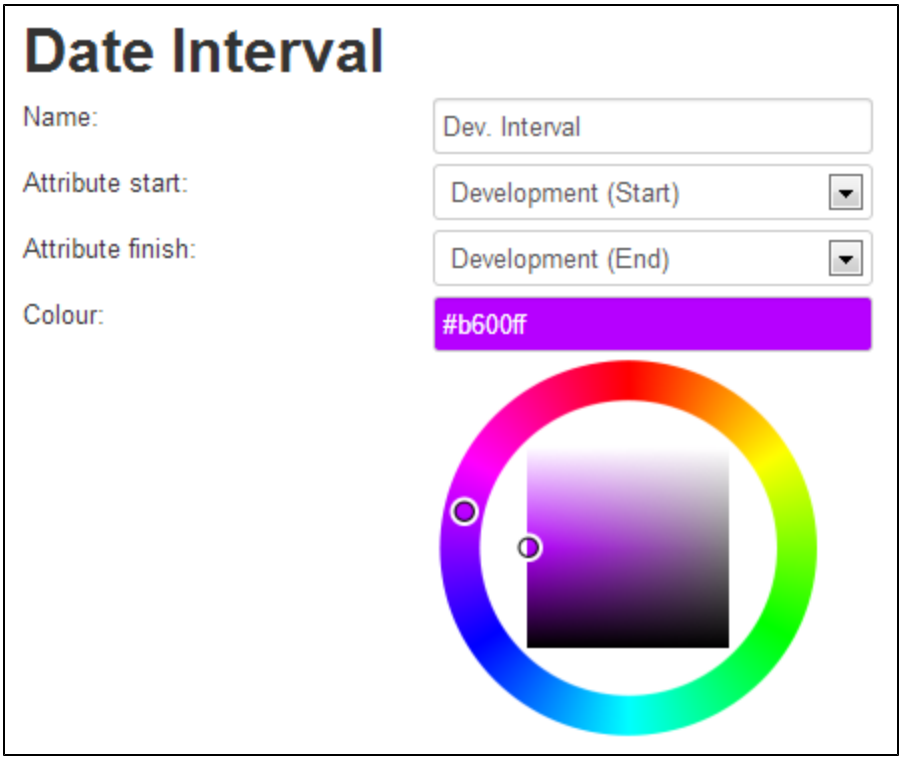


The access to these functionalities is granted to an user with the permissions to manage Attribute Types.

Once defined these Date Intervals, they can be later used and rendered in a Masterplan Diagram.

Before creating a new Date Interval, make sure you have a couple of Date Attributes you can assign to it (Administration -> Attribute Types). Those Dates will represent the logical start and end of the Date Interval.

In the create/edit form, you will see that there is a color to choose. This color will be later used to render the Date Interval in a Masterplan Diagram export.



Later, when you want to use this Date Interval in a Masterplan Diagram, you will have to add some Building Block Type associations to the Date Attributes used in the Date Interval.

To see how Date intervals are used in a Masterplan Diagram, check out the Masterplan Diagram page.

Bulk Updates

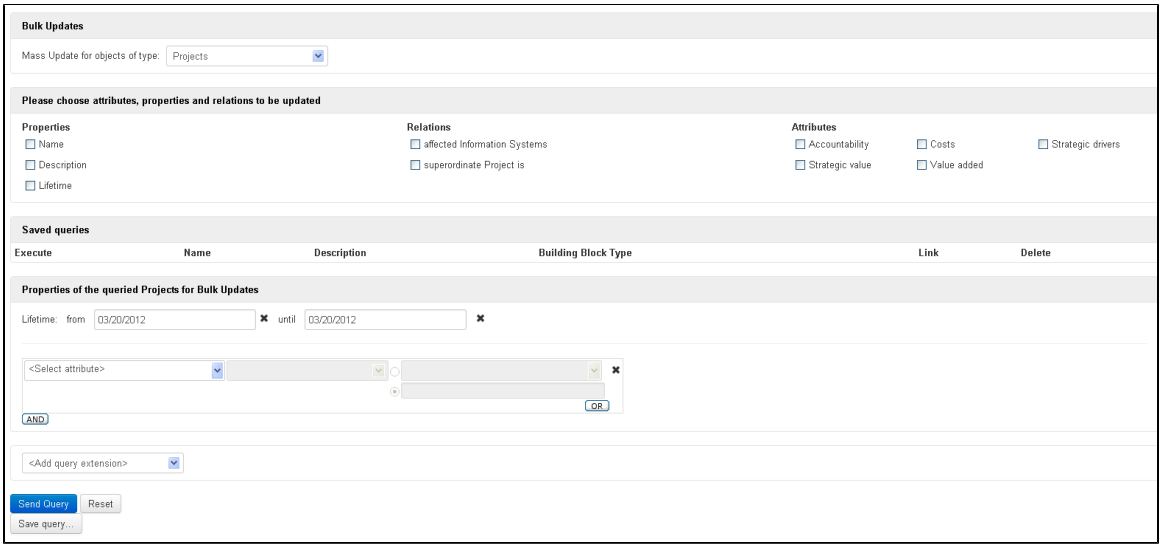
iteraplan provides two different ways of dealing with a big amount of Building Blocks at the same time. Next to a Bulk Delete (see section 4.17.2) there is a Bulk Update function with which you can edit an entire set of Building Block data.

Bulk Update

A Bulk Update permits you to modify properties, relationships and attribute value assignments for several Building Blocks of the same type simultaneously, speeding and simplifying maintenance for users.

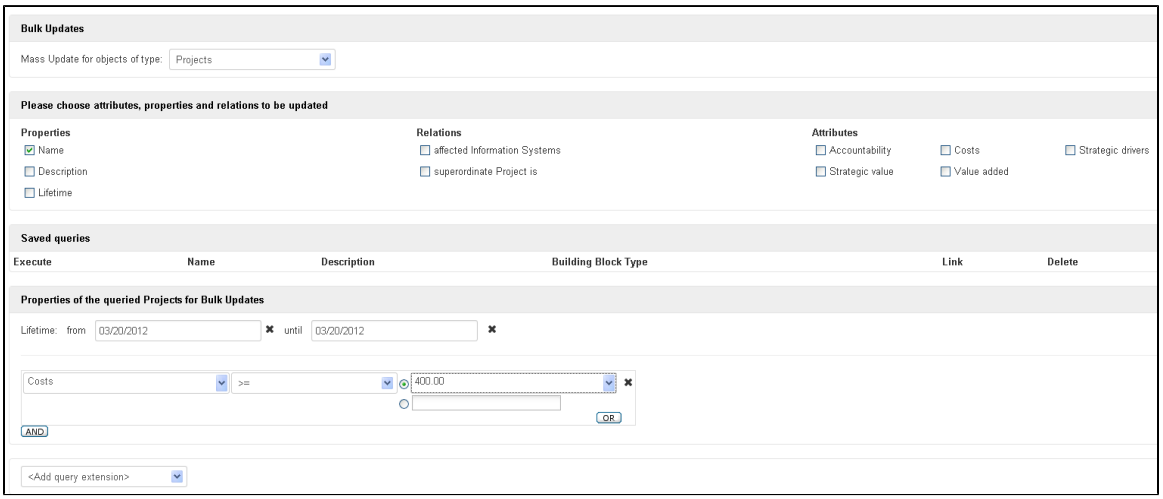
What to do:

- Select the type of Building Block (e.g. Information Systems) for which you wish to perform the Bulk Update, top section of form);
- Specify the properties, relationships and attributes you wish to update, **Please choose attributes, properties and relations to be updated**);
- If you wish, restrict the scope by specifying particular properties to be matched, e.g. **Properties of the queried information systems for Bulk Updates**, (see Formulating queries);
- Besides defining a specific query, it is also possible to load and edit a former query from the saved queries list. This list contains predefined queries you have saved before in a Spreadsheet Report (see Saving and loading spreadsheet reports for more details).
- Click **Send Query**;



Start page for bulk update

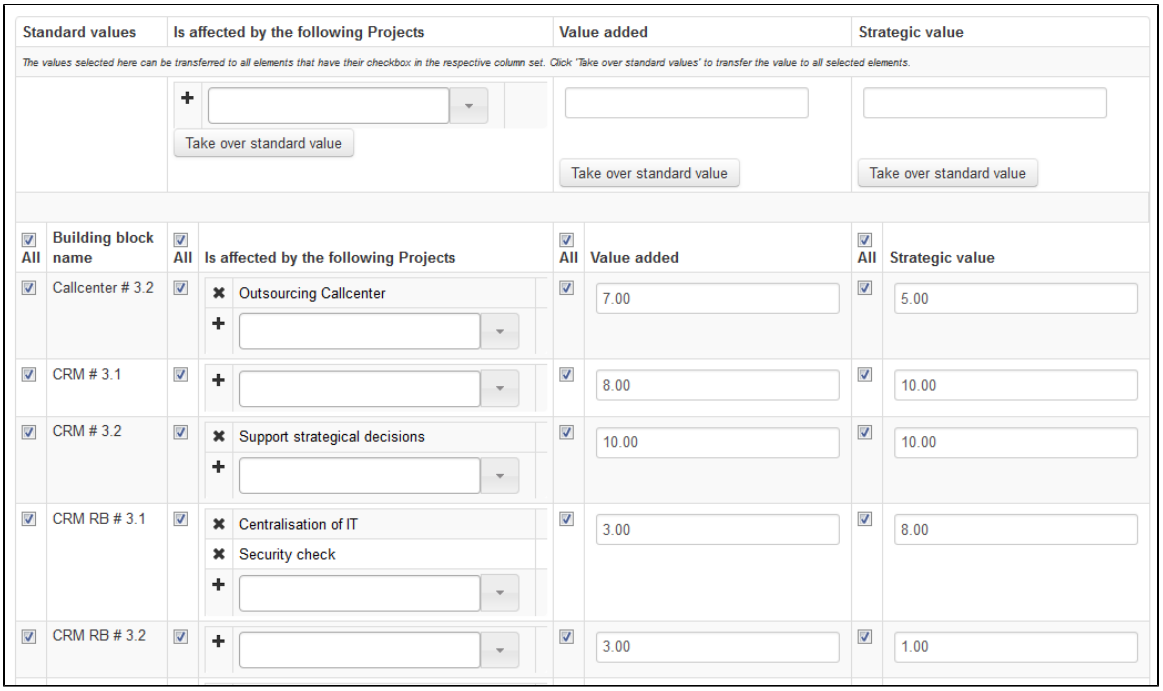
- If necessary, edit the results list (by setting or removing check marks next to the entries in the list; only the elements whose check box is activated will be transferred to the Bulk Update page);
- Click **Define Bulk Update for the selected elements** to confirm the selection of elements for the operation.



Preparing for the bulk update - refining the results.

There are two ways to define attribute values and associations for Building Blocks:

1. Select or define default values beneath **Standard values**. The values selected are confirmed when you click **Take over standard value** and are assigned to each element for which the check box **S** is activated.
2. Assign or define attribute values and associations individually for each Building Block in the list.



Page for Bulk Updates

The **Run bulk update** button (see screenshot above) executes the changes for every building block that has the check box **M** activated. Following the operation, the message "Update was successful" appears alongside each building block which has been updated correctly. If the update failed for any of the blocks, they are marked in red. This can happen, for instance, when another user altered the block in the database while the parameters for the bulk update were being defined. The bulk update has to be repeated for these building blocks.

Bear in mind that you cannot modify a Building Block if it inherits its attribute value from its superordinate block.

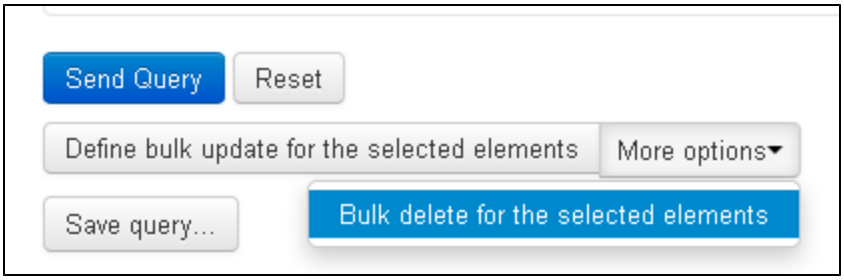
Bulk Delete

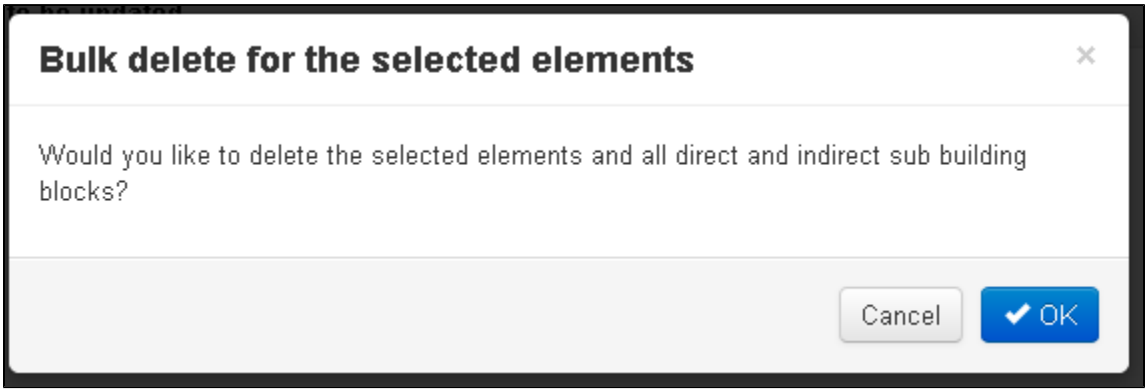
A Bulk Delete permits you to delete a selected amount of Building Blocks. Usually this functionality is only needed if a data import was wrong and you have to delete many elements all at once.

The procedure for this is very similar to the Bulk Update mechanism (see Bulk Update):

- Select the type of Building Block (e.g. Information Systems) for which you wish to perform the Bulk Delete, top section of form);
- If you wish, restrict the scope by specifying particular properties to be matched, e.g. **Properties of the queried Information Systems for Bulk Updates** (see Formulating queries);
- Besides defining a specific query, it is also possible to load and edit a former query from the saved queries list. This list contains predefined queries you have saved before in a Spreadsheet Report (see Saving and loading Spreadsheet Reports for more details).
- Click **Send Query**;
- If necessary, edit the results list (by setting or removing check marks next to the entries in the list; see bulk update); only the elements whose check box is activated will be deleted.
- You can find the button **Bulk Delete for the selected elements** under **More options**.
- Click **Bulk Delete for the selected elements** (see screenshot below). You will then be asked again to confirm the deletion of selected elements. By clicking **Okay** all chosen elements will be deleted permanently from the database.

Be aware of that at this point in iteraplan there is no way to restore the data again.





Confirmation of the Bulk Delete

Building Blocks in iteraplan

Business Domains

Business Domains serve to group a set of Building Blocks from the business landscape (Business Functions, Business Processes, Business Objects, Business Units and Products). For example, you can organise a Product and several Business Functions for this Product into a Business Domain.

Business Domains have the properties *Name* and *Description* and may also have attributes (if any have been defined and enabled). You can manage the following relationships:

- *Hierarchy - Superordinate Business Domain*: This is an ordinary "is part of"-relationship, as used in other Building Blocks;
- *Hierarchy - Subordinate Business Domains:* This is an ordinary "consists of"-relationship, as used in other Building Blocks;
- *Relations*: allows you to indicate which elements of the business landscape (Business Functions, Business Processes, Business Objects, and Business Units and Products) belong to the domain.

You can modify Business Domains in Edit mode, which you activate by clicking the **Edit** button. After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving.

For more information about copying a Business Domain see Copying a Building Block.

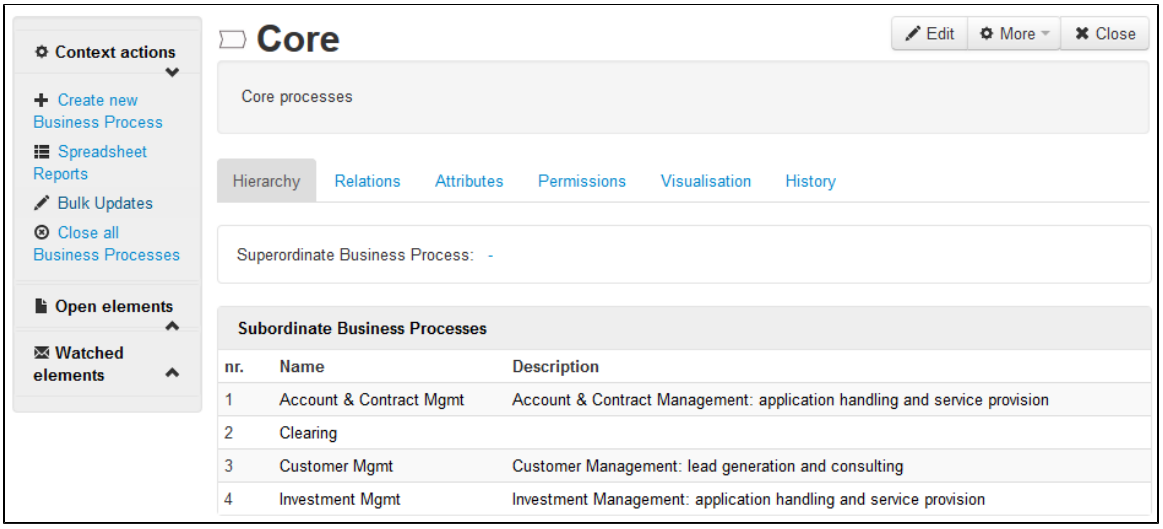
Business Processes

Business Processes have a *Name* and a *Description*, and may also have Attributes (if any have been defined and enabled). You can also specify one or more *subordinate Business Processes* and the sequence of these subordinate processes.

> ⚠ Each Business Process can have multiple subordinate processes, but cannot be assigned to more than one *superordinate Business Process*. Assigning a Business Process as a subordinate to another business process has the effect of deleting an existing assignment to a superordinate process.

You can modify the properties, attributes and relationships of Business Processes in Edit mode, which can be activated by clicking the **Edit** button . After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving. **Deleting** a Business Process has the effect of deleting its entire substructure, i.e. all its sub-business processes.

For more information about copying a Business Process see Copying a Building Block.



Page for editing Business Processes

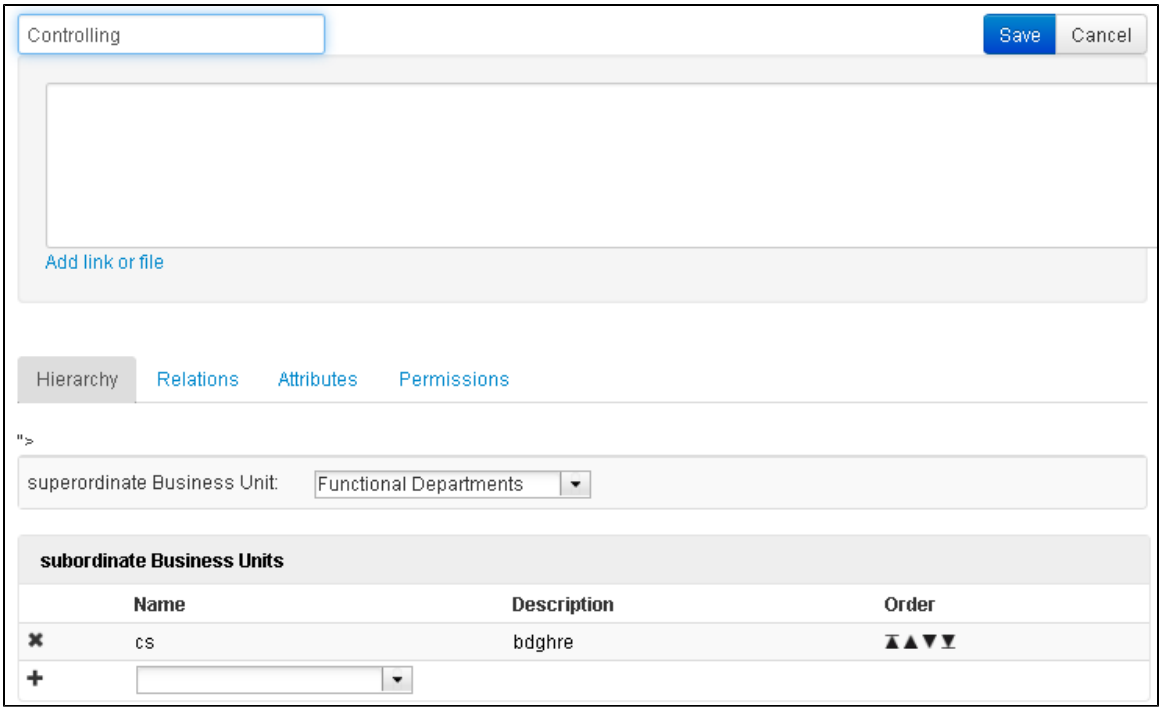
Business Units

Besides the Business Unit's *Name* and *Description*, you can also enter a *superordinate Business Unit* as well as several *subordinate Business Units*. To define a substructure, switch to the **Hierarchy** tab and choose the related super- and subordinate Business Units. The Business Unit's name is then prefaced by the name of its superordinate unit. To create a new relation, switch to the **Relations** tab and choose the related Business Domains.

A virtual business unit ("-") has been introduced to enable the sorting of Business Units at the root level of the hierarchy to be changed. This virtual Business Unit cannot be assigned any other properties, attributes or relationships.

You can modify the properties, relationships and attributes of Business Units in Edit mode, which you activate by clicking the **Edit** button. Save your changes by clicking **Save**, or click **Cancel** to discard without saving. Bear in mind that **deleting** a Business Unit will also have the effect of deleting its substructure, i.e. all its subordinate Business Units.

For more information about copying a Business Unit see Copying a Building Block.

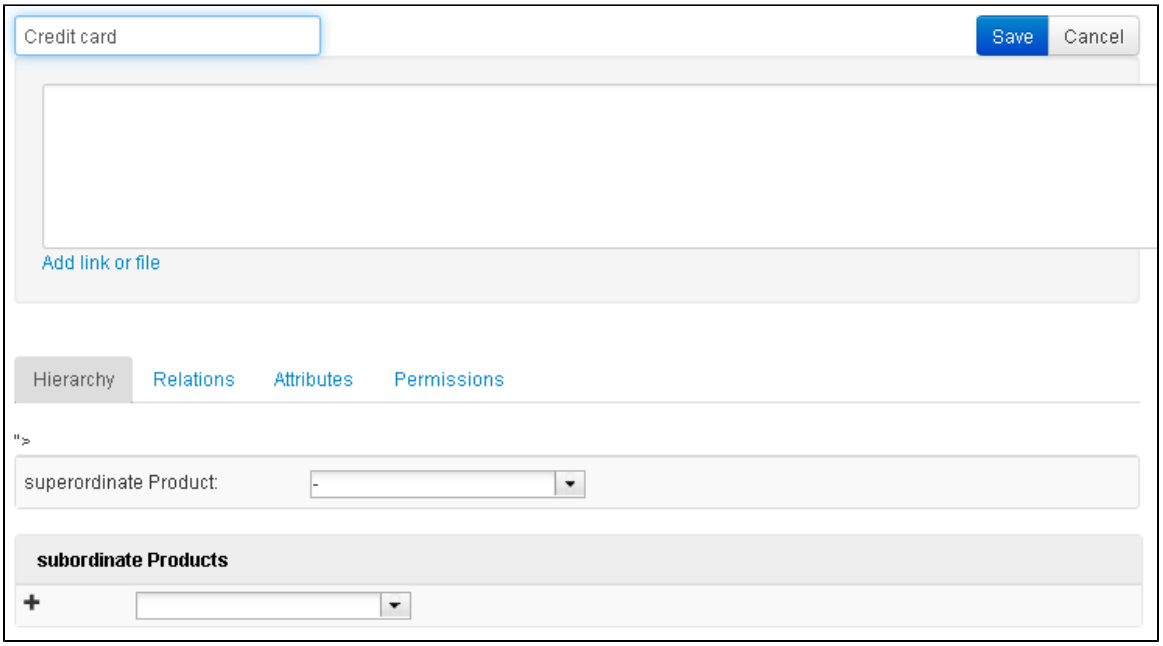


Page for editing Business Units

Products

Besides its *Name* and *Description*, a Product is defined by its association with a *superordinate Product* as well as several *subordinate Products*. The subordinate product's name is then prefaced by the name of its superordinate. User-defined attributes can also be enabled for Products. You can modify the properties, relationships and attributes of a Product in Edit mode, which can be activated by clicking the **Edit** button. You can save your changes by clicking **Save**, or click **Cancel** to discard without saving. **Deleting** a product has the effect of deleting its substructure, i.e. all its subordinate products.

For more information about copying a Product see Copying a Building Block.



Page for editing Products

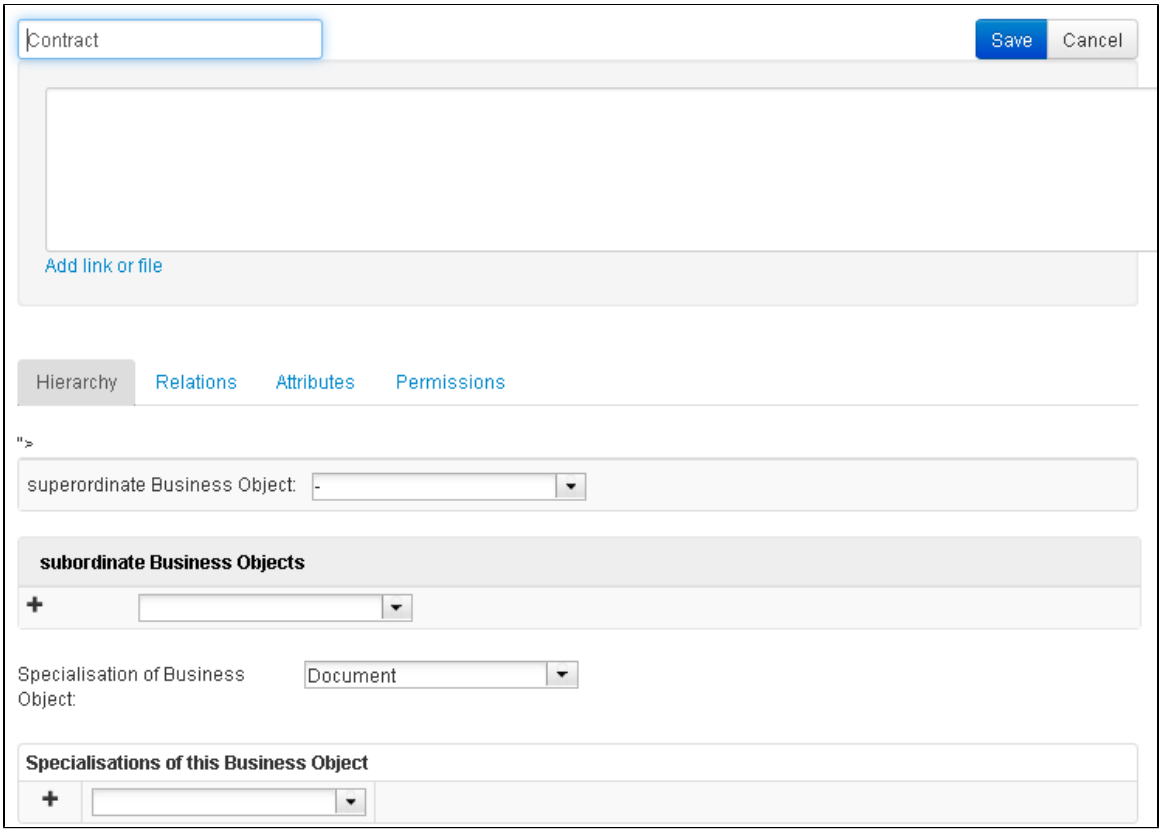
Business Objects

Business Objects have a *Name* and a *Description*, and may also have attributes (if any have been defined and enabled). You can also define these relationships:

- *Superordinate Business Object*: This is a ordinary "is part of"-relationship, as used in other Building Blocks, e.g. the Business Object *Addr ess Data* is a subobject of the Business Object *Customer Data*.
- *Specialises Business Object*: This is a specialisation relationship, for example:
  - The business object *Customer USA* specialises the business object *Customer*; vice versa, *Customer* is a generalisation of *Custo mer USA;*
  - The Information System-specific Business Object (information object) *Customer Data CRM* specialises the Business Object *Cust omer.*

You can modify the Business Objects in Edit mode, which can be activated by clicking the **Edit** button. After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving. **Deleting** a Business Object has the effect of deleting its entire substructure. The specialised Business Objects are retained, but the relationships which these objects have with the deleted object are deleted.

For more information about copying a Business Object see Copying a Building Block.



Page for editing Business Objects

Attributable Association: Information System Release to Business Object

The relation between Information System Releases and Business Objects can get attributes. To activate this option, at least one attribute type needs the "**Relation between information system and business object**"-type in the "**activated for the following Building Block Types**"-field. How to create or edit attributes is described here: Building Block Attributes. Next time, you refresh a Information System or a Business Object page, you will see attributes in the "Used by the following Information Systems" or "assigned Business Objects" table, corresponding to the Building Block. In edit mode, values for these attributes can be edited.

For example, making the complexity attribute for the "Relation between information system and business object"-Building Block Type available and hit refresh in the element itself. After that we will set some values for the attribute type in "Account-Sys RB # 3.1". Now, the "assigned Business Objects" for this Business Object, looks like this:

assigned Business Objects

Name	Description	Complexity
Account statement		high
Accounting entry		high
Check account		average
Contract		average
Savings book		low

Viewing the Account-Sys RB # 3.1 after adding attributes

As you can see, the attribute name (Complexity) appears in header, and for every relation, e.g. "Account-Sys RB # 3.1 to Account statement", a value can be chosen. The same value for Complexity can be found (and edited) on the other side: Business Object in "Used by the following Information Systems". There is a row with "Account-Sys RB # 3.1" and it's description, and also the common Complexity value.

Business Functions

Business Functions have a *Name* and a *Description*, and may also have attributes (if any have been defined and enabled). You can also specify a *superordinate Business Function* as well as one or more *subordinate Business Functions* and the sequence of these subordinate Business

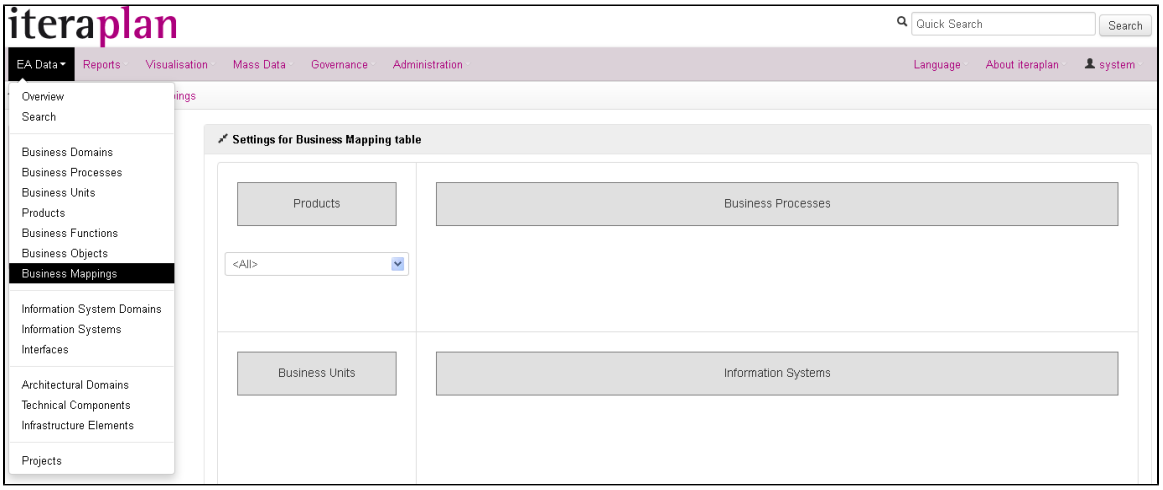
Functions.

You can modify the properties, attributes and relationships of Business Functions in the Edit mode, which can be activated by clicking the **Edit** but ton. After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving. **Deleting** a Business Function has the effect of deleting its entire substructure, i.e. all subordinate Business Functions.

For more information about copying a Business Function see Copying a Building Block.

Business Mappings

A Business Mapping represents the relation between four types of building blocks: **Products**, **Business Processes**, **Business Units** and **Information Systems**.
This page will not show a list of all Business Mappings. It will show a snippet of Business Mappings in form of a table which can be configured.

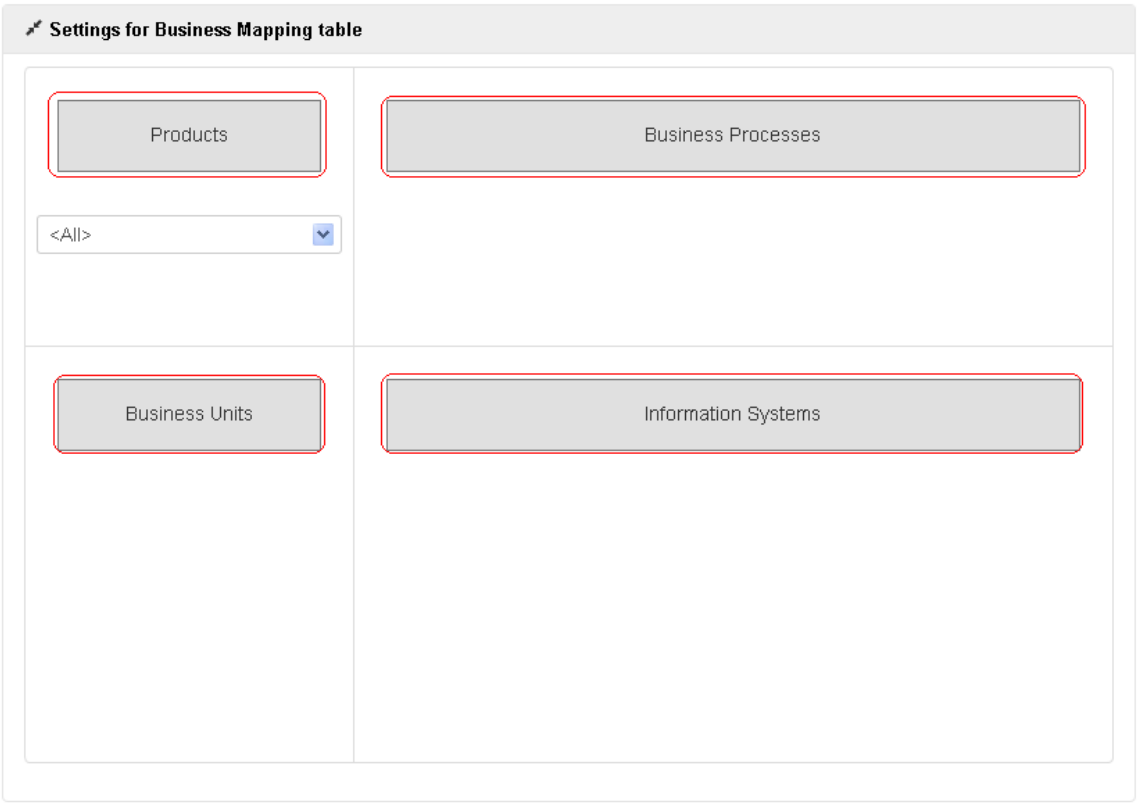


Settings for Business Mapping table

One Business Mapping consists of four different Building Blocks. Each type represents one dimension.

- There has to be one fixed building block. All Business Mappings displayed, after submitting, are connected with it. In the picture below, the type for this Building Block is **Product**. You can select the specific product with the drop-down-box.
- Another Building Block Type will be shown as columns. In this configuration the type is **Business Process**.
- The third Type will represent the rows. **Business Unit** for this configuration.
- The last type is for the content. Every cell of the table will contain a list of building blocks of this type. In this case **Information Systems** will be displayed in the cells.

To change this default configuration, simply **drag-and-drop** the boxes with the name of Building Block type, over an other box to swap them. After selecting the fixed Building Block, a button appears where you can load the table.



Now, you can see a table as shown in the picture below.

Each of the Building Blocks in the cells represents a Business Mapping.

For example, the second element in row "Capital & Risk" and column "Clearing" is "Risk Manager # 1.0". This means, that a Business Mapping connecting the following Building Blocks exists: " - / Clearing / Capital & Risk / Risk Manager # 1.0". The data comes from Product (the element we have selected in settings): "-" , Business Process (in the column): "Clearing", Business Unit (in the row): "Capital & Risk" and the Information System: "Risk Manager # 1.0".

	-	Account & Contract Management	Clearing	Controlling	Core processes	Customer Management	Customer Strategy	Employee Development & Satisfaction	Human Resource Management
-									
Business Customers						Deposits-Mgr # 2.0			
Capital & Risk									
Compliance									
Controlling									
Corporate Customers						Deposits-Mgr # 2.0			
Executive Board									
Finance		Account-Sys RB # 3.1							
Functional Departments									

View Mode

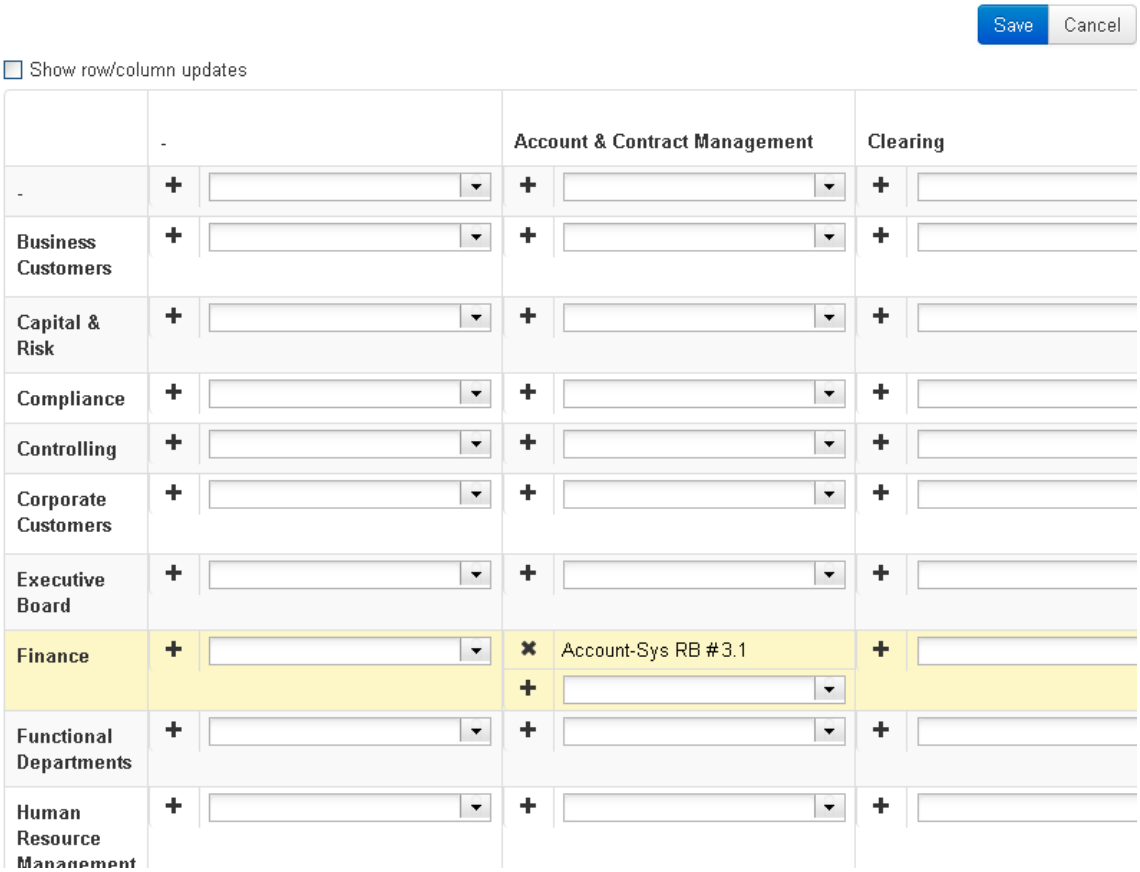
First you are in the view mode. The following operations are available here:

- The settings has collapsed. Click on "Settings for Business Mappings" if you want change something. After changes, the table will disappear.
- Only a part of the table is shown. If you want see other snippets, use the arrows above the table.
- The size of the table snippet can be configured with the fields beside.
- With the **Edit button**, you can edit this snippet. You will need a permission for that. For more information's about editing this table, see below.
- Click on the **Refresh button**, to reload the table.
- The **Close button** will make the table disappear, and sets the settings to default.

Edit Mode

Here you can add and delete Business Mappings. The table snippet is the same as, in the previous page. After making some changes, leave the edit by clicking one of these buttons:

- Save button: for committing the changes
- Cancel button: to revert your changes, and leave edit mode
- Close button: to revert your changes, and go back to default settings for the table



Row and Column update

This is kind of bulk updates only for business mappings. You can either update a row, a column or everything on the snippet you see. Only the visible cells are affected by row and column updates. To enable this feature fill the check-box over the table. A new row and column appears. Select values which should be taken over for all building blocks in the row/column, then click on the button below. The values are inserted into

the corresponding cells, where they get saved, after clicking on the save button. Values standing only in the updater, would not be saved.

To use the feature, the user needs **Read**-, **Create**- and **Update**-Permissions for the building block type "business mapping", as well as **Read**- and **Update**-Permissions for the building block type in the top left quarter (in the section "Settings for business mappings table") near the drop-down list (for example "Products" in the example images for this chapter).

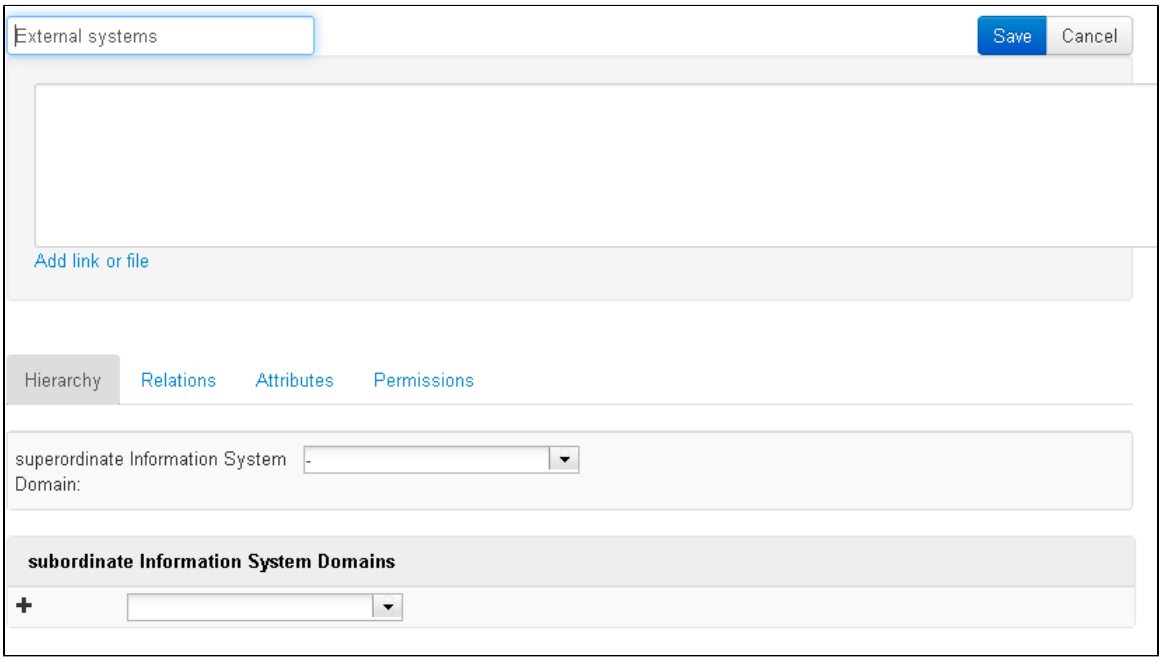
Information System Domains

Information System Domains serve to group Information Systems. For example, you can group all sales process Information Systems into one domain.
Information System Domains have the properties *Name* and *Description* and may also have attributes (if any have been defined and enabled). You can also define these relationships:

- *Superordinate Information Systems Domain*: This is a ordinary "is part of"-relationship, as used in other Building Blocks;
- *Subordinate Information System Domains*: indicates which information system releases belong to the domain.

You can modify Information System Domains in Edit mode, which you activate by clicking the **Edit** button. After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving.

For more information about copying an Information System Domain see Copying a Building Block.



Page for editing Information System Domains

Information Systems

Most work with Information Systems is done by creating or modifying their releases. Each Information System Release is one version of a specific Information System. The search results page shows the name of Information Systems in two different representations. The first column contains simple names, while second column shows complete hierarchical names, together with release information. The format is as follows:

```
[ superordinate information system release's hierarchical name : ] <name of
information system > # <release information>
E.g: SAP Classic-P10 : SAP Fi-P10 # 6.0
```

Alongside its simple *Name* (which may contain a release identifier) and *Description*, an Information System Release has the following main properties and relationships:

Productive
The productive timespan of an Information System Release is defined by two items of data: *productive from* and *productive until*. The lack of a start date for a release indicates it is not yet known when the release in question is due to go into productive operation; if the end date is missing,

the end of the release's productive operation has not yet been set. Any inconsistency in entries (e.g. a conflict with the set status) can be revealed by consistency checks (see Consistency Checks).

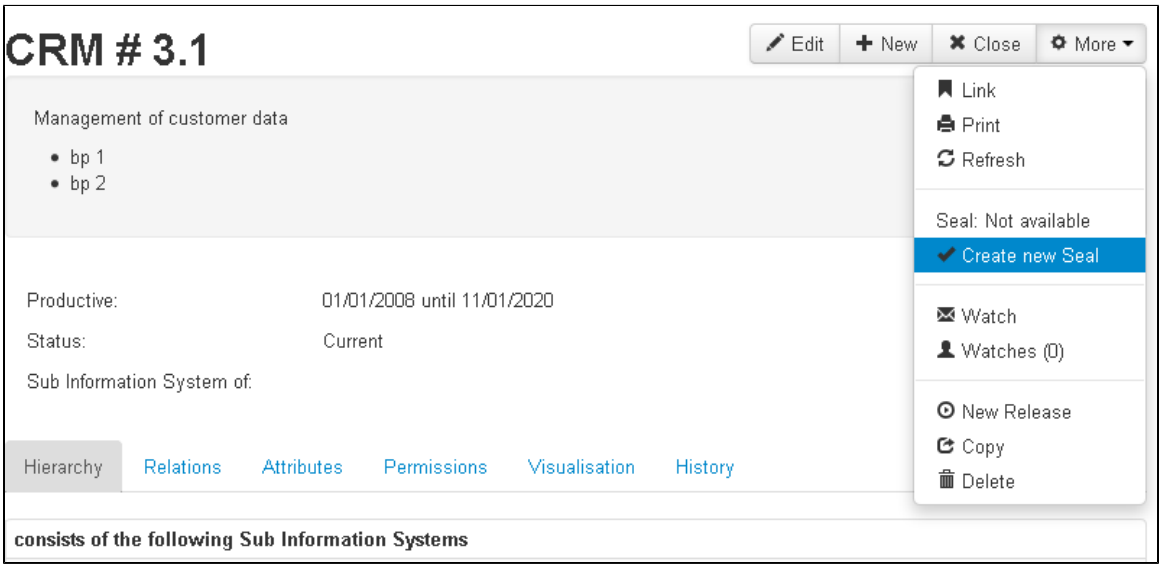
Status
The status of an Information System Release must have one of the following values:

1. *Current* denotes an Information System Release that is productive at the present time, i.e. the release is in productive operation and supports a Business Process;
2. *Planned* denotes an Information System Release that is either being developed or whose rollout is planned. This means there is an approved IT project addressing the implementation and/or introduction of this release;
3. *Target* denotes an Information System Release that is part of the future vision of the business landscape. The Information System in question is still at a draft planning phase. As yet there is no approved project which specifically addresses its implementation or introduction;
4. *Inactive* denotes an Information System Release that was in productive operation at an earlier time but is no longer in use.

EAM-Tip
iteraplan provides a convenient mechanism for handling different strategic scenarios based on your *Target*-information systems. Have a look.

Seal

Seals are used to mark verified Information System states and can be used as a means of data quality assurance. All iteraplan users can see seals and their current states, but for creating a new seal the functional permission '*Create Seal*' is required. The seals can be set and renewed at any time.



By clicking on the Seal-Entry, a table will pop up, showing all current and past seals.

The seal can have one of the following states:
(*Not available*) denotes an Information System Release that does not have any seals yet;
(*Valid*) denotes an Information System Release that has a valid seal set. This means, that this Information System was not modified since this valid seal was set;
(*Invalid*) denotes an information system release that was modified since the last valid seal was set;

(*Outdated*) denotes an information system release that has a valid, but outdated seal set. The seal expiration period is configurable in iteraplan.properties. The default period is 180 days.

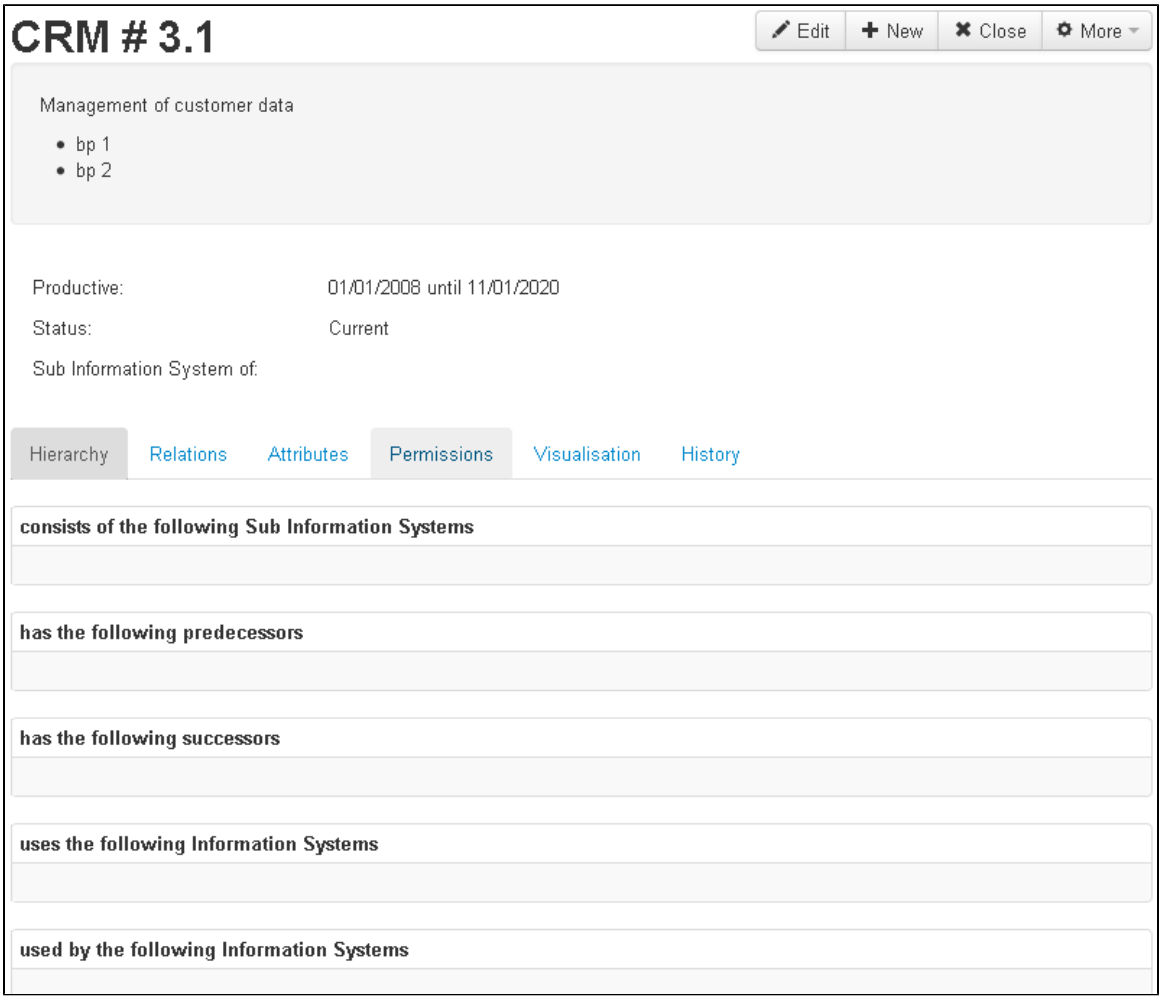
The current seal status can be exported in spreadsheet and diagram reports. Users can filter Information System Releases by seal state.

Sub Information System of
Each Information System Release can be part of another one. You specify a superordinate release by selecting herein its name. The drop-down list presents then the Information System Release prefaced by the name of its superordinate release. The elements of the name are separated by a colon ':'.

Once you have created an Information System Release, you can view and modify the properties in the general area of the edit forms (visible whichever tab you have open). Relationships and other user-defined attributes are managed on separate sections which you open by clicking specific tabs.

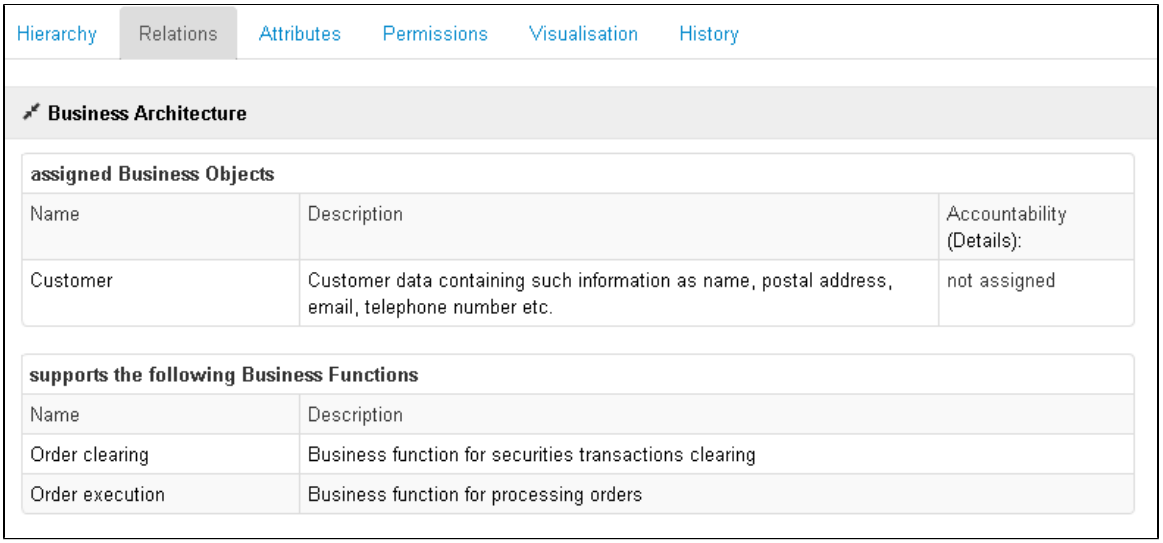
To modify information, click the **Edit** button at the top. Then you can make your changes. You can save your changes by clicking **Save**, or click **Cancel** to discard them without saving. Bear in mind that **deleting** an Information System Release will have the effect of deleting its substructure, i.e. all its subordinate releases.

The **Hierarchy** tab presents an overview of Interfaces of this Information Release.



Hierarchy tab on the screen for managing information system releases

Use the **Relations** tab to edit relationships for the following building blocks.



Relations tab on the page for editing Information System Releases

The Building Blocks are grouped by different architectures:

Information System Architecture

1. *Sub-Information Systems* : An Information System can consist of several sub-Information Systems. Such sub-Information Systems realise a part of the current Information System's functionality.
2. *Predecessors* and *Successors* : A predecessor-successor relation exists as a basis for relationships connecting Information Releases. To assign a particular release as a predecessor/successor to the release you are working with, select its name from the drop-down list.
3. *Information System Domains* : Each Information System Release can be assigned to one or more Information System Domains. To make this assignment, select the domain from the drop-down list
4. *Uses* and *Used by* : A uses-used by relation is another relationship connecting Information Releases. Values for this relation can be added from the drop-down list in a similar way to the predecessor-successor relation.

Technical Architecture

1. *Technical Components* :The Technical Components comprise elements (e.g. application servers or frameworks) on which the Information System Release is based.

Infrastructure Architecture

1. *Infrastructure Elements* : The Infrastructure Elements describe the operating platform (e.g. servers) that runs the Information System Release.

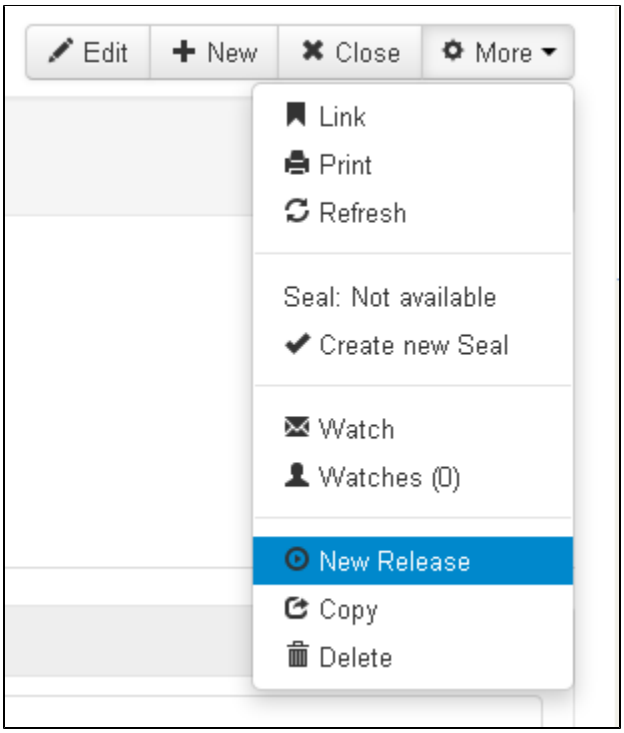
Project Portfolio

1. *Projects* : Use this section to manage information about the Projects concerning the Information System Release.

Business Architecture

1. *Business Objects* : Use this section to manage information about the main Business Objects handled by the Information System Release. This association can get attributes. Find more informations about attributable associations here.
2. *Business Functions* : Use this section to manage information about the main Business Functions supported by the Information System Release.
3. *Business Mapping* :
    a. *Particular Business Units specified :* the Business Process is only supported for the units specified. If there are no Business Units specified, the Business Process assignment is applicable without restriction in every Business Unit.
    b. *Particular Products specified :* the Business Process is only supported for the Products specified. If there are no Products specified, the Business Process assignment is applicable without restriction for every Product.
4. *Business Units* : To indicate that a Business Unit uses this Information Release (irrespective of which Business Process is involved), a new business support relationship must first be created for the release in question. Since the new relationship is to be valid irrespective of Business Process, you can model this with the virtual Business Process ("-").
5. *Products* : To indicate that a Product uses this Information Release (irrespective of which Business Process is involved), a new business support relationship must first be created for the release in question. Since the new relationship is to be valid irrespective of Business Process, you can model this with the virtual Business Process ("-").

The ***Attributes*** tab provides functions for editing attribute values (see *Attribute Groups and Attributes). The* **Permissions** tab provides functions for assigning explicit Building Block Permissions (see *Users, Roles and Permissions*). You create a new Information System, and by implication a new Information System Release for this system, with the **New** button. The button opens a form (see below) where you can enter all the properties, interfaces, relationships and permissions for the new Information System. An asterisk preceding a field indicates that an entry is mandatory.



Creating a new Information System via the "New Release" button

Similar to the copy function, a new release can be created. This will also take over the values from the actual release, but the name can't be changed, because the new release references an Information System which already exists. To create a *new Information System Release* based on an existing one, click on the **New Release** button. You can find more information's about copying building blocks here: Copying a Building Block.

In both cases the following information will be copied in the new Information System (Release):

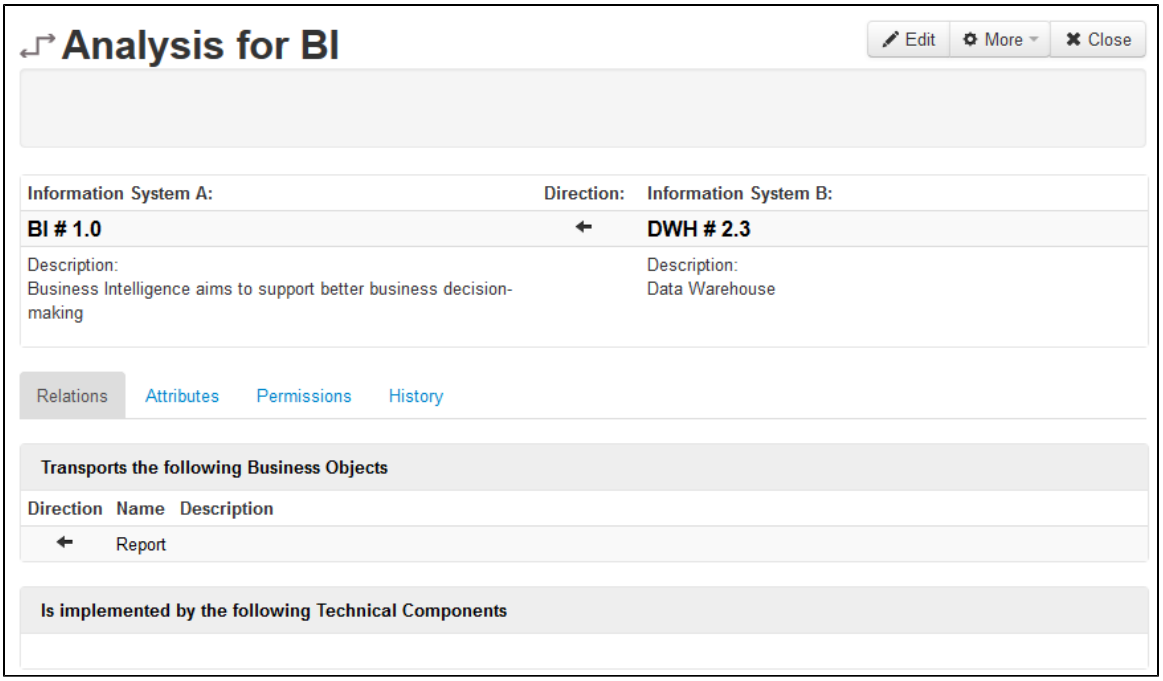
- Relations
- Attributes
- Permissions

Not only does this permit you to use much of the data already entered for an existing release, you can also enter the release name of your choice and create a new Information System.

Since Interfaces are a separate BuildingBlock they are not copied within here, but can be copied on their own, providing you with a great flexibility.

Interfaces

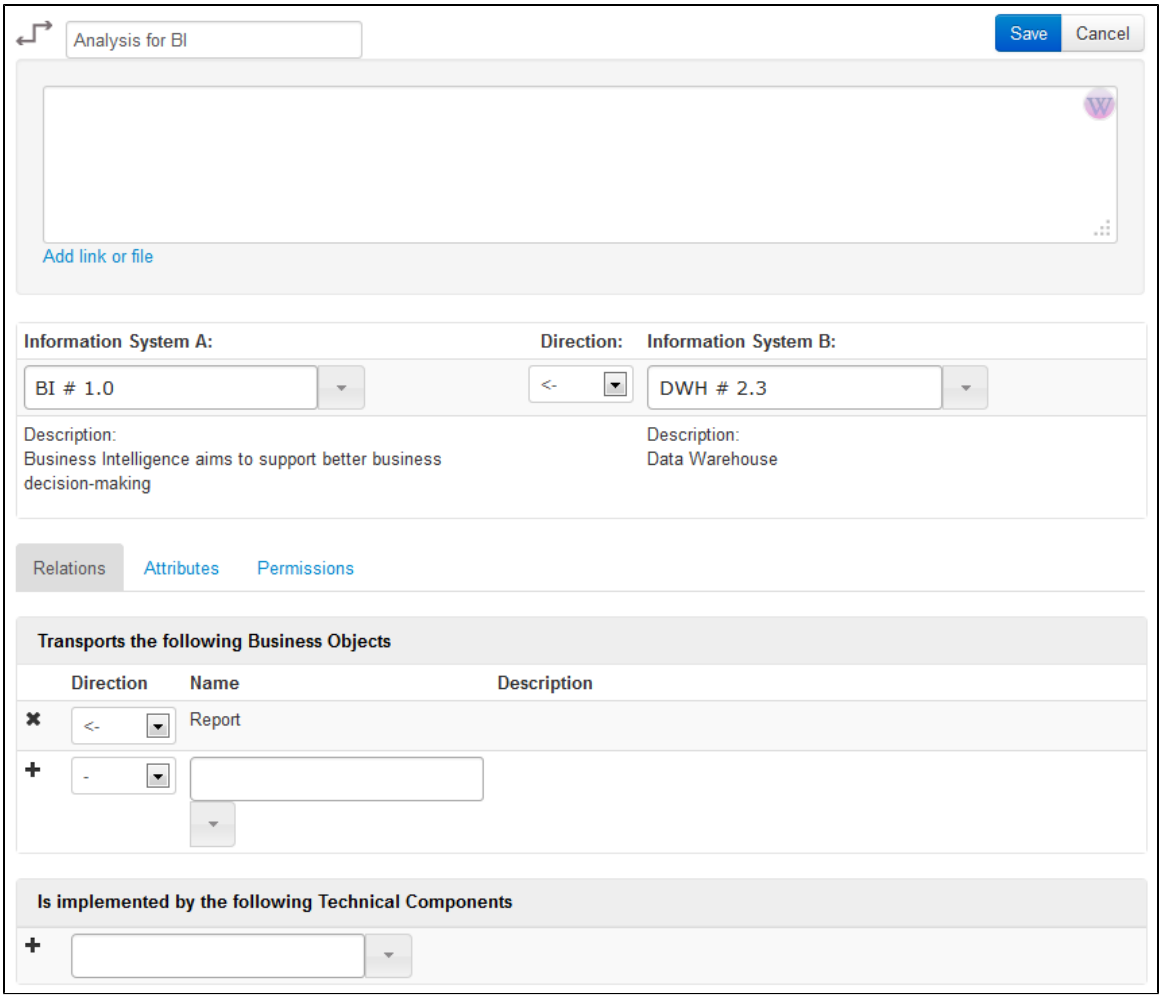
Similarly to the other existing Building Block types, an Interface contains not only the property *Description* (since release 2.8), but also a *Name* an d a *Transport direction*. An Interface has moreover relationships with the Business Objects it is transporting, and with the Technical Components on which it is based. Additional information about an Interface can be stored in user-defined attributes. To view or modify an Interface that already exists between two Information Systems, select it from the search results page. A searched text will be looked up in the name of the Interface as well as the names of the connected Information Systems.



Locating an Interface via connected Information System Releases

The detail screen for an Interface shows which Information Systems it connects, their respective descriptions and a description for the Interface. Click **Edit** to switch to Edit mode, where you can modify the associations with the following Building Blocks (see screenshot below):

- Informations Systems which are connected by the Interface;
- Technical Components by which this Interface is implemented;
- Business Objects transported via this Interface, and their direction of flow;
- Associated attribute values and permissions.



Page for editing Interfaces

Click **Save** to save your changes permanently; **Cancel** discards your entries without saving.

To copy an Interface click on **Copy** button. The following information will be copied:

- Relations
- Attributes
- Permissions

Architectural Domains

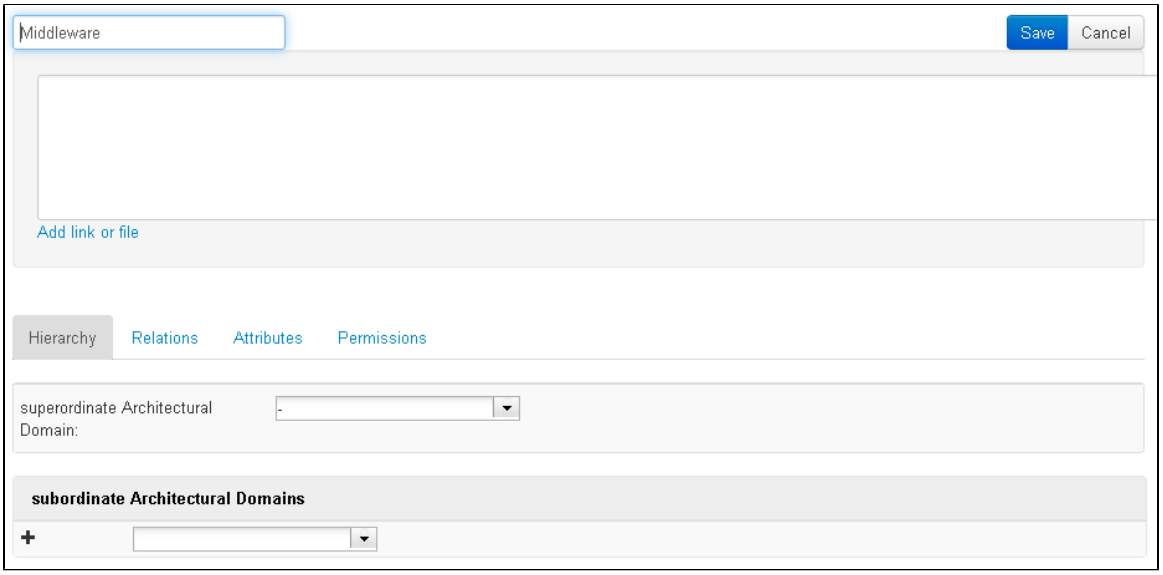
Architectural Domains serve to group Technical Components. For example, components such as MySQL and Oracle can be grouped into Databases, and BEA and JBoss into Application Servers.

Architectural Domains have the properties *Name* and *Description* and may also have attributes (if any have been defined and enabled). As for Information System Domains, you can also manage the following relationships for Architectural Domains:

- *Superordinate Architectural Domain*: This is a ordinary "is part of"-relationship, as used in other Building Blocks;
- *Contains the following Technical Components*: indicates which Technical Components belong to the domain.

You modify the Architectural Domains in Edit mode, which you activate by clicking the **Edit** button. After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving.

For more information about copying an Architectural Domain see Copying a Building Block.



Page for editing Architectural Domains

Technical Components

The Technical Component describes which Technical Components are in use by an Information System, e.g. databases or application servers, also programming languages or frameworks.

> **EAM-Tip**
> In most cases elements listed in standardization catalogs (also called blueprints) are modeled as Technical Components.

Alongside its *Name* (which might contain a release identifier) and *Description*, a Technical Component has the following main properties:

- *Productive* : The productive timespan of a technical component is defined by two items of data: *productive from* and *productive until*. The lack of a start date indicates it is not yet known when the release in question is due to go into productive operation; if the end date is missing, the end of the release's productive operation has not yet been set.
- *Status* : The status of the Technical Component can have any of the following values:
  - *Current* designates a Technical Component that is productive at the present time;
  - *Planned* designates Technical Component that is either being developed or whose rollout is planned;
  - *Target* denotes a Technical Component which is part of the future vision of the technical landscape. The component in question is still at draft planning phase, and there are as yet no firm plans for rollout;
  - *Inactive* denotes a Technical Component that was in productive operation at an earlier time but is no longer in use;
  - *Not assigned* denotes a Technical Component for which no status has yet been defined.
- *Available for Interfaces* : Check the box next to **Available for interfaces** if this component can be used for implementing Interfaces (see I nterfaces).

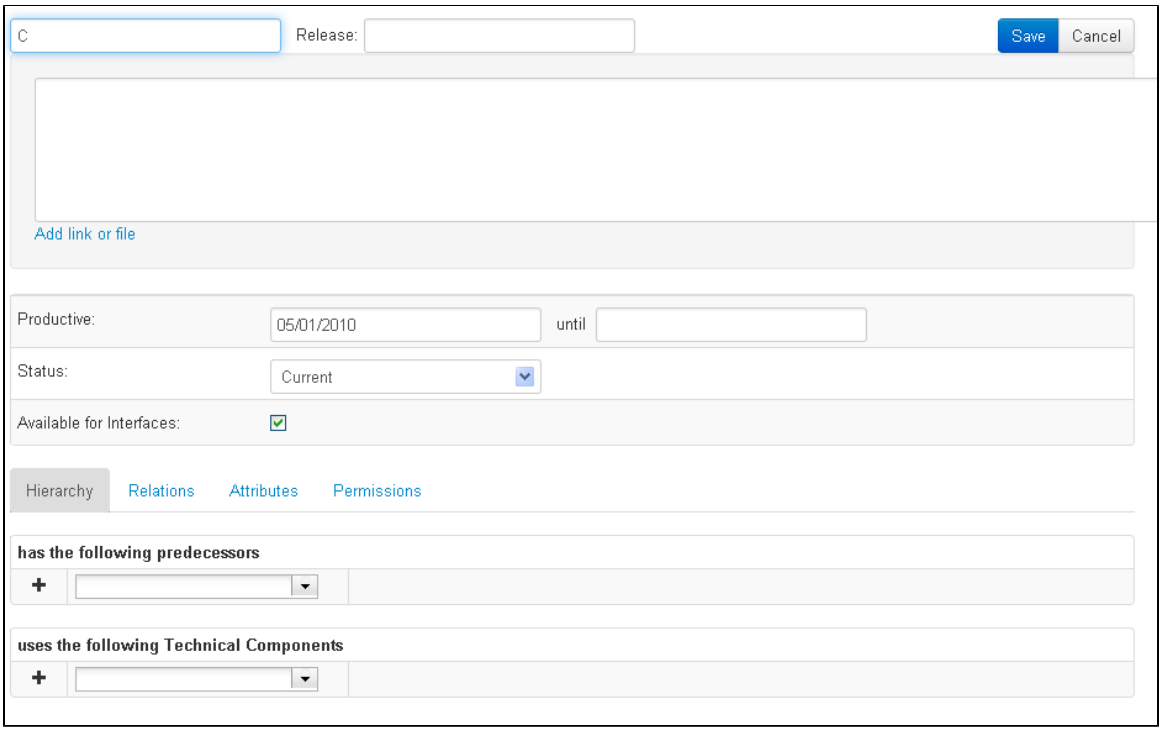
You can edit these properties, and the following relations and attributes, by switching to Edit mode and opening the tabs in question (Hierarchy and Relation):

- *Predecessors :* A predecessor-successor relation exists as the basis for relationships connecting Technical Components. To assign a particular component as a predecessor to the one you are working with, select the name of the predecessor from the drop-down list.
- *Architectural Domains :* Each Technical Component can be assigned to one or more Architectural Domains. To make this assignment, select the domain from the drop-down list.
- *Infrastructure Elements* : Each Technical Component can be assigned to one or more infrastructure elements. To make this assignment, select the element from the drop-down list. This association can get attributes, which works in a similar way than the association between Information Systems and Business Objects.
- *Uses the following technical components* : Technical Components may in turn use other Technical Components. You can specify these by entering their names.
- *Is the basis of the following Information Systems :* Technical Components can form a part of Information System Releases. This can be modelled by entering the releases here.

> **EAM-Tip**

The relation *Uses the following Technical Components* can be helpful in modeling connections between different Technical Components within a portal.

You can enable additional Attributes for Technical Components. To modify data, click **Edit** to switch to Edit mode. After making your changes, you can save them permanently by clicking **Save**, or click **Cancel** to discard without saving. You cannot delete a Technical Component while it is still being used by an Interface.



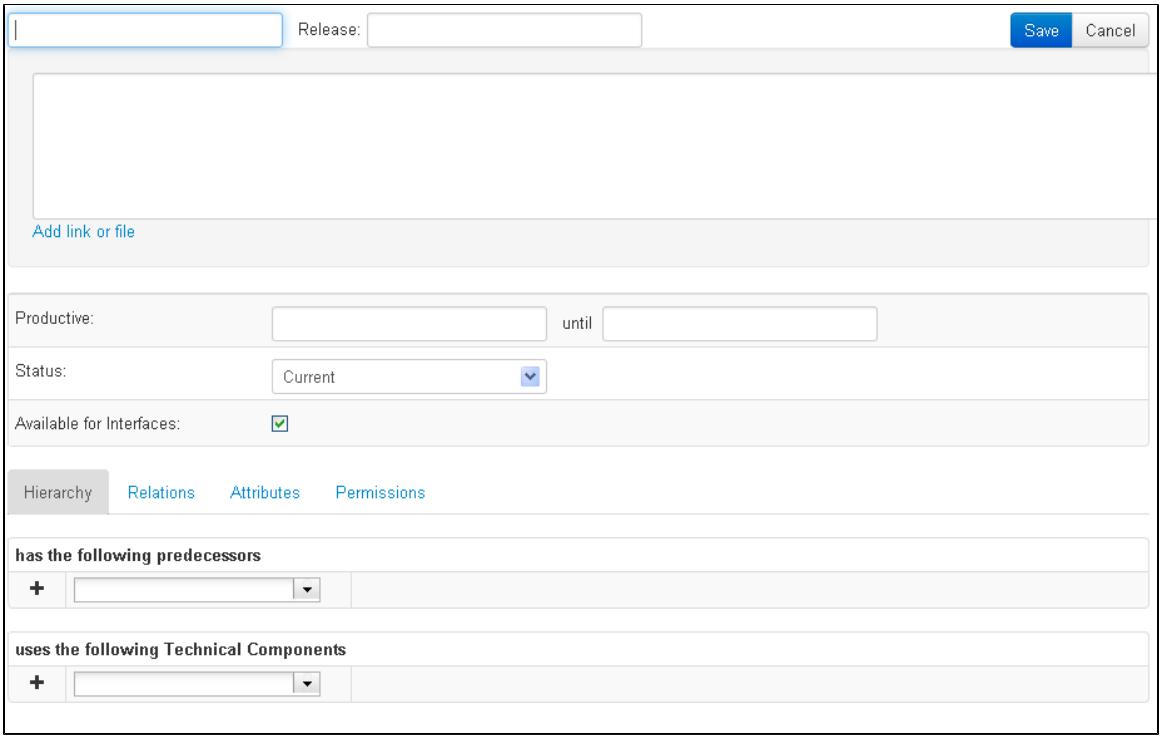
Relations tab on the page for editing Technical Components

The **Attributes** tab provides functions for editing attribute values (see Attribute Groups and Attributes).

The **Permissions** tab provides functions for assigning explicit Building Block permissions.

The **Visualisation** tab provides functions for a preview of the specific Technical Component.

You create a *new Technical Component* with the **New** button. An asterisk '*' proceeding a field indicates that an entry is mandatory.



Creating a new Technical Component via the New button

Similar to the copy function, a new release can be created. This will also take over the values from the actual release, but the name can't be changed, because the new release references an Technical Component which already exists. To create a *new Technical Component Release* ba sed on an existing one, click on the **New Release** button. For creating a new Technical Component at the same time you create the new release, click **Copy Release**. You can find more information's about copying building blocks here: Copying a Building Block

In both cases the following information will be copied in the new Technical Component (Release):

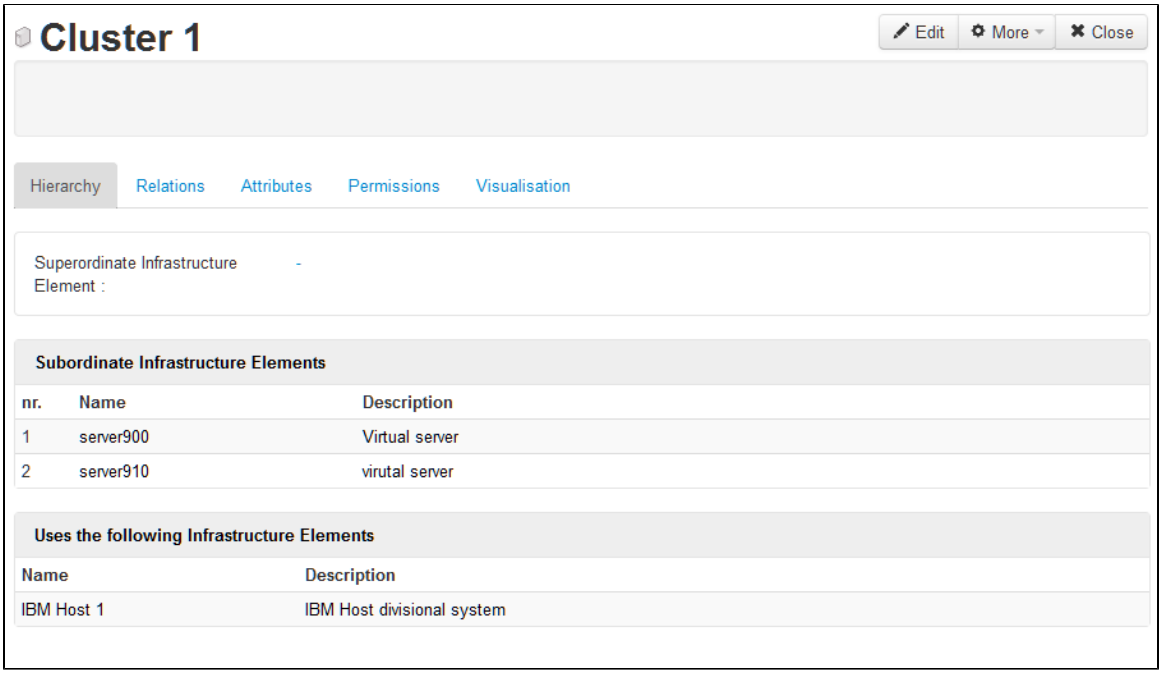
- Relations
- Attributes
- Permissions

To display an overview of the data of a Technical Component Release, switch to View mode and click the printer symbol at the top right of the form. This opens the print view.

Infrastructure Elements

Infrastructure Elements describe the operating platform (servers etc) on which the Information System Release is running.

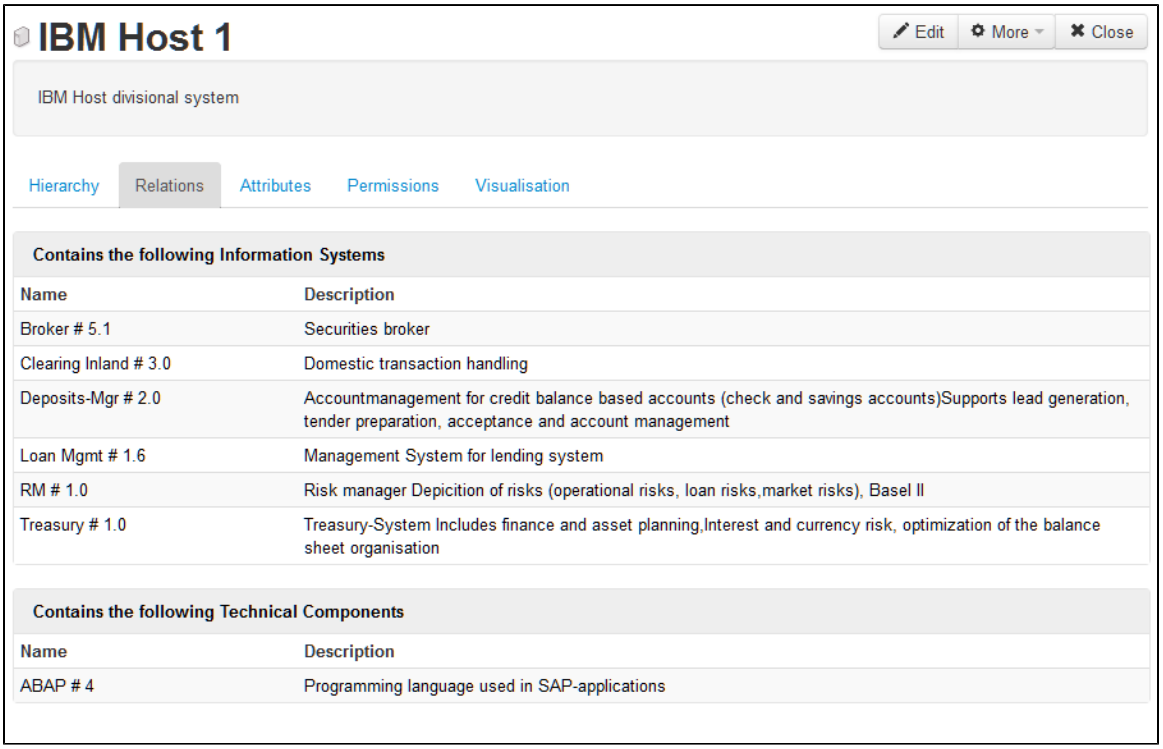
As well as specifying the *Name* and *Description* of an Infrastructure Element, you can also set different kind of relationships to other building blocks and copy the current Infrastructure Element. For more information about copying an Infrastructure Element see Copying a Building Block.



Hierarchy tab of Infrastructure Elements

On the **Hierarchy** tab you can see and set one *superordinate Infrastructure Element* and one or more *subordinate Infrastructure Elements*. The sequence of these subordinate elements can be altered at any time. The specified arrangement is used in the menus and in how the elements are presented in the system.

new
Uses and *Used by* relation: A uses-used by relation is another relationship connecting Infrastructure Elements. Values for this relation can be edited in a similar way to the subordinate elements relation.



Relations tab of Infrastructure Elements

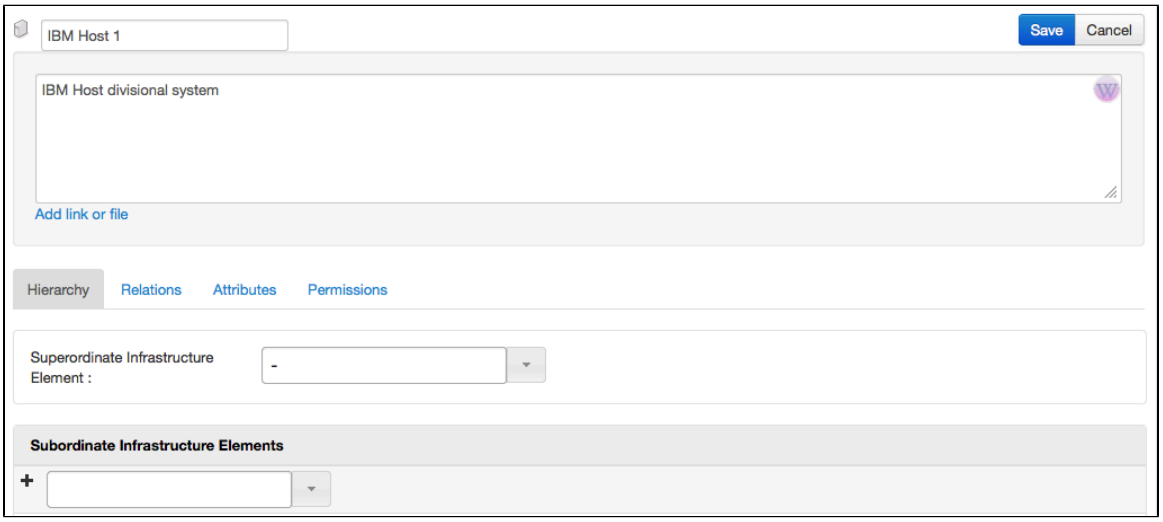
On the **Relations** tab you can see and set Information Systems and Technical Components contained in this Infrastructure Element. Values for

this relation can be edited in a similar way to other multivalued relationships like the uses relation.

On the **Attributes** tab you can, in the same way as for other building block types, assign attribute values for user-defined attributes which are available to Infrastructure Elements.

The **Permissions** tab enables you to assign explicit Building Block Permissions.

To do this or to apply other changes, click **Edit** to switch to Edit mode. You can save your changes by clicking **Save**, or click **Cancel** to discard without saving.



Page for editing Infrastructure Elements

Projects

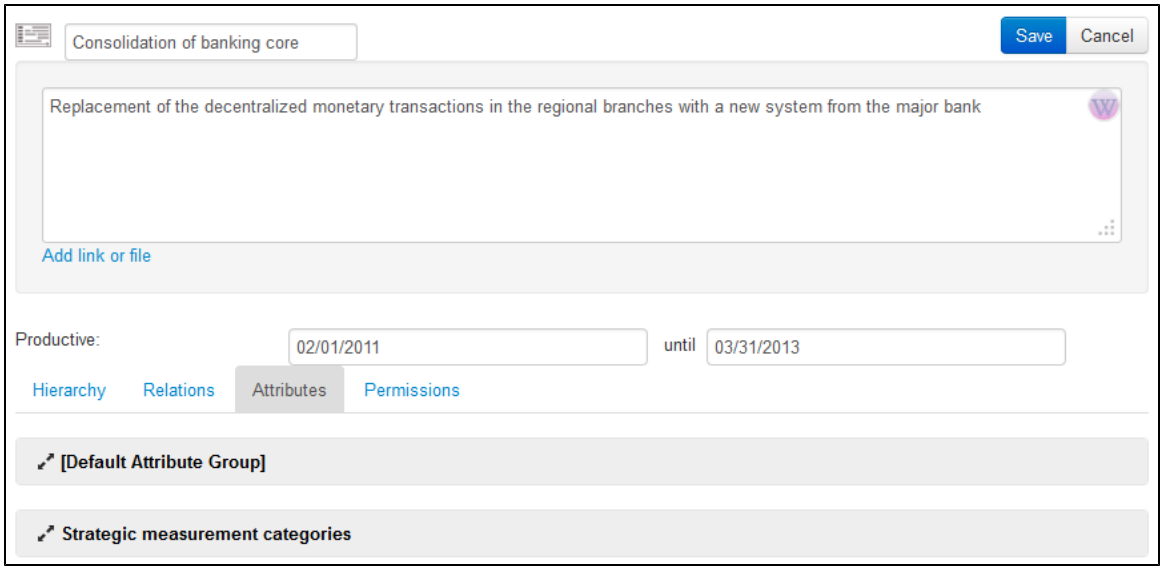
> **EAM-Tip**
> The element **Project** can be used to model current Projects as well as demands and change requests.

Alongside its *Name* and *Description*, a Project has the following properties:

- *Productive :* The runtime of a Project is defined by two items of data: *productive from* and *productive until*. The lack of a start date indicates no start time has yet been set for the project; if the end date is missing, the project end date has not yet been set.
- *Sub projects of :* Each Project can be part of another one. You specify the *superordinate Project* by entering its name here. Afterwards, the name of the Project is presented in the application prefaced by the name of its superordinate.

You can also enable user-defined attributes and assign attribute values. To modify properties and attribute values, click **Edit** to switch to Edit mode. You can save your changes by clicking **Save**, or click **Cancel** to discard without saving. **Deleting** a Project has the effect of deleting its complete substructure, i.e. all its sub-projects.

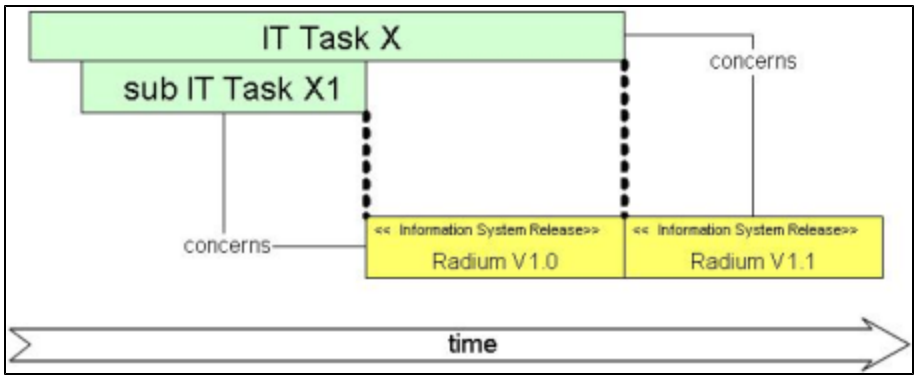
For more information about copying a Project see Copying a Building Block.



Page for editing Projects

In many cases the Project end date will coincide with the starting date of the associated Information System Release, as illustrated.

> Please bear in mind, however, that iteraplan will not automatically ensure that the two dates do in fact coincide. You can use a Consistency Check to verify that theses dates match (see Consistency Checks).



Relationship between Projects and Information System Releases

Analyzing Landscape Data and Generating Reports

One important purpose of IT landscape management is to create transparency in mapping as-is, planned and to-be enterprise architectures. iteraplan provides a wide choice of reports in spreadsheet and diagram format to help you analyse the landscape data stored in the application. The Spreadsheet Reports output query results as a list in iteraplan or to a file in Microsoft Excel or CSV format for editing later. Diagram Reports present at-a-glance overviews of the data in iteraplan and the interdependencies that exist between data of different types. The output of Diagram Reports is presented in Microsoft Visio format. This section explains the various report options at your disposal.

Diagram Reports - Visualisations

The menu command **Diagram Reports** enables you to create different types of diagrams:

- Custom Dashboard
- Landscape Diagram
- Cluster diagram
- Nesting Cluster diagram
- Information flow diagram
- Portfolio diagram
- Masterplan diagram
- Bar and Pie diagrams
- Composite Bar and Pie diagrams

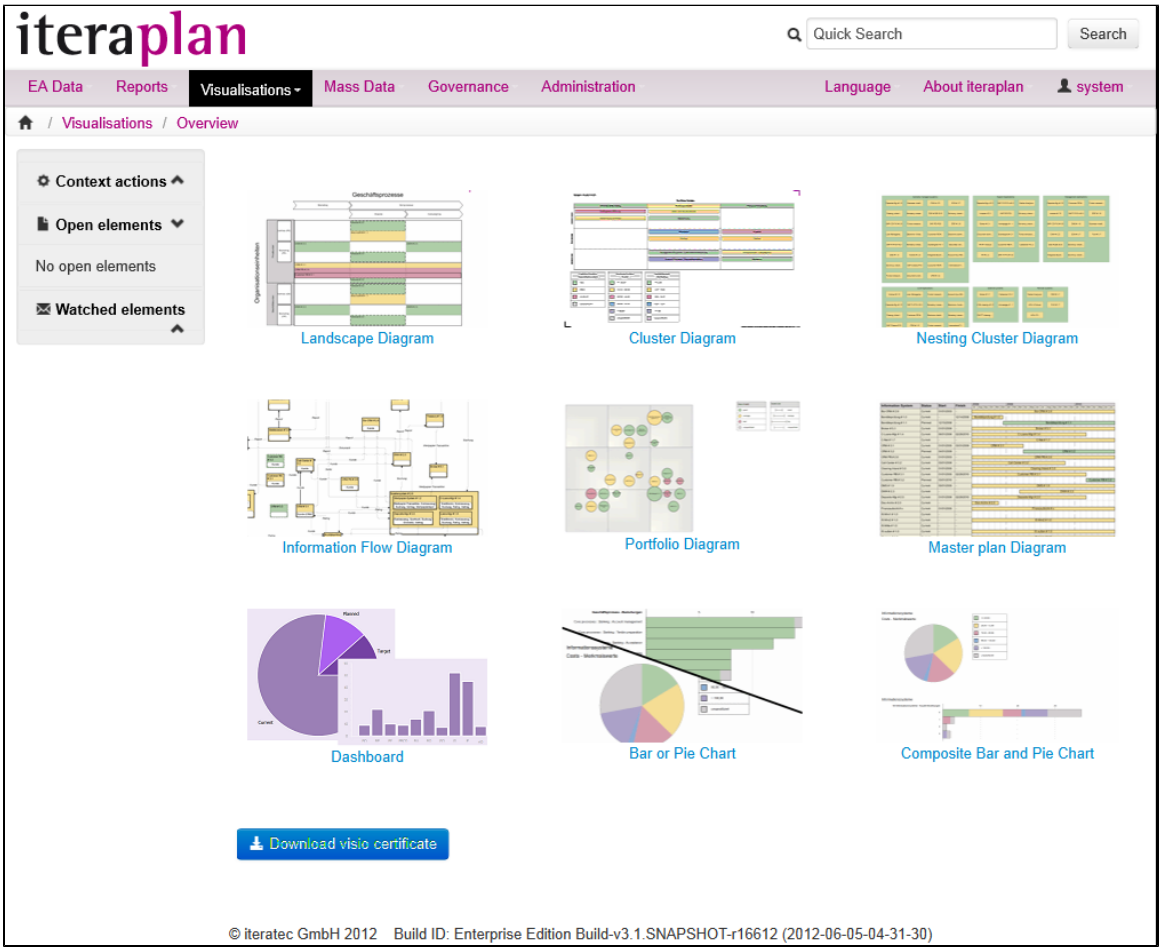


Diagram reports in the main menu

These diagrams can be generated in SVG, JPG, PDF, PNG (all diagrams), and in Microsoft Visio format (all diagrams, except for the Cluster, Bar, Pie and Composite diagrams).

Additionally you can directly execute saved queries from here. When you click a saved query in the dropdown list that opens from the **Directly execute saved query** Menu the query will be executed and the resulting file will be offered for download. The file format can be changed within the saved query.

To guarantee best appearance of the graphical representations, all boxes are of the same size. Whenever a label of an element doesn't fit into a box, an abbreviation is used with a number in square brackets. The numbers are explained in an additional legend.

This section explains how to create the provided diagram reports.

Limitations with Microsoft Visio

- The feature "assigning multi-value attributes to colors" is not available in MS Visio.
- Visio export requires MS Visio Professional 2003 SP2 or newer. See also our FAQs.

Custom Dashboard

new in 3.1
Custom Dashboards are available since release 3.1

A dashboard is a page easy to read and which contains diagrams of the current status and of historical trends. It enables the user to get an overview of a certain building block type. In iteraplan each dashboard is based on a specific building block type. A template contains the complete design, optionally some descriptive text and, of

course, diagrams. When creating an instance of a dashboard, you choose a filter to fill the diagrams with a different content.

Thereby, all dashboards for a specific building block type have the same design but might show different content.

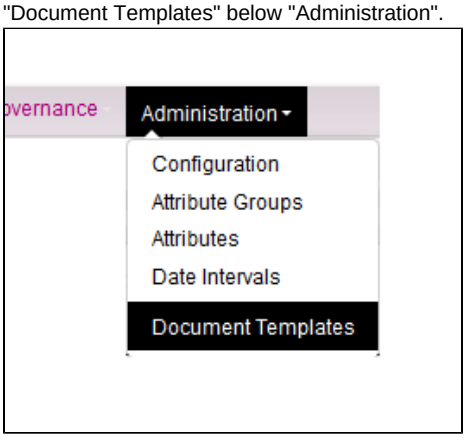
Dashboard Templates

Permission

The following permission is required for the creation of templates: "Add and remove template files"

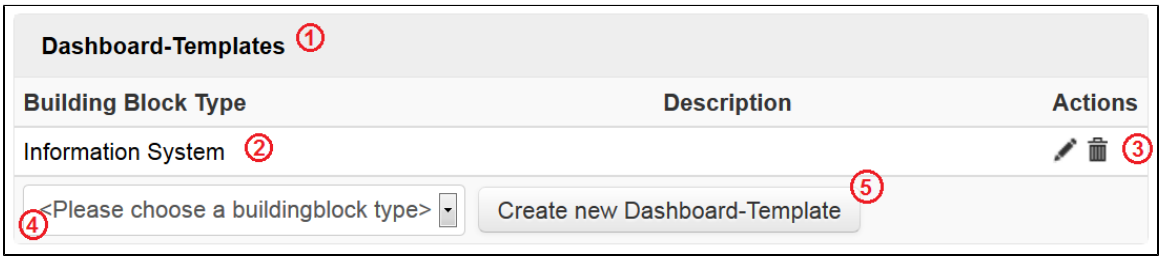
Menu entry

The function to create dashboard template is located in the page for "Document Templates". You can reach the page via the menu item "Document Templates" below "Administration".



Dashboard Template Overview

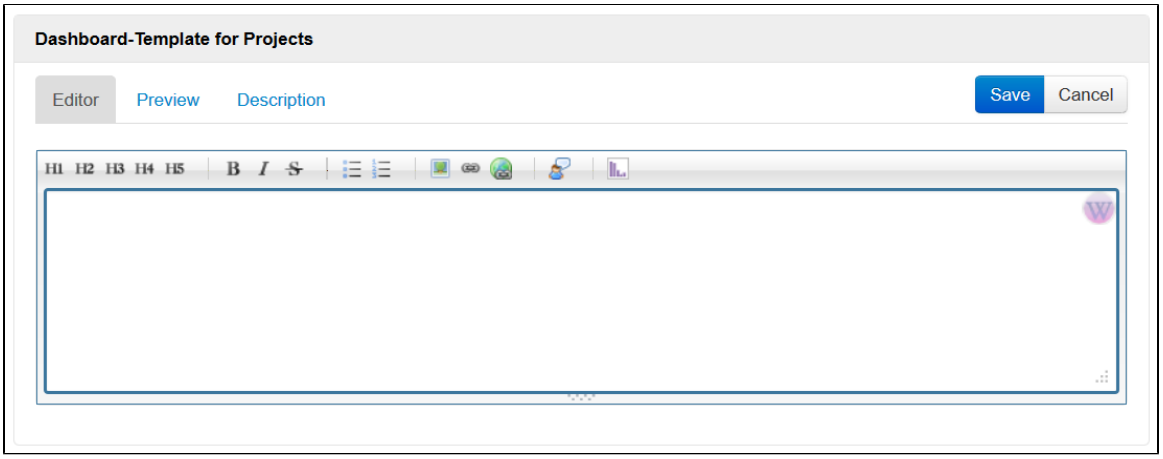
At the end of the page for "Document Templates" you will find the option for the creation of dashboard templates (1).



The table for dashboard templates contains all existing dashboard templates (1). Each existing template can be edited and deleted (3). To create a new template, you must choose a building block type first (4) and confirm with a click on the button "Create new Dashboard-Template" (5).

Create/Edit Dashboard Template

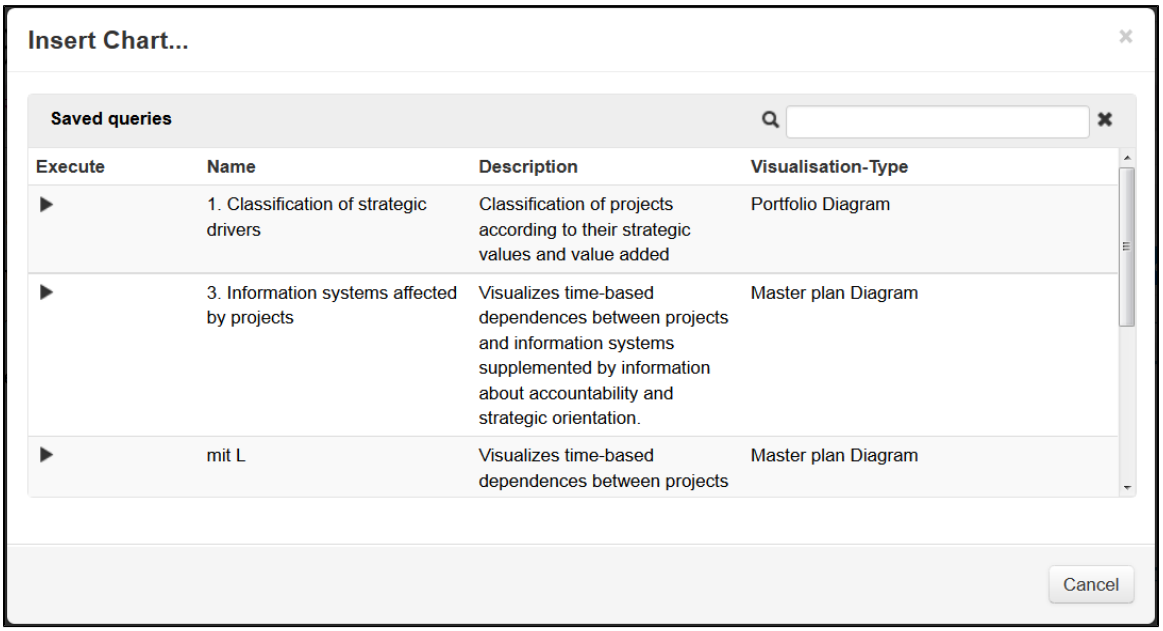
When you create a new template or edit an existing one, a panel is opened.



The panel contains three tabs: Editor, Preview, Description.

- ***Editor:***
  In the editor, you create the dashboard template.
  The rich text editor provides the most common elements of the XWiki syntax for direct selection. You can find a comprehensive description of the XWiki syntax here: XWiki Syntax

  In addition to the XWiki syntax elements, an option to **insert diagrams from iterplan** is available. The button opens the following menu.



  This lightbox/popup shows all available "saved queries" / diagrams which are based on the building block type the current dashboard supports. If no corresponding query is available, the list is empty.
  With a click on a saved query, the appropriate diagram is inserted into the editor, in form of a special syntax. A detailed description of this syntax is shown below.
  All diagrams are referenced by an ID, they aren't copied. Therefore all changes to the saved diagrams are immediately visible in the dashboards.

- ***Preview:***
  In the Preview tab, you can get a preview of the dashboard without saving it.
  All embedded graphics are represented as original without filtering, because no filter can be specified in the template.

- ***Description:***
  In the description tab you can insert an optional description which is then shown in the list of saved dashboards.

Syntax to insert a chart (Where the content is to be filtered)

The following syntax is used, if you insert a diagram from iteraplan: `<diagram>ID</diagram>`
ID is meant as a placeholder for the diagram id.

In addition, you can also specify a height or width. For example
`<diagram width="900">ID</diagram>`
or
`<diagram height="900">ID</diagram>`
But you can only use one of this. If both are simultaneously specified, only the width is taken into account and the height is scaled automatically.

If you use this syntax for inserting iteraplan diagrams, then the contents of the diagram (only in the dashboard instance) is replaced according to the used filter.

Please note that this syntax allows you to embed only those diagrams which use the same building block type as specified for the dashboard.

Inserting other graphics (Where the content is not to be filtered)
If you like to use other graphics or additional diagrams from iteraplan which should not be modified by a filter, then use the XWiki Syntax for inserting images [[image:http://...]]. To insert the syntax you can also use the following button: 
Before you can use a link from a stored query it is necessary to change the parameter **`output-Mode`** to the value **`inline`** and to add the following parameter **`resultFormat=SVG`**.
A valid link must look as follow:
`...url to iteraplan.../show/fastexport/generateSavedQuery.do?id=26&savedQueryType=reports_export_graphical_Masterplan&outputMode=inline&resultFormat=SVG`

To use size adjustment, you can use the following additional parameters. These must only be appended to the link.

- `&width=...`
- `&height=...`

The size is specified in pixels without specification of the unit (px).
Only one value can be supplied either height or width. The other value is always scaled to the appropriate size. If both values are specified, only the width is considered and the height is scaled automatically.
By default, all diagrams in iteraplan are provided with additional information like an QR-Code, Timestamp ...
With the parameter

- `&nakedExport=true`

it is possible to hide this informations.

Suggested best practice
Rename your Saved Queries / Diagrams for using within a dashboard by introducing a prefix. Saved Queries / Diagrams in dashboards will only be referenced. That way you can prevent that you accidentally delete a Saved Query / Diagram used in a dashboard.

Dashboard Instance

Permission
The following permissions are required to **create** custom dashboards:

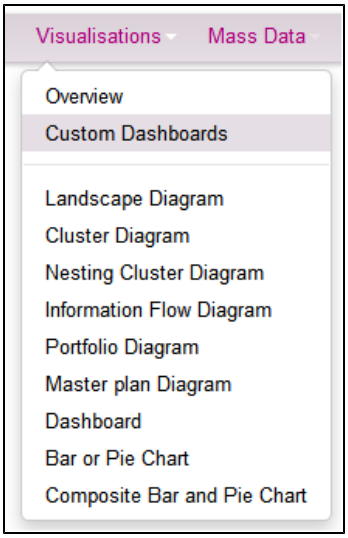
- "Create and execute queries for Diagram Reports" or
- "Edit (create, update and delete) and execute queries for Diagram Reports"

The following permission is required to **read/open** custom dashboards:

- "Execute saved queries for Diagram Reports"

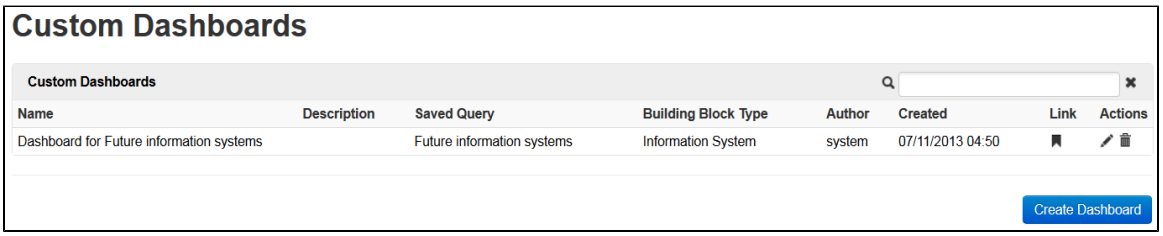
Menu entry

The function to open and to create custom dashboards is located on the page "Custom Dashboards". You can reach the page via the menu item "Custom Dashboards" below "Visualisations".



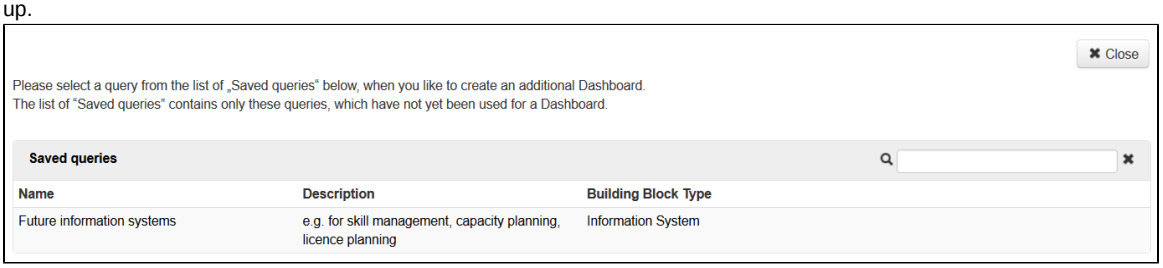
Custom Dashboard Overview

On the overview page all existing dashboards are listed. Click on one dashboard to open it. Additionally there are actions to create a link, to edit a dashboard and to delete a dashboard.



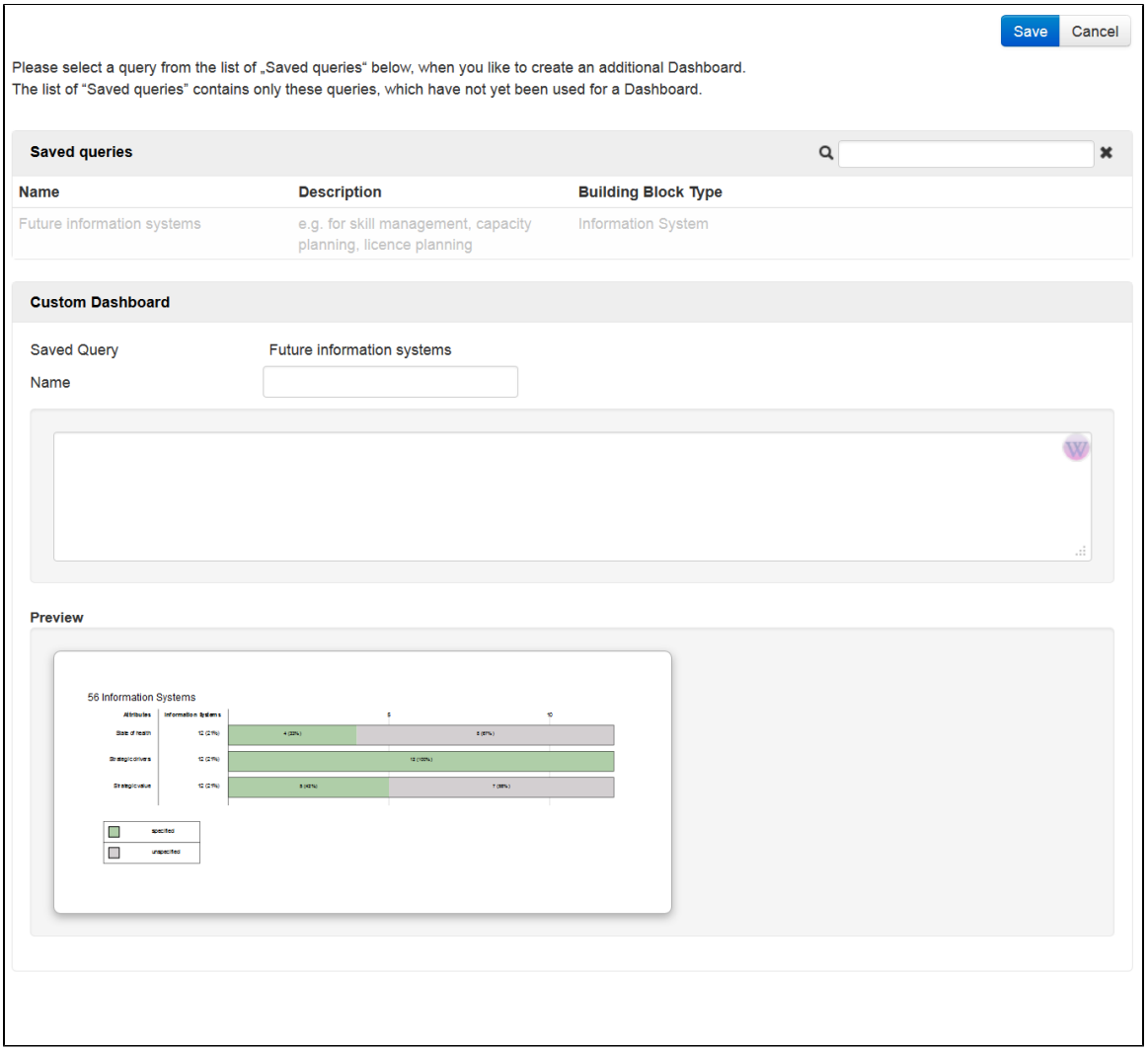
Create/Edit Custom Dashboard

With the Button "Create Dashboard" on the Custom Dashboard Overview page you can create a new dashboard. The following panel will show up.



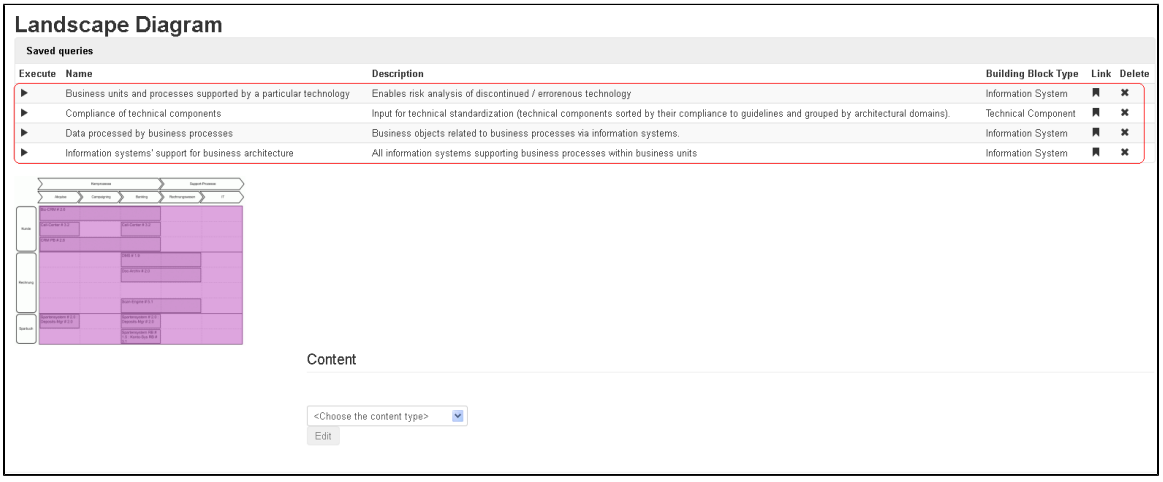
All saved queries (tabular reports) are listed for which a dashboard template exists. If you miss a saved query, please check if there exists a dashboard template for this query and check that the missing saved query is not already in use for another dashboard.

To create a custom Dashboard select a query from the list of „Saved Queries" and a "Dashboard Template". This will open a preview. You might enter a description for your Dashboard. Before you can save the dashboard, you have to provide a name for the Dashboard. By clicking on the Save-button, the dashboard is saved and is now available on the overview page.



Landscape Diagram

As for Information Flow Diagrams, iteraplan provides a predefined query for Landscape Diagrams. With a predefined query, iteraplan generates a matrix-view Landscape Diagram that presents all the assignments of Information Systems to Business Processes and Business Units. You can of course create and save queries on your own, if you have the necessary permissions (see Functional permissions).



Selection of predefined Landscape Diagram

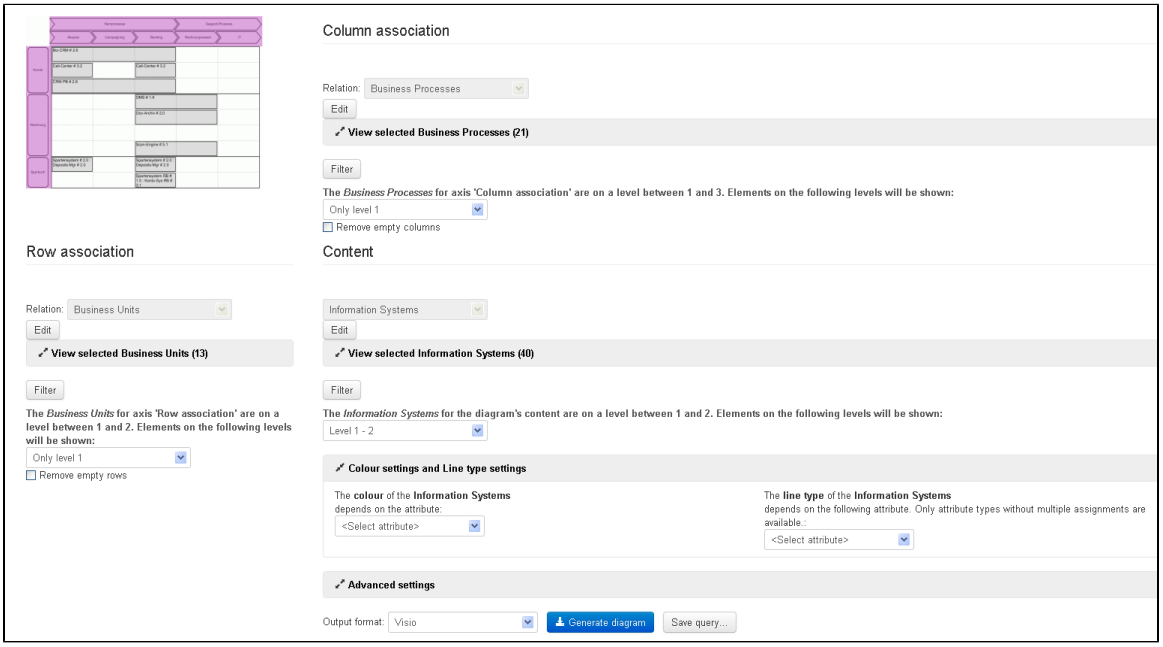
The excerpt of the screenshot below shows which Information System Releases are used in which Business Processes (X axis) by which Business Units (Y axis). The colour of the releases indicates their state of health; the line type indicates their complexity. The meaning of the different colours and line types (i.e., the Attributes they represent), are explained in a legend within the diagram. Due to calrity reasons, this

legend is not shown in the figure below.

Example section of the generated Landscape Diagram

Building Block types can be assigned in a flexible manner to rows, columns and the content of the diagram, enabling you to generate a variety of different views. You can also assign Attributes of the content elements to the rows or columns of the diagram, and map content Attributes to colours and line types in order to represent additional information.

This section takes you step-by-step through the process of generating a Landscape Diagram. After selecting the content and deciding what should be represented in the rows and columns, you can make changes to and refine your selection at any time.



Page for defining Landscape Diagrams

Selecting content

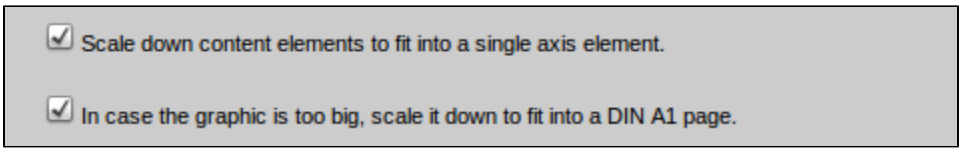
For the content of your Landscape Diagram, you can choose either Information System Releases or Technical Components. Once you have made this selection, iteraplan automatically selects all Information Systems or Technical Components which have the status *Current* and which are valid at present. You can restrict the scope of this selection by setting filters. To do this, click the **Filter** button: this opens a new query form where you can modify the selection. The query form for Landscape Diagrams is identical to the one used for Spreadsheet Reports (see Spreadsheet Reports). You can change the type of Building Block you wish to map to the Landscape Diagram by clicking the **Reset** button.

new in 3.0
You can now use color ranges for number attributes (for more information have a look at Portfolio Diagram)

Scaling Configurations

new in 2.9
Use the different scaling configurations of the Landscape Diagram to optimize the presentation of large amounts of data in a single graphic.

In the advanced settings of a Landscape Diagram (see Figure 3) you will find two scaling options. These are depicted in the picture below.



The first option, available in iteraplan since version 2.8, enables you to determine whether the content elements of a Landscape Diagram are scaled down (made thinner) so that their accumulated height (or width, if a vertical presentation is selected) equals the height of a single axis element. By default this option is enabled and results in a diagram as the one depicted in Figure 2. Alternately, if the option is disabled, the content elements have a fixed height and the axis elements are growing larger, so that each axis element can accommodate all of its content elements. Disabling this option can be useful, since it yields a much more readable graphic in case a given axis element has a lot of associated content elements.

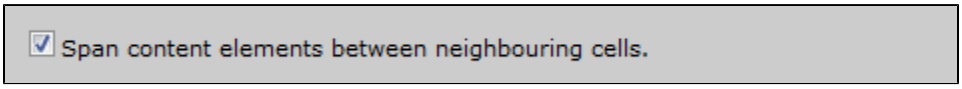
A Landscape Diagram, whose content exceeded the dimensions of a DIN A1 page, is as a default scaled down to fit into an A1 page (to optimize printing). For very large Landscape Diagrams this downscaling might render them unreadable. In this case use the new option (the second one in the screenshot above), which disables the global scaling and produces a diagram which occupies an arbitrary large page, while maintaining the original size of all graphic elements.

By using the different combinations of those two options you can optimize the produced diagram so that it can optimally fit your needs. Note that for both axes you can also remove the empty columns and rows, which enables you to additionally reduce the size of the resulting graphic.

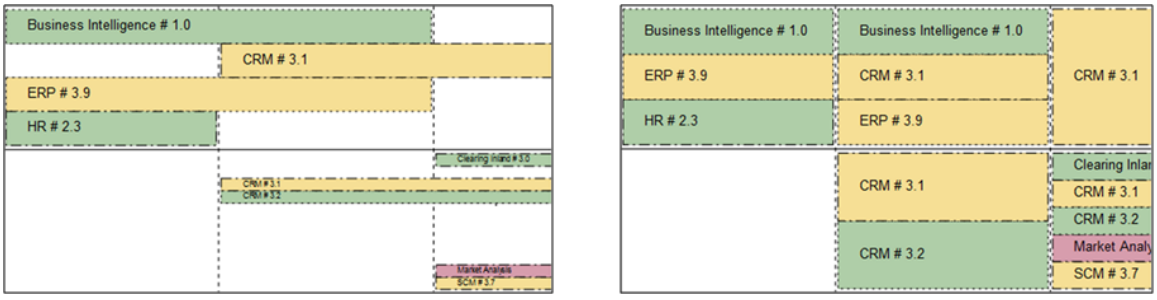
Content Spanning

new in 2.9
Use this option to improve the readability of a generated Landscape Diagram.

By default, the content elements of each cell are made as big as possible, so that they entirely fill it. This provides for maximal readability with respect to each particular content element. In the case of more complex Landscape Diagrams you might also want to visualize the relation between different axis elements implied by their respective associated content elements, i.e. which axis elements 'share' a given content element. This can be done by enabling the option, as depicted in the screen shot below.



When this option is enabled, a content element which occurs in neighbouring cells will be spanned over all those cells. Sometimes, this causes the content elements of a cell to be very small, while covering only a fraction of the cell. This is because some of the content elements are also spanned to (or referenced in) another cell, which has a greater number of content elements. The picture below depicts excerpts of a Landscape Diagram for which the spanning of content elements is enabled and a normally generated one.



A comparison between a Landscape Diagram for which the spanning of content elements between neighbouring cells has been enable d (left) and a standard Landscape Diagram (right).

Column association

Once you have selected the content of the Landscape Diagram, you can specify which elements you wish to map to the columns. You can have

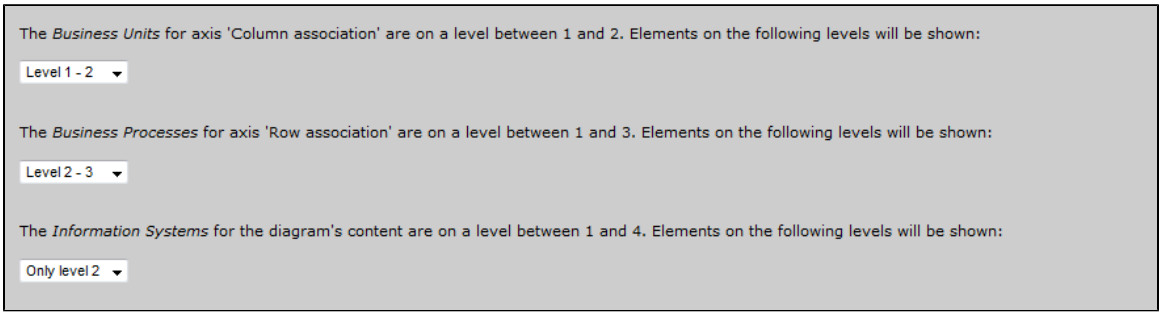
the columns represent either Building Block types or Enumeration Attributes of these blocks. Bear in mind that the scope of the Landscape Diagram is determined by the elements you choose for the rows and columns. The selected content elements will only be presented if they fall within the matrix drawn for the diagram. Once you have selected which Building Block Type you wish to display, iteraplan automatically selects all blocks of this type. If the type you have selected defines a timespan, only blocks will be selected which overlap the current date. If the block type has a status, the selected instances of this block type have the status *Current*. You can filter this preselection in the same way as for selecting Building Block types for the content of the diagram. You open the query for making filter settings by clicking the **Filter** button.

Once you have selected an attribute for one of the axes, iteraplan automatically selects all attribute values of this attribute. Bear in mind that you can only select attributes of the enumeration type. Click the **Reset** button to change the configuration which you have set.

Row association

Make the assignments for the rows in the same way as for columns.

Hierarchical Levels



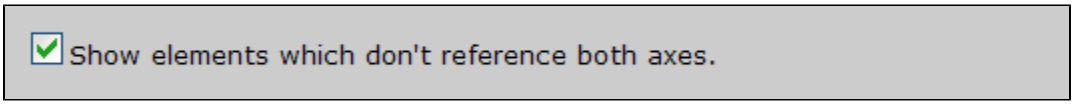
hierarchical levels selection

If Building Block Types you selected for axes have a hierarchical structure, you can choose which levels of this hierarchy are displayed in the diagram (see screenshot above). Elements of higher levels than selected for display are ignored, while the relations of elements of lower levels are aggregated to applicable higher-level elements. If for example an Information System has a relation to the Business Unit 'Treasury' which is a subordinate of top-level element 'Services', and you chose to display only level 1 elements on your Business Unit axis, you'll find this Information System under the 'Service' column (or row, depending on your axis assignment).

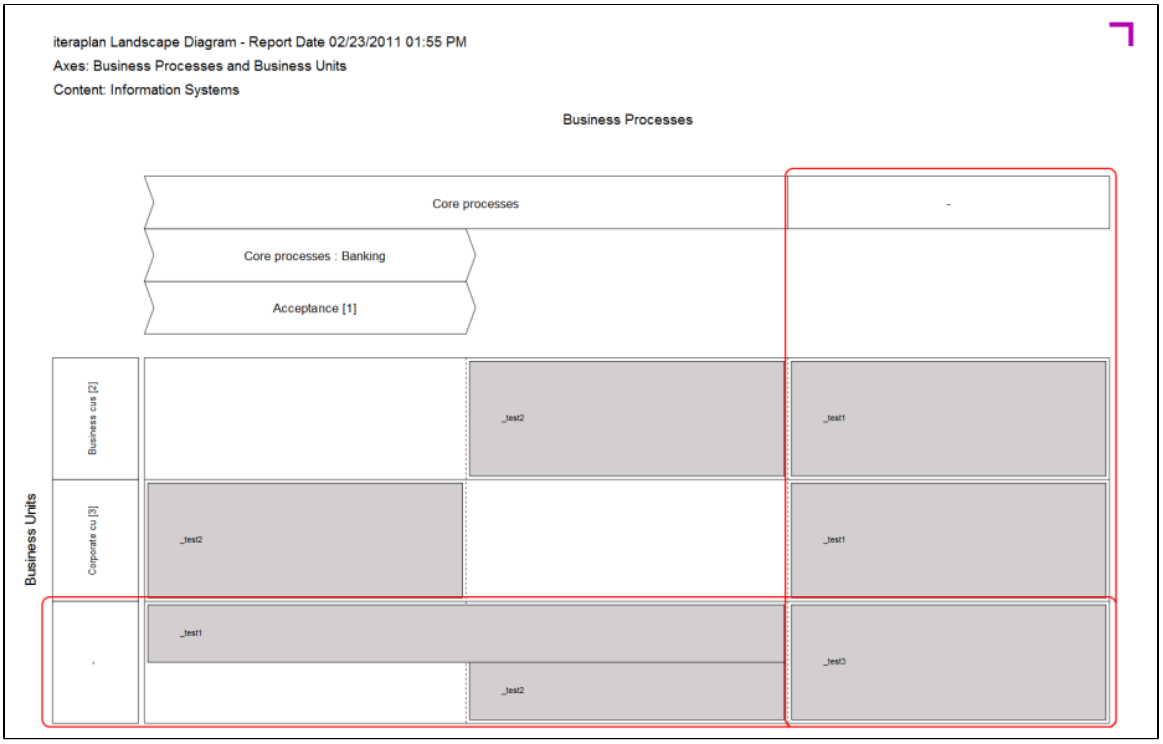
Content elements with hierarchical structure can now be displayed according to selected levels, too. Similar to axis elements, content elements of higher levels than selected are ignored, while elements of lower levels are represented in the diagram by their superordinate elements, where applicable.

Partially related content elements

Sometimes it is necessary to visualize content elements which have a relation to only one (or to neither) of the selected axes. In such cases you should check the box titled "Show elements which don't reference both axes" as it is shown below.



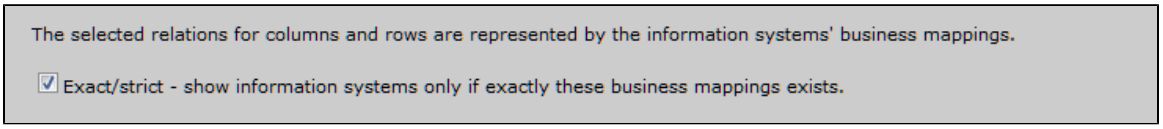
The "unrelated" content elements are displayed in the corresponding column and row, the exact location depending on whether they have a relation to either column or row or neither of them (see highlighted content elements in the iteraplanDocumentation303:graphic below).



display of partially related and unrelated elements - example

Business Mappings

Handling of Business Mapping in Landscape Diagrams has now become more flexible. By using the option shown below you can combine multiple Business Mappings for visualization of Information Systems.



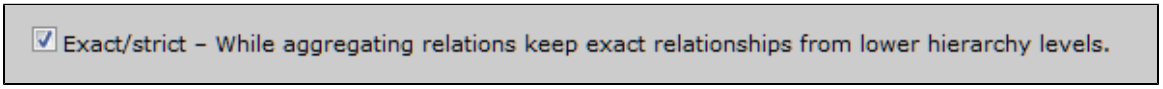
This option appears only if you selected Information Systems as content elementes and both rows and columns are assigned to one of the Business Mapping elements (Business Processes, Business Units, Products).

- If this option is selected, you will get an exact visualisation of all Business Mappings. This means an Information System is only placed into the appropriate cell on the grid; if it has **one** Business Mapping containing **both** the corresponding row and column elements.
- If this option is unselected, an Information System is placed into the appropriate cell of the grid if it has at least one Business Mapping containing the corresponding row element and at least one (not necessarily the same) containing the corresponding column element.

Example: Let's assume Business Mappings (IS#1, BP X, - , -) and (IS#1, -, BU A, -) and a Landscape Diagram with Business Processes as columns and Business Units as rows. If you select the option above; IS#1 won't appear in the result diagram. Only with this option deselected will iteraplan combine the Business Mappings and put IS#1 into the result diagram.

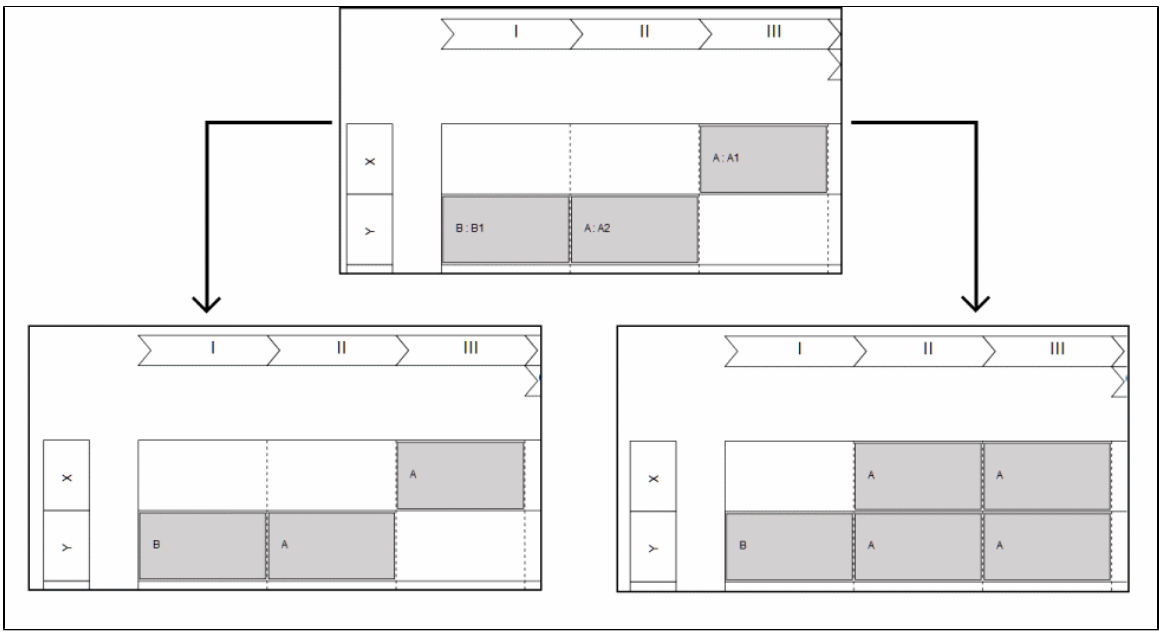
You can also combine this option with the one for iteraplanDocumentation303:partially related content elements to visualize Information Systems in Business Mappings containing unspecified elemenents.

Exact/Strict Relations



new in 3.0
The option to evaluate the relations of content elements to axis elements in a more strict way is now also available for non-business-mapping relations and affects the behaviour of the diagram when you choose to not display lower hierarchical levels (see also Hierarchical Levels).
When not using the "exact/strict relations"-evaluation, the relations of not-displayed elements on lower hierarchical levels are aggregated to the higher-level elements in an additive manner, treating the higher level elements as if they'd possess the accumulated

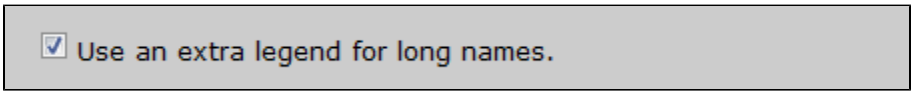
relations of their sub-elements. In the example below on the right side you can see the effect of that: Information System "A" is treated like having the relations to the top axis' elements "II" and "III" and to the side axis' elemente "X" and "Y" of its children added together. Using "exact/strict relations" on the other hand results in a diagram like on the left side of the example, placing the higher-level element only in places of the grid where actual sub-elements exist.



Example fo reducing the original diagram to have only top-level elements shown. On the left using the "exact/strict relations" option, on the right without.

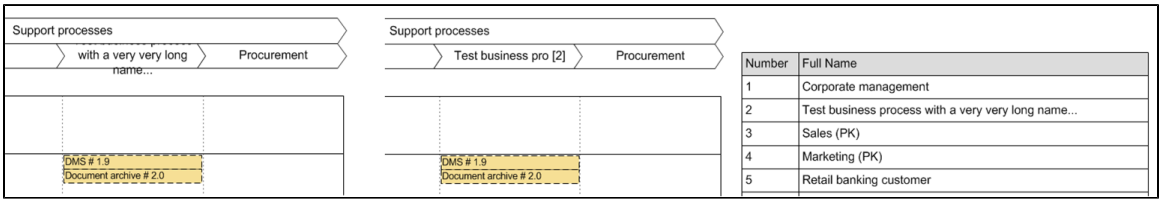
Legend for long names

You can choose whether you want to use a legend for long names or not.



Using the legend, the name of Building Blocks is cut if it does not fit into its box (at least four characters remain).

Not using the legend, the hole name of Building Blocks is used. This may overflow its box (see graphics below).



not using extra legend **using extra legend**

new in 2.9
To improve the readability of diagrams in visio format the boxes of the last level of hierarchical axis have been made higher. This way the names on this level fit into the corresponding box completely.

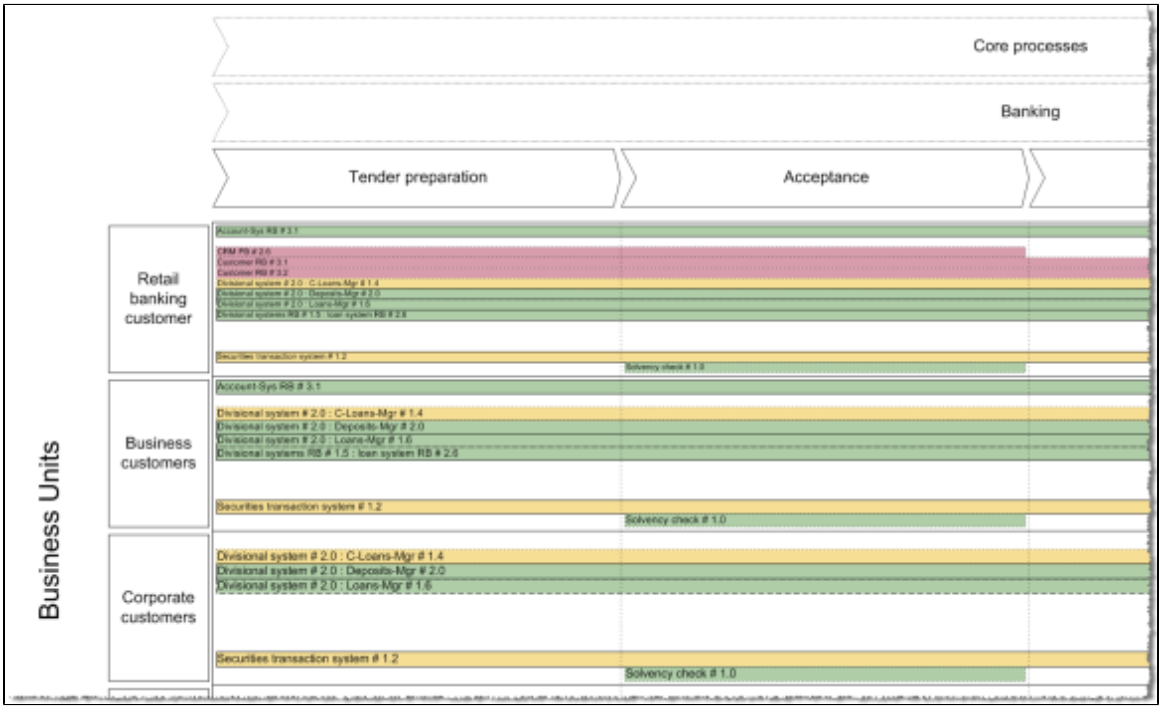
Generating the diagram

Before generating the diagram, you can select two additional attributes for the Building Block Type which will form the diagram content: colour coding and line type. These settings can only be made for Enumeration Attributes which do not have the multiple-value option selected.

The alignment of the content elements can be either along the columns or along the rows. The use of blocks of the same type across multiple columns or rows can then be presented as a bar.

If the row or column elements have a hierarchical structure, you can also indicate how many levels you wish to include in the diagram. Bear in mind that elements assigned only to the upper levels of the hierarchy will not be shown if you elect to diagram just lower levels. However, if you choose to omit the lower levels, elements on these levels will automatically be mapped to the lowest hierarchical level actually included in the diagram.

The diagram is generated with the **Generate diagram** button (you must first have indicated what content the diagram is to have, and what is to appear in the rows and columns). The following screenshot shows an example of a Landscape Diagram. Its content are Information Systems; Business Processes are mapped to the columns and Business Objects to the rows. The values of two Information System Attributes (*state of health* and *complexity*) are represented by the colour and line type of the content element.

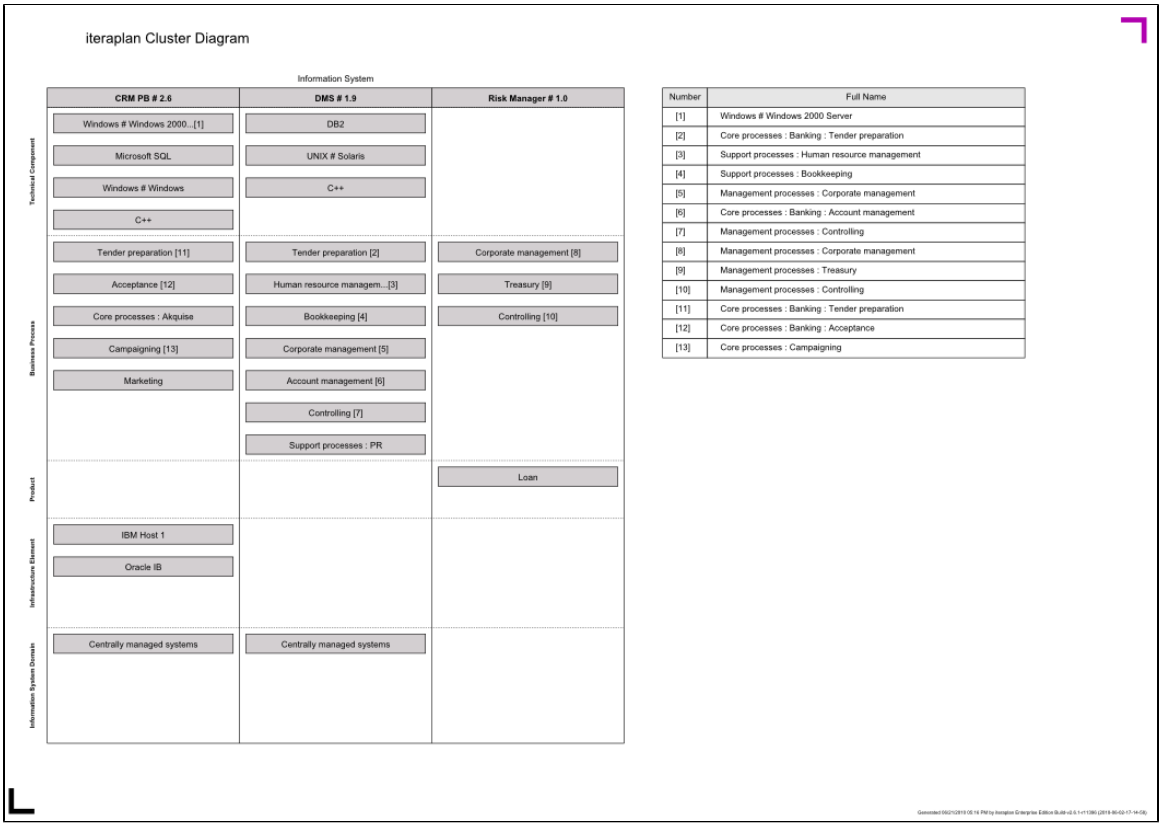


Example Landscape Diagram (three levels for columns, one level for rows, horizontal alignment)

- For tips on editing a Visio Landscape Diagram, please see our FAQs.

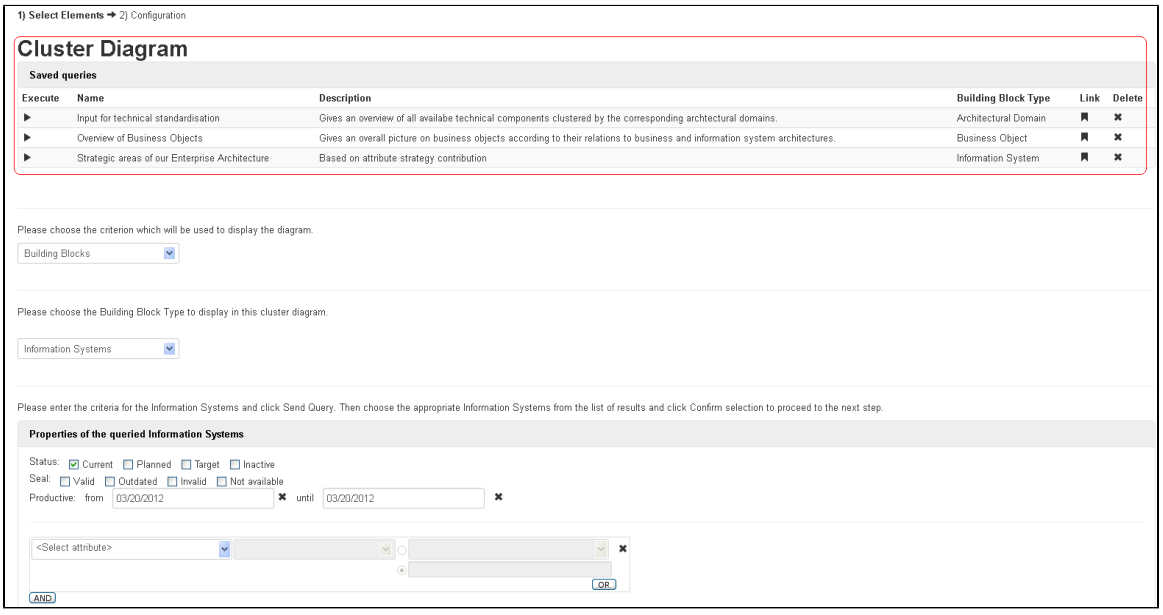
Cluster diagram

Another type of diagram you can generate is the cluster diagram. In iteraplan there are two different ways how you can group building blocks in a cluster diagram. First one is to cluster elements by a specific type of building block. In this case all building blocks with a relationship to the chosen one are gathered in one cluster. The second way is to group building blocks by values of a chosen attribute. All building blocks assigned with the same attribute value, or in case of number attributes, where the attribute value falls into the same range (see number attribute ranges), are then grouped in the same cluster. The screenshot below illustrates a simple cluster diagram of information systems listing all infrastructure elements, projects, technical components, information system domains and business objects related to each information system.



Simple cluster diagram

Analog to the previous diagram types you can either generate cluster diagrams using predefined queries or create and save your own queries to your specific needs.



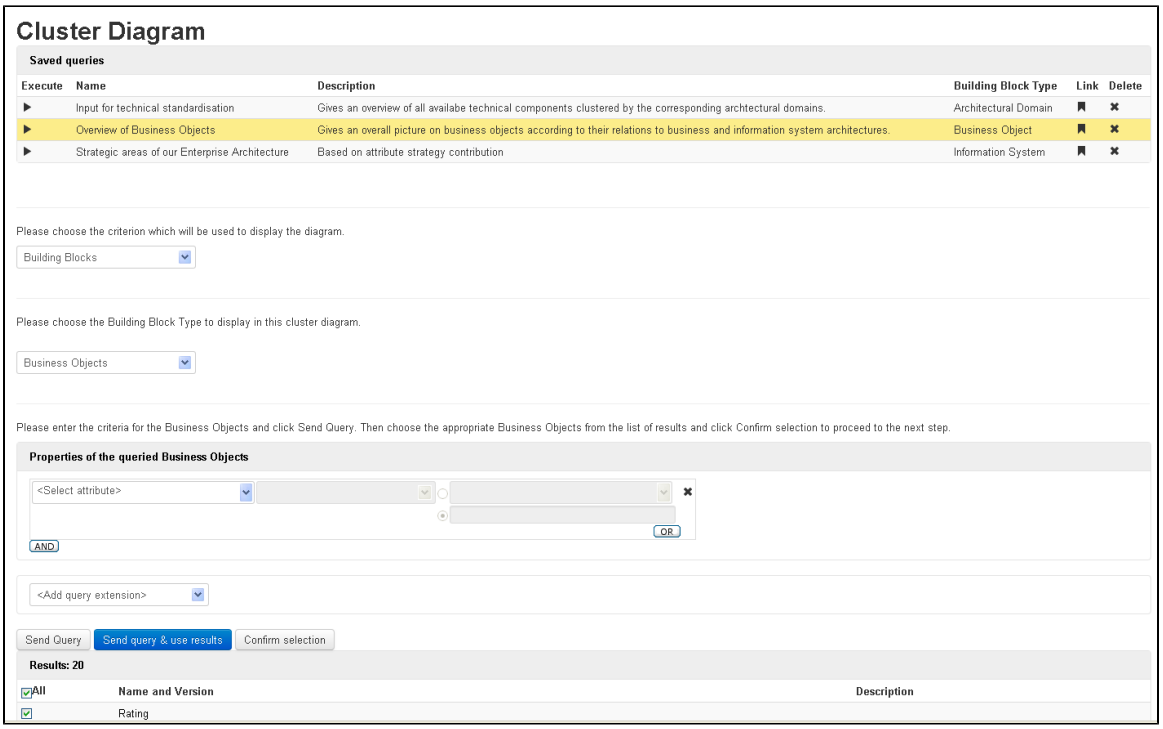
Selecting the predefined cluster diagram

This section explains the procedure for generating a cluster diagram. Similar to the masterplan diagram it is a three-step operation:

- Select the type of building block or attribute as cluster type for the diagram
- Configure your query and query extensions, or select the attribute values/ranges to incorporate, depending on whether it's a building block based or attribute based diagram.
- Configure presentation options

Here is the first step how to start. At the top of the page you can select if the diagram should be clustered by building blocks or attributes. Then

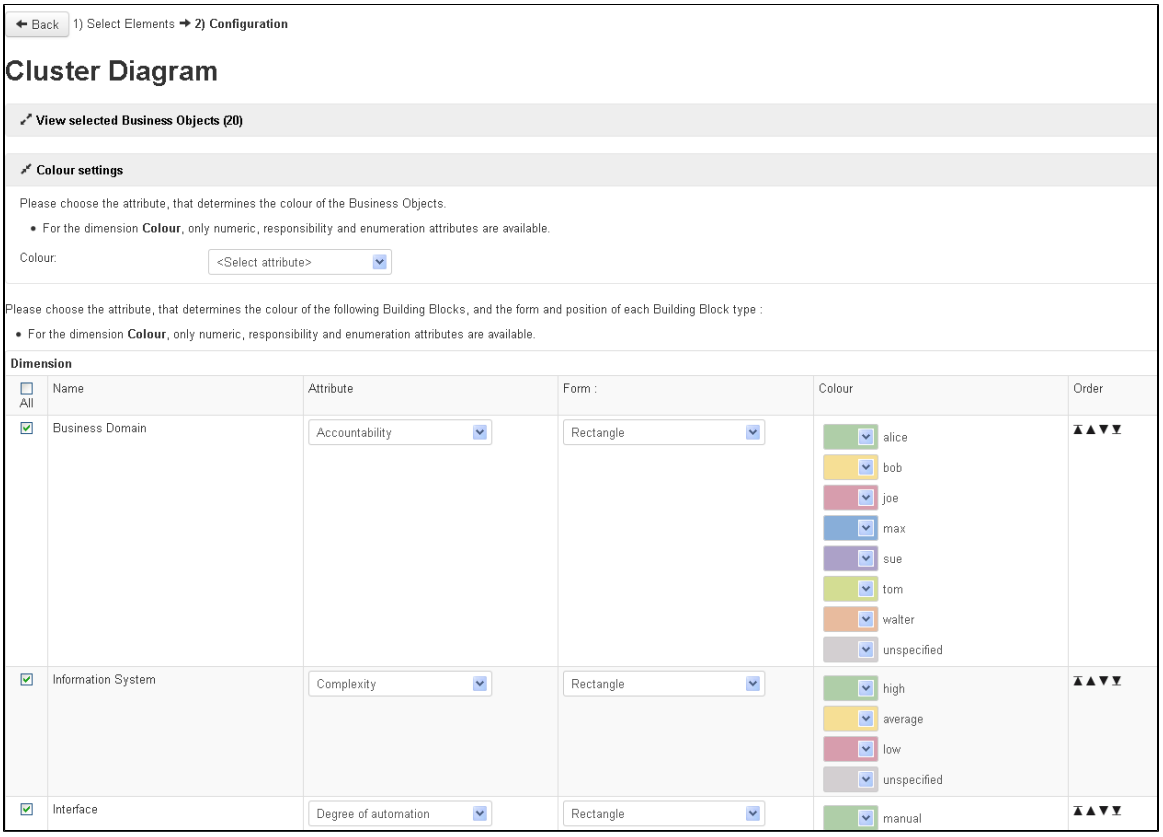
choose the desired type of building block or attribute you are interested in.



Query of information systems for the cluster diagram

The query form below is the same as the one used for spreadsheet reports (see Spreadsheet Reports) and important to refine the selection of elements for the diagram. After defining your query and query extensions click on **Send query** to receive the result set. When clustering by an attribute was chosen, there will be a form to select the attribute values or ranges which should be incorporated in the diagram. As soon as the list of results contains all desired elements, click **Confirm selection** to open the configuration page for the diagram.

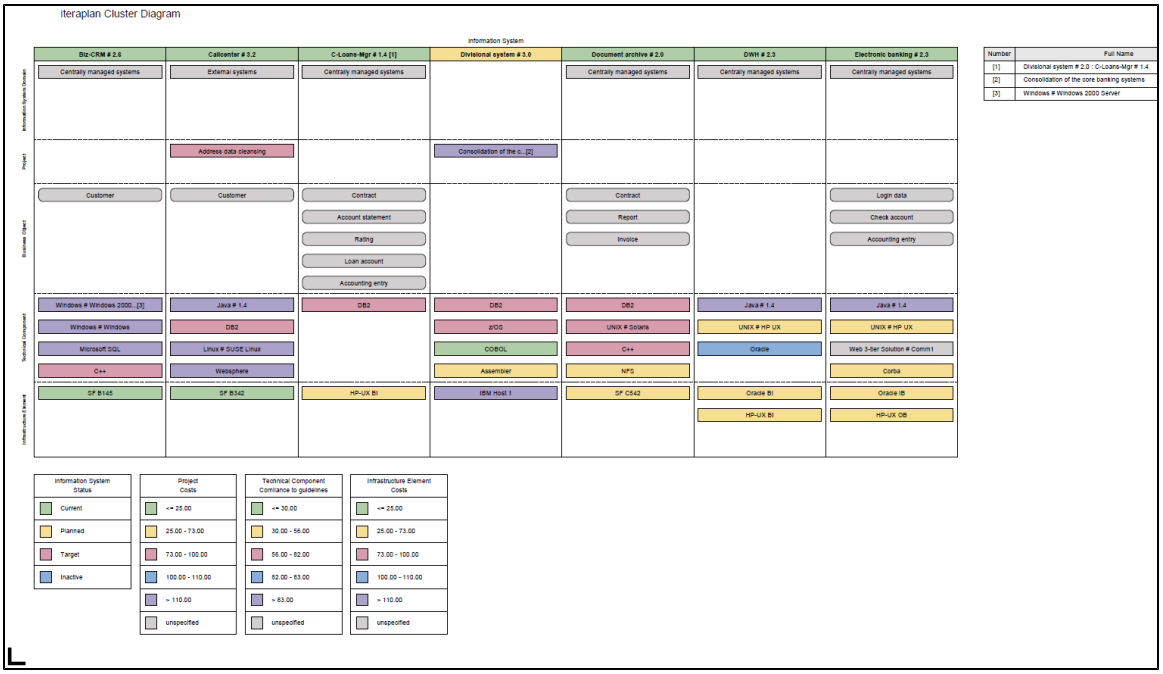
If you click **Send Query & use results** you go automatically to configuration step 2 with all building blocks matching your criteria selected.



Configuring the cluster diagram

The configuration page presents several options how the elements of a cluster should appear in the diagram. You can select how many hierarchical levels should be considered in the cluster. You can also choose one attribute (only numeric, responsibility or enumeration attributes available) to color-code the different clusters. Through the check mark below, you can specify whether each content element of the diagram should have the same position in every cell of the cluster (Swimlanes). Using this option provides a clearer arrangement of the elements, but requires more space. The list underneath shows all types of building blocks having a relationship to the one selected in step 1, as well as its attributes. In case of an attribute-based cluster diagram the list shows all building blocks for which the selected attribute is available. By adding or removing check marks you can define, which building blocks should appear. Additionally you have several options to configure the layout of each building block belonging to the cluster like color, shape or their order.

Finally choose your appropriate output format and click **Generate diagram** to generate your cluster diagram. The iteraplanDocumentation303:screenshot above shows a cluster diagram of information systems configured with different shapes and colors. Below the diagram you can find four different keys. The first one represents the state of health for each information system colored in the head of each cluster. The keys of technical components, infrastructure elements and projects indicate which attribute values are represented by each building block. In case the amount of possible attribute values (minus the default for unspecified value) in any of these keys exceeds 7, to save space only the values actually appearing in the diagram are listed in the according keys. Otherwise all possible values will be displayed.



Example section of a cluster diagram

new in 2.8
You can choose wether you want to use an extra legend for long names (for more information have a look at Portfolio diagram)

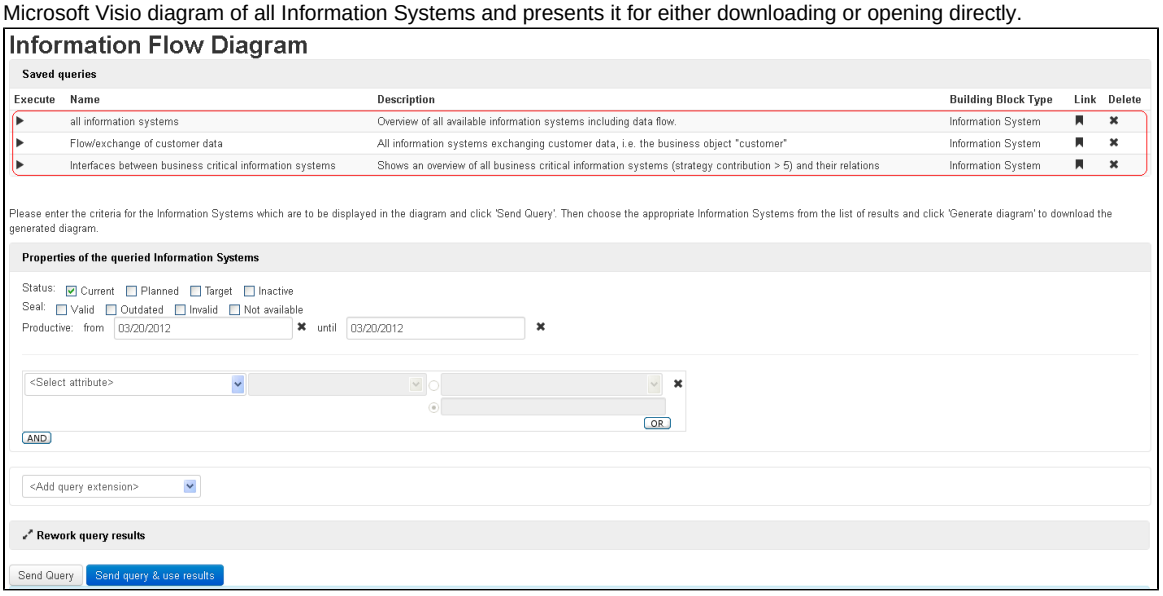
new in 3.0
You can now use color ranges for number attributes (for more information have a look at Portfolio Diagram)

Information Flow Diagrams

The *Information Flow Diagram* represents the Interfaces between Information System Releases, and the flow of Business Objects via these Interfaces between various releases.

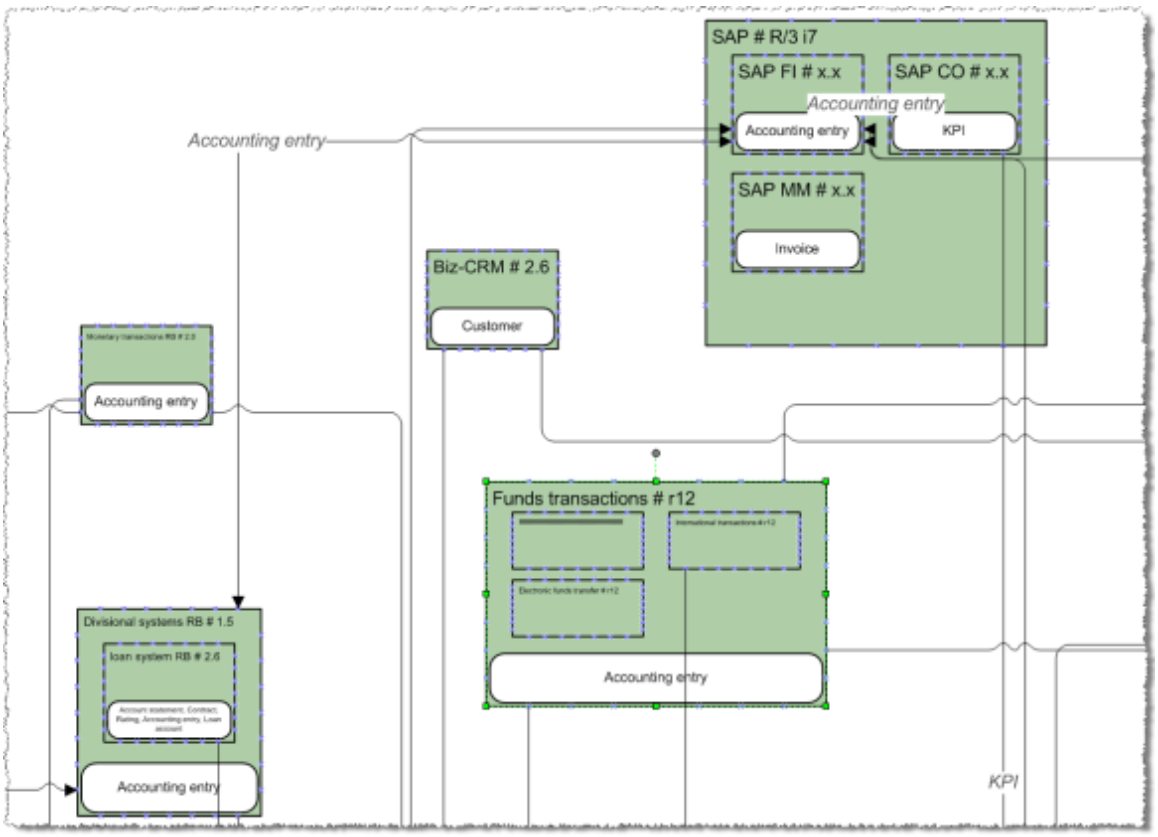
Saved Queries

To generate a diagram, you can either use a predefined query or define a query configuration of your own. In this example, iteraplan generates a Microsoft Visio diagram of all Information Systems and presents it for either downloading or opening directly.



Loading a predefined query might fail, if the user has no privileges on Business Objects.

Selecting predefined Information Flow Diagrams



Example section of an Information Flow Diagram

The boxes in the Information Flow Diagram represent Information System Releases, as illustrated above. You can choose coloring of the Information System Releases by Status (see Step 2). In this example, the *current* status (as-is) is indicated by green.

> In Visio diagrams, the line style of the Information System's boxes indicates their status: "inactive":none; "current":solid; "planned":dashed; "target":dotted

The Business Objects processed by the Information Systems are shown inside the release as boxes with rounded corners, used Information System Releases are similarly shown in light grey boxes. Sub-Information System Releases are presented inside their super ordinate releases.

The arrowhead connections between information system releases represent the interfaces. The annotation states the type of business object which they transport.

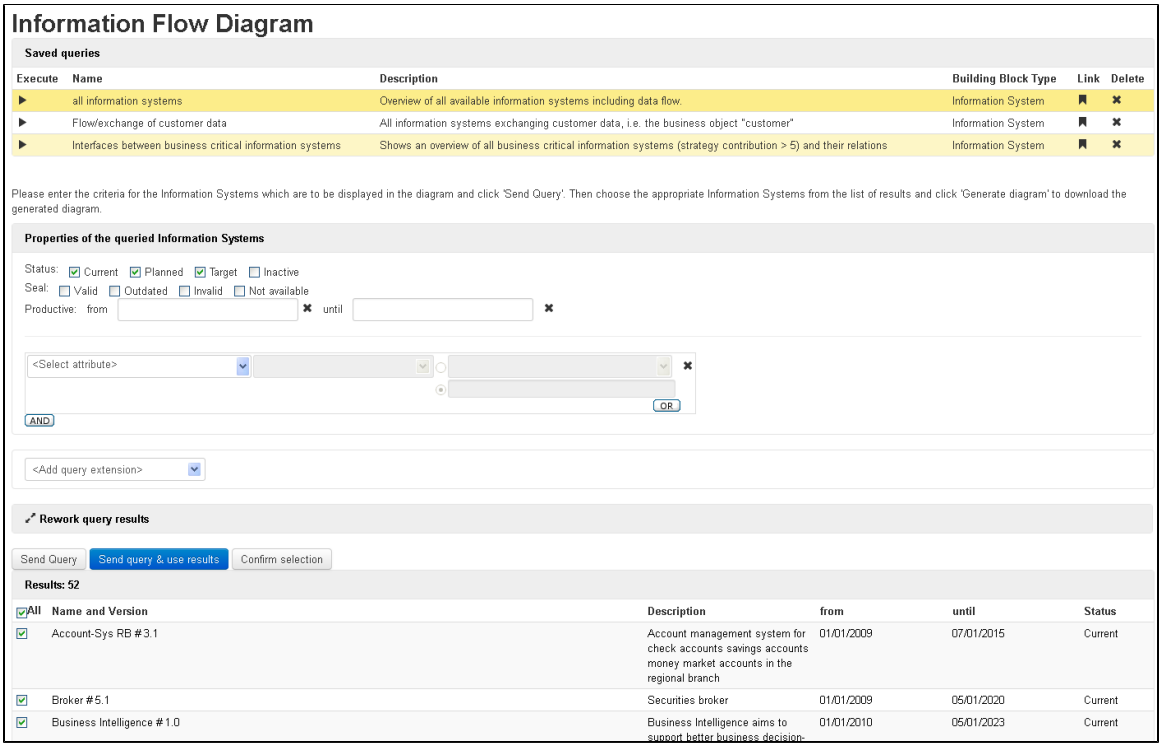
Step 1: Query Generation

To generate your own query, select the required Information System Releases as follows (see screenshot below):

- Select the properties you wish to use, e.g. status and productivity time span
- Define the conditions (see Formulating queries);
- Define any query extensions you wish to use (see Formulating queries).

Click **Send Query** to display the results. iteraplan lists the Information System Releases which match your criteria. Before you generate the diagram, you can filter the set of Information Systems by adding or removing the check marks as appropriate in the results set. You can also refine the query using the **Add query extension** options, as explained in Formulating queries. Then, only selected Information System Releases are taken into consideration for the Information Flow Diagram you are generating.

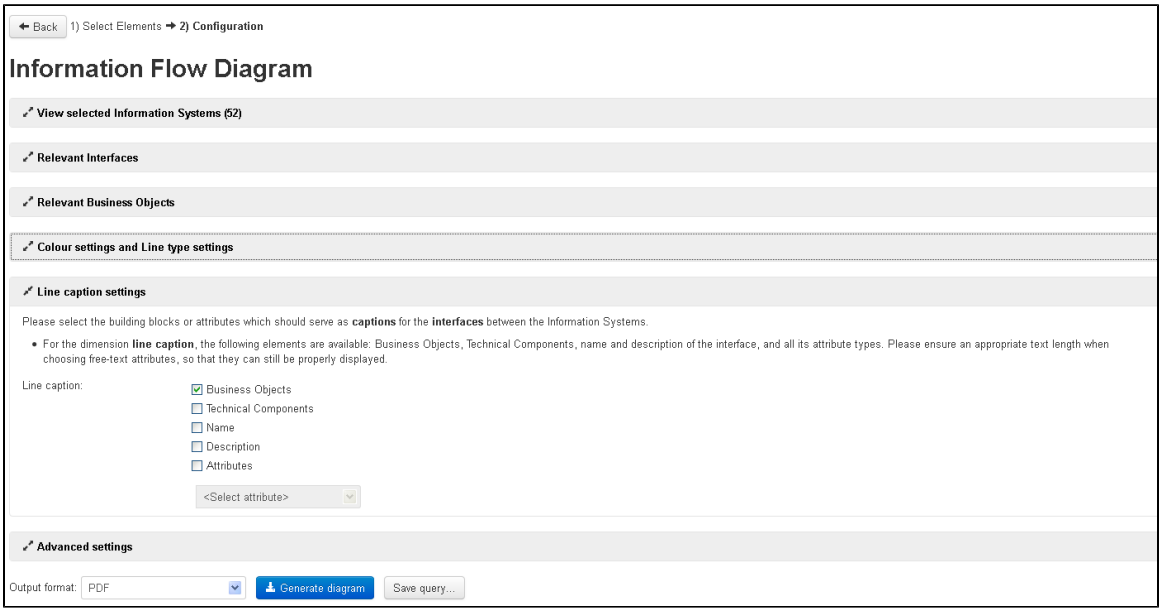
> If you click **Send Query & use results** you go automatically to configuration step 2 with all Information System Releases matching your criteria selected.



Example selection for the Information Flow Diagram

Step 2: Configuration

When you click **Confirm selection**, iteraplan opens the configuration page.



Second step – diagram configuration

Here you have a variety of options which enable you to further refine the information to be included in the diagram. The first two check box options specify whether the Information Systems should be depicted with their assigned Business Objects and/or their used Information Systems. By disabling the check boxes one can reduce the complexity of the diagram.

The next option, which also focuses on the Information Systems, is the assignment of a colour coding to one of a set of possible attributes. Using this option provides a mechanism for encoding further information into the diagram. Every Information System is colored in accordance with its assigned value with respect to the selected attribute. If you select an attribute which can have assigned multiple values, the Information Systems are accordingly colored in multiple values as well.

Multi-colored information systems

- This doesn't work with Visio output format, which chooses the first of the attribute values for coloring the Information System, when multiple values are assigned.

The same can also be done for the Interfaces between Information Systems by specifying an Attribute whose values are then depicted through different line patterns.

You can further select a relation that is to be used for the captions of the Interfaces. By default the transported Business Objects are depicted. Alternatively, you can choose between the assigned Technical Components, the description of the corresponding Interface or a further user-defined attribute.

The last bit of configuration concerns the presentation of the diagram. Herein you can choose the output format of the diagram, for example Microsoft Visio or Adobe PDF, and select a preferred layout for the Information Systems. You can choose between the standard Spring-Force Layout, the KK Force Layout or the Circle Layout. The former and the intermediate are very similar and can be used equally. Each element in the graph is displayed as far as possible from the others, so there are as few crossing edges as possible. The circle layout, in contrast, distributes the elements (nodes) in a circle.

You can choose whether you want to use an extra legend for long names. Color ranges for number attributes are also available. For more information have a look at Portfolio Diagram.

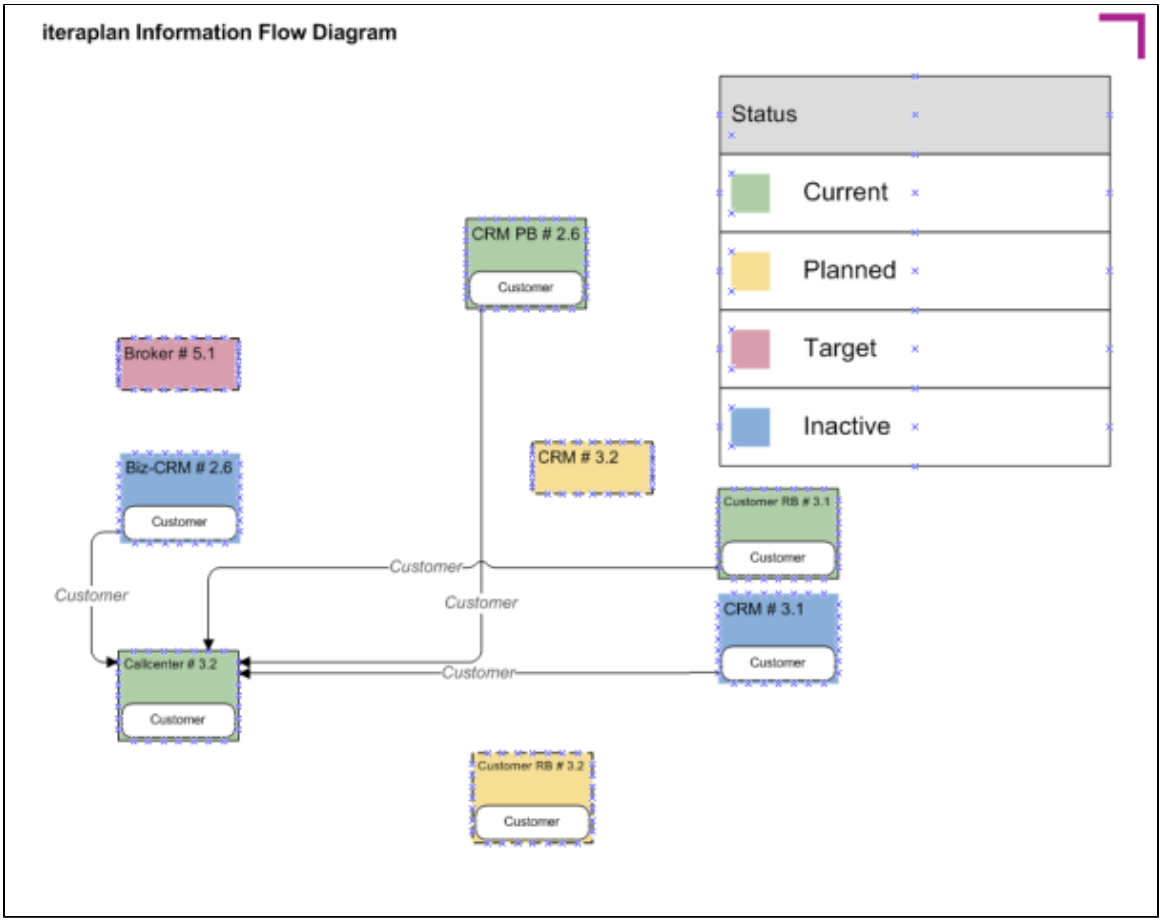
Automated Layouts with random component

- The layout algorithms have a random part and therefore Information Flow Diagrams will always be different, even if the data is exactly the same.
- Export in Microsoft Visio format only supports Standard Spring-Force Layout. If you choose one of the other layouts, it will still render the graph as Standard Spring-Force.

Filtering Interfaces

You can choose which interfaces, should be rendered. In some cases the number of interfaces is huge, because of the number of information system. To show only important interfaces, click on the "Filter Interfaces"-Button, which could be found in Advanced Settings. There could be defined a query, which filters your interfaces. After that, a Button labelled "Reset" appears, if you want undo your filter.

You can also save the query together with all its settings, and retrieve it later from the list of saved queries (see Saving and loading Visio diagrams). The **Generate diagram** button generates the Information Flow Diagram with the selected Information System Releases in chosen format and presents it for you to download or to open directly. The figure below shows an Information Flow Diagram generated with iteraplan. Here you can see eight Information System Releases together with their Interfaces and the assigned Business Objects. As indicated by the color of each box, some Information System Releases in this diagram have the status *Current*, while others are *Planned*, *Inactive*, or in status *Target*. Boxes with rounded corners within an Information System Release's box depict the Business Objects assigned to the Information System Release. The lines respectively arrows between Information System Releases represent Interfaces between these systems. Arrow labels and flow direction indicate which Business Objects are transported through an Interface and in which direction.

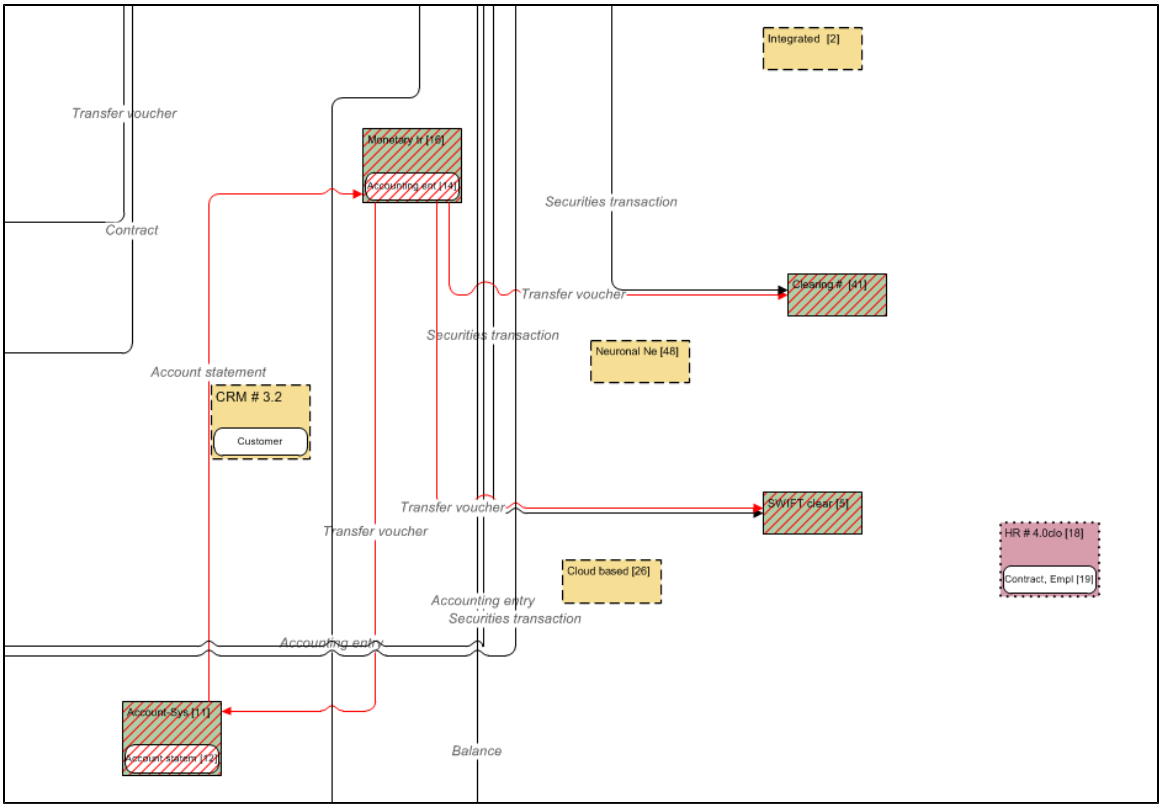


Example of a generated Information Flow Diagram with key

To reduce the necessary generation time of the Information Flow Diagram (and make some very complex diagrams possible on limited memory environments), there is a limit of Interfaces per Information Systems up to which orthogonal edge routing is done for SVG (and the derived JPEG, PNG, PDF formats). By default it is set to 12. This means, if an Information System has assigned more than 12 Interfaces, the edge routing will be simplified for this Information Flow Diagram. You may change this limit by editing the property "maximum.export.svg.interfaces.for.informationsystem" within the iteraplan.properties file. You can find the properties file in <*project_home*>iteraplan\apache-tomcat-6.0.20\webapps\iteraplan\WEB-INF\classes.

Impact Analysis

available in the Enterprise Edition
If you export your Information Flow Diagram in Visio format, you can right-click any Information System on the diagram and select "Mark associated Elements of this IS". Each Information System connected to the clicked one (by one or a series of outgoing Interfaces) and the selected one, will be highlighted to represent the impact of changes to the selected information system. Related interfaces will be highlighted, too.
You can repeat this procedure with several Information Systems without removing the highlighting of the ones before.
Right-clicking any Information System on the diagram and selecting "Reset all impact highlightings" will remove all highlightings.



Example of the Impact Analysis highlighting. Highlighted Information Systems and Interfaces in red.

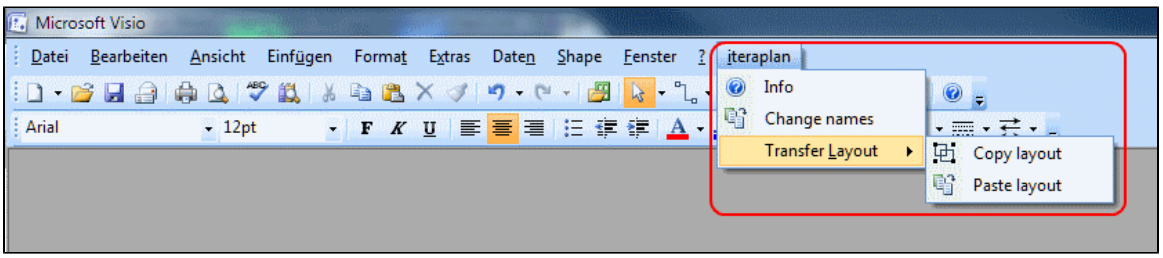
Copy Layout from existing Information Flow Diagram

available in the Enterprise Edition

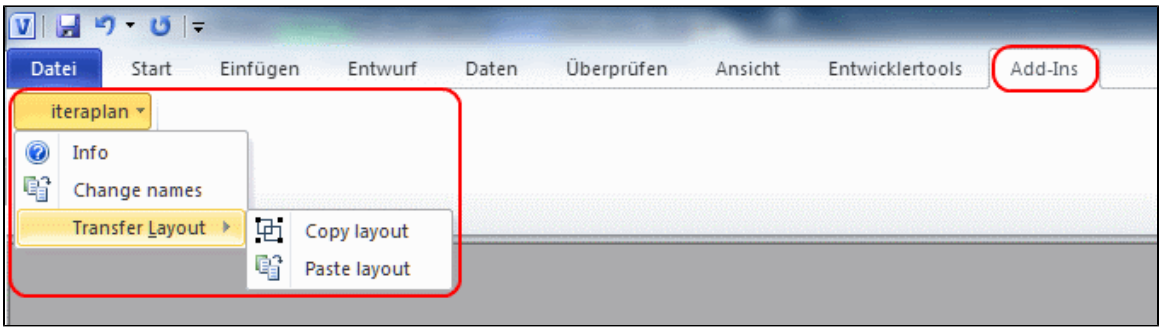
It is now possible to copy the positions of Information Systems and Interfaces from one Information Flow Diagram to another with help of a Visio stencil set, provided for users of the Enterprise Edition with the file iteraplan_layout.vss.

To use the feature, proceed as follows:

- First open two Information Flow Diagrams in Visio, one with the layout you want to copy, the other one as target to apply the copied layout.
- Now open iteraplan_layout.vss as stencil. In Visio 2007 you do this by clicking File->Shapes->My Shapes (if you copied the stencil set into your MyShapes folder) or by clicking File->Shapes->Open Stencil and opening iteraplan_layout.vss from the folder it is saved in. In Visio 2010 you'll find 'My Shapes' and 'Open Stencil' in the Shapes-window under 'More Shapes'.
- After you opened the stencil set, a new menu entry should appear, see screenshots below.
- To copy an Information Flow Diagram's layout, make sure the window of the according diagram is active, then click 'Copy layout' as seen on the screenshots below.
- To apply a copied layout to a diagram, make sure the window of this diagram is active, then click 'Paste layout'.



Added menu entry in Microsoft Visio 2007



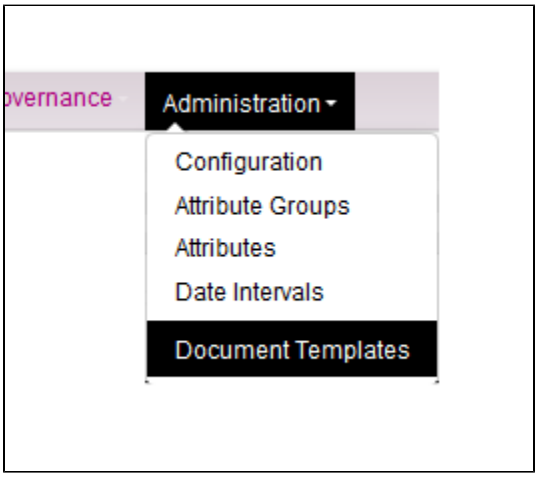
Added menu entry in Microsoft Visio 2010

Using an existing Information Flow Diagram as template (Enterprise Edition)

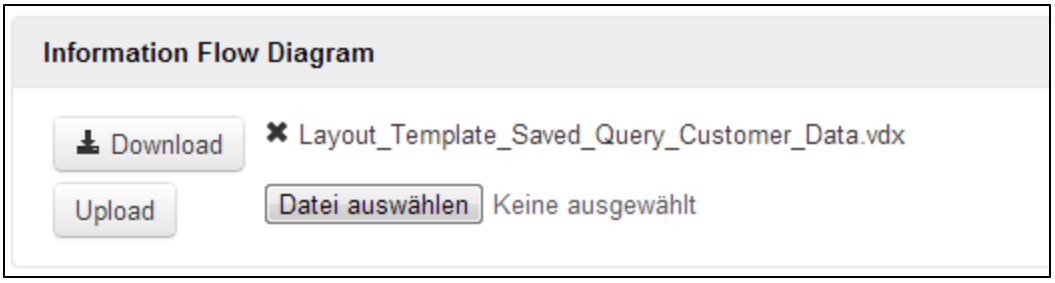
You can upload an existing Information Flow Diagram to iteraplan and use this diagram as a template for the layout of a newly created Information Flow Diagram.

If you choose an uploaded diagram as layout template for a new diagram, all Information System Releases will be placed on the same spot as in the template, if they are entailed in the template.

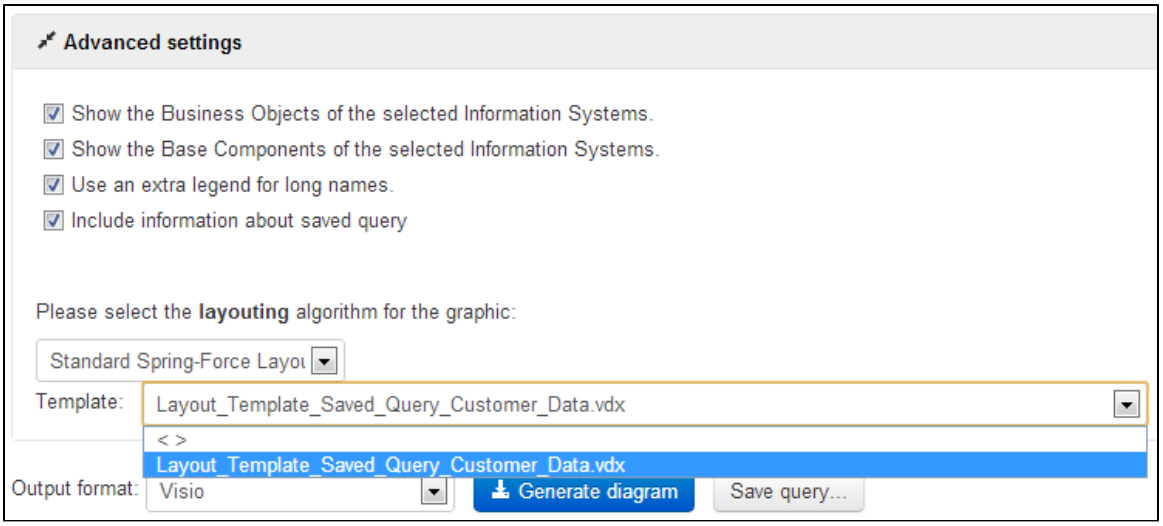
To upload a layout template, please choose "Administration" -> "Document Templates" from the upper menu.



In the section "Information Flow" you can upload or delete template files as necessary.



After either loading a saved Information Flow Diagram or creating a new configuration you can choose the template in the "Advanced", just above the "Generate Diagram" button.

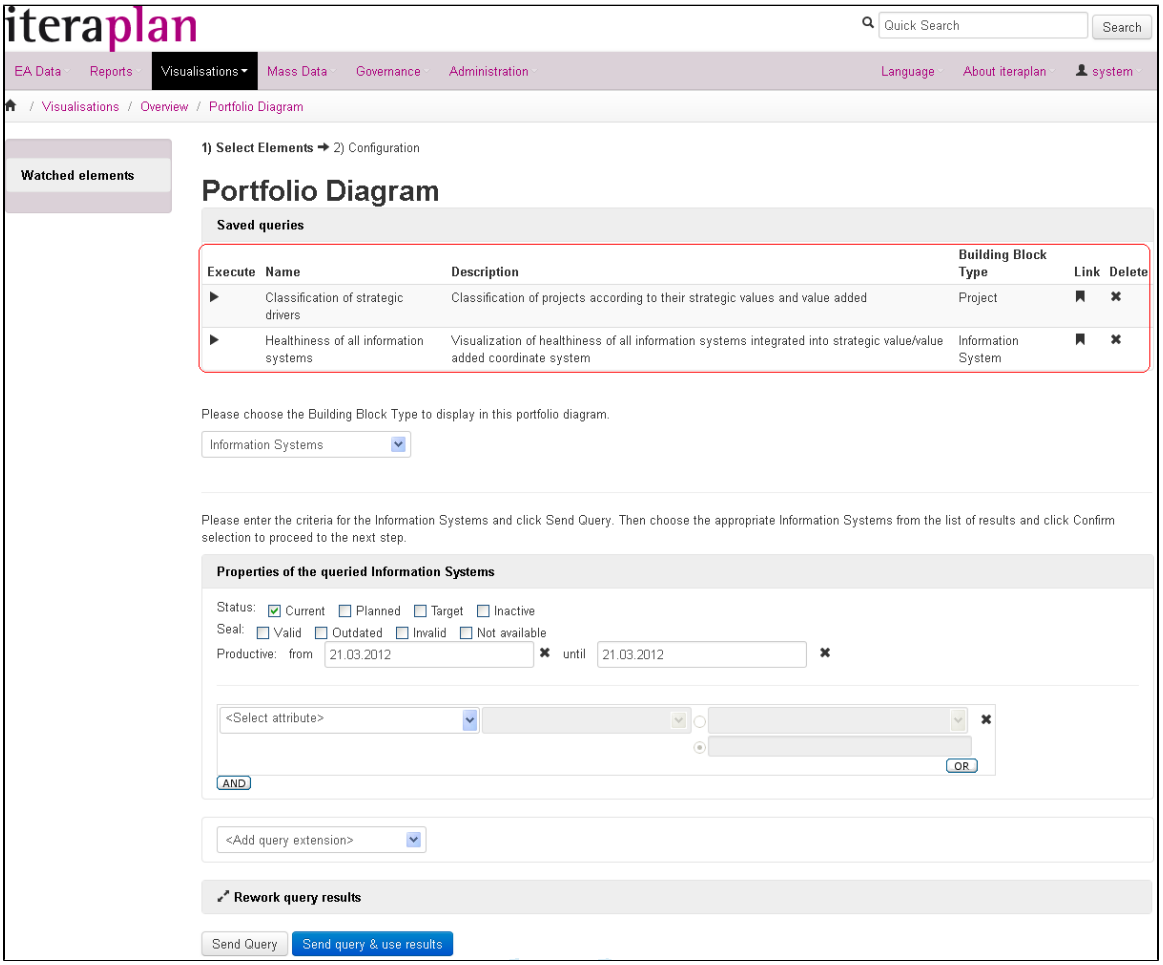


Portfolio Diagram

Portfolio graphics assist strategic decision-making processes in the enterprise – for instance, they can provide valuable input for deciding which Information Systems to decommission. A portfolio Diagram presents up to four different attributes of Building Blocks of a selected type:

- Horizontal positioning (along the X axis)
- Vertical positioning (along the Y axis)
- Circle size (circle)
- Circle colour (colour).

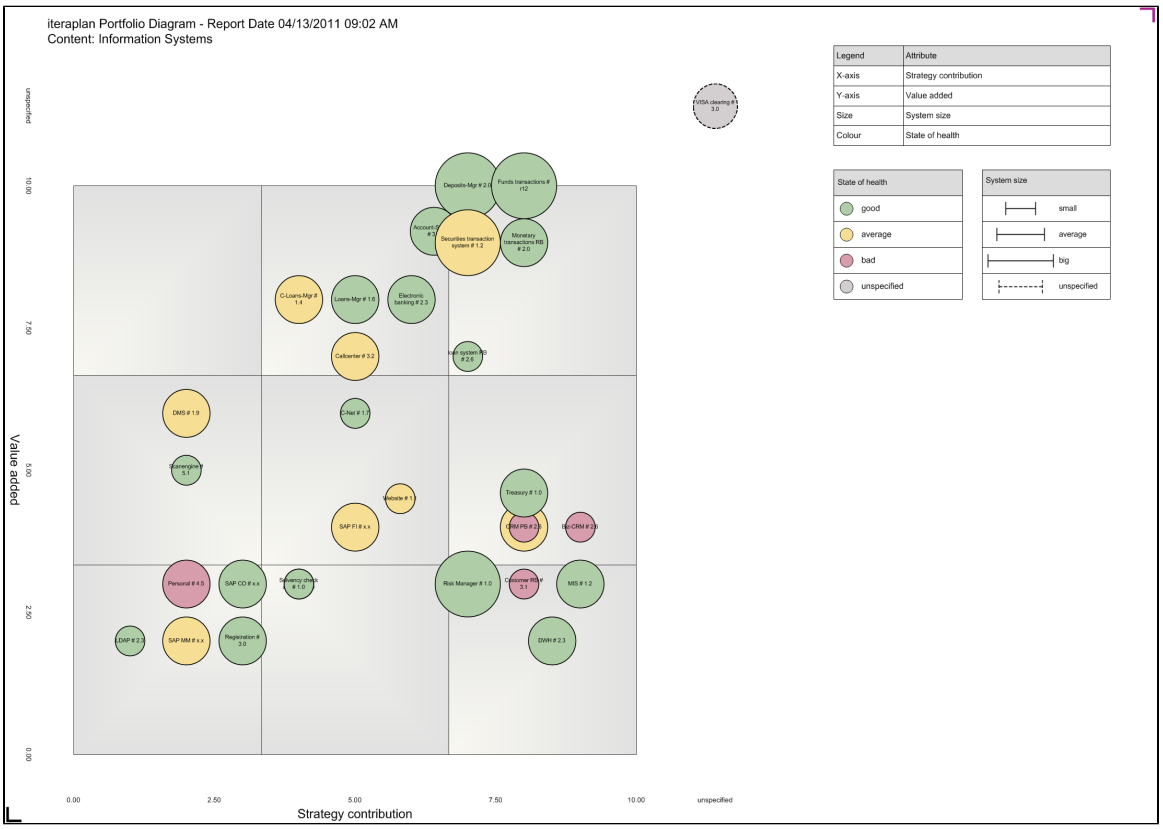
You can generate Portfolio Diagrams using a predefined query.



Selecting the predefined Portfolio Diagram

The screenshot below shows the result. The Information Systems are distributed between the quadrants in accordance with their strategy contributions and value-added. The size of the circles represents the size of the Information Systems and the colour indicates the state of health of the system in question (see key).

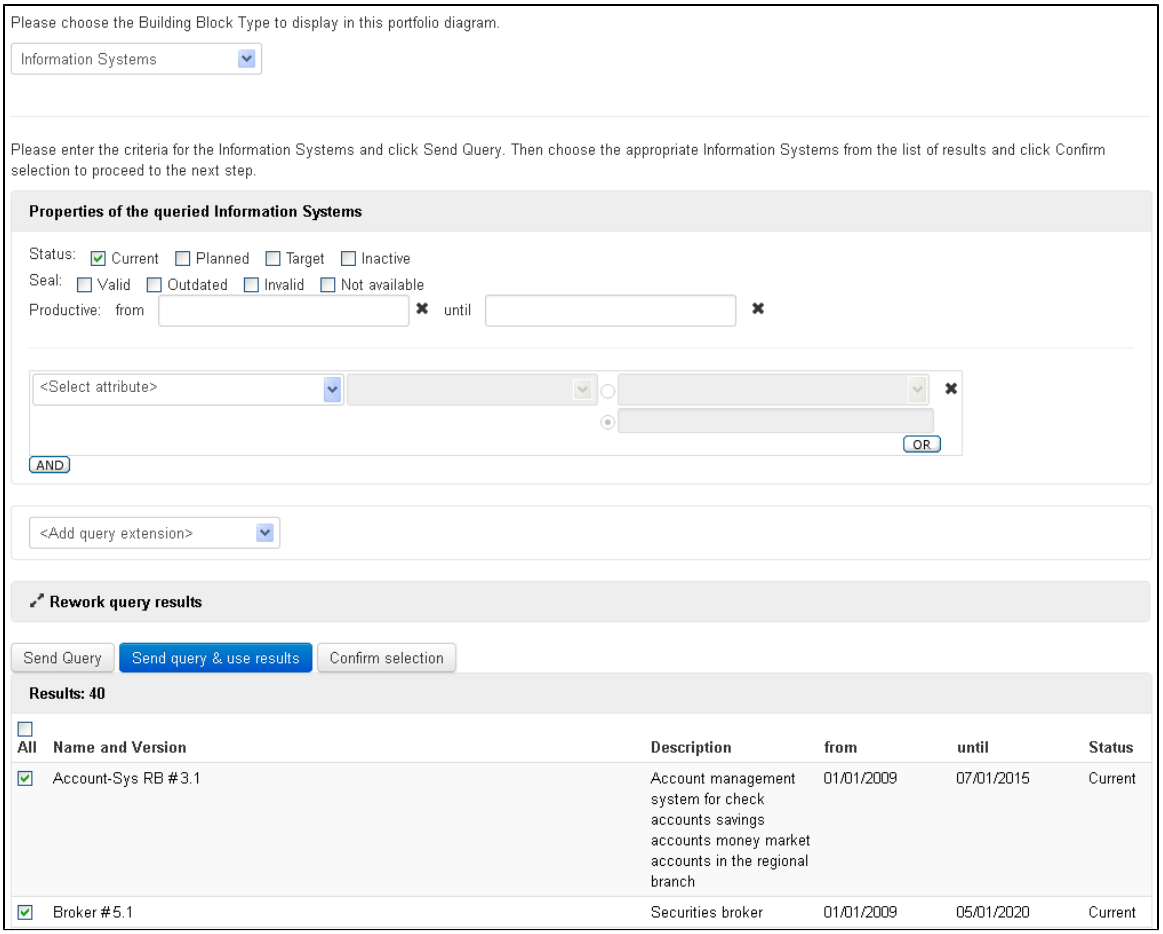
> Note that Information Systems may overlap if they are equal according to their Attributes. Therefore this happens often with "unspecified" Information Systems.



Example of the generated Portfolio Diagram

Configuration

Any of the Attributes assigned to the Building Block Type can be used as Attributes in the Portfolio Diagram. This section explains the procedure for generating a Portfolio Diagram. The screenshot shows the first step for generating the diagram.



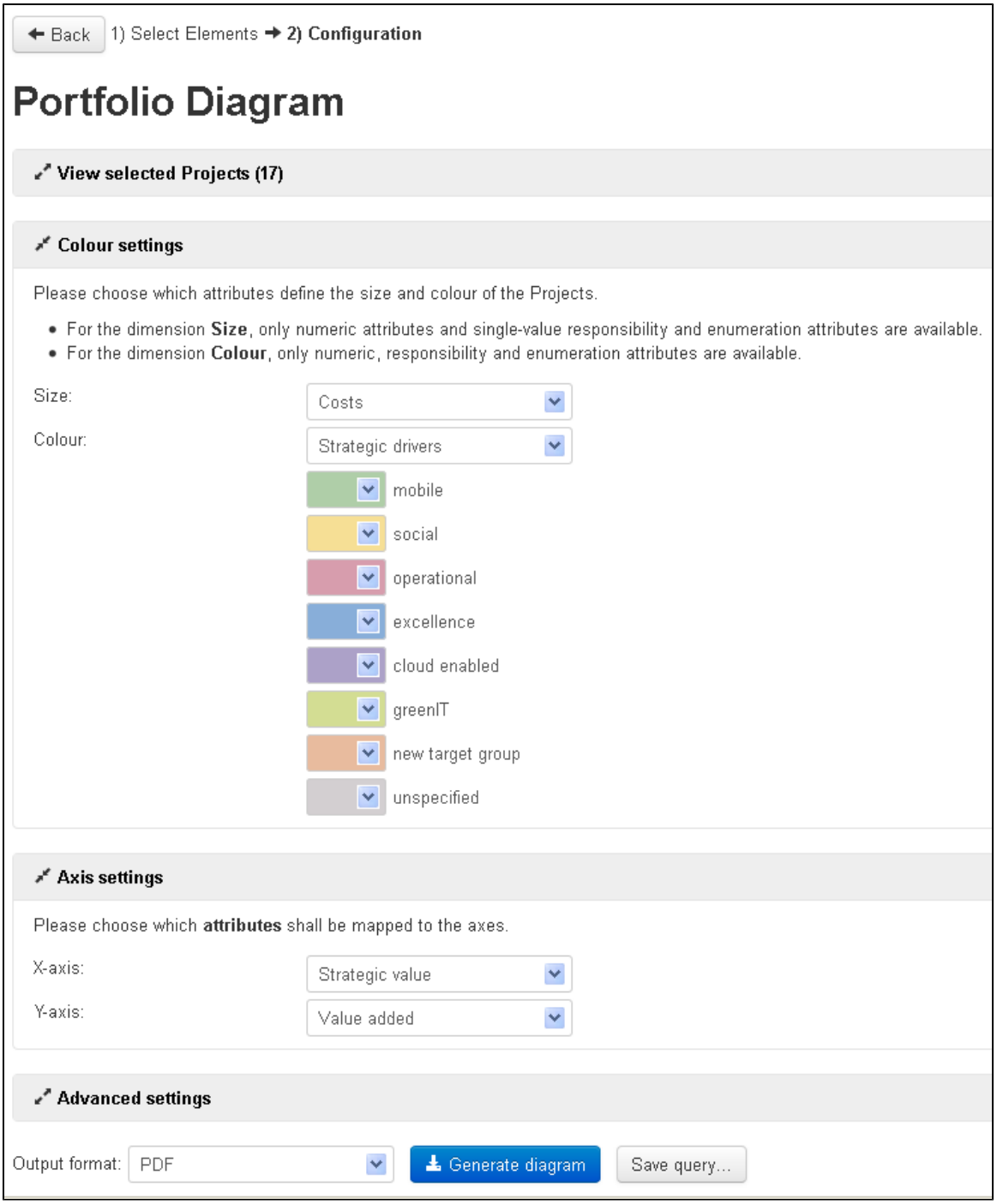
Step 1 for generating a Portfolio Diagram (selecting the Building Block Type, configuring a query and restricting the results set by activating and deactivating checkboxes in the results list)

At the top of the page you can select what type of Building Block you wish to map to the diagram. You are then presented with the query form for this block type, enabling you to refine the selection of elements for the diagram. The query form is the same as the one used for Spreadsheet Reports (see Spreadsheet Reports). You can further restrict the results set by activating and deactivating the checkboxes in the list of results. When you are happy with your settings, **click Confirm selection**. iteraplan then opens the page for configuring the diagram.

> If you click **Send Query & use results** you go automatically to configuration step 2 with all Building Blocks matching your criteria selected.

Assignment of attributes to dimensions can be done as follows: dimension 1, the *X-axis*, is assigned the attribute *Strategy contribution*. Dimension 2 (*Y-axis*) is assigned the attribute *Value added*. Dimension 3 (*Size*) is assigned the attribute *Costs*, and dimension 4 (*Colour*) is assigned *State of health*.
Bear in mind that you can assign any attribute to the *X-axis* and *Y-axis* dimensions. However, dimensions *Size* and *Colour* do not permit the assignment of text or date attributes because it is not feasible to map this information in graphic form.

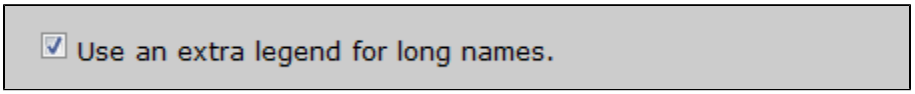
By choosing the scaling-option you can adapt the value range for visualized attributes. "No scaling (global context)" will show the complete range whereas "scaling (local context)" will adapt the visualized range to the actual attribute values.



Defining the axes and dimensions for a Portfolio Diagram

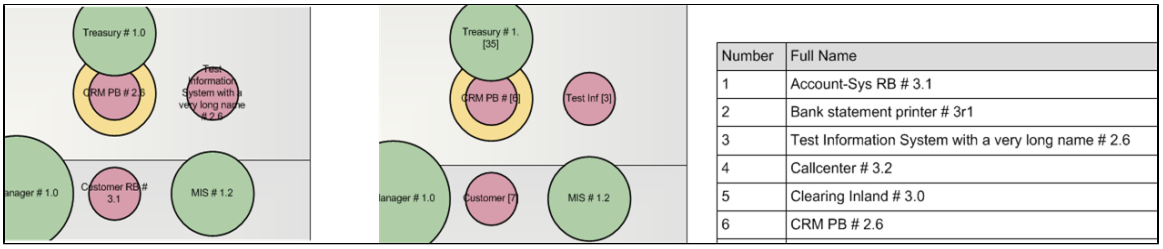
Names Legend

You can choose whether you want to use a legend for long names or not.



Using the legend, the name of Building Blocks is cut if it does not fit into its box (at least four characters remain).

Not using the legend, the hole name of Building Blocks is used. This may overflow its box (see graphics below).

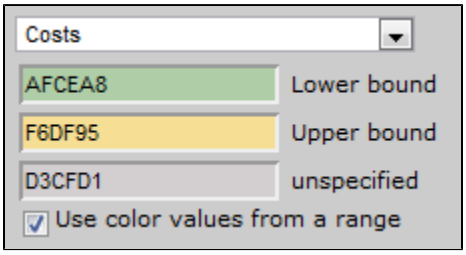


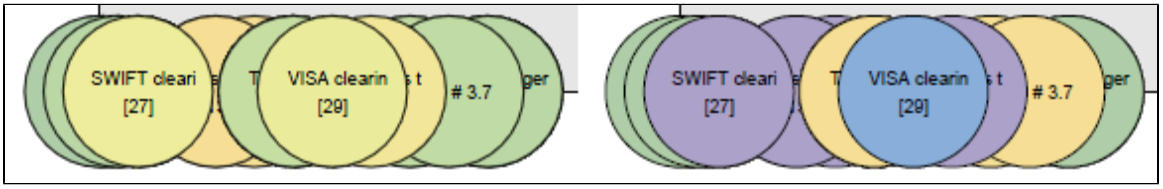
Number	Full Name
1	Account-Sys RB # 3.1
2	Bank statement printer # 3r1
3	Test Information System with a very long name # 2.6
4	Callcenter # 3.2
5	Clearing Inland # 3.0
6	CRM PB # 2.6

not using extra legend **using extra legend**

Color Ranges

You can define color ranges for number attributes. Instead of choosing color values for "<= 38.00", "38.00 - 100.00", "100.00 - 600.00", "600.00 - 700.00" and "> 700", you could also check the box "Use color values from a range" and define only two values. The lower and upper bound. All values would be mapped to a color within that region. Move your mouse over these values to show an color picker, or enter the hex value per hand.





using color ranges **not using color ranges**

The Diagram

The **Generate diagram** button generates the diagram in Microsoft Visio format and presents it for you to download or view directly in iteraplan.

A Portfolio Diagram generated with iteraplan complete with key and colour coding was presented iteraplanDocumentation303:earlier in this section. How the elements are presented within the coordinate system depends on the attribute value which they are assigned. Each axis is annotated with the Attribute which it represents. Building Blocks are presented outside the coordinate system if they do not have an attribute value for the Attribute assigned to the x or y axis in the diagram.

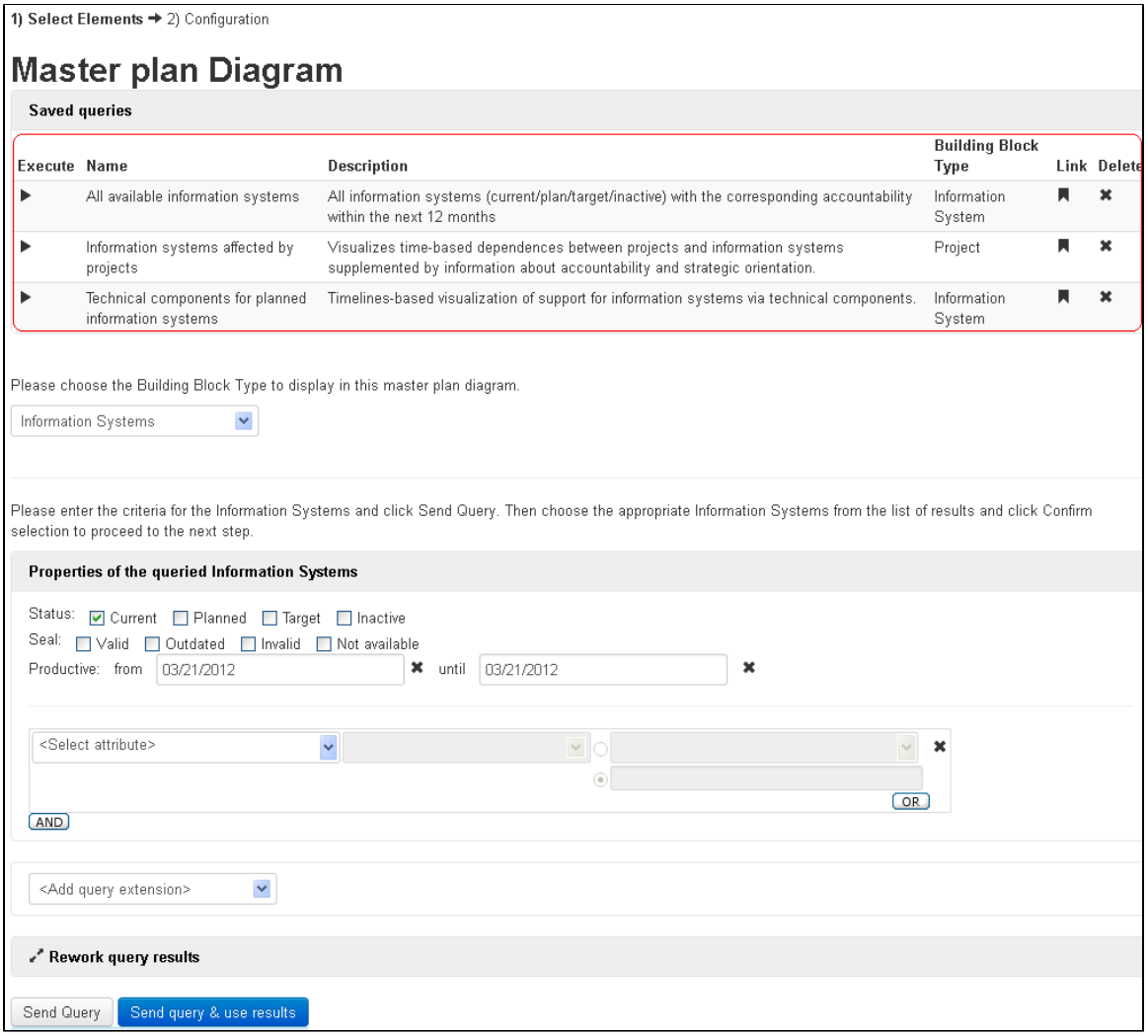
On the right of the diagram are three keys. The first, *Dimension*, lists the Attributes represented by each of the various dimensions (X axis, Y axis, size and colour). The *Colour* key indicates which attribute value is represented by each of the colours. The *Size* key indicates which attribute values are represented by each of the circle sizes. For Numeric Attributes, the diagram generates only the lowest and highest attribute values, and up to three values in between.

Masterplan diagram

The masterplan diagram serves to represent productive timespans and status information of all Building Block types which have the Runtime-Period property and/or Date-Intervals. This gives you an overview of rollout and decommissioning dates as well as of the runtime of all the Building block types.

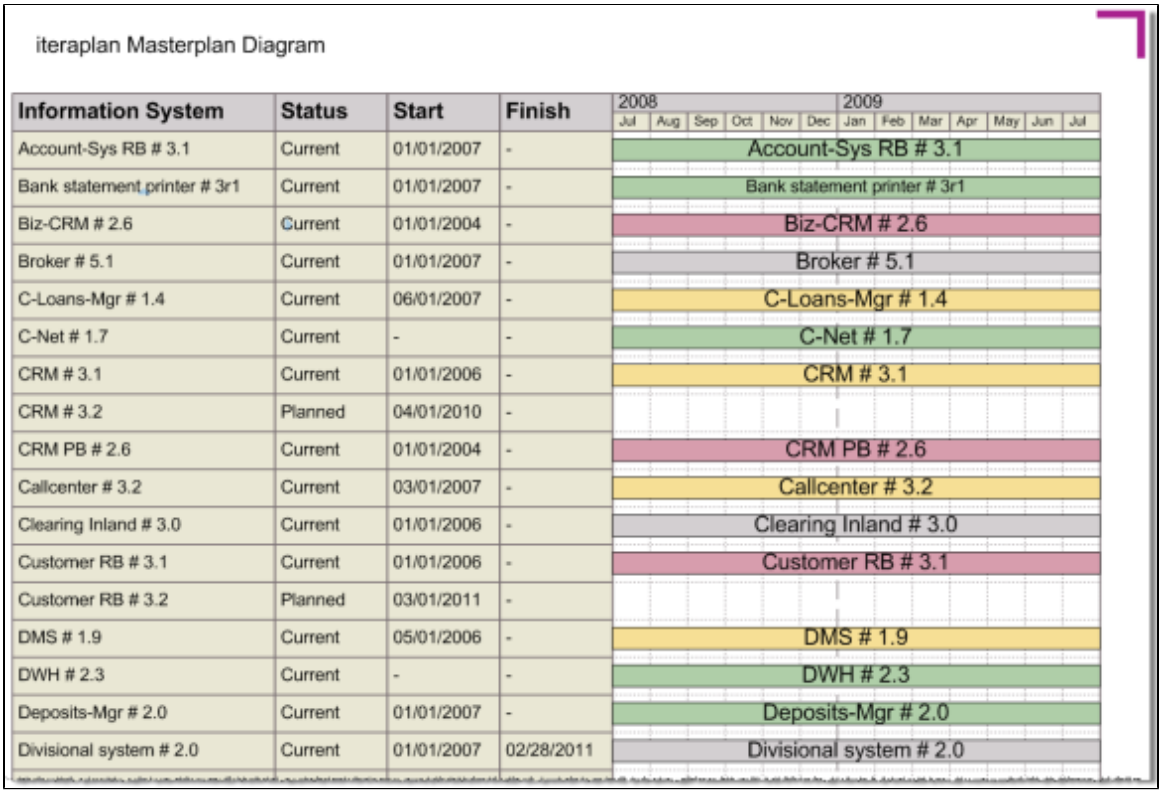
new in 3.1
Now you can choose among all Building Block types, in the previous version you could have choose only among information systems, projects or technical components.

iteraplan provides a predefined query for masterplan diagrams. This generates a diagram which presents the productive timespans of all the instances of the selected Building Block type over the last six months and the following six months from the current date.



Selecting the predefined masterplan diagram

Here is a part of a generated masterplan for information systems. The colour of the bar indicates the status of the system in question (current, planned, target).



Example section of a generated masterplan diagram

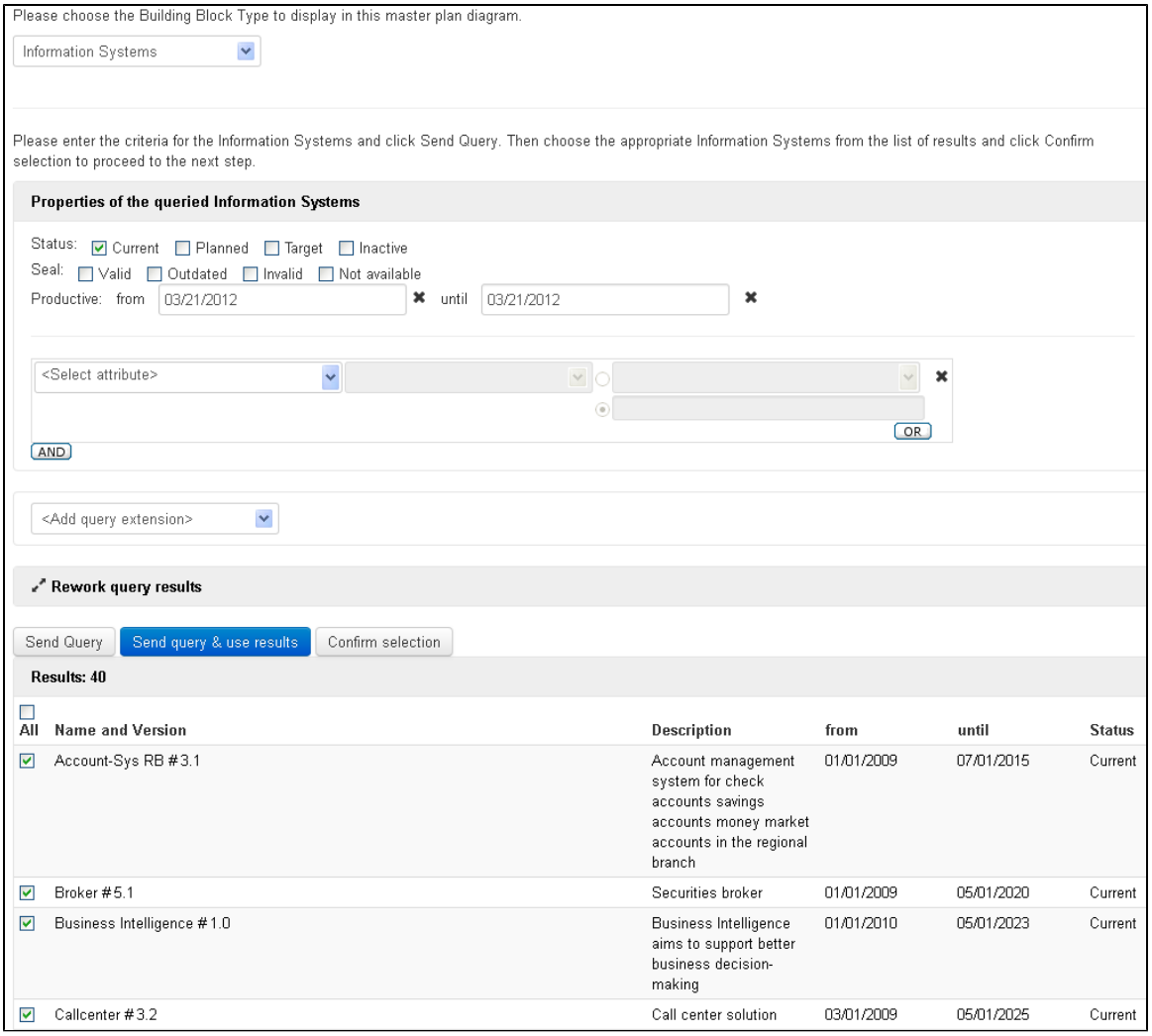
Generating the diagram is a three-step operation:

- Select the type of building block you want to include in the diagram
- Configure your query and query extensions
- Configure presentation options

In the first step, you select the type of building block you wish to present in the diagram.

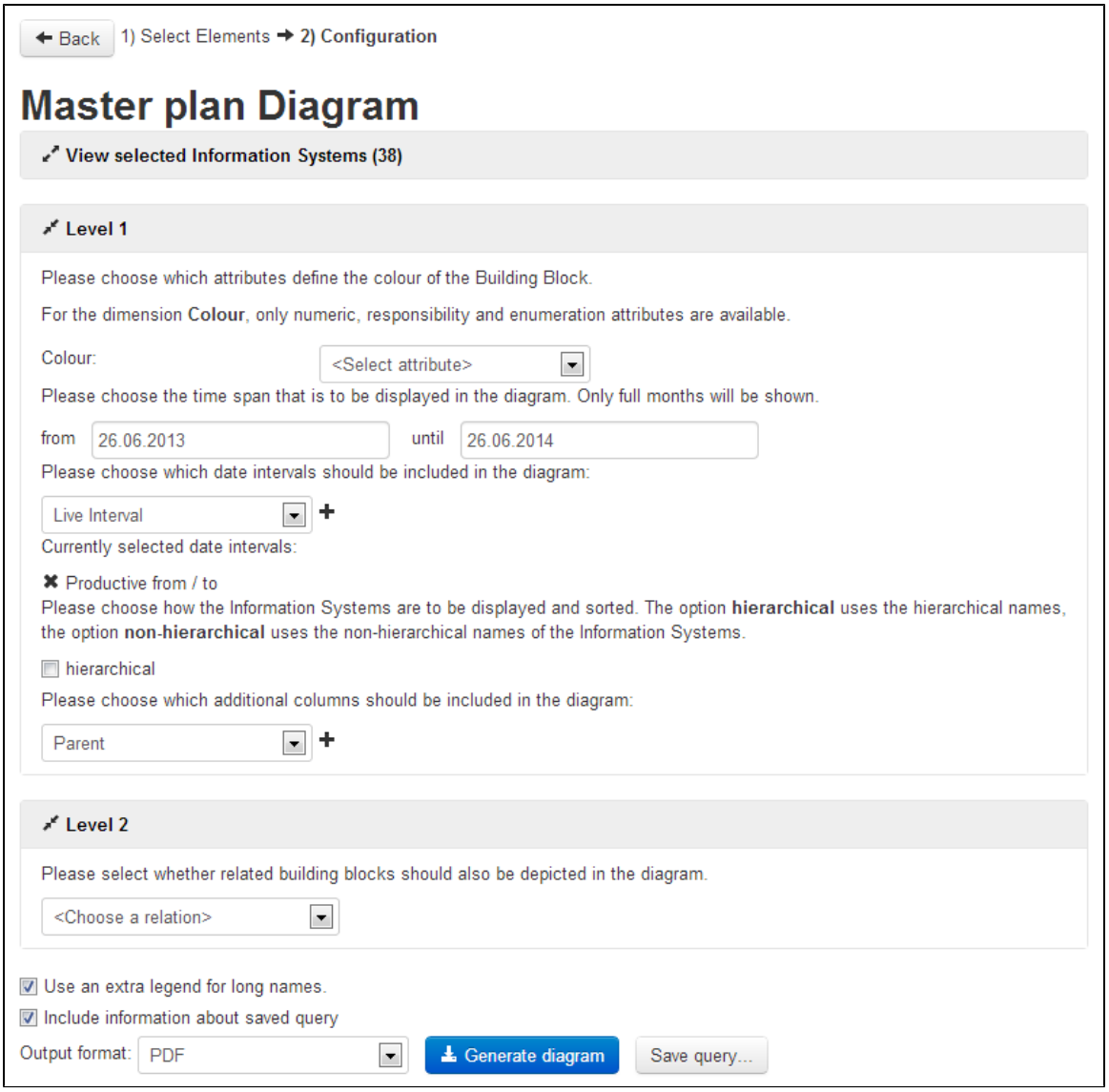
Then define the query and query extensions (see Formulating queries). Clicking the **Send query** button returns the set of results, which you can then refine by activating and deactivating the checkboxes in the list of results. When you are happy with your settings, click **Confirm selection**. iteraplan then opens the page for configuring the diagram.

If you click **Send Query & use results** you are automatically taken to configuration step 2, with all building blocks matching your criteria selected.



Query of information systems for the masterplan diagram

The configuration page presents options for selecting what time period you want to consider, and how the results are to be sorted.



Configuring the masterplan diagram

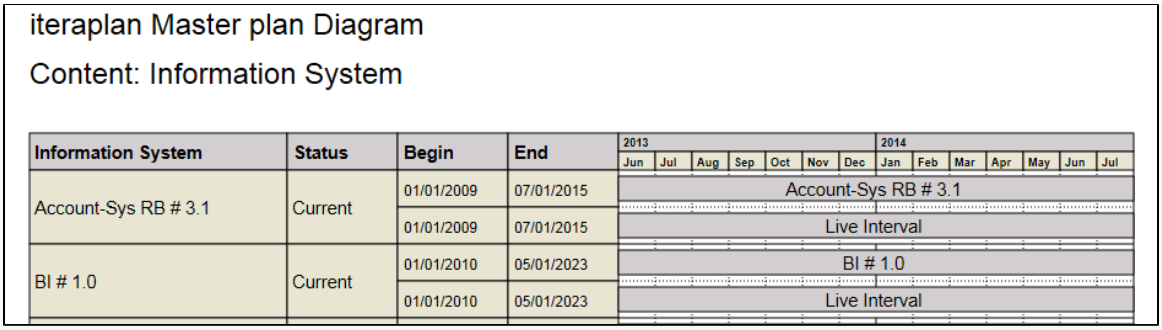
> **new in 3.1**
> We have introduced the concept of **Levels**, referring to related Building Block that will be displayed in the diagram. There are up to 3 levels of relations. The second and the third level are optional.

Inside a Level, under **Colour** you can pick an attribute type that should be taken as a basis for colouring the element bars on the time axis.

> **new in 3.0**
> You can use colour ranges for number attributes (for more information have a look at Portfolio Diagram)

In the **Time span** fields, you define start and end month of the time axis within which the selected elements will be shown.

> **new in 3.1**
> You can select and add as many **Date Intervals** to the diagram as you like. The Productive Period of Information Systems, Technical Components and Projects can also be selected or removed as part of the list of Date Intervals.
>
> When the diagram is generated, extra time bars representing those Date Intervals are rendered, on every Building Block that has an association with the Date Intervals selected.



Within each **Level** section you can refine additional diagram characteristics with the following options:

You can choose to present the masterplan entry names hierarchically or non-hierarchically. If you choose hierarchical presentation, element names will always be prefixed with the name of their superordinate element. Non-hierarchical presentation results in an alphabetical list of element names on the lowest level, as selected on the previous page.

With the next option you can choose to enrich the diagram with lines for related building blocks, e.g. information systems that are affected by a project. The default setting is not to display any related building blocks.

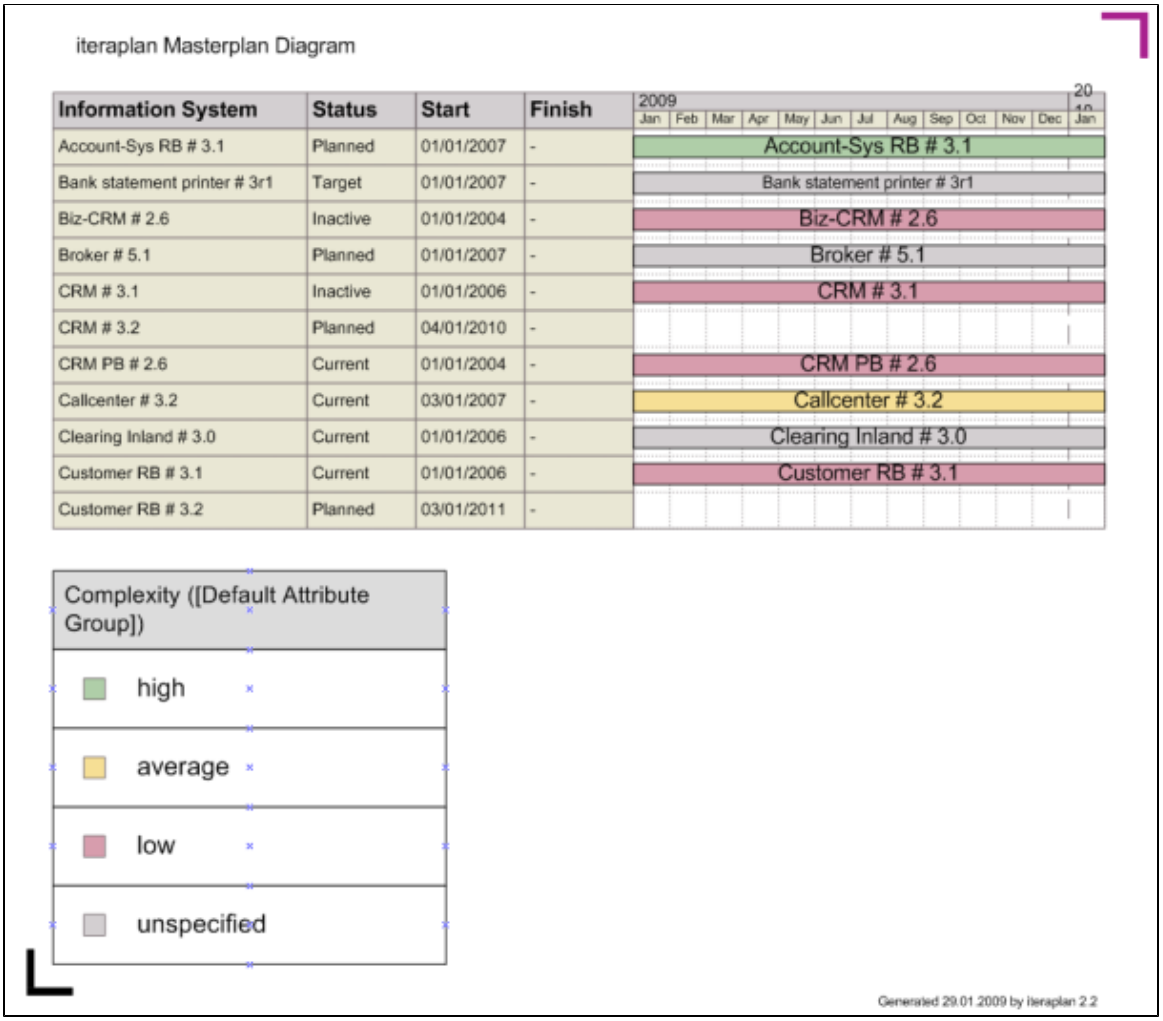
If you select a self-relationship, like the predecessor-successor or parent-child relation, to display as additional lines, there will be an additional option to include elements which are reachable through multiple passes over the selected self-relatationship. If you, for example, select the predecessor relation for the related building blocks in an information system masterplan diagram, by default only the direct predecessors of each information system are displayed as additional lines. If you enable the option *Also include elements which are reachable through multiple passes over the selected self-relatationship*, the predecessors of the predecessors etc. will be displayed as well.

You can choose to include extra columns for each diagram Level, in order to show detail information as you need it. You can pick values for up to three columns. Attribute types and relations are available for inclusion in the diagram.

If you disable the option *Use an exra legend for long names*, building block names will not be truncated and listed in a legend, but printed in full length. However, this may mean that names overflow the boxes of their elements.

Finally, you generate the diagram by clicking the **Generate diagram** button.

Here is an example masterplan diagram.



Masterplan diagram

Information systems are shown along a horizontal time axis that indicates their productive timespans. Each horizontal bar is also colour-coded based on the selected attribute of the information system it represents, and labelled with the system's name. The colour coding is explained in the key beneath the diagram. To the left of the timeline is a tabular overview of the information.

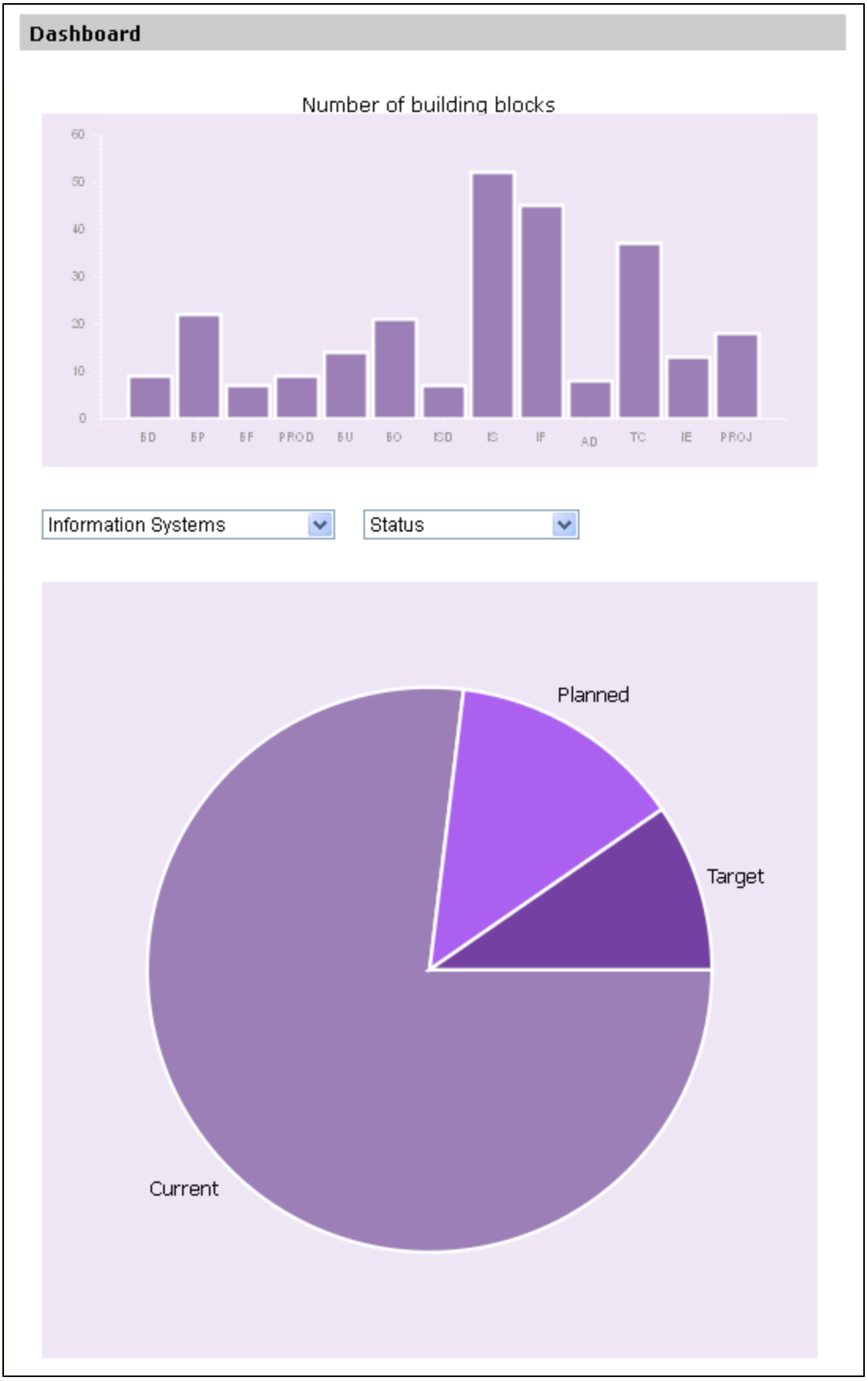
- You can select attributes with possibly multiple values assigned for the colouring. The time-bar will be coloured with horizontal stripes accordingly. Note that multiple-value colouring does not work with Visio output format, which chooses the first of the assigned attribute values to determine the colouring
- For tips on editing a Masterplan diagram in Visio format, please see our ?FAQs.
- For exporting to MS Project or Gantt Project, please check the Spreadsheet reports for Information Systems, Projects and Technical Components

Dashboard

A new dashboard dialog was added for helping the user visualise the basic diagrams of the application landscape data.

The Bar Chart displays the number of elements for each of the corresponding Building Blocks. Furthermore, additional information regarding the bar charts is provided by hovering over them, respectively by clicking on them.

The Pie Chart shows the value distribution for a certain Building Block Type and Attribute. The parameters to create the Pie Chart can be selected via the two dropdown boxes. The first one characterizes the Building Block Type, and the second one shows the single dimension attributes according to the selected Building Block Type.
The values surrounding the Pie Chart represent the set of values for an Attribute for all Building Blocks of a certain type, e.g. Information System Release and Costs. The size of a pie piece indicates the amount of how many Building Blocks share the same value.
Each Pie slice is linked with the Spreadsheet reports. To show the exact elements behind a specific Pie slice in Spreadsheet reports click on the specific Pie slice.



Bar and Pie Chart

- If no values were provided for the selected attribute for all of the elements of the chosen Building Block Type, then an information message appears and no diagram is displayed.
- If no attribute type with read permission for the current user, or only attribute types with multi value assignments are possible for the selected building block type, the list of attribute types to select from will be empty and no diagram can be created.

Two tables provide the user more information regarding Information Systems and Technical Components. More precisely, the first table displays the five Information Systems that contain the most Interfaces, whereas the second table displays the five most used Technical Components. With a simple click on the expansion button, the most used Technical Components clustered by Architectural Domains are also displayed.

Complex Information Systems (# Interfaces):	
DMS # 1.9	8
DWH # 2.3	6
CRM # 3.1	6
Funds txs # r12	4
Callcenter # 3.2	4

Top used Technical Components:	
Oracle # 10g	16
High Level Assembler	16
z/OS	12
COBOL	12
UNIX # Solaris	11

Two tables showing information about the most important Information Systems and Technical Components

Within the entire Dashboard, you get redirected to every displayed Building Block with clicking on it.

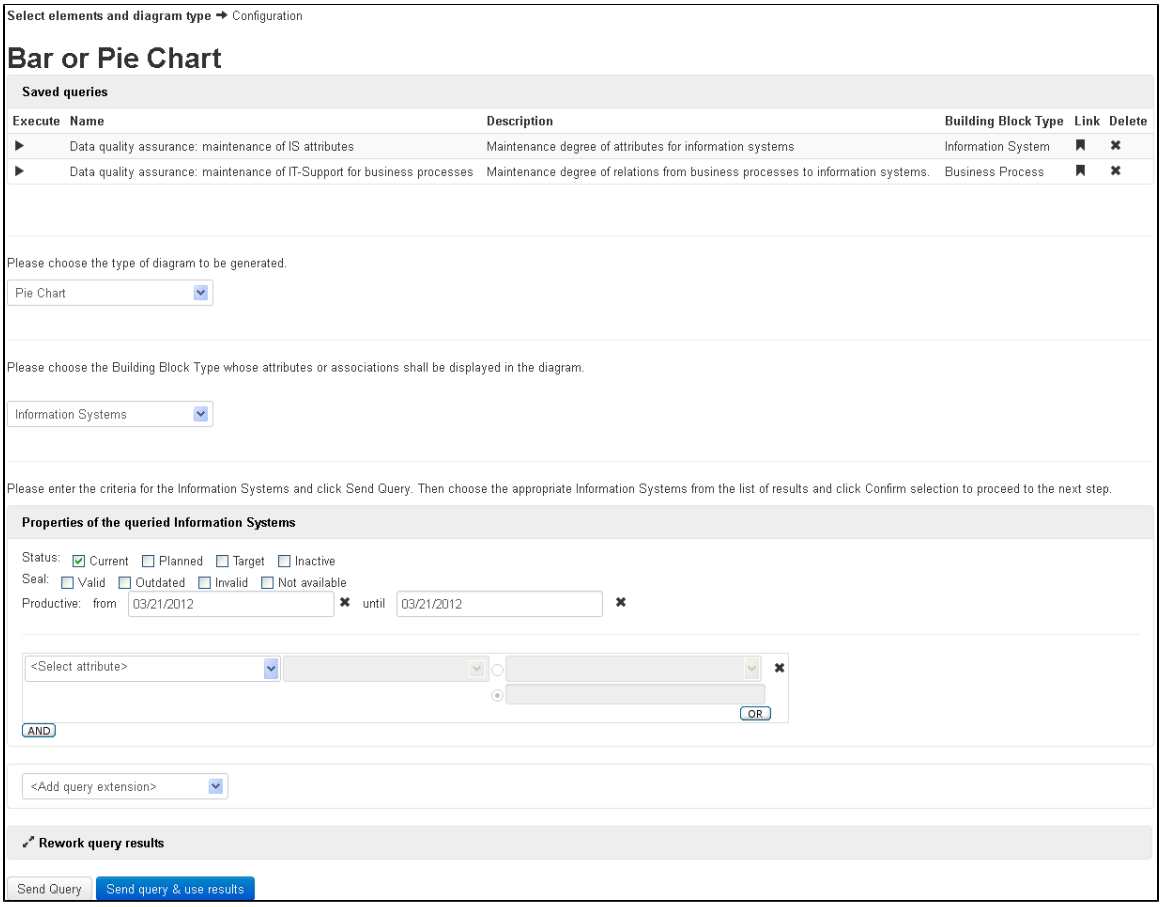
Pie and Bar-Charts

New in version 2.9

- With iteraplan 2.9 another type of graphical report will be introduced. You can now display various overviews of the selected building blocks' state of attributes or relations in pie or bar chars.

Step 1

Similar to most other graphical reports you start with selecting the building blocks you wish to include into the report. After selecting the type of building block of your interest you can refine the selection of building blocks by formulating an according query (see Spreadsheet Reports). In this step you also choose if you want to create a pie chart or bar chart. You can further restrict the results set by activating and deactivating the checkboxes in the list of results.



When you are happy with your settings, click Confirm selection. iteraplan then opens the page for configuring the diagram.

If you click **Send Query & use results** you go automatically to configuration step 2 with all building blocks matching your criteria selected.

Step 2 - pie charts

Types of pie charts

When creating a pie chart you can choose displaying information about an attribute or an association of the selected building blocks. You can display the distribution of the selected building blocks according to:

- their assigned attribute values (or Defining Ranges for Numeric Attributes, in case of number attributes) of the selected attribute (only if the attribute isn't of a multi-assignment type, meaning you can assign only one value to each building block).
- the number of assigned attribute values of the selected attribute (only if the attribute type is capable of multiple assignment).
- whether the selected attribute is maintained (at least one value is assigned) or not.

- the number of assigned building blocks for the selected association
- whether the selected association is maintained or not.

Types of partitioning the pie

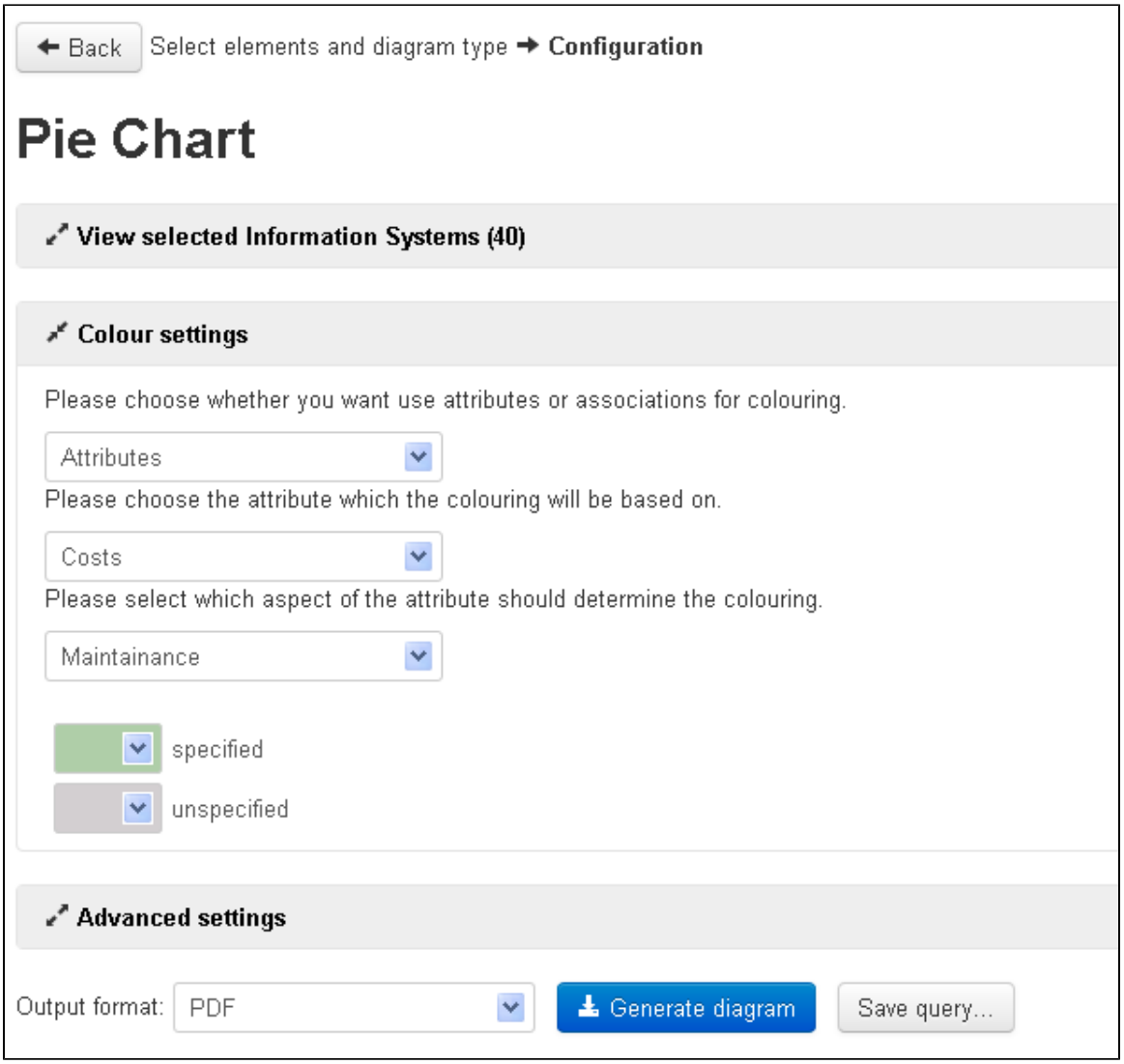
Depending on the attribute type or association you selected, you have different options to choose from for the pie's composition.

- Maintenance: This option is available for all attribute types and associaten and divides the pie into two sections depending on the amount of elements who have values set for the selected attribute/association and who have not.
- Attribute Values: This option is only available for attributes which aren't capable of multiple assignments. The result is a partition of the pie depending on the values set for the selected attribute of the elements.
- Number of assignments: This option is available for all associations and for attributes with multi-value assignments. The result is a partition of the pie depending on the number of assignments the elements have for the selected attribute/association.

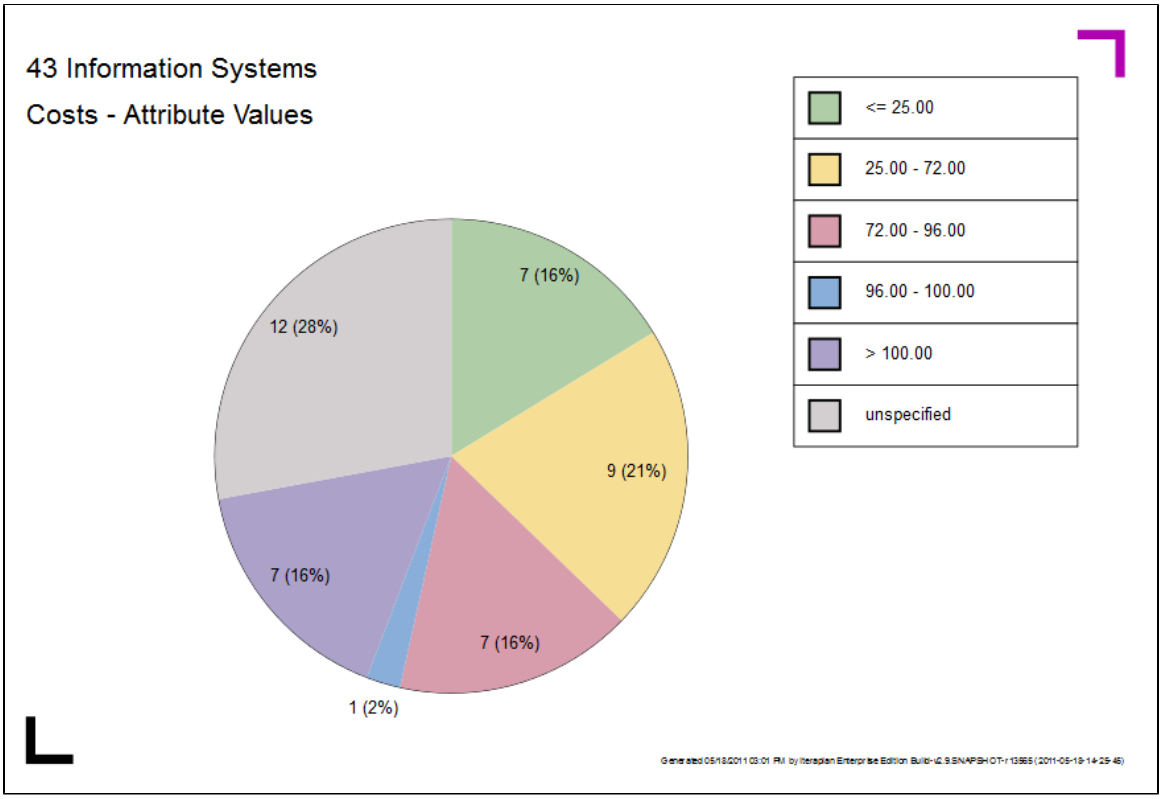
You can also choose to display labels showing the absolute and relative size of the segments. The according checkbox is found at "Advanced settings", see also iteraplanDocumentation303:example configuration below.

Example

Example of a iteraplanDocumentation303:pie chart configuration - step 2 and the resulting iteraplanDocumentation303:pie diagram



pie diagram configuration



pie diagram

Step 2 - bar charts

Types of bar charts

When creating a bar chart you can choose between 5 ways to divide your selected building blocks into bars:

- according to an attribute's values (or Defining Ranges for Numeric Attributes in case of number attributes). You'd choose this type to, for instance, answer the question "How many information systems are there for each value of the attribute Costs?".
- according to the number of assignments of the selected attribute (only possible for attribute types capable of multiple assignments). An example question to be answered by this type would be "How many technical components are there dependent on the amount of people responsible for them (the number of assignments of the attribute Accountability)?".
- according to the associated elements of the selected association. Example: "How many information systems are there for each business process, which are associated to said business process?" (see iteraplanDocumentation303:example below) In this case you can also choose which hierarchical levels of the associated elements you wish to display. Associations to building blocks with lower-than-selected level are aggregated to their ancestors, associations to building blocks which are of higher level than selected are ignored.
- according to the number of assignments of the selected association; analogue to 2.

For these 4 types you can choose to select an additional attribute or association to show stapled bars according to the attribute's assigned values, the attribute's or association's maintenance status (at least one value assigned: yes/no) or the number of assigned values (only for attributes capable of multiple value-assignments).

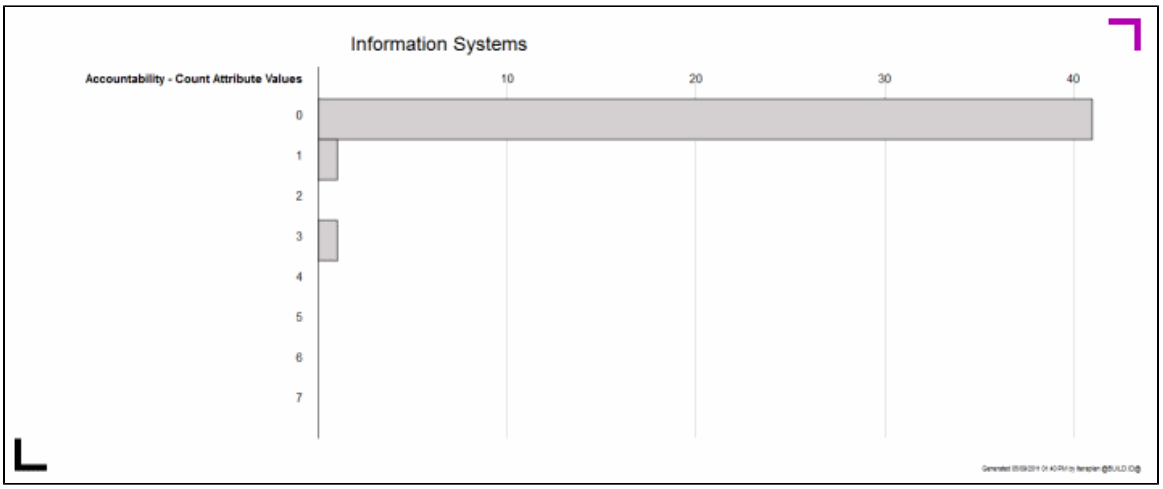
- The last type of bar diagram shows an overview over all attribute types of the selected building blocks. You can choose to display the maintenance status of each attribute or the distribution of each attribute's values.

Note: when a multi-value assignment attribute is chosen to display its values as bar-segments, the total length of the bar might exceed the according number of building blocks. Only the length of each bar segment is guaranteed to be accurate in this case.

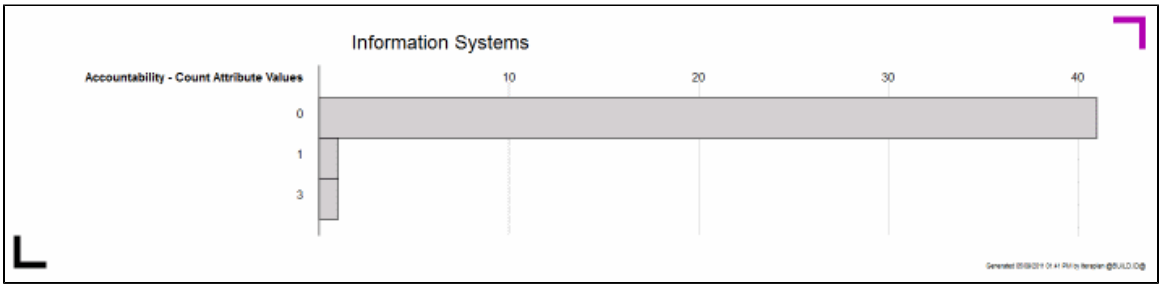
Advanced settings

- When you uncheck the check-box to show empty bars at the advanced setting, empty bars which could appear when showing elements dependent on the number of assigned attribute values or associated elements, won't be displayed (see the iteraplanDocumentation303:comparison below).

- You have the possibility to display the number of elements each bar represents as additional info included in the bar's label, when checking the box "Show number of assigned elements per bar". The percentage based on the number of all selected elements is displayed as well.
- Similar to pie charts you are also able to display labels showing the absolute and relative (as percentage of the whole bar) size of the segments, when checking the box "show segment labels".
- You have 3 possible ways to sort the bars:
    - the "Default"-order uses the order in which the attribute values or associated elements are defined within the iteraplan data, or, in case of displaying the number of assignments, an ascending numerical ordering.
    - the "Alphanumeric"-order sorts the bars based on their labels
    - the "Size"-order sorts the bars according to their size, from larger to smaller.



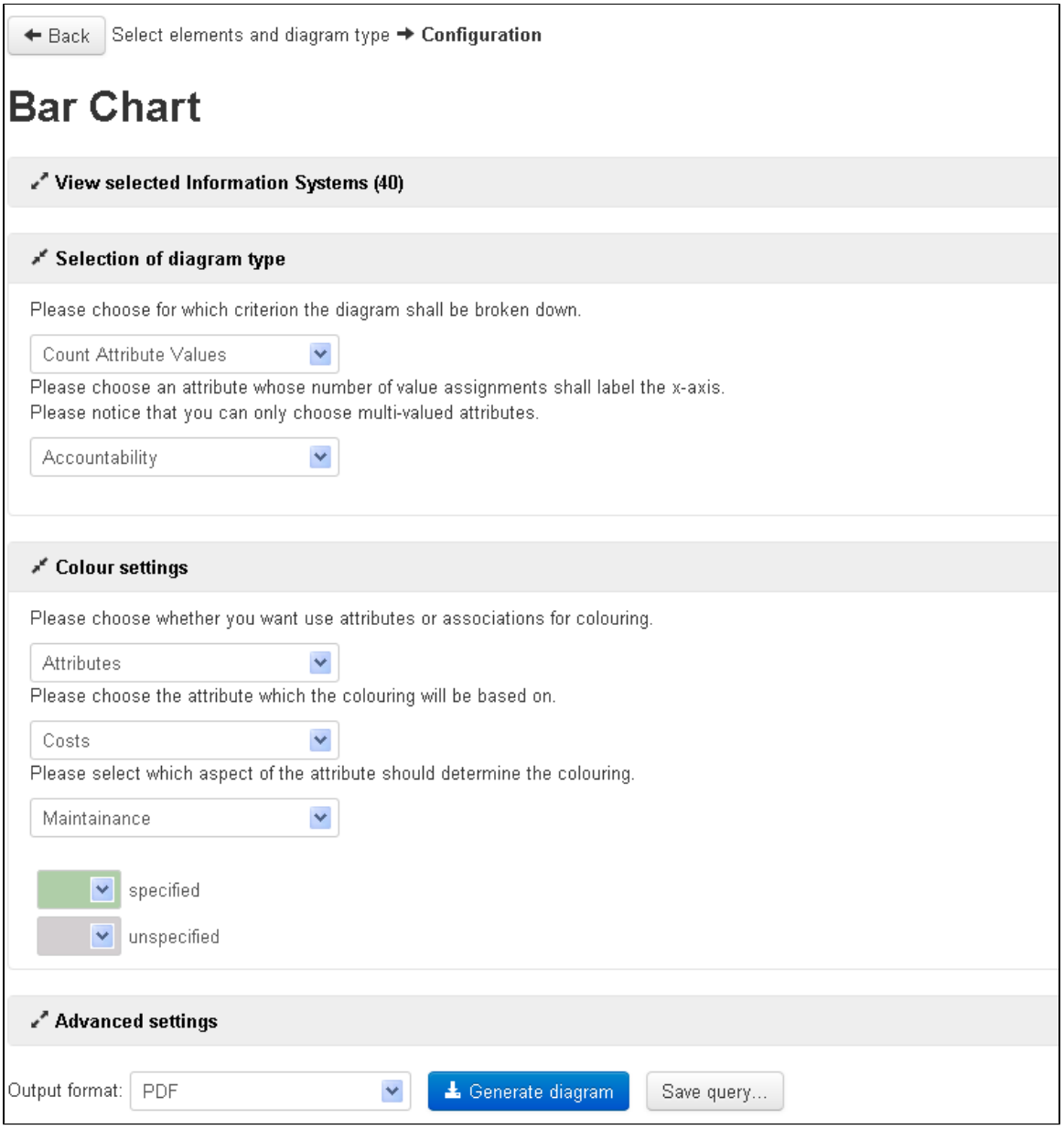
with empty bars



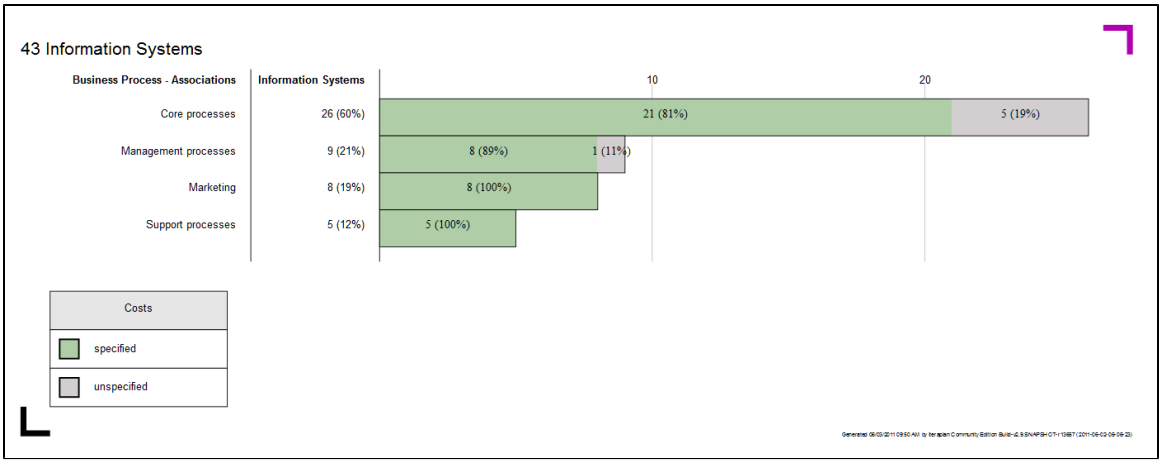
without empty bars

Example

Example of a iteraplanDocumentation303:bar diagram configuration - step 2 and the resulting iteraplanDocumentation303:bar diagram



bar diagram configuration



bar diagram

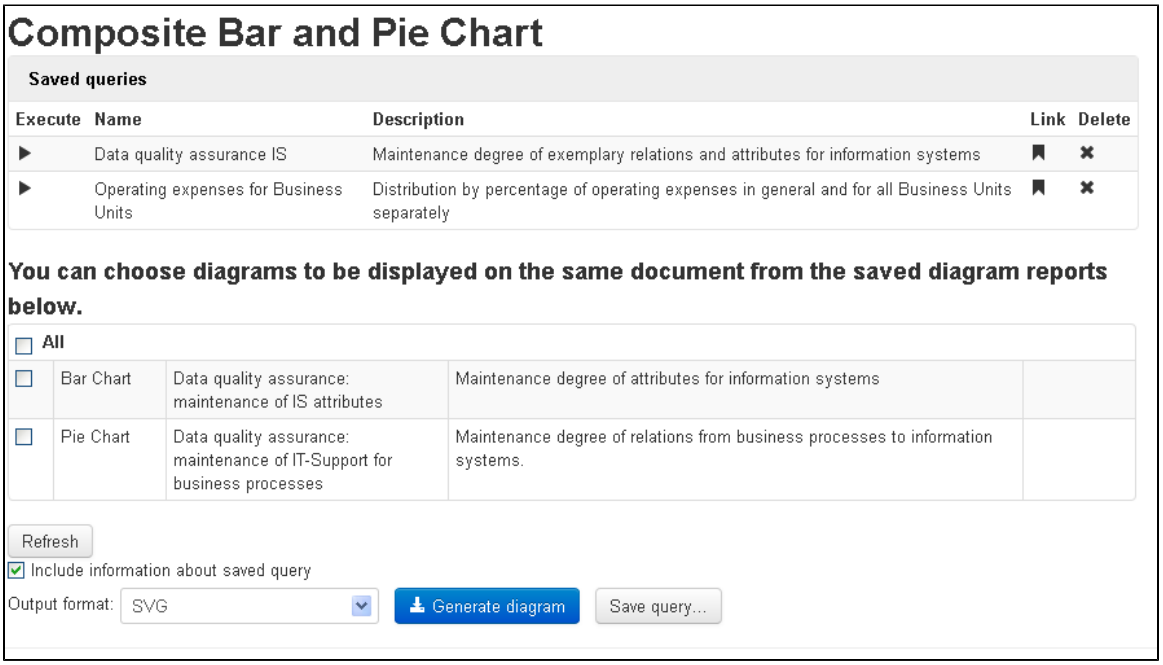
new in 3.0
You can now use color ranges for number attributes (for more information have a look at Portfolio Diagram)

Composite Bar and Pie diagrams

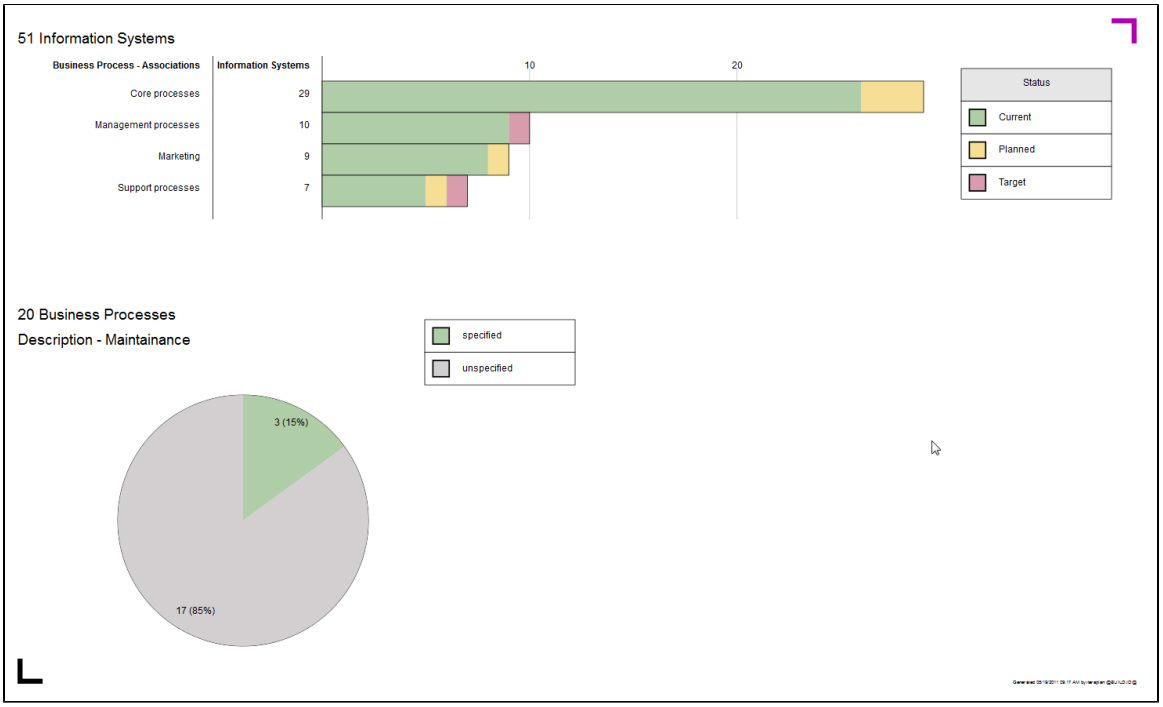
Composite Bar and Pie diagrams allow you combine several Bar and Pie diagrams you defined beforehand. By selecting from the list of saved bar and pie diagram reports and then clicking "Generate diagram" you get a single diagram with the selected reports in the order you defined. You can also select an output format independent from the one which you saved for each of the bar and pie diagrams.

The "Refresh"-button (see example configuration below) updates the list of available diagrams to choose from, in case something changed since you accessed the composite diagram page.

You can also save your composition of diagrams in the same way you can save other diagram reports. When loading such a saved composite report, involved bar and pie diagrams whose saved queries don't exist anymore are ignored, and only the list of still existing diagrams is loaded, including possible changes which were made to them since the saving of the composite report.



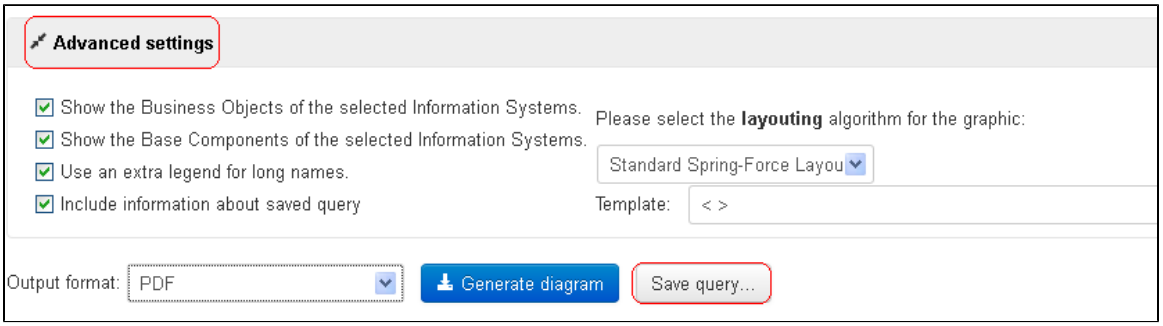
Example configuration of a composite diagram



A simple composite diagram example

Saving and loading graphical reports

Each diagram report has a submenu titled **Save query**.

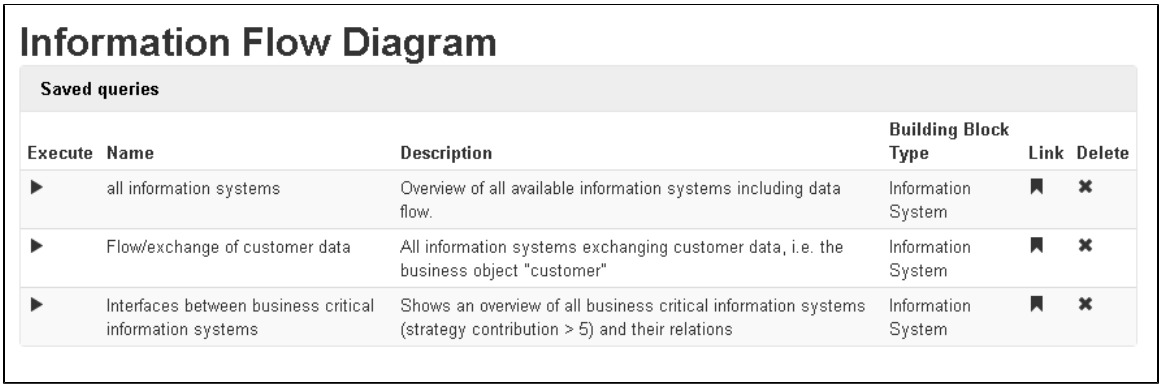


Saving a query

You must assign a query a unique name in order to save it. An entry in the **Description** field is optional – this can be any text of your choice. Once a query has been saved, iteraplan presents it in the list of saved queries on the home page for the type of diagram in question. The pre-defined queries delivered together with the iteraplan installation can not be modified.

> ⚠ If a query of the same type and with the same name you've chosen already exists, that query will be replaced by saving the new one.

On the first page of each diagram report type the already saved queries for this diagram type will be shown. You can directly execute a saved query by clicking the arrow button in the execute column. You can get a link for the directly executed query by clicking on the link icon on the right.



Overview of all saved queries

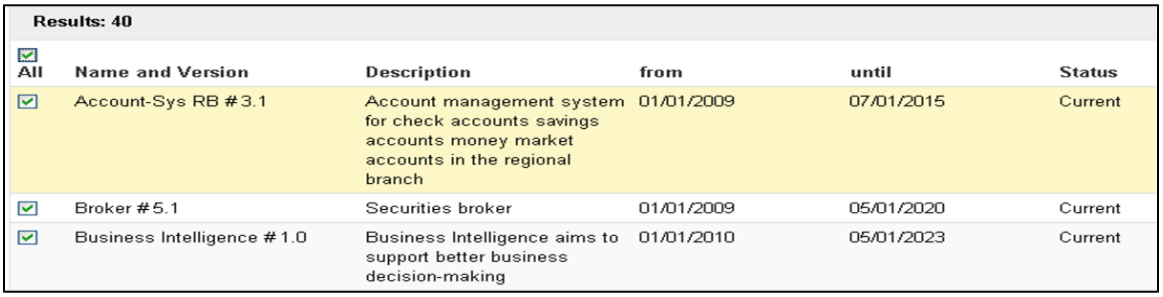
> You can also directly execute queries from the diagram reporting overview page (see Diagram Reports).

If you want to modify the saved query before executing it, you can load it by clicking on the name column. If you want to modify the saved query and then save it again, then the old one will be overwritten by the modified query. User-saved queries can be deleted if they are no longer needed.

There are two different flavours of saved queries:

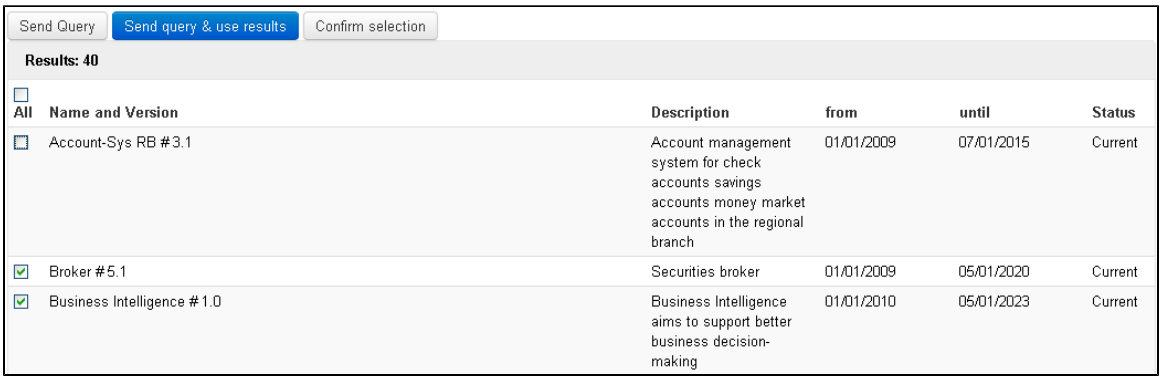
- only the filter is saved in the query and will be applied to the database upon every query execution to find the elements matching this query.
- the saved query not only contains the applied filter, but also the set of selected elements. When the query is executed, only the previously selected set will be included in the diagram, regardless of any new elements that would also be matched by the filter.

To use the first execution possibility the user has to check the **all** results check-box before storing the query. Every time the stored query is run, the full set of matching elements will be used for graphic generation.



Overview of all selected elements

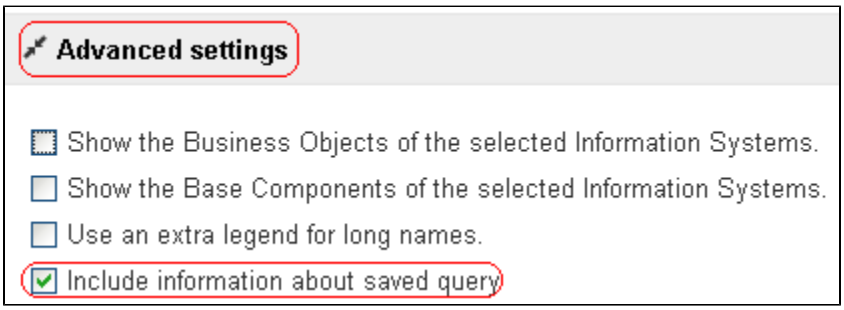
If the user is interested only in a subset of found elements, only those should be selected before storing the query. In this case only the chosen elements will be re-used for graphic generation.



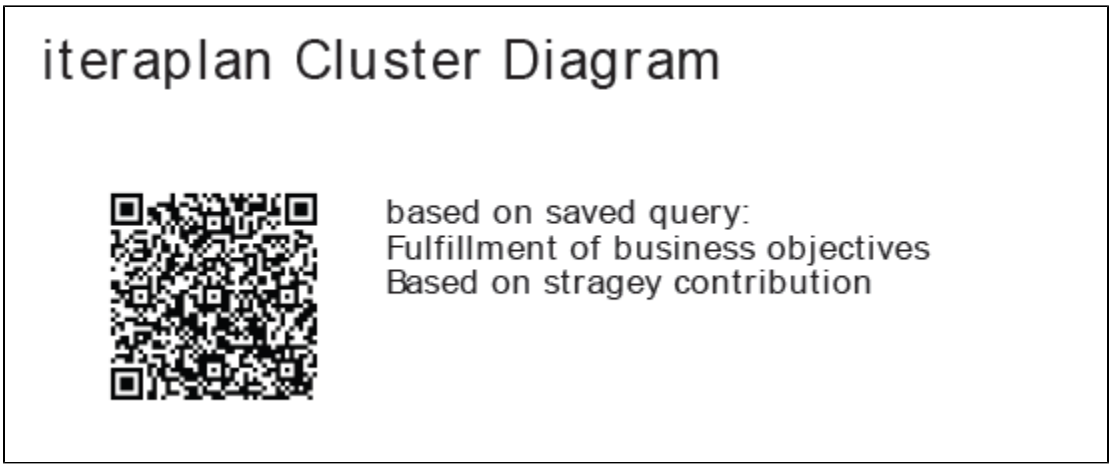
Overview of some selected elements

Saved Query Info

In version 2.9 it will be possible to include information about the saved query on the diagram document. If you saved your diagram configuration, or loaded it from an already saved query, selecting the according checkbox of the configuration's advanded settings (see configuration example below) will display name and description of the saved query on the diagram page. For SVG-based export (SVG, PDF, PNG and JPG) a scannable bitmap code with the URL to execute the saved query with the current data will be included as well (see example below). For SVG and PDF export this info also is clickable, leading to the same URL.



Saved Query Info checkbox



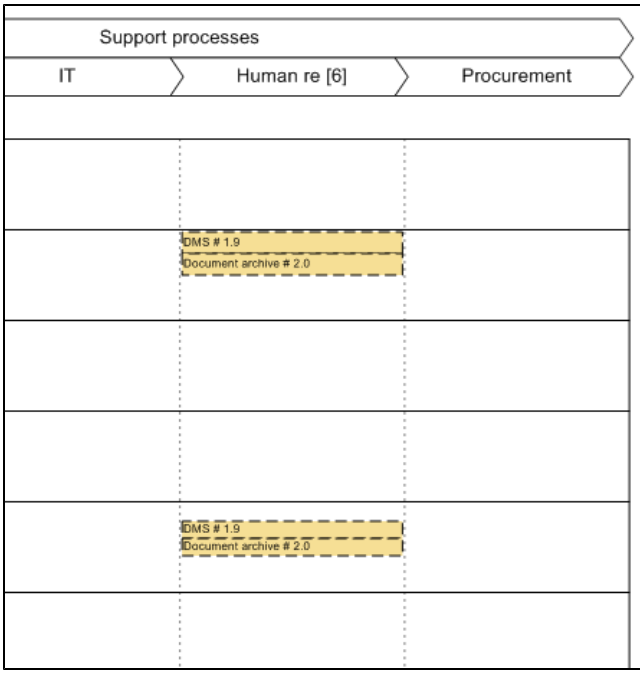
Saved Query Info example

Legends

There are several types of legends added to each type of graphical report. Here you can read more details about each type of legend.

Name Legend

To preserve readability in a diagram report it is sometimes necessary to visualize only a part of the full name of a visualized element. In such cases you will find an index in square brackets behind the name of the element in the report and the corresponding full name in the name legend. In the figure below the process "Human resource management" is visualized this way.



Number	Full Name
1	Corporate management
2	Product management
3	Tender preparation
4	Account opening
5	Account management
6	Human resource management
7	Sales (PK)
8	Marketing (PK)
9	Retail banking customer
10	Sales (GK)
11	Marketing (GK)
12	Sales (FK)
13	Marketing (FK)
14	Treasury
15	Bookkeeping
16	Controlling
17	Personal

Name legend of a landscape diagram

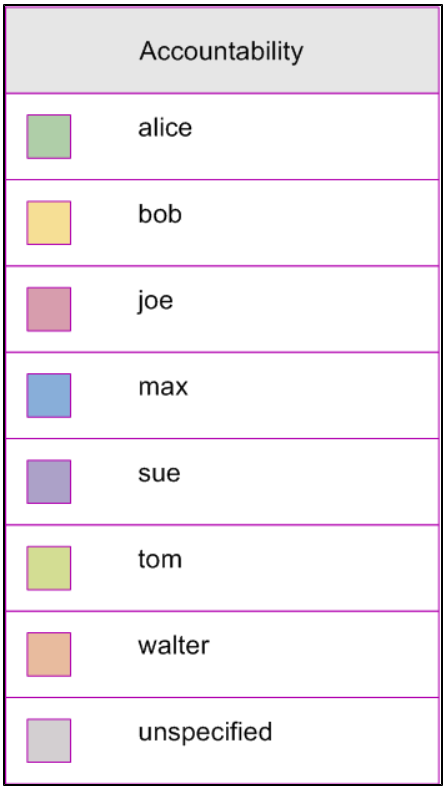
The name legend is available for all diagram reports and is shown only if required. The name legend is optional: You can uncheck the option "Use an extra legend for long names". Please note, that the text might overlap or not show up completely.

Exception with information flow diagrams
Usually each element appears only once in a name legend. With information flow reports a multiple occurrence is possible if

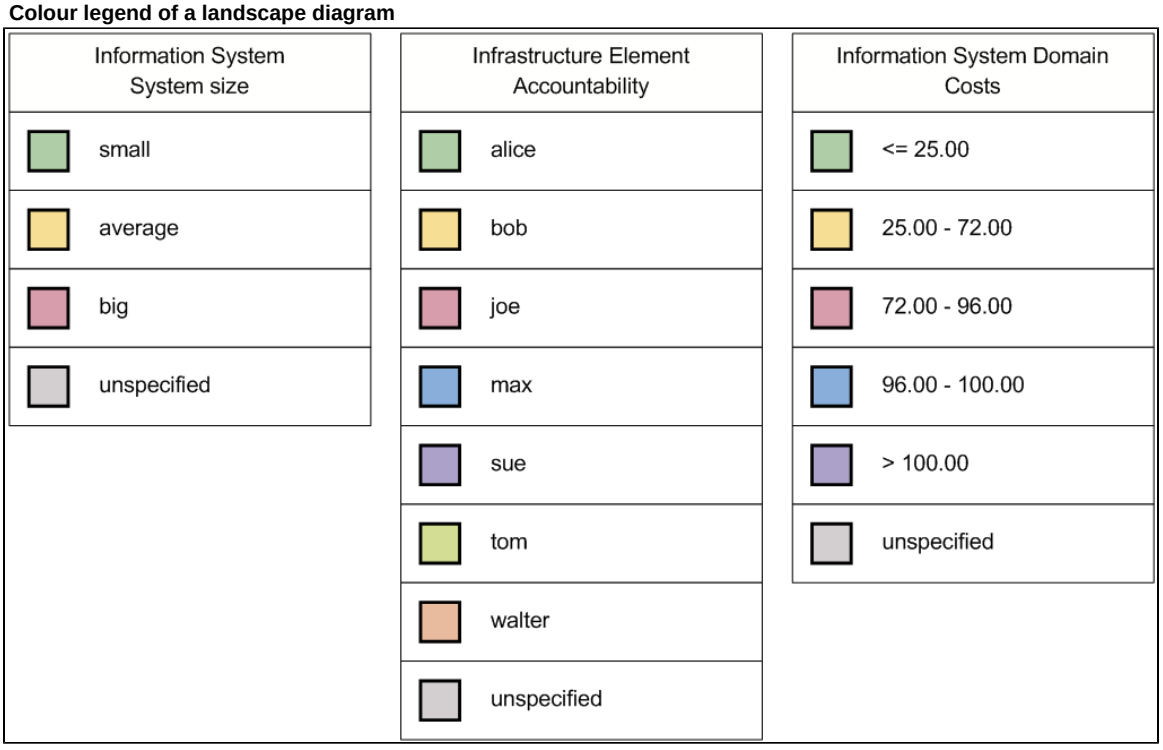
- an information system is **used by** multiple information systems, or
- if the legend contains the caption of a connection between two information systems and there exist another caption with the same semantics but another order of the listed elements.

Colour Legend

To visualize attributes with a range of possible values, such as numeric, enumeration or responsibility attributes, iteraplan provides the possibility to choose a colour range. The semantics of the colours is then summarized in the colour legend (see figure below).



Colour legend of a landscape diagram



Colour legends of a cluster diagram

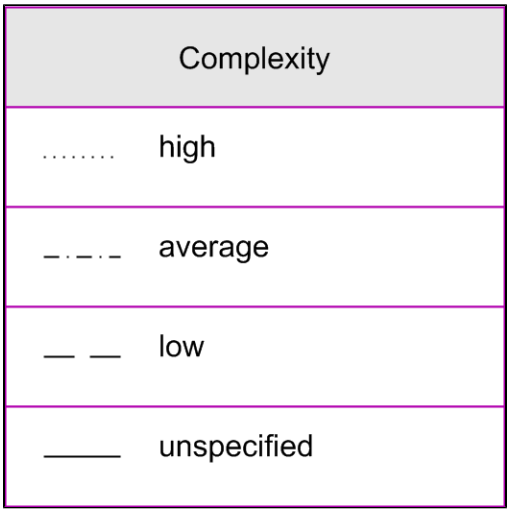
Cluster diagrams
There is an extra handling of enumeration and responsibility attributes in cluster graphics. If the number of possible values for such attributes is greater eight only those values are listed in a colour diagram which are indeed assigned to at least one of the visualized building blocks.

Additionally the header of colour legends in a cluster diagram contains the building block type for easier differentiation between multiple colour legends, as it is shown above.

The number of possible colours for one attribute is set to seven per default. If required you can change the default colours by customizing iteraplan-properties as it is explained in configuration.

Type of Line Legend

Sometimes it is necessary to visualize two different attributes in one diagram report. iteraplan helps you here by providing colour and type of line as the two visualization possibilities. As it is shown in the figure below you can choose different type of line for different attribute values. The type of line legend provides a summary of all selected values.

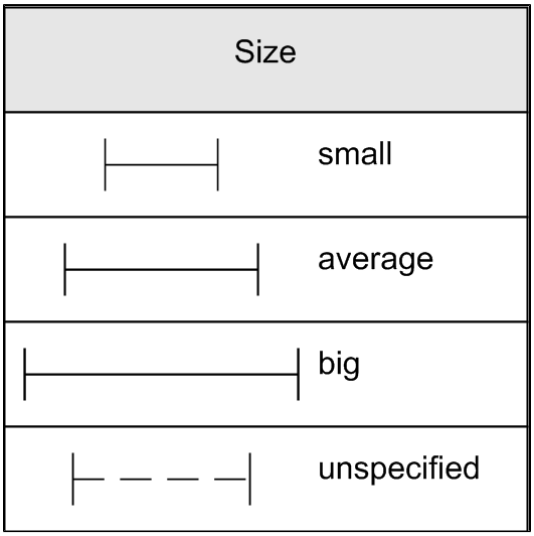


Type of line legend of a landscape diagram

> **Note**
> Due to visibility only a subset of all attributes with a value range are suitable for visualization via line type.

Size Legend

In portfolio diagrams you can use the size of circles instead of line type to visualize an additional attribute. The corresponding mapping is summarized in the size legend.



Size legend of a portfolio diagram

Attribute Mapping Legend

In addition to all legend types described above another one is provided for portfolio and information flow diagrams: the mapping legend. Among other information it contains the mapping for X- and Y-axis to attributes in portfolio diagrams and the mapping for line caption to the corresponding element for information flow diagrams.

Legend	Attribute
X-axis	Strategy contribution (Strategic measurementcategories)
Y-axis	Value added (Strategic measurement categories)
Size	System size ([Default Attribute Group])
Colour	State of health (Strategic measurement categories)

Attribute mapping legend of a portfolio diagram

Legend	Attribute / Building Block
Colour	Status
Line type	Complexity
Line caption	Business Objects

Attribute mapping legend of an information flow diagram

Availability of legends in diagram types

This table gives on overview which legend types are available for which diagram types. Please consider that the legends appear only if they are really required.

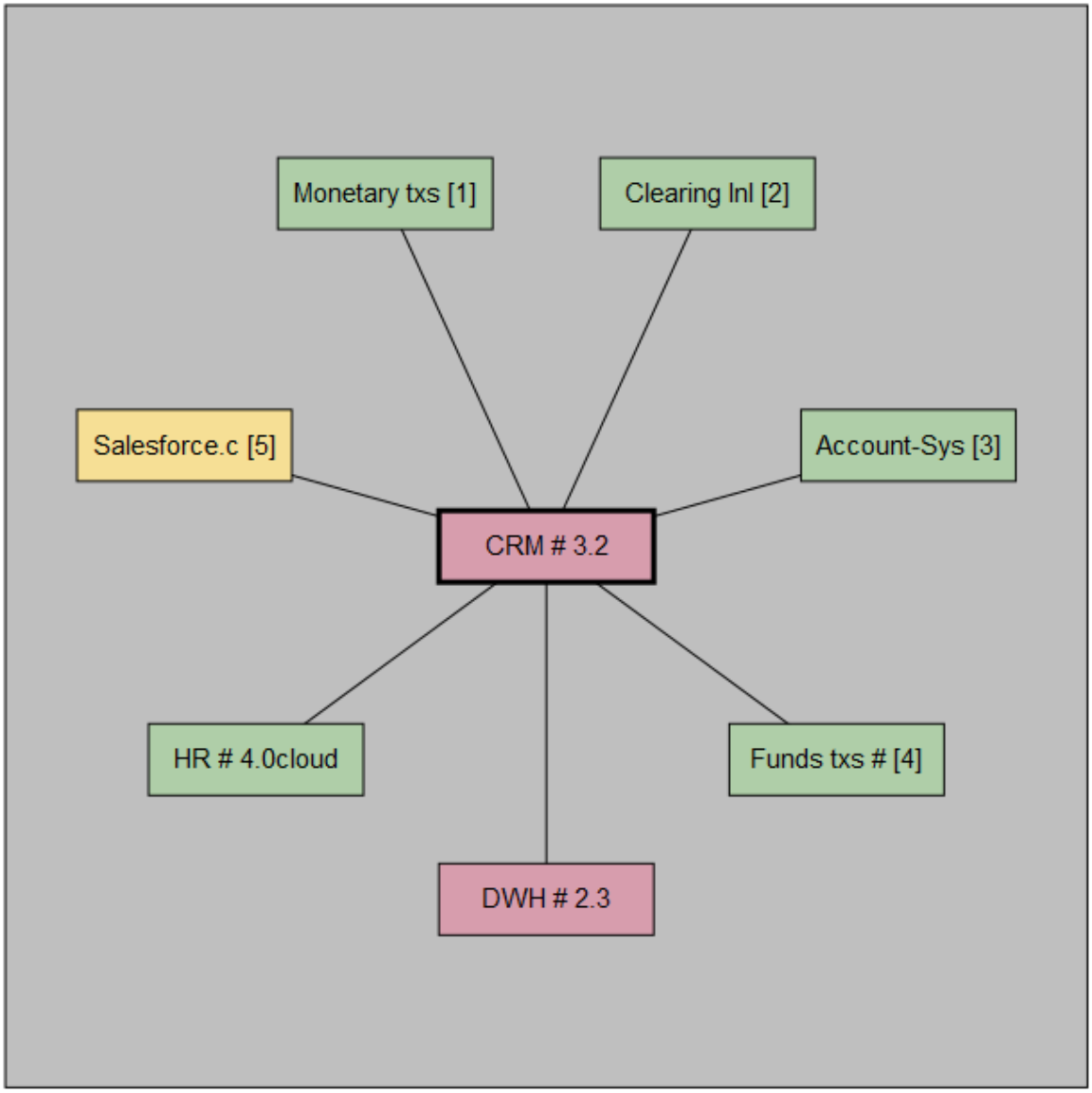
Type of diagram report	Name Legend	Attribute Mapping Legend	Colour Legend	Type of Line Legend	Size Legend
landscape diagram	X		X	X	
information flow diagram	X	X	X	X	
cluster diagram	X		X		
portfolio diagram	X	X	X		X
masterplan diagram	X		X		

Neighborhood Diagram

New in 3.2
The Neighborhood Diagram as a new form of context visualization was added

What is a neigborhood diagram?

The neighborhood diagram is a new type of context visualization. It should provide a fast overview which information systems are connected (over at least one interface) with the current selected information system. All connected information systems are positioned as a circle around the centralized information system. The information sytems are connected by a single line, with the current information system, regardless of the number of connections.



Selection of the connected information systems through status

Type of Status	current	planned	target	inactive
current	X	-	-	-
planned	X	X	-	-
target	X	X	X	-
inactive	X	-	-	X

The vertical Type of Status displays the value of the current information system.

The horizontal Type of Status shows which connected information systems are displayed according to their Status.

The 'X' marks the displayed information sytems

Colour of the information sytems

All displayed information systems are colored depending on their status.

Status
Current
Planned
Target
Inactive

Long Names

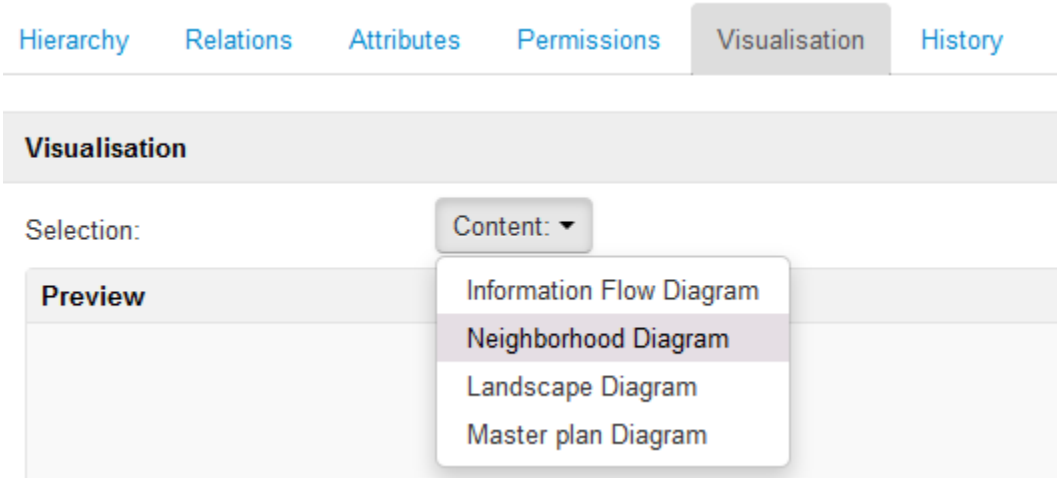
Names which are too long for the boxes are shortend and displayed in full length in an extra legend on the right side of the graphic.

Number	Information System
1	Monetary txs RB # 2.0
2	Clearing Inland # 3.0
3	Account-Sys RB # 3.1
4	Funds txs # r12
5	Salesforce.com

How to display the Neighborhood Diagram

To display the Neighborhood Diagram of an information system you have to:

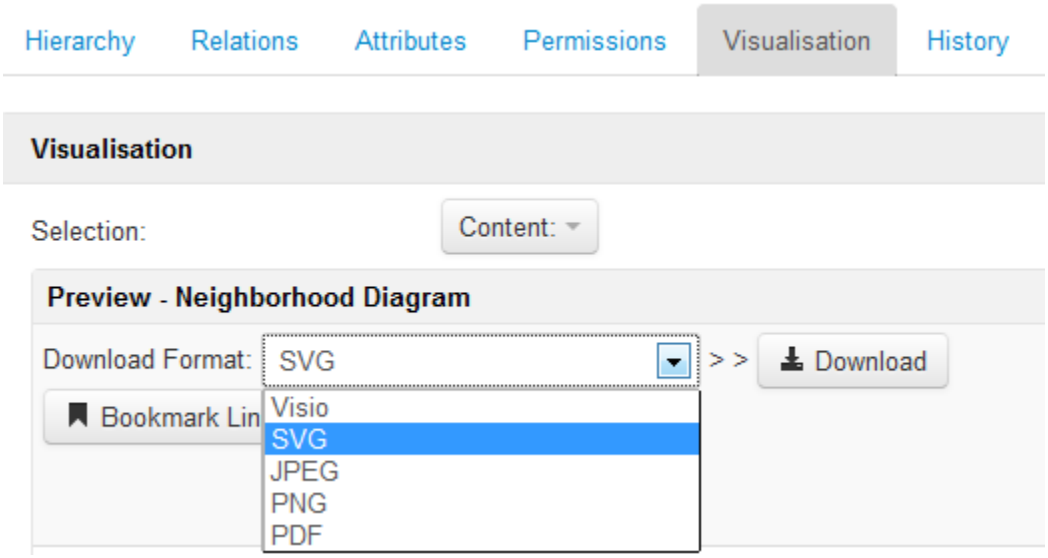
1. Open the information system in view mode
2. click on the 'Visualizations'-tab
3. On the 'Visualizations'-tab click on Content and select 'Neighborhood Diagram'
4. The Neighborhood Diagram will appear in the preview area of the 'Visualizations'-tab



How to save the Neighborhood Diagram

To save the Neighborhood Diagram of an information system you have to:

1. generate a neighborhood diagram, as discribed in 'How to display a Neighborhood Diagram'
2. (optional) select the format the downloaded Neighborhood Diagram should have (supported formats are: SVG, JPEG, PNG, PDF, MS Visio)
3. click on the 'Download'-button



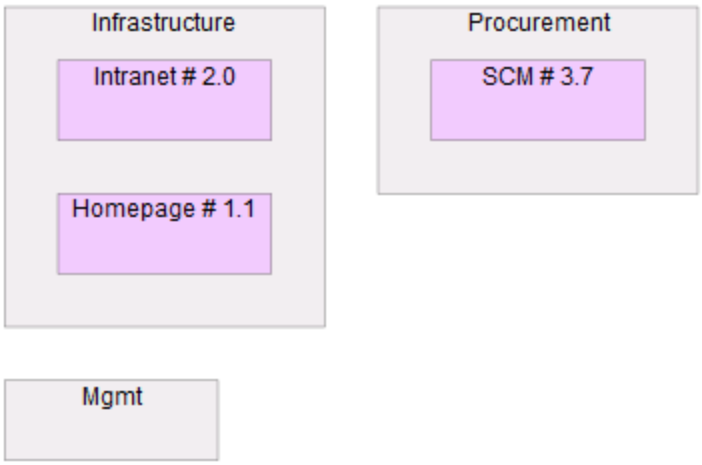
How to save a link to a Neighborhood Diagram

To save a link to a Neighborhood Diagram you have to:

1. generate a neighborhood diagram, as discribed in 'How to display a Neighborhood Diagram'
2. click on the button 'Bookmark Link to Graphic'
3. a new window appears with a link to copy in it

Nesting Cluster Diagram

Overview



A Nesting Cluster Diagram visualizes the relationship between Building Blocks of two different types (an outer and an inner type) by displaying them as boxes and nesting them into each other according to the relationship.

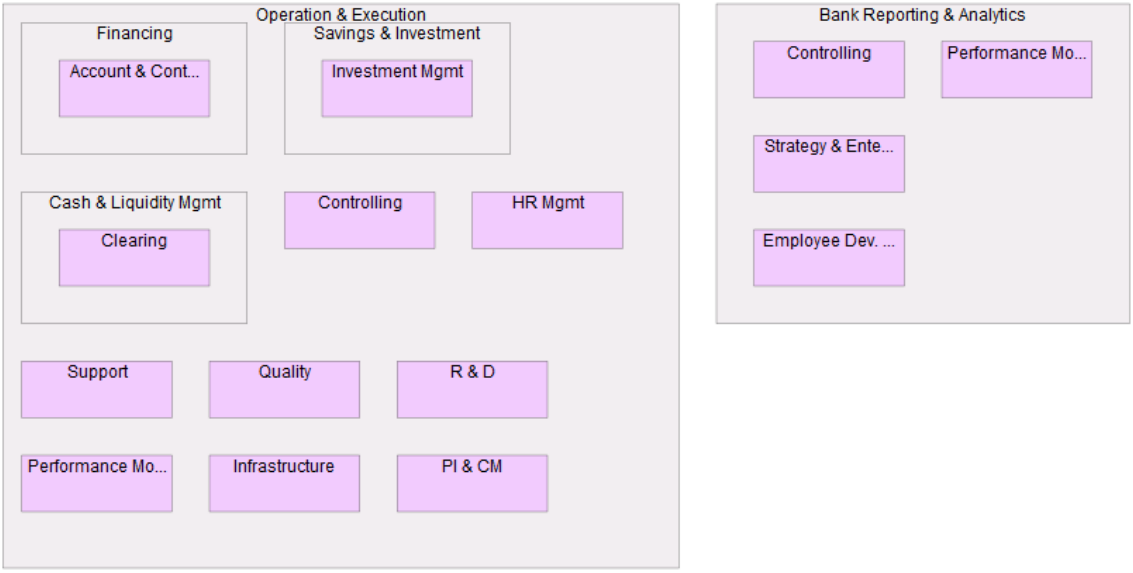
Additionally it allows you to nest Building Blocks of the types recursively into each other with regards to a self-relationship. The coloring of the boxes may be static or dependent on an attribute of the Building Block they represent. You may also filter the Building Blocks to be drawn in the diagram by using iteraplans well known filter mechanisms. For the inner Building Block Type you may specify to draw Building Blocks without a relationship to any outer Building Block within a separate cluster.

Structure of a Nesting Cluster Diagram

A Nesting Cluster Diagram depicts instances of an outer Building Block Type and instances of an inner Building Block Type, which are related via a Relationship. The Building Blocks are represented by boxes. A box of an outer Building Block contains a box for each Building Block of the inner type it is directly related to via the relationship.

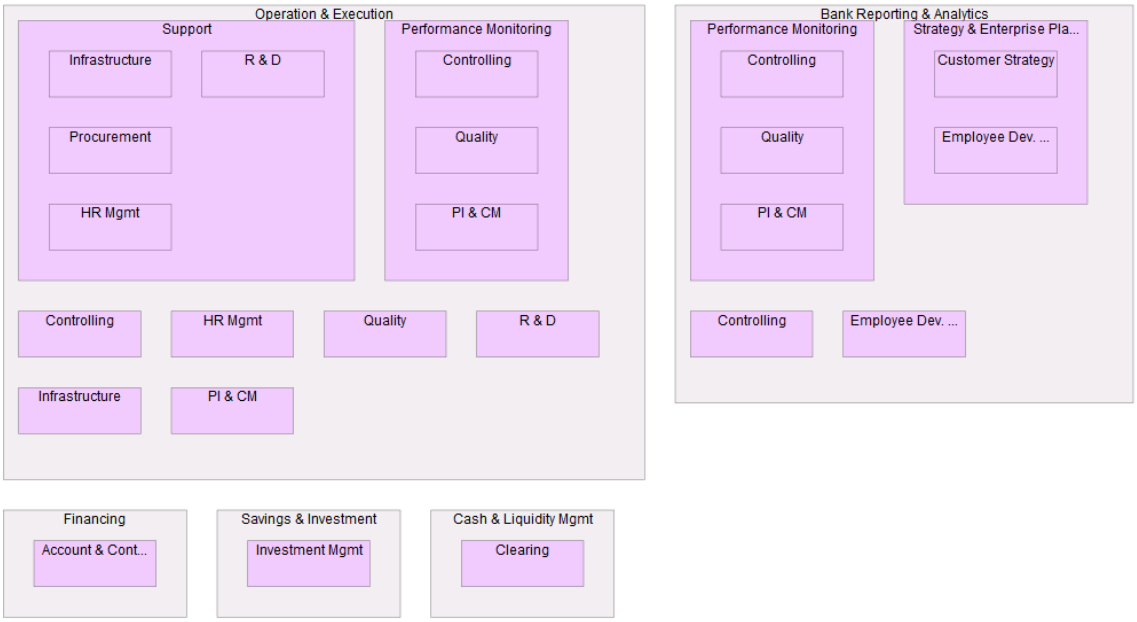
Simple Nesting Cluster Diagram

Outer and inner Building Blocks may additionally be clustered by a self-relationship.



Nesting Cluster Diagram with outer Building Blocks nested by the self-relationship "children"

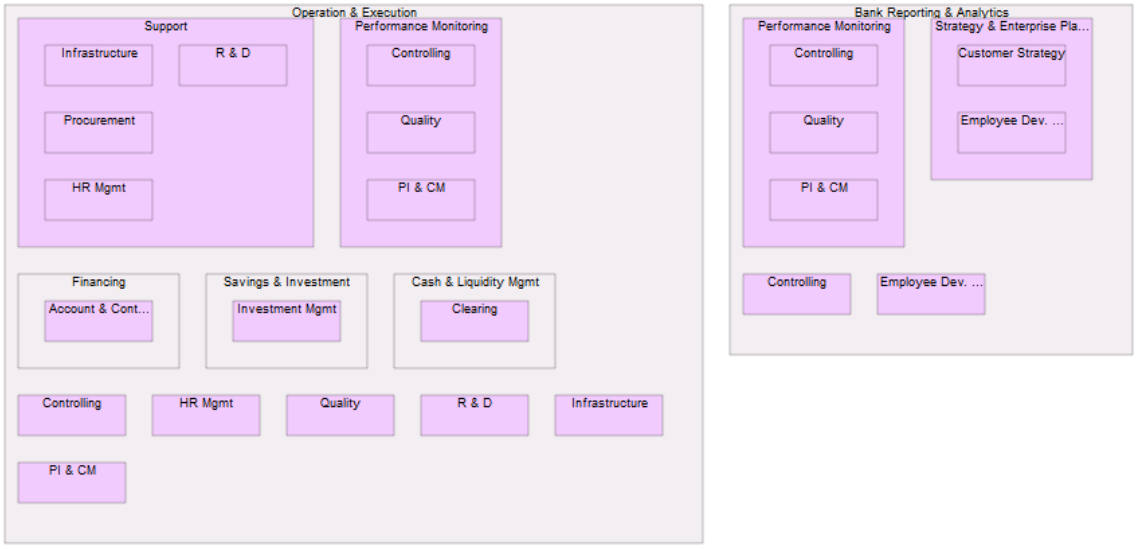
As you can see, the boxes "Financing", "Savings & Investment" and "Cash & Liquidity Mgmt" are now nested within the box "Operation & Execution" because they are children of "Operation & Execution".



Nesting Cluster Diagram with inner Building Blocks nested by the self-relationship "children"

In the Nesting Cluster Diagram above, the inner Building Blocks are nested by the self-relationship "children". This results in the boxes "Infrastructure", "Procurement", "HR Mgmt" and "R & D" being nested into the box "Support".

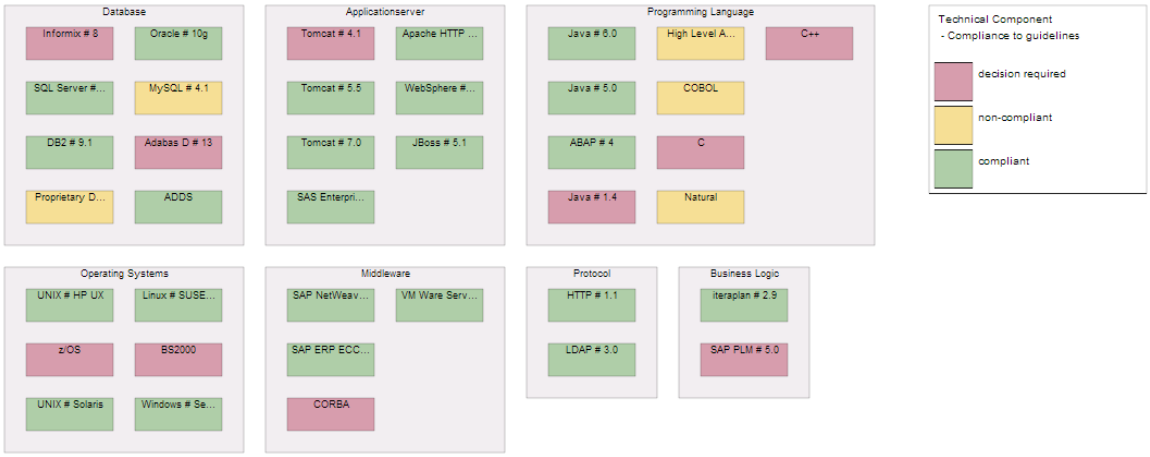
Of course the combination of recursively clustering the outer and recursively clustering the inner Building Blocks is also possible and leads to the following Nesting Cluster Diagram:



Nesting Cluster Diagram with outer and inner Building Blocks nested by a self-relationship

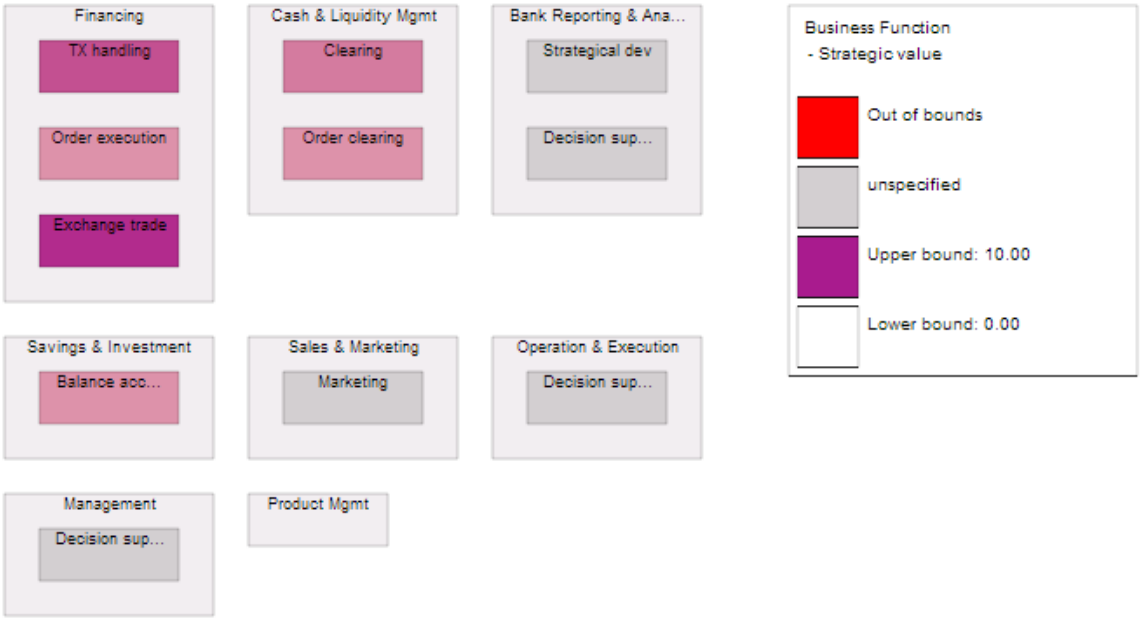
EAM Usage Examples

A common application of the Nesting Cluster Diagram is the technical blueprint. It groups all technical components according to their architectural domain and shows their compliance to standards:



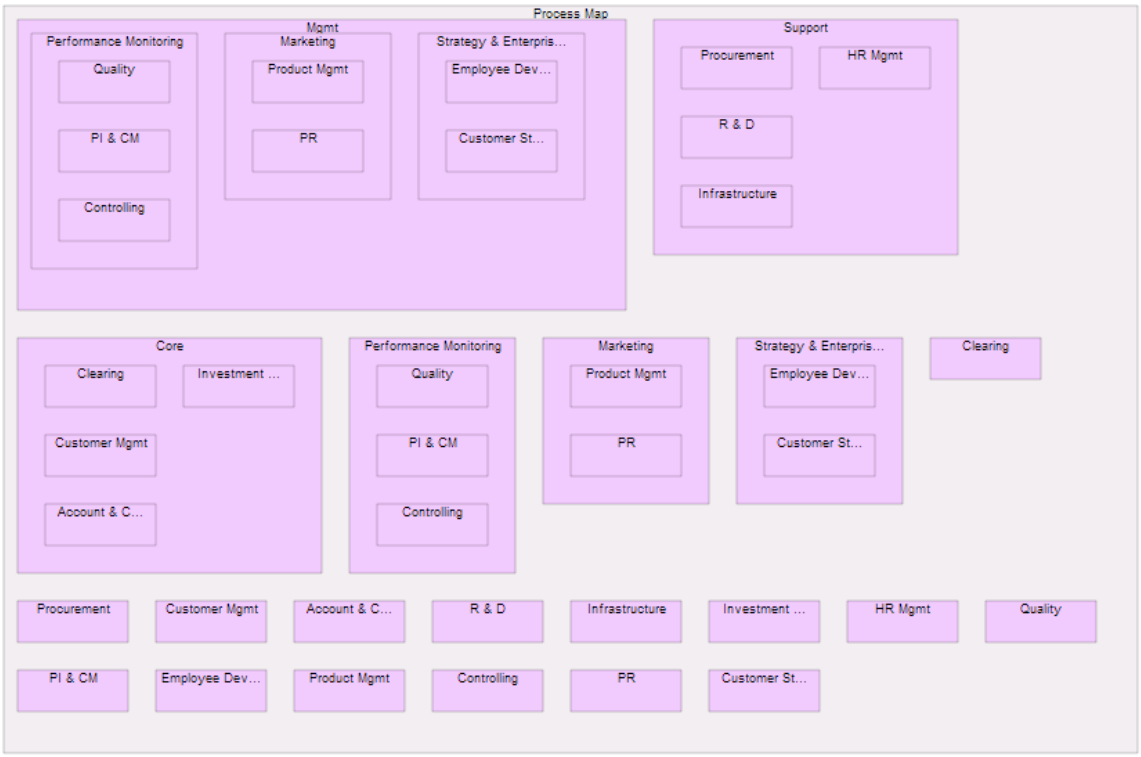
Technical Blueprint

A business domain model is another view which is based on the Nesting Cluster Diagram. Such a domain model i.e. depicts all business functions independent from business processes or business units:



Business Domain Model

On the business side, a Nesting Cluster Diagram can as well be used to create a process map, giving an overview of the first hierarchy levels of the business processes:



Process Map

Configuration

Simple Nesting Cluster Diagram

1. Select an outer Building Block Type by dragging it from the "Candidate outer-tags" onto the outer configuration box
2. Select an inner Building Block Type by dragging it from the "Candidate innter-tags" onto the inner configuration box

3. Generate the diagram by clicking onto the "Generate diagram" button.

Note that iteraplan automatically selects the most suitable relationship between the outer and inner Building Block Type as soon as you selected both of them.

Selecting Attributes of a Building Block Type to be displayed as Clusters

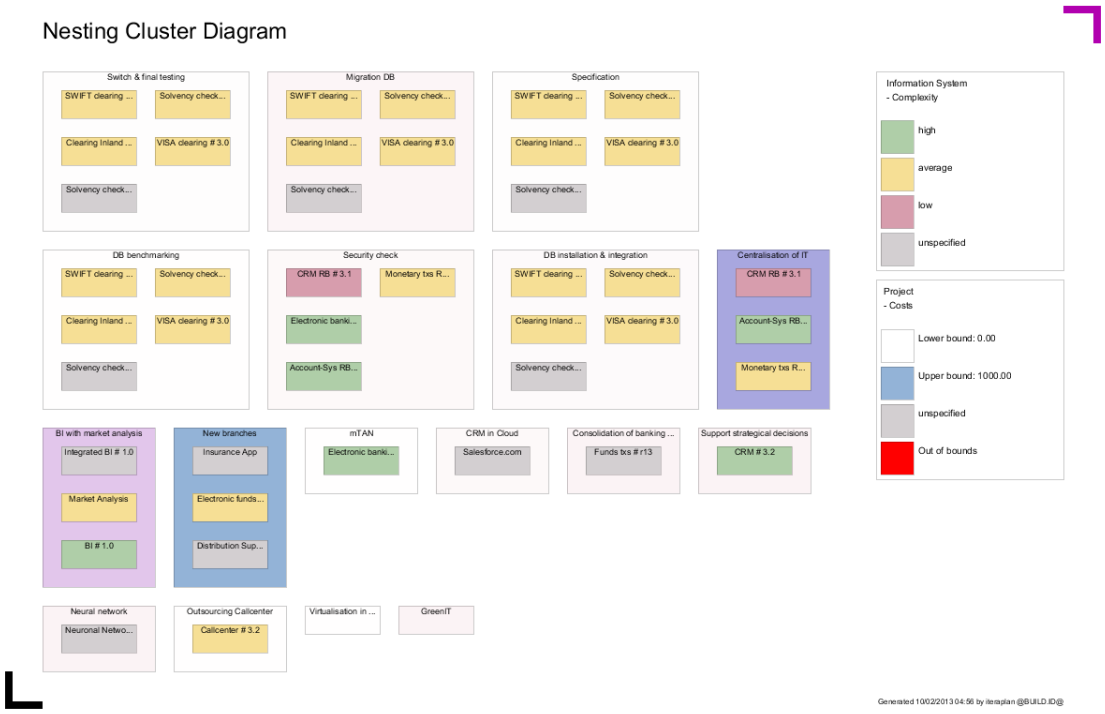
Apart from selecting a Building Block Type you may select an Attribute whose values should be displayed as boxes in the resulting Nesting Cluster Diagram

Optional Configuration Steps

The Nesting Cluster Diagram provides a number of additional configuration options, which are presented in the following sections.

Coloring

Like all other iteraplan diagrams, Nesting Cluster Diagrams can be configured to encode additional information through coloring of the elements of the diagram. For example, the diagram presented below shows Projects colored with regard to their costs and their related Information System Releases, coloured in accordance with their complexity:



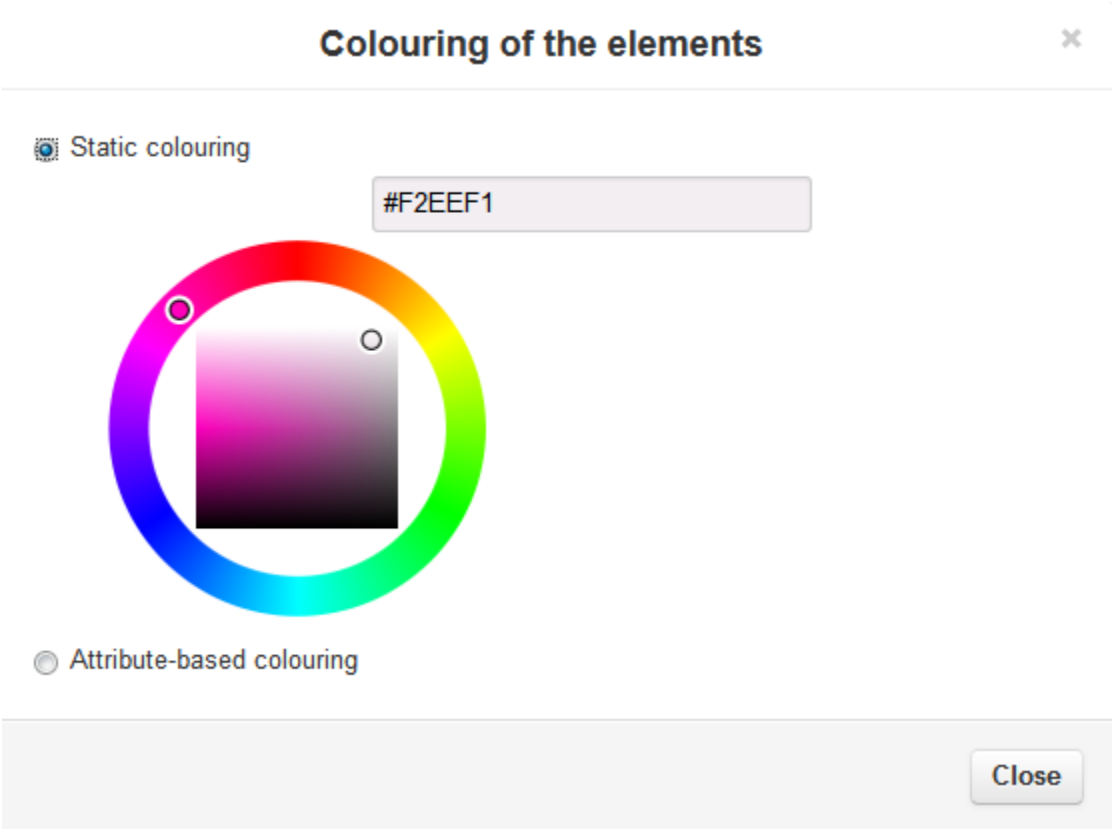
The coloring of elements of the Nesting Cluster Diagram can be accomplished after a Building Block Type has been dragged into its corresponding box, as usual. Separate color configuations are available for outer and inner Building Block Types, and can be accessed through the droplets in the upper left corners of the selected element configuration panels (see screenshot below).

Type Configuration Panel

Once the droplet is clicked upon, the color configuration panel is displayed and provides the user with the choice of using either static or attribute-based coloring.

Static Coloring

Static coloring is the default setting for both the inner and the outer Building Block Types. In this coloring mode, a single user-provided color is used for the coloring of all inner or outer Building Blocks in the resulting diagram. The particular color to use can be selected through the color chooser displayed in the color configuration page when the static coloring radio button is activated (see screenshot).



Static Colouring Configuration Panel

Attribute-Based Coloring

Attribute-based coloring can be activated by clicking on the corresponding radio button in the color configuration panel. Note that this additional radio button is only displayed when the selected Building Block Type has at least one admissible attribute: in the current version of the Nesting

Cluster Diagram, only numeric and enumeration attributes are admissible. Once the attribute-based coloring mode is activated, the user is presented with a drop-down containing all admissible attributes, and the first attribute in the list is selected automatically. The attribute used for coloring can be changed by selecting a different attribute from the drop down menu (see screenshot below). The configuration page is adjusted automatically, depending on whether the attribute is numeric or an enumeration.

Numeric Attribute Configuration

When a numeric attribute is selected, the following configuration options are presented:

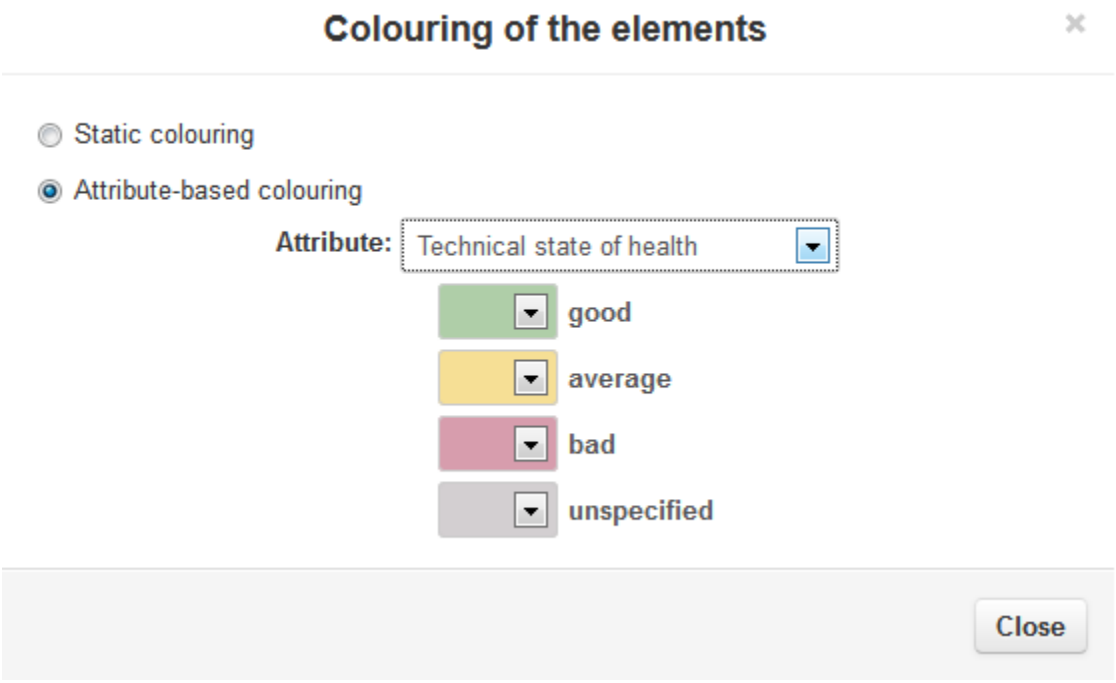
Configuration Options for Numeric Attributes

The first two color choosers - lower bound and upper bound - make it possible to specify a color range in which the lower bound corresponds to the lowest admissible value for this attribute, or the lowest value set over all Building Blocks, if the selected attribute has no specific lower bound set. The upper bound color corresponds with either a predefined or a set maximum value over all Building Blocks, accordingly. The out of bounds

color chooser can be used to highlight Building Blocks whose attribute value is above or below the maximal or minimal values defined for the attribute.

Enumeration Attribute Configuration

When an enumeration attribute is selected, the standard iteraplan color options are used to make it possible to select a color for each literal of the enumeration, as well as a color for Building Blocks for which no value is set with regard to the selected enumeration attribute. An example enumeration attribute configuration is depicted on the screenshot below.



Configuration Options for Enumeration Attributes

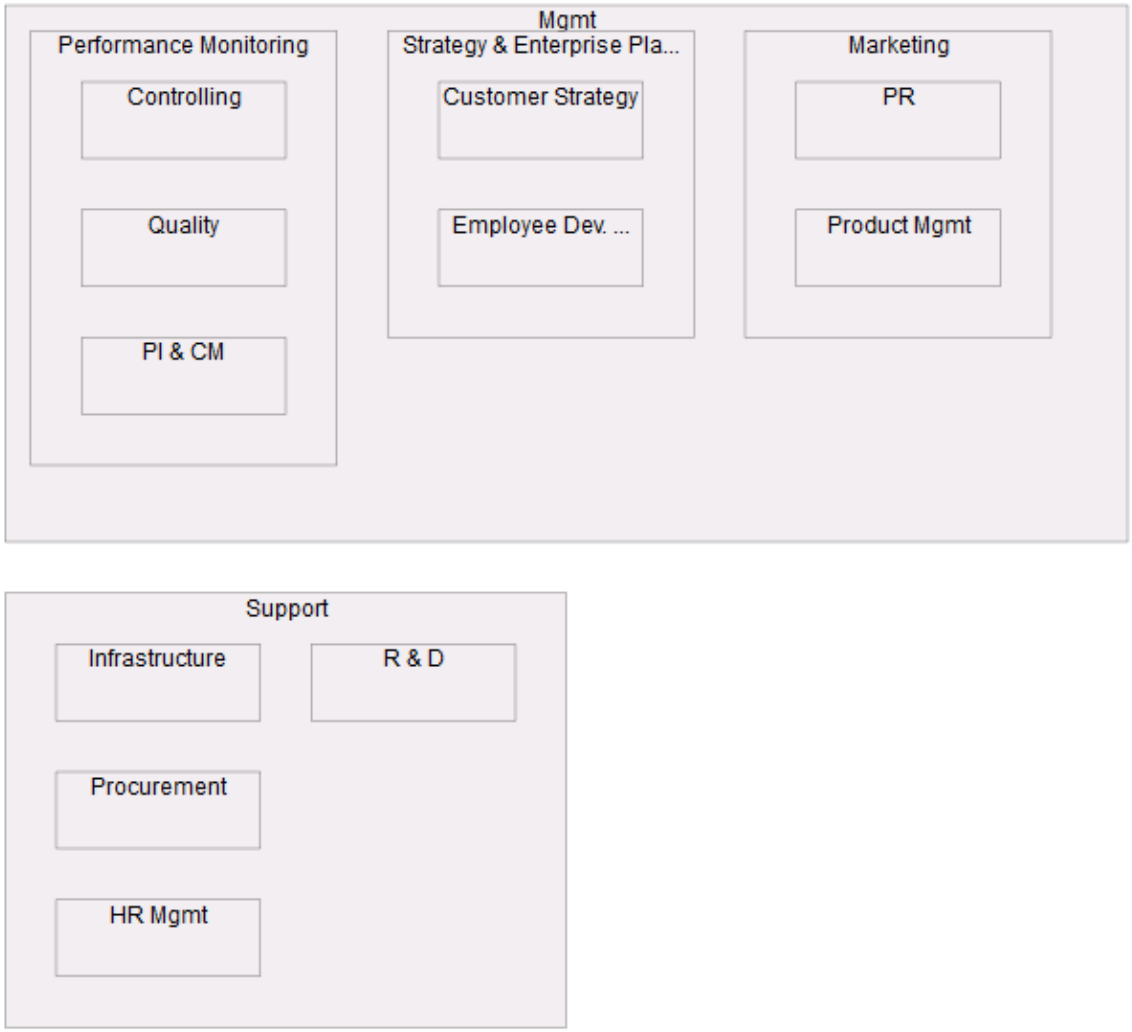
Filtering

You may filter the Building Blocks to be displayed within the Nesting Cluster Diagram as soon as you have selected a Building Block Type by dragging it onto a configuration box as usual. The resulting Diagram will only contain the Building Blocks that fulfill the filter criteria.

If you chose to recursively cluster the selected Type by a self-relationship the filter is applied recursively, preserving the self-relationship.

Example:

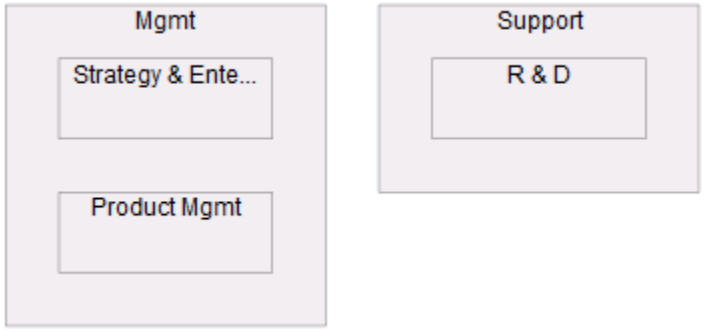
Consider the following structure of Business Processes (with regards to their self-relationship "children"):



Structure of Business Processes

Furthermore consider you filtered the Business Processes to just show "Mgmt", "Product Mgmt", "Strategy & Enterprise Planning", "Support" and "R & D".

If you chose to cluster Business Processes within a Nesting Cluster Diagram by their "children" relationship, the structure displayed in the Nesting Cluster Diagram will be as follows:



Structure of Business Processes with applied filter

Note that when applying a filter the nesting does not neccessarily represent the absolute nesting within the total set of Building Blocks (see "Product Mgmt" above). The nesting is preserved so that "Product Mgmt" remains a child of "Mgmt" although "Marketing", the original parent of "Product Mgmt", is no longer displayed. On the other hand, it is no longer guaranteed that nested boxes are on the same level within the hierarchy ("Product Mgmt" and "Strategy & Enterprise Planning" are not on the same level although the diagram suggests this).

Nesting by self-relationship

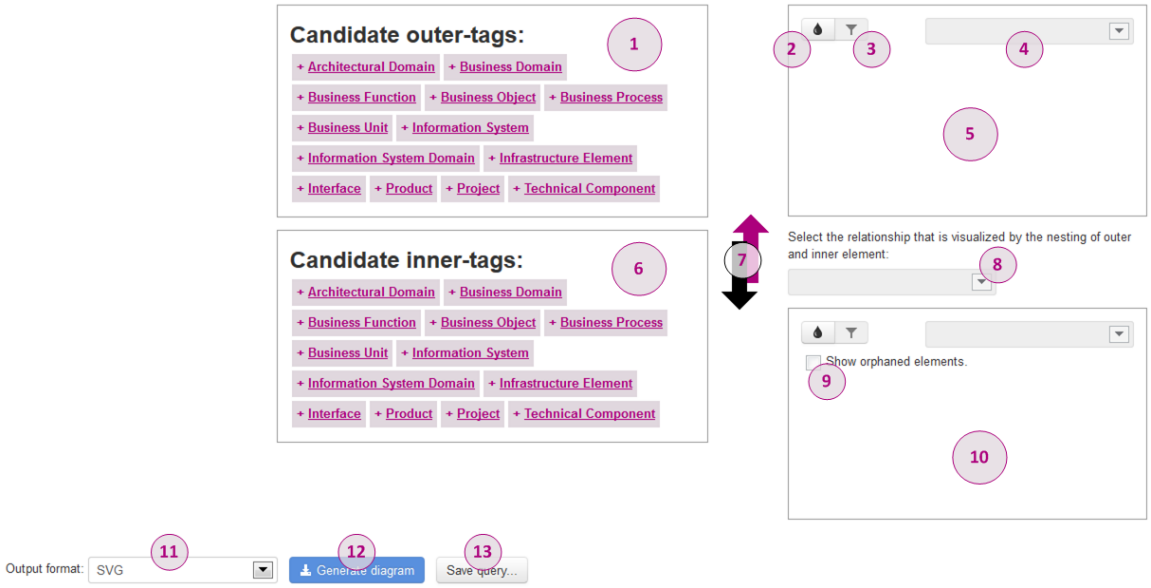
You may configure the Nesting Cluster Diagram to nest elements of the selected type with regards to a self-relationship by selecting the corresponding self-relationship.

Include orphaned inner Building Blocks

In some cases it might be of interest to include inner elements that are not connected to any outer element into the diagram. You may configure such a behavior by checking the corresponding checkbox in the configuration box of the inner type.

An additional 'outer' cluster will be drawn containing all inner elements that fulfill the filter on the inner element type but are not connected to any of the outer elements displayed within the diagram.

UI-Reference



Configuration Page for Nesting Cluster Diagrams

1. List of candidates to be used as outer element type
2. Coloring-Button
3. Filter-Button
4. Selection of the outer self-relationship to nest outer elements by
5. Selected outer element type
6. List of candidates to be used as inner element type
7. Button to switch configuration of the outer with the configuration of the inner element
8. Relationship between the outer and inner element type to be used for nesting inners into outers
9. Option to show orphaned inner elements within a separate cluster
10. Selected inner element type
11. Selection of the output format
12. Button to generate the Nesting Cluster Diagram
13. Button to save the configuration

Spreadsheet Reports

iteraplan provides various Spreadsheet Reports with detailed configuration options for matching reports to your needs, enabling you to provide in-depth information in response to wide-ranging questions concerning current, planned and target enterprise architectures. Spreadsheet Reports are a simple way to obtain an overview of the current status of data in iteraplan.

Currently supported export formats:

- **Simple list** - will be shown direcly in browser
- **Microsoft Excel**
- **XML**
- **CSV** (available only for information systems)

- **Microsoft Project 2003/2007 XML Format**
- **Gantt Project MPX Format**
- **Microsoft Project 2003/2007 XML Format** including Subordinated Buildingblocks
- **Gantt Project MPX Format** including Subordinated Buildingblocks

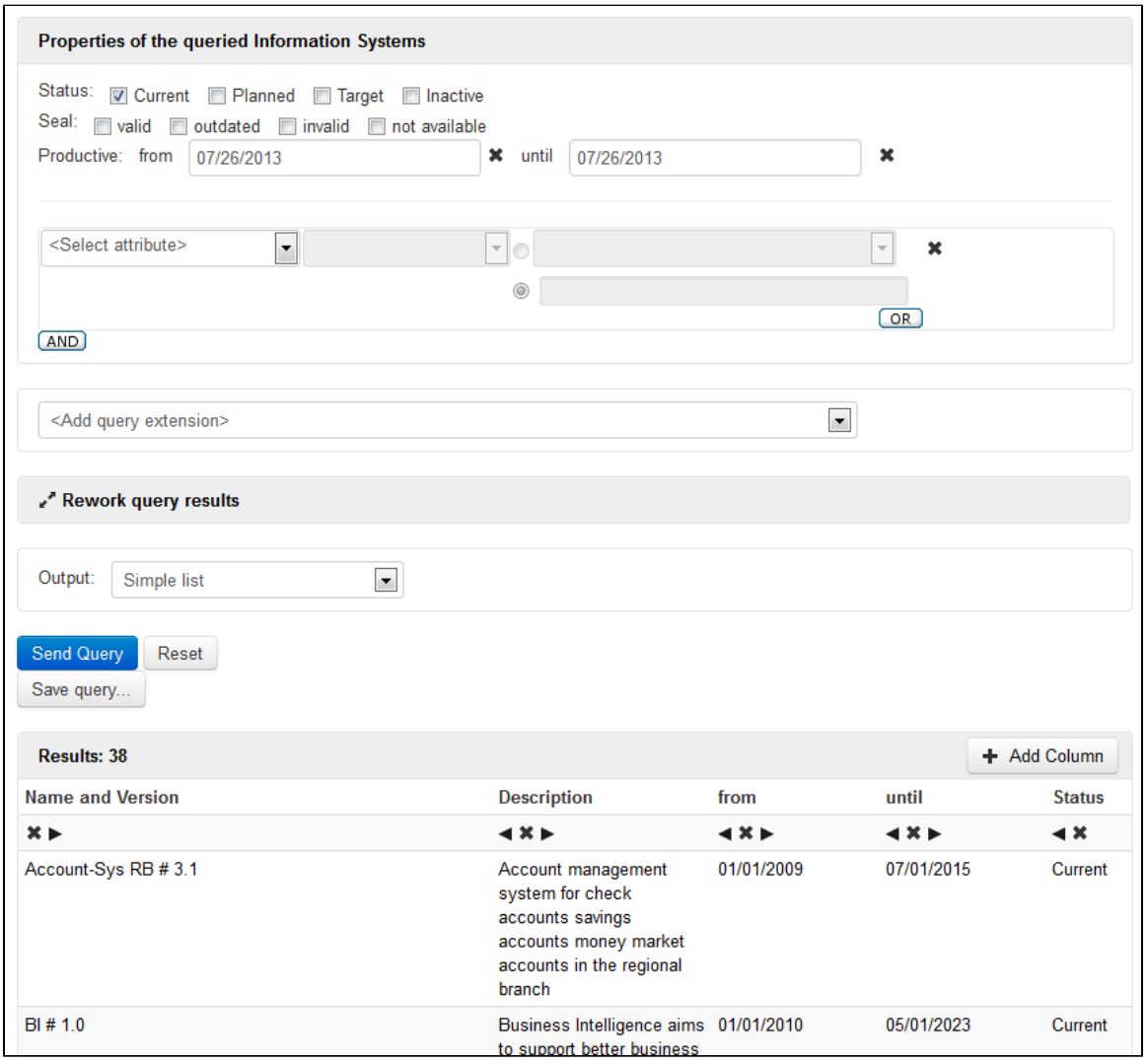
For a **full** model export and import, that is complete lists of all building blocks and all all types and relations, see How to use Import and Export an d REST API. These integration features are available in the Enterprise Edition only.

Formulating queries

The **Spreadsheet Reports** menu command opens the query manager. There is a separate query tab for each type of Building Block. The default tab shown when you open the query manager is **Information Systems**, but you can select the block you wish to work on by clicking its tab title. When you select **Information Systems**, the query manager returns *Information System Releases* as its results. One possible query could be:

> *According to their productive timespans, which Information Systems are in operation on 11/16/2011?*

The upper section of the form in the screenshot below shows this query; the lower section the query results. The query is submitted with the **Send Query** button.



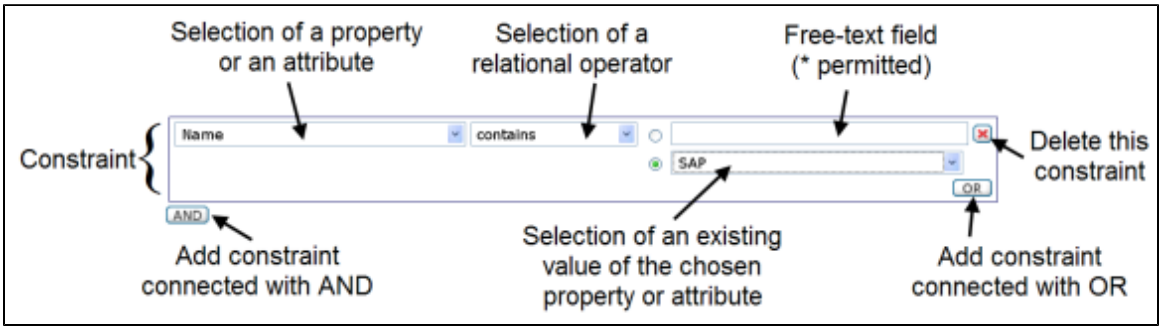
Query for Information System Releases which were productive on 07/26/2013 according to their productivity timespan settings

Defining result types

You can submit queries for any type of Building Block. Select the type of Building Block by clicking the appropriate tab at the top of the query manager page. In the example (see iteraplanDocumentation303:screenshot above), the **Information Systems** tab is selected.

Defining conditions

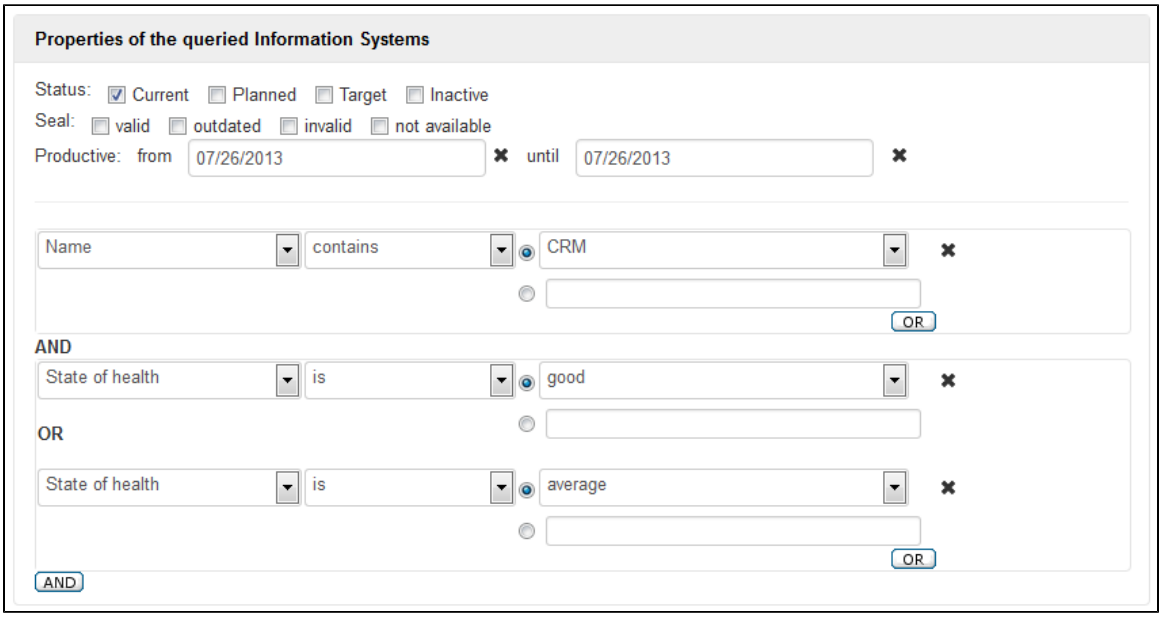
You can define conditions (constraints) to refine your query. Conditions can be specified for the Building Block on the active tab, and also for other blocks which are directly or indirectly related with those of the type under consideration.



Elements of a condition (constraint) for spreadsheet queries

In the section **Properties of the queried Information Systems**, enter the conditions which Building Blocks of the Information System Release type must fulfill. You can enter conditions pertaining to properties or attributes. If a Building Block has a status, you can also use the status as a condition: iteraplan will then return only Information System Releases which match the status settings. With multiple-value selection, the status values are linked with the logical OR. If you wish to use **productive from... until** information as part of your query, you can either key in the dates or select them from a calendar. If the time periods recorded for a Building Block of the queried type overlap the specified period by at least one day, the block will be included in the results provided it also fulfills the other conditions.

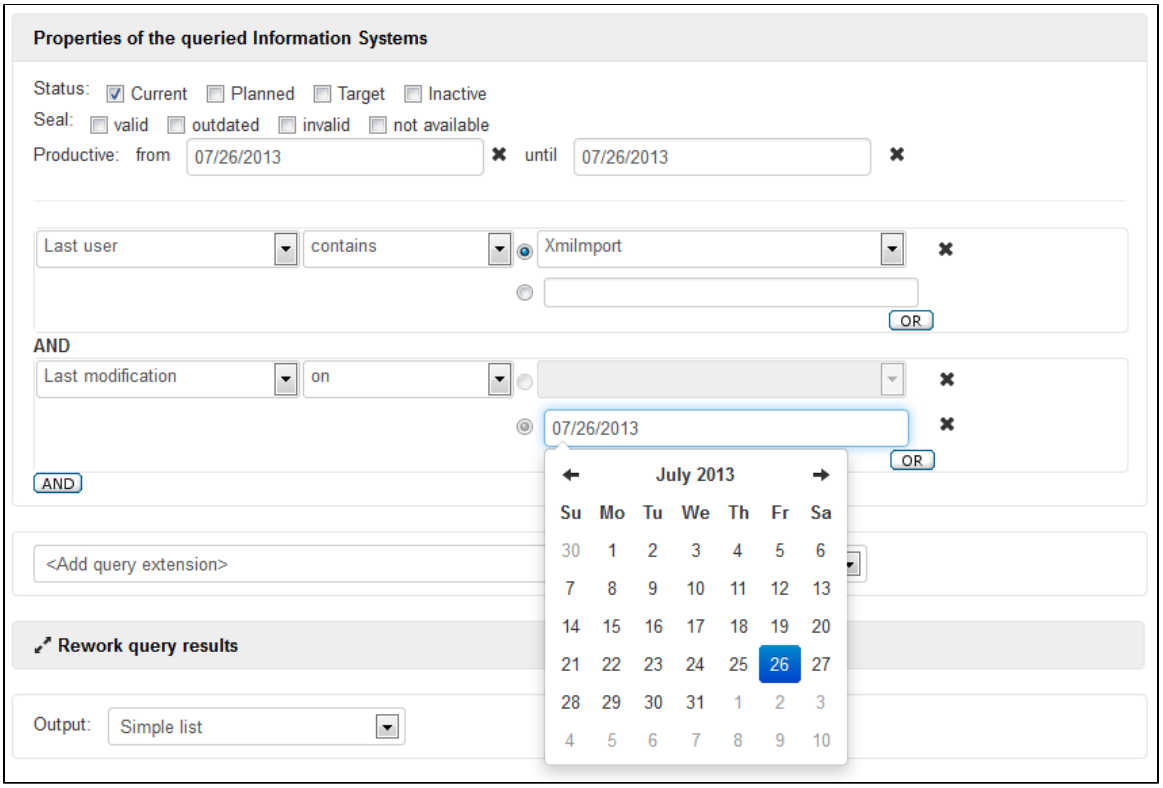
The drop-down list with the default entry **<Select attribute>** shows the list of properties and attributes. Use the AND and OR buttons to add conditions and define relational operators linking with the conditions already specified for the query. The button with the cross serves to remove the condition from the query.



Logical linkage of conditions for Building Blocks of the Information System Release type

Tracking last modifications in the system

Provided Last Modification Logging is activated for iteraplan (see Installation Guide), you can submit queries to track changes made to landscape planning elements. There are two special attributes available for this purpose: **<last user>** and **<last modification>**. For example, you can display all elements modified in the month of March, or all elements whose last modifications were carried out by a particular user.



Query pertaining to last modifications in the system

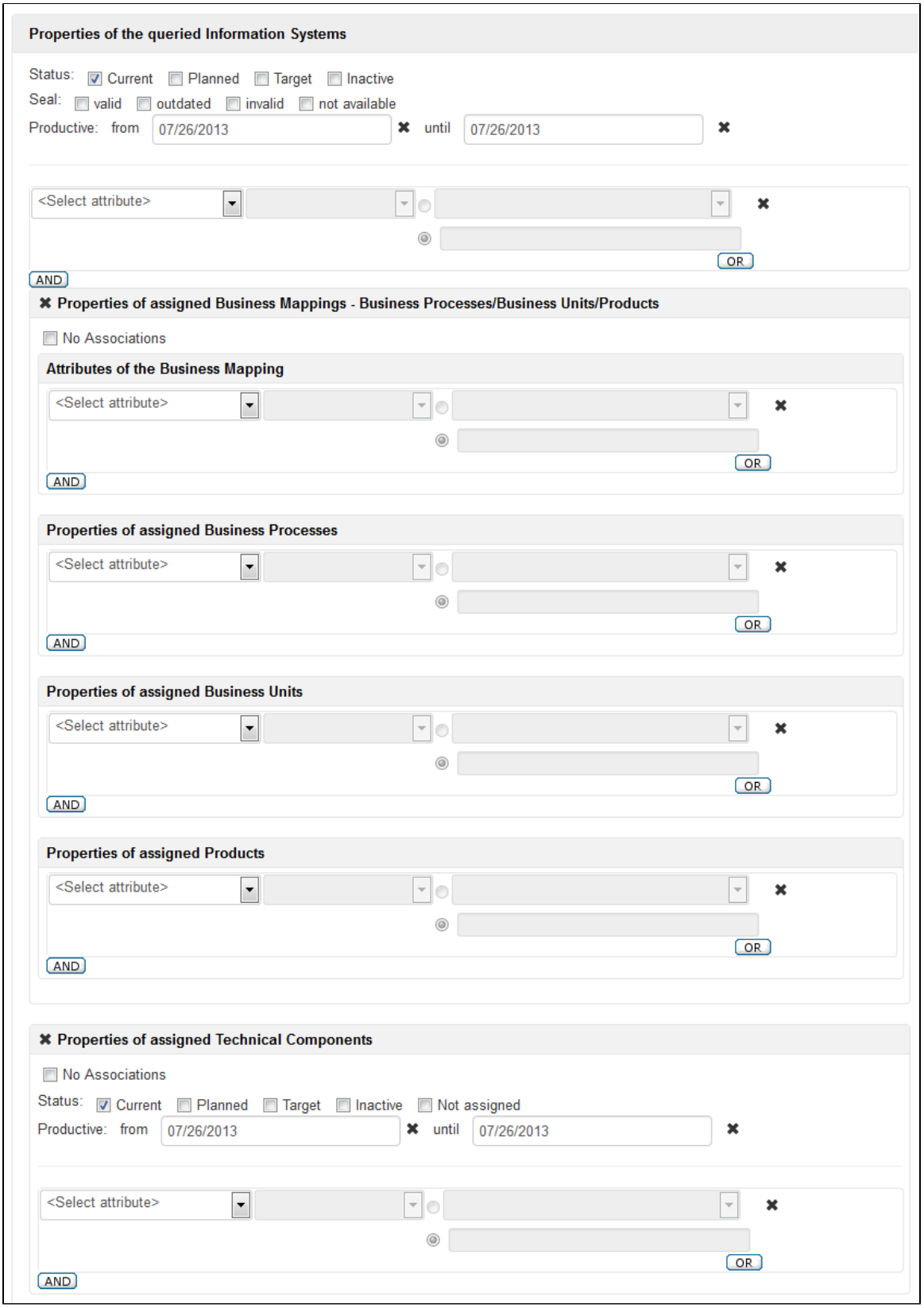
Defining query extensions

You can use the drop-down list with the default entry **<Add query extensions>** to specify conditions for directly and indirectly adjacent Building Blocks ("adjacent" in terms of their position on the iteraplan model shown in iteraplanDocumentation303:iteratec Best-Practice-EAM#_Figure2).

iteraplanDocumentation303:The screenshot below shows the extended query form after the query extension **Attributes of the Business Mapping** has been selected. You could now configure the query to retrieve only Building Blocks of the selected type (in this case Information System Releases) which support the Business Process "accounting". If, on the other hand, you were to check the **No Associations** box, iteraplan would then only retrieve Building Blocks of the specified type which specifically have *no* association with *any* Business Process.

> Bear in mind that checking the **No Associations** box means iteraplan will search specifically for "has *no* relationship". It is not equivalent to omitting the query extension.

Properties of assigned Business Mappings - Business Processes/Business Units/Products enables you to restrict the query by specifying properties of assigned Business Units, Attributes or the properties of Products assigned to the Business Process. This is a special filter option, available only for the Business Mapping query extension. Checking **No Associations** causes iteraplan to retrieve Building Blocks valid for all business units or for all products. Take note of the different semantics here – **No Associations** is not equivalent to omitting the query extension.



Query extended by Properties of assigned Business Processes and properties of assigned Technical Components

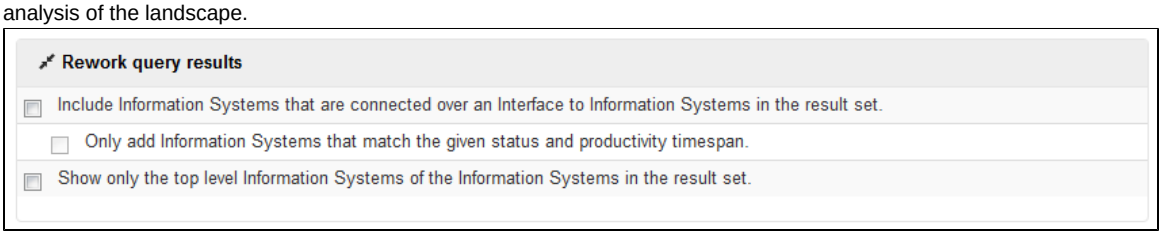
Another example shown in the iteraplanDocumentation303:screenshot above is a query extension for **Properties of assigned Technical Components**. Query extensions can be combined to suit your circumstances. Within a query extension, you can use the **AND** or **OR** buttons (or a combination of the two) to define logical links between multiple conditions.

iteraplan enables you to use multiple query extensions in each query. These are linked with the logical AND.

Editing query results

After submitting your query on Information System Releases, you can edit the set of results returned by iteraplan, i.e. refine the query for the Building Blocks in the results set. You can expand and contract the list of rework options by clicking the toggle button (plus or minus symbol) for **R ework query results**. There are two options available for reworking:

1. **Include Information Systems that are connected over an interface to Information Systems in the result set** Use this function to add all Information System Releases that have an interface with an Information System Release in the result set. This option can be valuable in tasks such as impact analysis. You can also restrict the scope of Information System Releases by specifying they must match the status and productive timespan specified in the selection.
2. **Show only the top level Information Systems of the Information Systems in the result set** This function projects all Information System Releases in the result set to their root-level releases in the hierarchy. The resulting abstraction can be useful for helicopter-view analysis of the landscape.

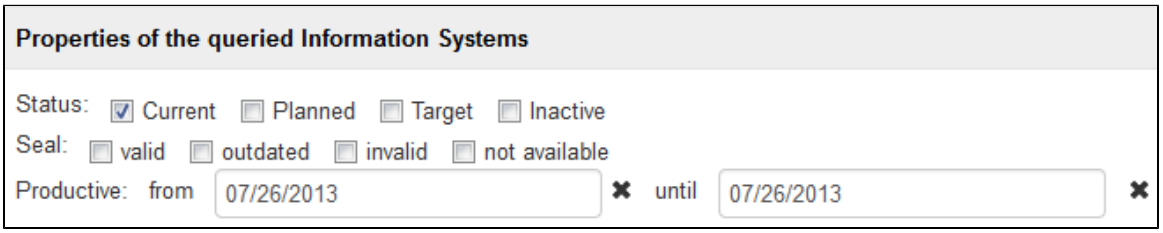


Optional post-editing of query results, selection of output formats

You have to submit the query again to display the fresh set of results. Since 2.8 you can combine these post-editing options, while so far you had to choose one of them.

Productive Timespan queries

When querying for Information Systems or Technical Components you can also enter the start and end date of the productive timespan:

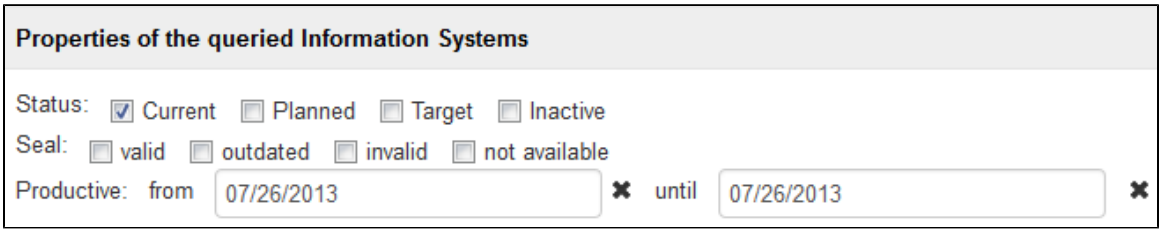


Productive Timespan for Information System queries

iteraplan interprets these timespan queries as follows: When you query for a productive timespan the result will contain all elements that overlap with this timespan, i.e. are active **at some point** during the queried timespan.This means that the entities can start **before** or **at some point** withi n the provided timespan and can end **after** or **at some point** within the provided timespan. Further the result set may contain entities, that have no start or end date, or no dates assigned at all.

Seal-based queries

new in 2.9
With iteraplan 2.9 you can filter information systems using seal information. More about seals read here.



Filtering of Information Systems using seal information

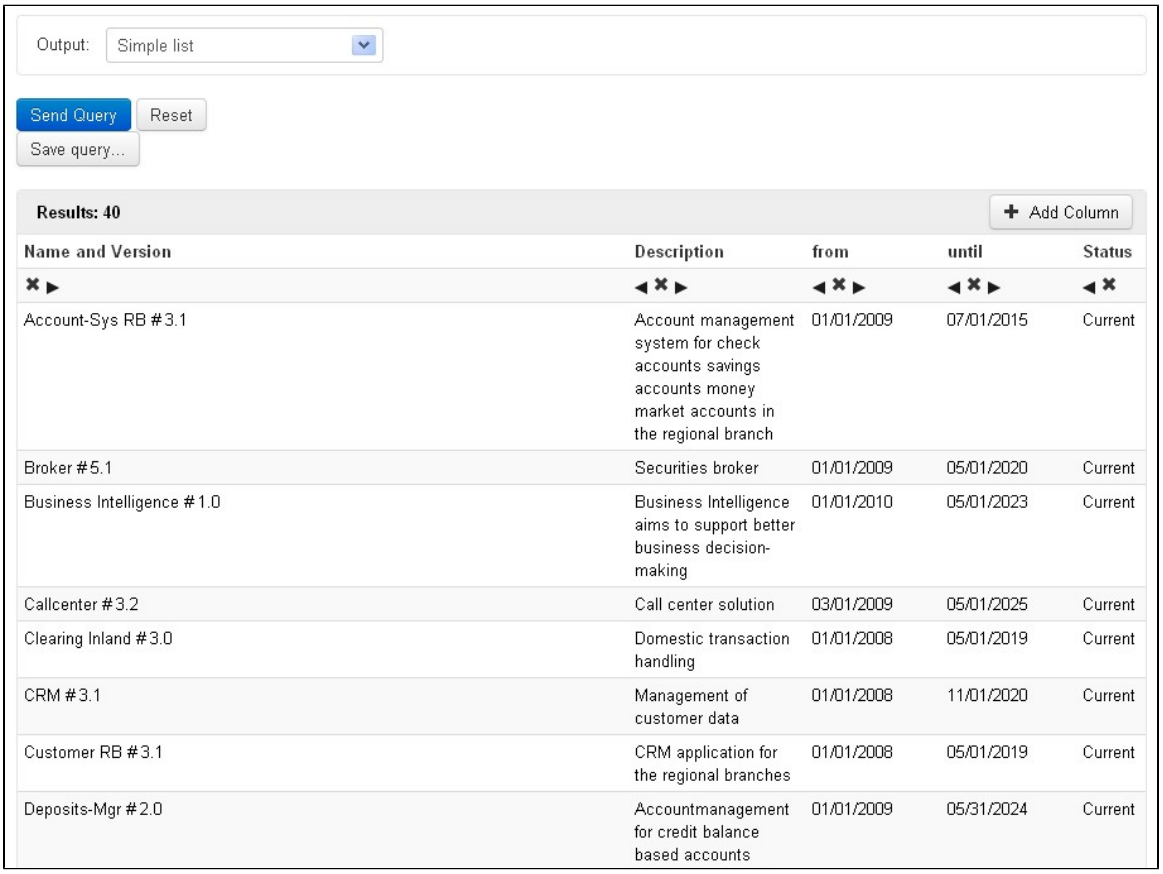
You can choose different values for the filter as it is shown in the figure above. If no value is chosen all possible values are considered, i.e. the bahaviour is the same as choosing all values.

Output formats

As well as displaying results in the web browser, iteraplan also enables you to export results in spreadsheet or CSV format. To display spreadsheet data, you must have the version Microsoft Excel 2000 or higher.

Simple list

The default output format is a non-hierarchical list in the web browser. The list contains only the most salient Attributes of the Building Block. This output format is available for all types of results. After executing the query you are able to adjust the result list on your special needs. It is possible to add additional columns to the result list by choosing them from the *Add Column* dialog. Unnecessary columns can also be removed from the result list by clicking the cross icon in the table header. Aside from that, you are able to rearrange the order of the columns via the black arrows located in the table header.



Name and Version	Description	from	until	Status
Account-Sys RB #3.1	Account management system for check accounts savings accounts money market accounts in the regional branch	01/01/2009	07/01/2015	Current
Broker #5.1	Securities broker	01/01/2009	05/01/2020	Current
Business Intelligence #1.0	Business Intelligence aims to support better business decision-making	01/01/2010	05/01/2023	Current
Callcenter #3.2	Call center solution	03/01/2009	05/01/2025	Current
Clearing Inland #3.0	Domestic transaction handling	01/01/2008	05/01/2019	Current
CRM #3.1	Management of customer data	01/01/2008	11/01/2020	Current
Customer RB #3.1	CRM application for the regional branches	01/01/2008	05/01/2019	Current
Deposits-Mgr #2.0	Accountmanagement for credit balance based accounts	01/01/2009	05/31/2024	Current

Results of Spreadsheet Report as a simple list

Microsoft Excel

You can export the report of any query to an Excel file. Figure "Excel export" illustrates what the exported Excel file looks like for a report on Technical Components. The landscape data pertaining to the result set is distributed over multiple worksheets in Excel. In this report, all the properties and attributes, and all the relationships, are included, grouped together on the various tabs for each Building Block Type.

Counting Enumeration Values
If you require the number of all exported Enumeration Attributes in one cell apply the following matrix formula for excel:

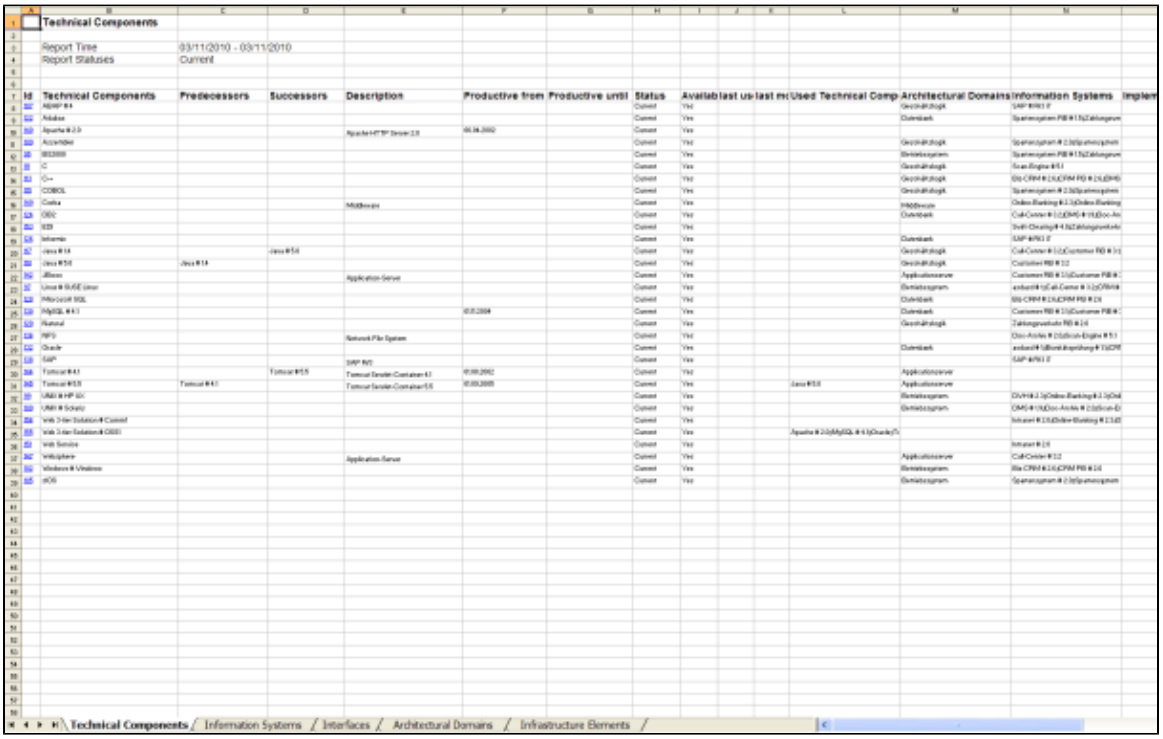
```
=SUMME((TEIL(N8;SPALTE(1:1);1)=";")*1)+WENN(N8<>"";1;0)} (for German-based Excel)
```

```
=SUM((MID(N8; COLUMN(1:1);1)=";")*1)+IF(N8<>"";1;0)} (for English-based Excel)
```

Use this formula as a matrix formula in Excel (Control+Shift+Return). N8 is the corresponding cell in your Excel sheet and ";" is the

separator.

The first column in an Excel report contains the ID of the corresponding Building Block. The hyperlink behind the ID links to the detail page of this Building Block.



Excel export for a query with results of the Technical Component Type

Offhand it is not possible to link directly to a Building Block although the hyperlink is correct. See our FAQs for more information on this issue.

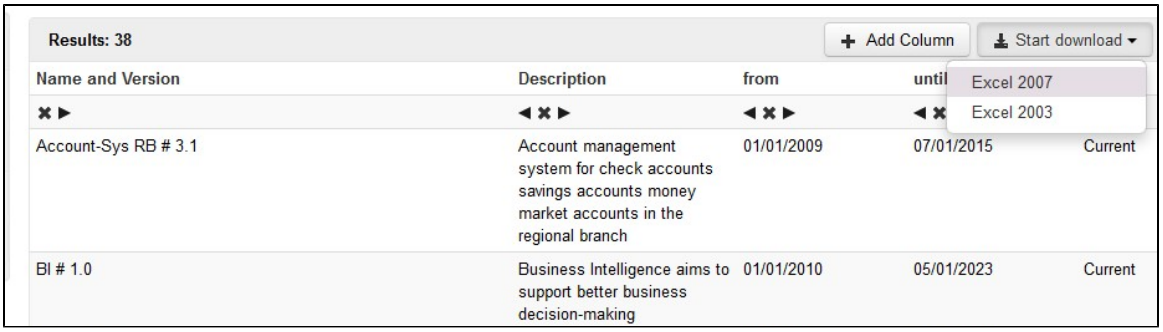
CSV

You can also export the output of Information System queries in CSV file format. This export file contains all properties, attributes and relationships in structured form.

List-Export as Excel

New since iteraplan 3.3: List-Export

Besides the normal report with Excel output format, you can create a Excel List-Export by clicking on the "Start download" button and selecting a Excel format.



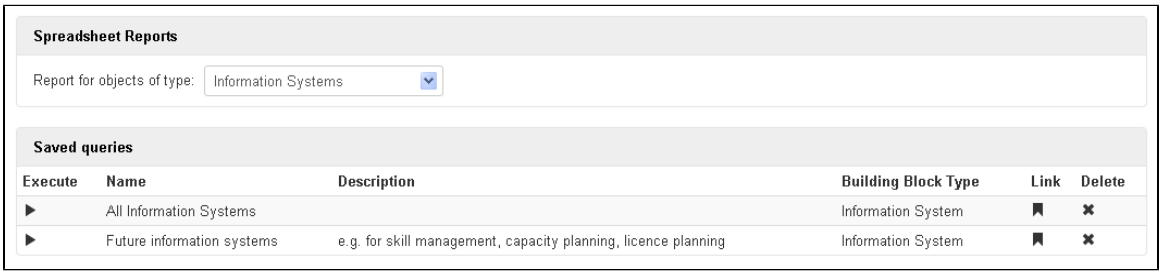
For details please refer to the page Building Block Lists and Search page.

Saving and loading Spreadsheet Reports

Similar to Diagram Reports (see Saving and loading graphical reports) each Spreadsheet Report has a submenu titled **Save query**. You must assign a unique name to a query in order to save it. An entry in the **Description** field is optional – this can be any text of your choice. Once a query has been saved, iteraplan presents it in the list of **Saved queries** on the home page for the type of Building Block in question. User-saved queries can be reloaded by clicking on the name of the query in the list. Should you wish to update an existing query with different parameters, save your modified query again under the same name. Finally, it is also possible to delete queries if they are no longer required.

Save query for Spreadsheet Report

To load a saved query, simply click on its name in the saved queries table on top of the tabular reports page.



List of saved queries

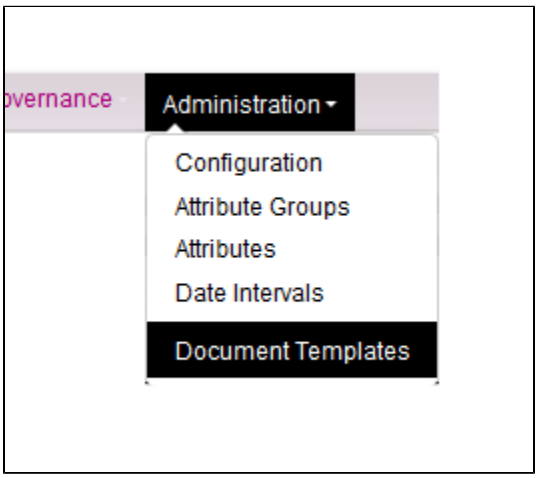
You can directly execute a saved query by clicking the play button in the execute column. You also can generate a link to the query result by using the bookmark button in the link column. If the saved query returns a graphics file, you can also use the link to embed the diagram in other pages, such as a corporate wiki.

Customizing Spreadsheet Reports

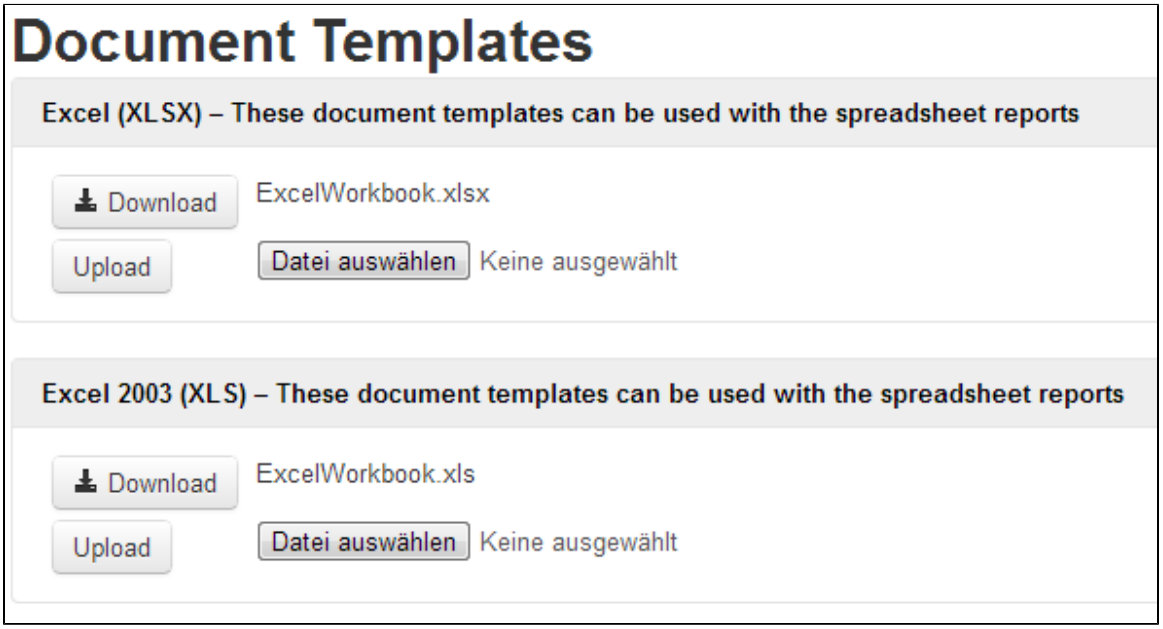
With the Enterprise Edition you can customize the Spreadsheet Reports by uploading templates for the Excel 2003 and/or Excel 2007 output

formats.

To achieve this please navigate to the "Administration" -> "Document Templates" section in the upper menu.



On this page you can either download the existing templates, upload new templates or delete your custom templates.



The uploaded templates can then be chosen during the creation of a Spreadsheet Report with an Excel output format via a dropdown box.

The template files are stored in the ***WEB-INF/classes/templates/excel*** directory.

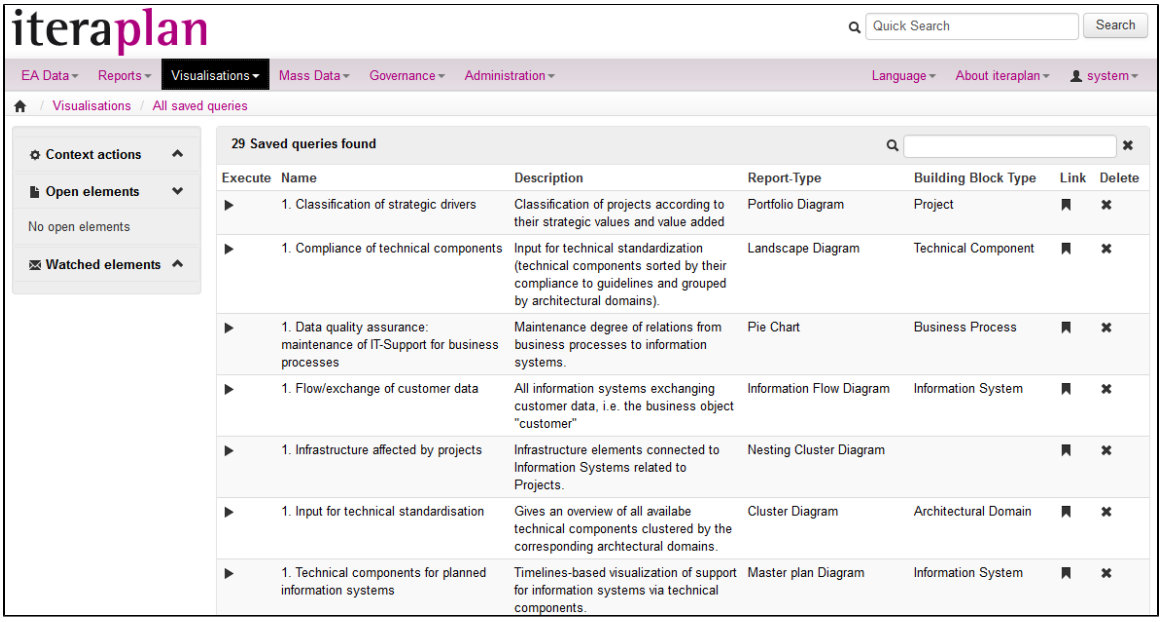
As a customization you could, for example, add macros to the Excel file, add sheets with prepared charts or create a custom overview page in your CI.
Please do not add sheets with the same name as the sheets that will be generated by iteraplan (e.g. "Information Systems", "Business Processes", ...).

View all saved queries

New in 3.3:

You can view, run and delete all saved queries (tabular and diagram) on one page.

When you open the Saved Queries overview in the menu with "Visualization/All saved queries" or "Reports/All saved queries", then you get to a page, where all saved queries are shown in alphabetical order.



On this page you can execute, load, link or delete the saved queries, according to your permissions.

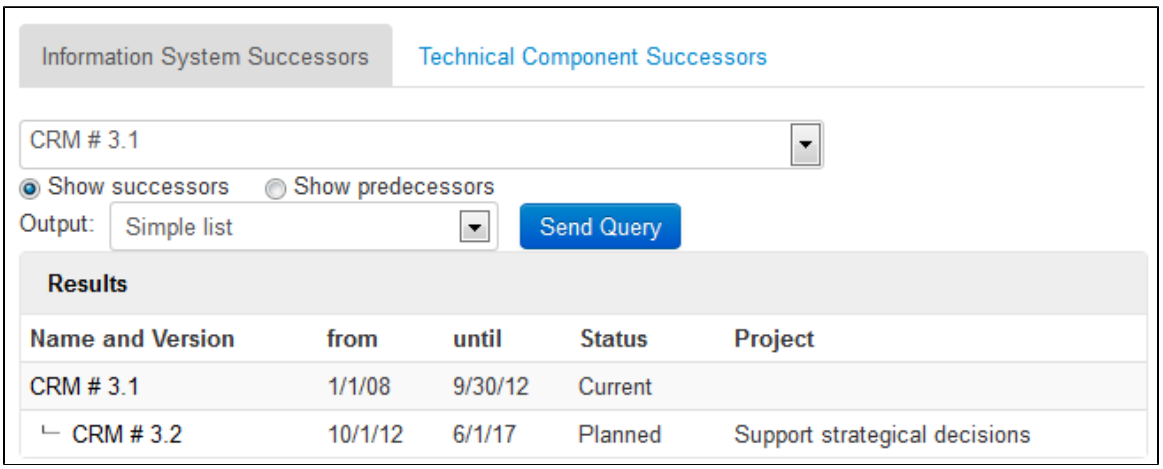
You can filter the list by typing into the input field on the right top of the queries list.

Successor Reports

iteraplan offers a reporting functionality to display the transitive successor graph for information systems and technical components. This graph is based on the predecessor-successor relationship which you can manage for these two building block types. You select the information system or technical component that you want to analyse from the drop-down respective box. These boxes do only list those elements which have at least one successor or one predecessor system. Hence, these drop-down boxes can remain empty if these relations are not maintained in your data.

The report is either returned as a table in your web browser (Output option *Simple List*) or as a spreadsheet file (Output option *Excel*) which you can open in Excel. The spreadsheet includes additional worksheets with information about adjacent building-block elements (see iteraplan meta model).

This screenshot shows the **Successor Reports** page with an example query.



Report about successor information systems and technical components

Consistency Checks

iteraplan essentially aims for maximum flexibility in data management and editing, the objective being to place as few constraints as possible on users. The overarching strategy is one of iterative refining and quality enhancement. iteraplan offers a range of **consistency checks** to help raise the quality of landscape data and verify that the data makes sense in terms of how it mirrors your business. You should perform these checks regularly so as to flag violations of functional or business consistency that can arise in the process of working with landscape data. All checks can also be accessed directly via webbrowser URL, so it is possible to link to results from external systems. iteraplan provides consistency checks for the following areas:

Information system landscaping

1. *Are there interfaces between releases of Information Systems which are not productive according to their dates at the same time?*

This consistency check tells you whether you need to conduct more detailed investigation of the interfaces and the information systems they connect. The productive timespans of the connected information system releases must match up.

For example, let us assume information system release A is productive from 15.10.2007 until 15.10.2008 and release B from 15.11.2009 with an open end date. An interface between these two systems would therefore never be productive.

2. *Are there releases of Information Systems that are longer in production, according to their dates, than their superordinate release?*

The productive timespans of superordinate and subordinate information system releases must match up. For example, it makes no sense in real-life business terms to have the productive timespan of a release longer than that of its superordinate release.

3. *Are there projects that are running at least x days longer than the starting point of operation of a connected information system?*

A project is an undertaking concerned with the rollout of a new release of an information system. This consistency check serves to verify that the project ends around the time the information system release goes into productive operation.

Not all the entries in this report necessarily indicate business-landscaping inconsistency. Some projects are in fact designed to last longer than the rollout date of a release: some include a maintenance phase, for example.

This consistency check expects a positive integer as an input parameter. If the parameter is omitted, iteraplan uses the default value displayed.

4. *Is there more than one release of an information system that has the status "Current"?*

Working on the assumption that only one release of an information system should have the status *Current* at any one time, it is advisable to verify that you do not have multiple current releases. Generally, multiple current releases point to inconsistency in business-landscaping terms in the information system releases managed in iteraplan, but this is not always indicative of a problem. For example, there might be overlaps when the enterprise conducts a release upgrade, or deliberate concurrent operation of two information system releases with different functionality or at different sites.

5. *Are there releases of information systems which are productive, according to their dates, but do not have the status "Current"?*

The productive timespan of information system releases must be kept consistent with the status. Sometimes you will have to adjust the status or productive timespan to ensure landscaping data remains consistent in real-life business terms. Two possible inconsistencies can arise over time.

Example 1: you will need to change the status of an information release from "*Planned*" to "*Current*" if it is being used in productive operation and its specified productive timespan is equivalent to current, productive use.

Example 2: if the information system release is still at "*Planned*" status and not yet in productive operation (despite this being indicated by the specified productive timespan), you can postpone the productive timespan of an information system release into the future.

6. *Are there Information Systems that have the status "Current" or "Inactive" but will, according to their dates, only be productive in the future?*

The status of information system releases should be kept consistent with their productive timespans in terms of business use.

7. *Are there Information Systems that were, according to their dates, productive in the past, but do not have the status "Inactive"?*

The status of information system releases should be kept consistent with their productive timespans in terms of business use.

8. *Are there any releases of Information Systems with the status "Planned", but without associated projects?*

A project is an undertaking concerned with the rollout of a new information system release. Each information system release at "*Planned*" status should be assigned a project in iteraplan, enabling users to obtain further information if necessary about the information releases being planned – e.g. to give pointers for more detailed analysis (project names, contacts, etc.).

9. *Are there any releases of Information Systems with associated projects, but with a status other than "Planned"?*

Like the preceding query, this check investigates the consistency between the status of the information systems and the projects (but works in the opposite direction,).

10. *Are there any Information Systems connected with interfaces which are not both active right now?*

This consistency check determines whether there are any information systems with the status "*Current*" which are linked by an interface with an information system at a different status (e.g. "*Planned*", "*Inactive*", "*Target*").

Technical landscaping

1. *Are there technical components linked with an information system which, according to their dates, not at all times productive during the production timespan of the information system?*

This consistency check identifies technical components which serve as the technical basis for an information system release but which are not used (or must not be used) throughout the release's entire productive timespan. Information system releases flagged by this consistency check must be investigated in greater detail. It might be necessary to move to a new release of a technical component or adjust the productive timespan for the component you are using.

2. *Are there releases of technical components that do not have a status?*

As a rule, technical components should be assigned a status in order to clearly indicate whether they are part of the as-is, planned or to-be landscaping. This consistency check shows all technical components which have the status "-".

3. *Is there more than one release of the technical components that has the status "Current"?*

Working on the assumption that only one release of a technical component should have the status "*Current*" at any one time, it is advisable to verify that you do not have multiple current releases with the status current. This is not always a sign of business-landscaping inconsistency. For example, different releases of a technical component can feasibly be in productive operation in different contexts.

4. *Are there releases of technical components which are productive, according to their dates, but do not have the status "Current"?*

The productive timespan of technical components must be kept consistent with the status. Sometimes you will have to adjust the status or productivity timespan to ensure the landscaping data remains consistent in business terms. Two possible inconsistencies can arise over time.

Example 1: the status of a technical component has to be switched from planned to current if this component is actually used in productive operation and the specified productivity timespan is equivalent to current, productive use.

Example 2: if the technical component is still at "*Planned*" status and not yet in productive operation (despite this being indicated by the specified productive timespan), you might elect to postpone its specified productive timespan into the future.

5. *Are there technical components that have the status "Current" or "Inactive", but will, according to their dates, only be productive in the future?*

The status of technical components must be kept consistent with the productive timespan. This check works in the same way as the one for information systems outlined in the previous section.

6. *Are there technical components that were, according to their dates, productive in the past, but do not have the status "Inactive"?*

The status of technical components must be kept consistent with the productive timespan. This check works in the same way as the one for information systems outlined in the previous section.

7. *Are there technical component sharing no architectural domains with their children?*

If a technical component uses another technical component, both should belong to the same architectural domain.

8. *Are there technical components sharing no architectural domains with their successors?*

Successor technical components should share at least one common architectural domain.

General landscaping

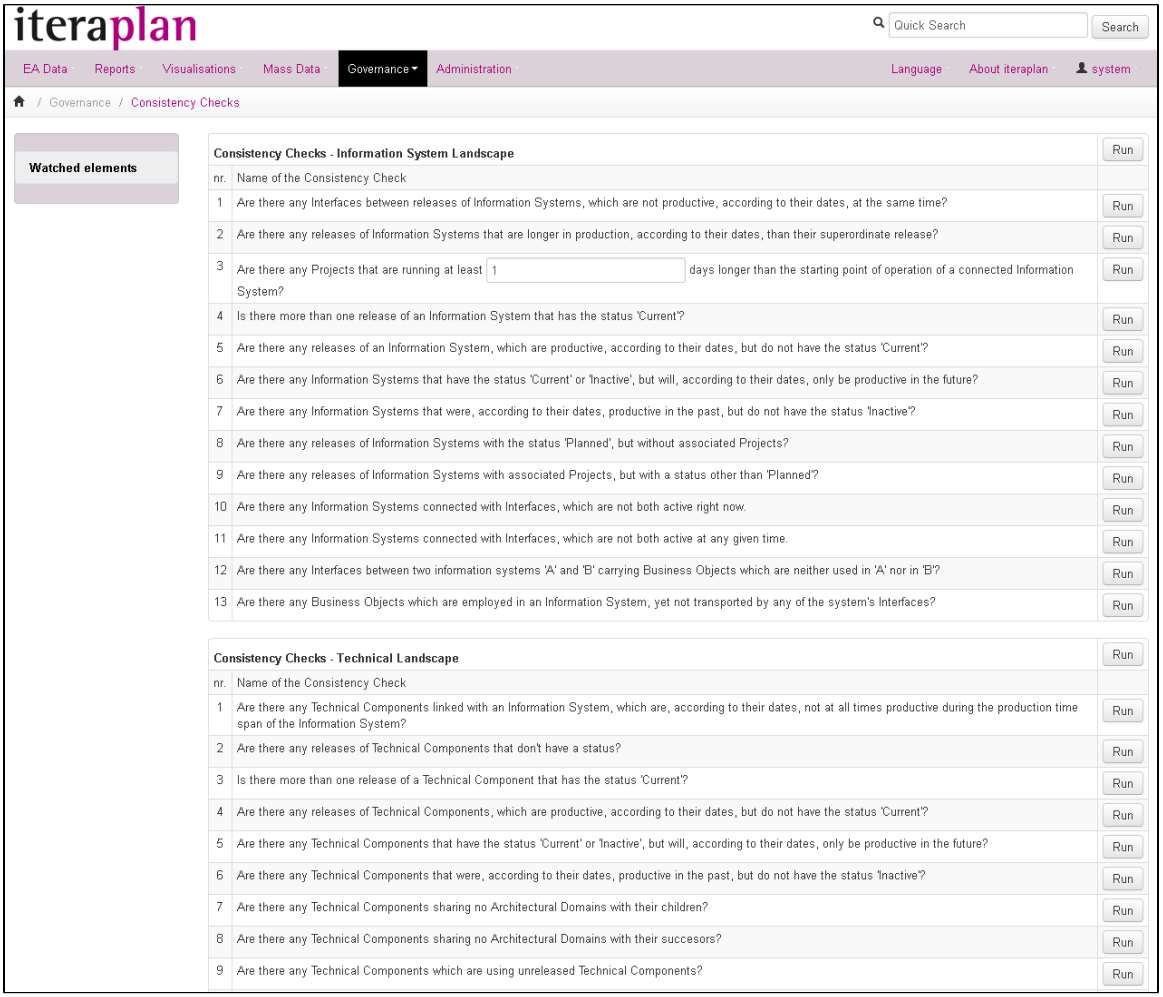
1. *Are there building blocks of type < drop-down list for building blocks> that do not have all mandatory attributes filled out?*

An attribute can be declared as mandatory. However when building blocks are edited, there is no check to verify mandatory attributes have in fact been assigned a value. This consistency check enables you to list building blocks which have mandatory attributes for which values are missing.

2. *Are there building blocks of type < drop-down list for building blocks> that have number attribute values which are out of bounds?*

A range of valid values – an upper limit and lower limit – can be defined for numeric attributes. However, when building blocks are edited, there is no check to verify attribute values are in range. This consistency check enables you to list building blocks which have out-of-range values assigned for numeric attributes.

The **Consistency Checks** menu command opens the Consistency Checks page. To execute a particular check, click its adjacent **Run** button. You can also execute all the checks of one area by clicking the **Run** button of the specific section. To run all consistency checks together, click **G enerate full report**. After running one or more consistency checks the results will be displayed at the bottom of the same page.



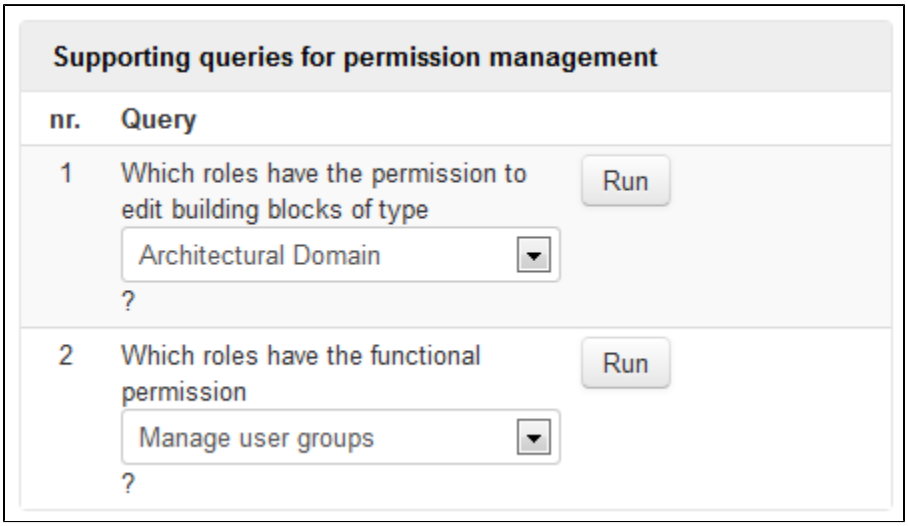
Verifying information consistency with consistency checks

Supporting Queries

The **Supporting Queries** menu command enables you to submit queries pertaining to assigned permissions. There are two queries available:

1. Which roles have the permission to edit building blocks of type <*drop-down list for building blocks*>?
2. Which roles have the functional permission < *drop-down list for functional permissions*>?

You execute these queries with the **Run** button. The query returns a list of the roles which have the designated permissions.



Supporting queries

Integration - Import and Export

The section below describes iteraplan's Import and Export capabilities.

How to use Import and Export

General Remarks

iteraplan supports two file formats for exporting/importing data and meta-model:

- XML Metadata Interchange (XMI) as standardized by the Object Management Group (OMG)
- Excel with the file format as described in Excel EA Data Format

Both formats can be used to export data as well as the underlying meta-model and to import it into iteraplan (preferrably with an "empty" iteraplan database). The XMI format is particularly designed for interoperability with other modeling tools. The Excel format in turn is intended for human data maintenance activities. In the Excel format sheets, columns, and rows can be adapted to fit the needs of the data maintenance activity at hand. In particular, bulk edits of a subset of the available data can be conducted using the Excel file format to export and re-import the updated data. The mechanisms for "round-tripping" are detailed subsequently.

Exporting Meta-model and Data

The export function of iteraplan is available in the main menu "Mass Data" / "Export/Import". iteraplan supports two distinct modes of export: *meta-model export* and *data export*. In the mode meta-model export only the structure of the data modeled in iteraplan, but no actual data is exported. The structure describes, which types, properties and relationships exist in iteraplan. Depending on the intended format (Excel or XMI) different buttons are used to trigger the different export modes, as the subsequent table shows.

	File format: *Excel*	**File format: *XMI***
Export mode: *meta-model*	"Download Excel Template"	"Ecore"
Export mode: *data*	"Download Data"	"ZIP (Ecore + Xmi)"

Importing Data

The import function is available in the main menu "Mass Data" / "Export/Import". To import data, use the "Import Data" section, choose a file, and click the "Upload" button.

A wizard screen guides through the steps of an import. During these steps iteraplan performs different checks on the supplied data with respect to format, content, and consistency.

> ⚠ **Concurrent changes**
> Changes done to the iteraplan database after the import wizard has been started and before the import has finished completely can lead to unexpected results or errors during execution of the actual data write.

The steps are:

1. **Read file and check plausibility of the data:** iteraplan checks whether the file is a correct data file, and whether it adheres to the iteraplan format definition (for the format of the Excel file see Excel EA Data Format). The checks especially ensure that the supplied data is internally consistent, i.e. that no "dangling" references or unknown values exist. In case an inconsistency is detected, iteraplan reports the detected issue to the user and cancels importing.
2. **Determine Metamodel changes:** iteraplan checks whether all properties described in the file are already present, or whether it is possible to add missing properties to the iteraplan meta-model. iteraplan presents a list of missing properties and allows the user to decide whether to apply the neccessary meta-model changes, i.e. create the properties, or cancel the import process.
3. **Write Metamodel changes:** iteraplan creates the missing properties, if any. The newly created properties are not assigned to attribute groups, but are created in the default attribute group.
4. **Validate data:** iteraplan compares the data to import with the data already present in iteraplan. Different checks are applied to detect any inconsistencies, i.e. duplicate names, that would arise from importing data into the iteraplan database. iteraplan presents the result of the comparison to the user, who can then decide to proceed with the next steps or to cancel the import. Note: Canceling the import at this point does not roll-back the meta-model changes applied in the previous step.
5. **Write data:** iteraplan applies the data changes identified in the preceeding step to the database.

Above five step process is used to make sure that the user always knows which changes are applied to the meta-model or database of iteraplan.

Therefore, the user has to acknowledge each step by checking the information given by iteraplan and clicking "Next" to go to the next step.

Steps 1-4 are usually executed fast (seconds for smaller or medium sized imports, still within only a few minutes for larger data sets).

Step 5 may take some time, especially for larger data sets. Aside from concurrent edits to the database or technical difficulties, e.g. caused by a server outage, successful completion of step 4 ensures that step 5 will also succeed. Therefore, the user should carefully follow steps 1-4, such that step 5 can run unattended.

Import Strategies

There are several ways how the data in the imported file can be interpreted and different changes will be applied to the iteraplan data depending on the selected import strategy, see Import Strategies.

Additional information/technical limitations

Updating element hierarchy
It is not recommended to mix updates to the elements' hierarchy with other updates, like renaming or adding new elements.

Furthermore, a switch of hierarchies, i.e. A is parent of B to B is parent of A, cannot be achieved by a single import run, even though there may not be an error shown in the validation step (step 4). This is also true if B is a more distant descendant of A.

Workaround:

The Tree View of the corresponding building block type can be used to switch hierarchies. Furthermore, it is possible to perform two separate import runs, i.e. the first run to remove B from parent A and the second run to attach it otherwise.

Limitations

- Due to technical constraints there is currently no way to create attributes of the type "Responsibility Attribute" by adding the attribute to the excel file to be imported. When exporting landscape data or downloading the template, each responsibility attribute will be exported as text attribute. Importing data like this into a database where the according responsibility attribute already exists works correctly, but if the attribute does not exist already the attribute type and its values will be ignored: the attribute is not added to the building block type and the attribute values are not imported.
- Adding a new building block type or relationship by adding an according sheet or column to the excel file is not possible and the attempt will result in an according error message.
- You cannot change whether an attribute is mandatory or not with the excel import.
- The ordering of the children of a hierarchical element may be lost after importing.

Import Strategies

One of several import strategies has to be selected before a file can be imported.

When using the import from the GUI, a select field for the strategies is displayed next to the upload button.

When using the import via REST, a parameter can be used to select an import strategy.

As default, the additive import strategy is selected, which corresponds to the import behavior in iteraplan 3.2.

Additive Import Strategy

The additive import strategy is designed to add data to iteraplan.

You can update or insert data, but not delete any data from the iteraplan database. Changing a "to-one" relationship, e.g. "parent", will nevertheless overwrite the old value with the new one.

Updating building blocks works as described below:

There are two distinct cases when updating attribute values:

1. **single value assignment attribute:** The importer can change the value of the attribute to another value, or set the value if it was

previously unset. It cannot remove the attribute value to go back to the unset state. If the according value field in the file to import is empty it will simply be ignored.

2. **multi value assignment attributes:** The importer can only add additional values to the list of current values (or leave it unchanged). If, for example, the attribute "Accountability" for Information System "BI" is already set to "Alice" and "Bob" in the database, importing a file with "Accountability" set to "Alice" and "Max" does not remove the entry "Bob", but simply adds "Max" to the list. Result: "Alice", "Bob" and "Max"

There are also two distinct cases when updating relationships:

1. **to-one relationships**: These are relationships which, in an Excel template, are represented by a column on the sheet of the building block type they belong to, for example the parent relation. Each building block can have at most one parent.

   > **Attention**
   > These relationships are a special case, as, in contrast to single value assignment attributes, you can not only change, but also unset values. In the example of the parent relationship, this means the importer can remove the parent of a building block, to make it possible to move a building block to the hierarchical top level.
   >
   > However, both changing an assigned building block in such a relation to another building block, as well as the removal of the assignment will only be performed by the import, if all involved building blocks are listed in the import file. This means not only the changed building block and the newly assigned one, but also the formerly assigned building block need to be present in the import file.

2. **to-many relationships**: These are relationships where a building block can be connected to a list of several other building blocks by the same relation, for example the "based on these Technical Components"-relation of Information Systems. The handling of such relationships by the importer is similar to the handling of multi-value attributes described above: The importer can only add additional connections between building blocks, not remove existing ones. If your database contains a connection between Information System A and Technical Components B and C, but the Excel file to import only contains a connection between A and another Technical Component D, rather than replacing the existing connections A-B and A-C, the importer will add the connection A-D to the already existing A-B and A-C.

   > **Special Case**
   > There is one case where the connection between two building blocks can be removed by the importer.
   >
   > Example: There are Information Systems A, B and C in your database. A is the parent of B. The importer can change B's parent to C, which also results in B being removed from A's children.

Overwrite Strategy

The Overwrite Strategy is made for updating data in iteraplan, but also for adding new building blocks or deleting existing ones.

- If an existing building block is not present in the imported file, it will be deleted
- Existing building blocks will be updated so that their new state is as described in the imported file.
  - Relationship assignments such as "parent" in Information System or assigned Information Systems in Business Functions will be updated to represent the state described in the imported file. This means that connections existing in iteraplan, but missing in the imported file, will be removed when importing.
  - Attribute values will be set as they are in the imported file. Assignments of attribute values not mentioned in the imported file will be removed in iteraplan.
- Building blocks described in the imported file, which are not present in iteraplan, will be created.

> **Exceptions to updates and deletions**
> - When importing a previously exported (and possibly modified) file, building blocks will only be deleted or updated when they were not changed since the time of the export. If the file to import does not contain information about the time of the export, such as files manually created from the downloadable Excel template or exported files from older iteraplan versions, all changes will be applied.
> - Removal of assigned attribute values or building block connections will only be done if the according attribute or relationship as such is mentioned in the imported file. If, for example, an attribute column and relationship sheet in an Excel file are removed completely, the import leaves values of that attribute and assignements in that relationship untouched in iteraplan.

Excel EA Data Format

The Excel file format represents the meta-model of iteraplan, i.e. the types, properties and relationships backing the data stored in iteraplan.

An Excel file consists of the following kinds of sheets:

- An "Introduction" page: carries an iteraplan logo, the date when the file was exported, and the version number of iteraplan used for the export.
- One sheet per type, e.g. one sheet for Business Domains, one for Information Systems
- One sheet per relationship type, e.g. one sheet for Business Mapping and other relationships that hold properties
- One sheet per relationship, e.g. one sheet for the relationship between Technical Component and Information System
- One sheet per enumeration type, e.g. one sheet listing the admissible status for the state-of-health of Information Systems

Important: any additional sheets existing in the Excel file potentially lead to errors during the import.

General structure of a sheet

Each sheet consists of two sections, the "header" (rows 1 to 7) and the "body" (starting at row 8). The header supplies the name and the description of the kind of element described in the sheet. In particular, any column used in the sheet, is described in the header. A column can either represent a property of the corresponding (relationship) type or a "to-one" relationship to another type. The hidden rows in the header section (rows 3 to 5) contain technical information that details the used meta-model. The contents of these rows should not be altered and neither the rows nor their content should be removed to ensure integrity of the data file. In case of changing the ordering of columns, please make sure that the hidden rows are always moved alongside. The importer only uses the values of the hidden rows to assign values. The headings in row 7 are provided for readability only.

Type Sheets and Relationship Type Sheet

The body of a (relationship) type sheet is structured along following rules:

- Text, number, and date properties are described in a single column as plain values; Properties of type "runtime period" are split in two columns (start and end) which hold plain date values.
- Properties of type "responsibility", or other enumeration typed properties which can have a multi-value assignment are given in one cell. The single values are separated with a semicolon (";").
- Columns holding a relationship to another type, e.g. another building block type, reference the corresponding entity via the value of its "name" property.

Note: A relationship to another entity can only be established, if this entity is also described in the same Excel file; otherwise, the file would be considered inconsistent and would therefore be rejected upon import.

Relationship type sheets adhere to the same structure. Sheets of such type are for example used to describe business mappings.

Relationship Sheets

Relationship sheets describe relationships between types. A relationship sheet supplies only two columns, each one referencing to one of the types that are related. Each row in the body of a relationship sheet describes a relationship between the two entities referenced by the names supplied in the row. In this sense, the rules for referencing are the same as for the "to-one" relationships described in the type sheets.

Enumeration Sheets

Enumeration sheets describe enumerations, i.e. are used to define the values an enumeration-valued property can take. The body of an enumeration sheet supplies for columns. The first column ("persistent name") supplies a unique technical name for the corresponding value. The columns "name", "description", and "abbreviation" supply the screen name, the descriptive text and the short name for each defined value.

Consistency

It is important that an Excel file which is uploaded to iteraplan for import (see How to use Import and Export), is **consistent**. This in particular applies both to relationships as well as to enumeration-valued properties. For a relationship, the referenced entity must also be described in the Excel file. For an enumeration-valued property, the Excel file must define all used values and assign unique technical names. Another consistency rule applies to the values used in the columns "name" and "id", respectively. The values supplied in these columns must be unique per type, i.e. no two Information Systems having the same id or name may exist in the same Excel file.

An Excel file obtained from iteraplan either via "Template Export" or "Complete Data Export" (see How to use Import and Export) contains all types. Further the set of described entities is complete, such that the file is consistent.

In case an Excel file is exported and re-imported to perform a bulk update of only a subset of the types and properties, we highly recommend to adhere to following procedure:

1. Obtain a complete export using "Complete Data Export"
2. Remove all sheets that contain (relationship) types and relationship, which should not be updated. Please make sure that relationship sheet are removed, whenever a sheet containing one of the referenced types is deleted.
3. Remove all columns that contain information, which should not be updated. Further, remove all columns that describe "to-one" relationships, of which the corresponding target type was removed in step 2.

XMI EA Data Format

File-Format

The content of a valid XMI file schould look as follows:

Headers

```
<?xml version="1.0" encoding="UTF-8"?>
<iteraplan:Container xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:iteraplan="urn:iteraplan">
  <!-- put your data here -->
</iteraplan:Container>
```

It is strongly recommended, that the imported XMI file follows this structure. Otherwise the import will fail!

Content

```
<contents xsi:type="iteraplan:InformationSystem" id="1" name="Example IS #
1.0" description="Examplary information system" </-- other attribute values
for type InformationSystem -->/>
<contents xsi:type="iteraplan:InformationSystem" id="2" name="Example IS #
1.1" description="Examplary information system with reference to parent
system" parent="#1" </-- other attribute values for type
InformationSystem-->/>

<contents xsi:type="iteraplan:BusinessUnit" id="1" name="Example BU"
description="Examplary business unit" </-- other attribute values for type
Business Unit-->/>

<!-- other data (as "contents") -->
```

Each line of the XMI file defines a single Object, its attributes and its references as they are defined in the corresponding ecore file.

The content of the XMI file can be edited or enlarged by additional Objects. Thereby it is strongly recommended, that every referenced Object is defined in the XMI file as well. In the example above, the content defines two **Information Systems** and one **B usiness Unit**.

References can only be resolved if the referenced instance is defined within the same file.

To ease modifying of XMI files it is useful to use an just exported XMI file as a basis for editing. The complexity of the development of an XMI file from scratch can increase rapidly.

Use-Cases

Editing data:

You can edit the content of XMI files using a XML editor. Importing an edited XMI file will update all affected persisted objects by overriding the instances with the same ID.

> ⚠ Please only change IDs in order to solve conflicts between XMI data and already persisted objects. Otherwise it could be possible that the import leads to undesirable changes. When changing an ID inside the XMI file, you have to assert that the change is made for every occurrence or the import may fail.
>
> If an object is referenced by another one, you have to edit the ID in this reference as well!

Creating new data:

Creating XMI files with a XML editor is an easy way to create new **Building Blocks**, **Attributes values** and objects of all other classes defined in the iteraplan meta model. Before starting the import-process, you should assure that every recommended attribute (an attribute is recommended if its definition in the Ecore file defines a "lowerBound" of "1") is set for all created instances.

When creating data, be sure to leave the id attribute blank (Note that no other attribute may be left blank). If the id attribute were set in the XMI, iteraplan would update existing objects matching those IDs rather than creating new ones.

During the import, iteraplan always assigns the next unused ID to each created object. Whenever objects are created in this process, you should download the latest XMI file before applying any further changes.

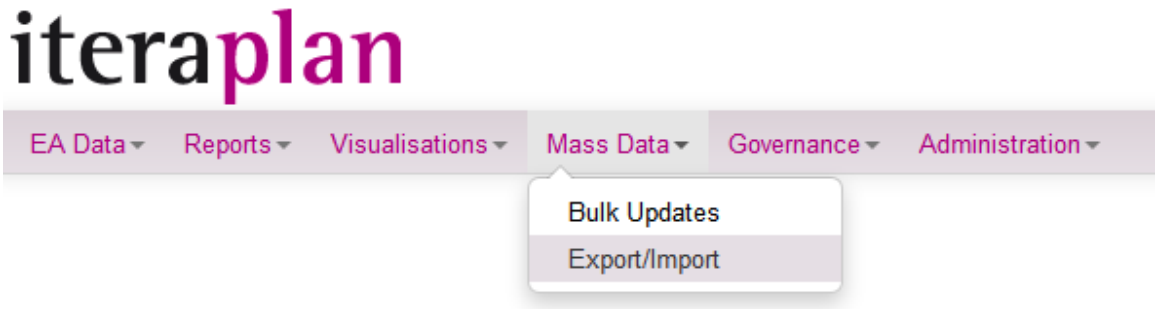
Partial Import/Export

Introduction

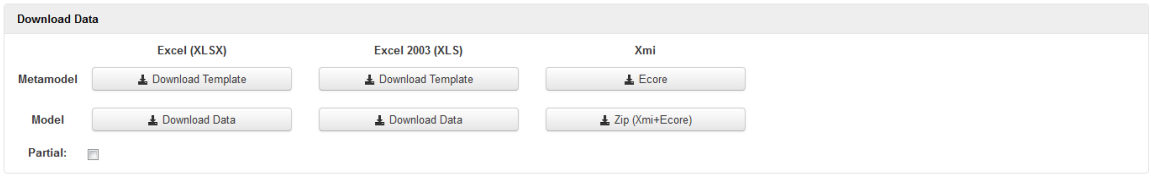
One of our aims is to provide our users with the ability to maintain and manipulate their data conveniently even outside iteraplan. Among the most accessible and common ways to do this is the representation of the enterprise architecture as a Microsoft Excel file, in which different sheets cover aspects of the architecture such as types or relations. While iteraplan has provided this feature all along, it has now been improved so that is not only possible to obtain the total of the user's data at once, but also to obtain an extract of the data tailored in accordance with the user's current needs. These data excerpts can, for example, be sent to responsible persons for data maintenance. Once the responsible persons have edited their corresponding Excel files, these can be imported back into iteraplan by the person in charge, to produce a unified updated picture of the enterprise architecture. This part of the iteraplan user's guide addresses the steps of this process in detail.

Creating a Partial Export

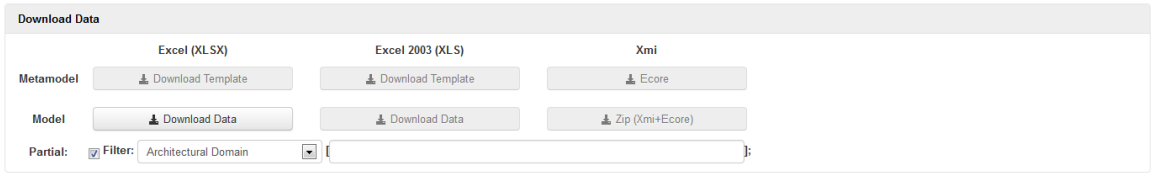
The first part of the described scenario is the creation of a partial Excel export. This can be accomplished through the standard iteraplan Import/Export Menu available under Mass Data.



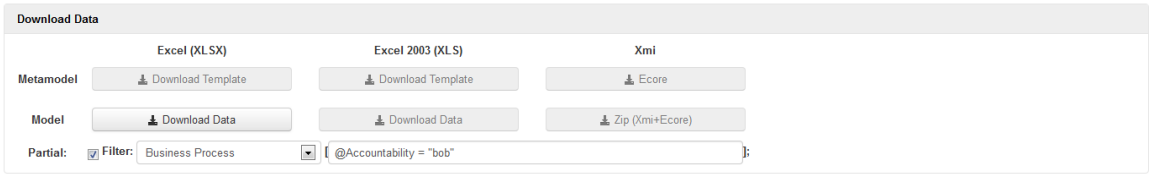
Once the page is opened, the user can activate the partial functionality by enabling the check box 'partial' under the buttons for the different export formats.



When the partial mode is activated, the partial export options are presented to the user, as depicted in the figure below.



In the initial configuration, the partial export is based on the Architectural Domain building block type. The building block around which the export is centered can be changed by selecting a different type from the drop-down menu. Furthermore, the export can be additionally refined by specifying an iteraQl filter condition in the text box provided to the right of the drop-down box. For example, the configuration



will specify an export in which only the Business Processes for which bob is responsible are contained. Note that while arbitrary iteraQl statements can be used, we recommend to use filters based only on local features of the specified building block type.

After those few steps, the configuration of the partial export is complete and the Excel file can be downloaded. Note that once the partial configuration is enabled, the only available download is in the XLSX Excel format.

Excel File Structure - What to Expect

As with the original iteraplan Excel export, the file may contain sheets representing different aspects of the enterprise architecture's metamodel:

- Type sheets: Sheets containing instances of building block types. Each row contains an instance of a building block type, with its properties.
- Relationship sheets: Sheets representing relationships between types of building block. Each row contains a pair of instances of the respective building block types.
- Enumeration sheets: One of these sheets is provided for each enumeration used by a type sheet. The sheet contains the enumeration values, or literals, of the respective enumeration.

In partial exports, the number of type, relationship and enumeration sheets is greatly reduced. Consider the Excel file obtained in the last section:

iteraplan

Date	4.3.2014
Version	3.3
Main export type	BusinessProcess[@Accountability="bob"]

Types	
Business Process(Accountability	Business Process[Accountability="bob"]
Business Mapping (BM)	Business Mapping

Relationships	
BD-BP	(Business Domain) - (Business Process)

Enumeration Attributes

Note that the file contains only two type sheets: one for Business Processes and one for Business Mappings, and a single relationship sheet, for the relationship between Business Processes and Business Domains. Recall that the partiality of the export was defined on the basis of the Business Process type of building block.

In general, a partial export is constructed by iteraplan through the following rules:

- One sheet is created for the type of building block selected by the user as the basis of the partial export. This type is denoted as **main export type**.
- If a filter is provided, the **main export type** is additionally restricted in accordance with the filter. Note that not specifying a filter is equivalent to specifying a filter which allows all instances through.
- If the **main export type** has association building block types related to it, all of these also obtain sheets.
- All relationships of the **main export type** are each provided with a sheet.
- Finally, sheets are created for all enumeration attributes of the exported types.

In the example above, the Business Process (filtered by accountability) is the main export type. The Business Mapping sheet is included because of the Business Mapping being an association type of building block. Finally, a single relationship sheet is included, as the Business Process type has only this one relationship to Business Domains.

Manipulating Data

Having obtained a partial Excel export, the next step of the example scenario is the actual data maintenance. In this aspect, the partial Excel file is no different than the original full Excel file provided by iteraplan and modifications can be carried out on both type and relationship sheets as described below.

In type sheets, the modification of the value of a cell represents the modification of the feature designated by its column for the building block instance represented by the corresponding row. The deletion of a row in a type sheet represents the deletion of the building block instance. Bear in mind that the deletion or renaming of instances can cause a consequent import attempt to fail, if the instance is referenced in another sheet. Finally, new rows in a type sheet are interpreted as newly created instances of the building block type.

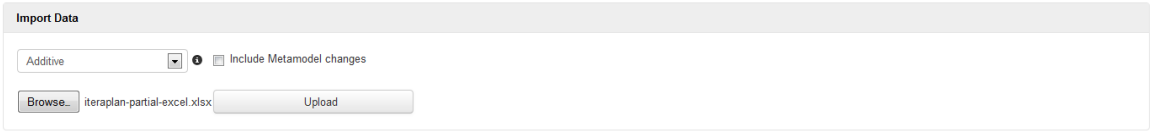
In relationship sheets, each row represents an association between two building block instances. The deletion of a row in a relationship sheet thus represents the removal of the relation between the two instances. An update of one of the cells of a row represents the update of the association, i.e. the old association is replaced by the new one. Finally, a new row represents a new association between two building block instances.

Remarks:

1. The semantics of the modifications to the Excel sheets described here are valid under the assumption that the Excel file is interpreted as an enterprise architecture model. In this sense, while the described operations have the provided meaning in the scope of the Excel file itself, their interpretation when importing them into iteraplan may deviate, depending on the import strategy selected. To illustrate this, consider updating one of the cells of a row in a relationship sheet. The intended meaning of this operation is to *replace* the old association with the new one. Nonetheless, if the Excel file is imported with an Additive strategy, the old association will *not* be removed. Instead, both the old and the new associations will be available in iteraplan.
2. The partial Excel file contains a number of hidden sheets, rows and cells. These should not be modified, since they contain important context information, necessary for the interpretation of the visible data when importing the Excel file. Modifying this part of the Excel file may corrupt it and render it un-importable, or might have undesirable side-effects.
3. With renaming building blocks using the import, we strongly advice to do this in an extra import run with no other changes, using the additive mode. Please also take care to adjust the names in cells referencing the renamed building block, for example in the parent column of the building block sheet or on relationship sheets.

Importing a Partial Excel File in iteraplan

Once an Excel file is modified in accordance with the user's demands, the data can be re-imported into iteraplan. This is achieved by adding the file through the Browse button of the Import Data section. iteraplan will automatically detect that the uploaded file contains a partial Excel export.

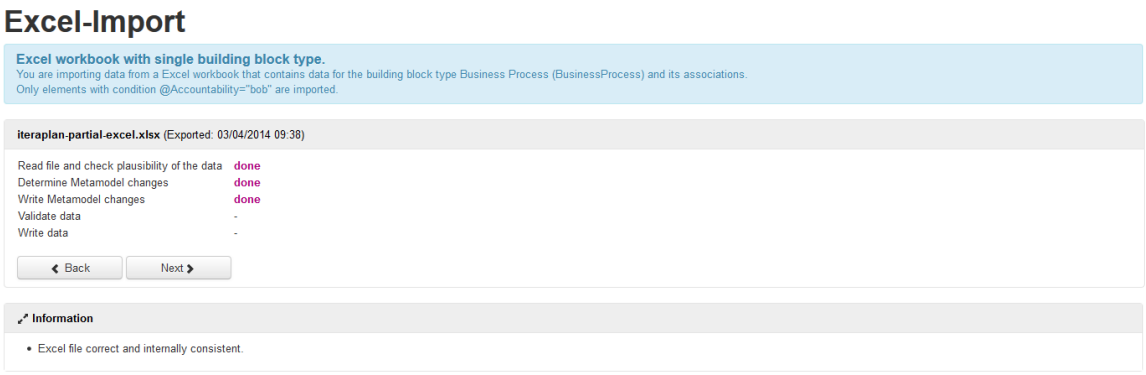


At this point, the user can select the desired import strategy. Currently, iteraplan supports the Additive and Overwrite strategies. These are explained in more detail in the Import Strategies page.

Furthermore, the Import Data section offers a check-box, which enables the user to also import Metamodel changes from an Excel file. This check-box will **not** work for partial Excel files. If the check-box is enabled, iteraplan will ignore it with a corresponding message to the user.

The user can start the import process by clicking the Upload button.

The import process then navigates the user through a number of steps. As already mentioned, the partial Excel import does not allow for changes to the iteraplan metamodel (creation of attribute types, in particular). Thus, in case the imported file is internally consistent, the process directly proceeds to the data comparison, while informing the user that a partial Excel import has been initiated (depicted in the blue message field in the figure below).



Note that from the imported file, iteraplan extracts the date and time at which the export was made. This information is relevant, since only entities which were not modified since the export are considered when updating the enterprise architecture in iteraplan. In other words, the timestamp of the export guarantees that the information being updated is not newer than the information contained in the Excel file.

Running the Validate data step of the import process provides the user with a summary of the changes which will be applied to the enterprise architecture, in accordance with the selected import strategy and the actuallity of the data contained in the Excel file with regard to the current enterprise architecture. If the changes are to the user's satisfaction, the last step of the import process can be triggered. This step performs the actual import of the Excel data into the iteraplan database. Once the step is finished, the user obtains a summarized report of the data written. With this the import process is completed.\

Note: If a user has only read permissions for the main export type of a partial Excel file, trying to import the file will result in an error message, due to the user's lack of permissions. The same error message can also be observed, if the user's metamodel at the time of import is missing a feature which was used for the definition of the partiality specification when the Excel file was generated.

REST API

Integration via the REST API

You can integrate iteraplan with other repositories or planning tools using its REST API. The REST API is available since iteraplan 3.2. It is available in the Enterprise Edition only.

A tool or integrating component can access the building block data and the configuration in the form of resources, and these resources in different formats. The tool must authenticate itself and have sufficient permissions.

Resources

Four types of resources are provided:

- metamodel
- full model, that is EA data
- lists of Building Blocks
- single Building Blocks

Formats and Media Types

Two formats are currently provided:

- MS Excel ("xls" or "xlsx")
- JSON ("json")

The Excel format of the REST API is identical to the format used in Export/Import How to use Import and Export. Only the full model is available in Excel. For the JSON format, see REST API Resource Structure.

The format of a resource is controlled, in order of precedence, by

- query parameter "format"
- HTTP request header "Accept". Supported options for the "Accept" header are:
  - "Accept" header is missing
  - "Accept" header with empty value
  - */*
  - application/json
  - application/vnd.ms-excel (for Excel XLS)
  - application/vnd.openxmlformats-officedocument.spreadsheetml.sheet (for Excel 2007 XLSX)
  - any media type containing "html" (= browser request)

"Accept" headers allow multiple values and quality parameters. The iteraplan REST API will use the first supported media type out of the list (ordered by highest quality) of the "Accept" header's media types.

For example, the Excel XLS format would be used with a "Accept" header like this:

```
Accept: application/wordperfect-6.0;q=0.8,application/json;q=0.4,application/vnd.ms-excel;q=0.6
```

In this example, the media type "application/wordperfect-6.0" is not supported, but the media type "application/vnd.ms-excel" has a higher quality than the media type "application/json".

On requests with an unknown format, iteraplan returns a 4xx HTTP response code.

Format for Exploration via Browser

You can explore the REST API by accessing it from your browser. To make this easy, the format "html" is interpreted as the default JSON format. It is recommended to use a contemporary browser to explore the API. To use Internet Explorer 8, set the format query parameter explicitly. This is necessary because IE 8 uses different Accept header values.

Permissions

A user must be logged in and have a role with the permission "Access iteraplan via REST API" in order to access it. If missing, iteraplan responses with a HTTP 401 (Unauthorized) code. Note that the permission "Execute iteraQL power queries" is not required to use iteraQL queries via the REST API.

In the resources provided by the REST API, a user has only access to building blocks that he can access via the user interface. This access control is on the level of building block types. For example, if a user has read permission (at least) for the type Architectural Domain, he can see building blocks of this type in his resources, and in other building blocks he can see associations to Architectural Domain building blocks. In the same example, if the user does not have the read permission, both the AD building blocks themselves and all associations in all other building blocks to AD building blocks are missing in the resources as seen by the user in question.

Note that all attribute groups of a building block type are part of the resource, even if the user cannot access them via the web user interface. In a

typical role and permission setting, a user with access at a technical integration level has access to types and all attributes anyway.

Note that object-specific permissions are not enforced in the REST API either.

Initialization and Synchronization

The first access to the REST API can lead to a small but noticeable delay because the query framework is initialized. Changes in building blocks via the EA Data user interface may not be visible at once in a REST API resource, only after a short delay due to synchronization. This is the same behavior as the synchronization delay noticeable in the iteraQL query console.

API Versions

When the metamodel evolves, the structure of the resources changes accordingly. For example, a new attribute is visible in the metamodel resource, and the attribute values are part of the model resources. In addition, details of the resource structure and the formats may change in future versions of iteraplan, For example, new fields may be added, or the order of fields may change.

In general, the REST API will be extended in a compatible way, provided that a client program ignores new elements as fields or values that it does not know.

If the API has to be extended in a non-backward-compatible way, the old version will be provided under the same support policy as file formats in other parts of iterplan. Old versions of the API will be identified by a version identifier in the resource URI. Resources without a version identifier will refer to the recent version.

Old versions of the API will be phased out using the same policy as for other features of iteraplan, for example file formats.

Automating Data Export and Import

Scripts or other tools can use the REST API for automated data export and import.

The following examples demonstrate this. They are based on the command-line tool curl, and executed in a command shell that uses Unix-style forward slashes for file names.

Example for **basic authentication**:

```
#Upload / Import
curl -X POST -k -v --user "USER:PASSWORD" --data-binary
@<XLS_UPLOAD_FILE_PATH> <ITERAPLAN_URL>/api/data

#Download / Export
curl -X GET -k -L -v --user "USER:PASSWORD" -o <XLS_DOWNLOAD_FILE_PATH>
<ITERAPLAN_URL>/api/data?format=xlsx
```

Example for **"form"-based authentication** with **cookies**:

```
#Login
curl -k -v --data "j_username=<USERNAME>&j_password=<PASSWORD>"
<ITERAPLAN_URL>/j_iteraplan_security_check --cookie-jar cookies.txt

#Upload / Import
curl -X POST -k -v --cookie cookies.txt --data-binary
@<XLS_UPLOAD_FILE_PATH> <ITERAPLAN_URL>/api/data

#Download / Export
curl -X GET -k -L -v --cookie cookies.txt -o <XLS_DOWNLOAD_FILE_PATH>
<ITERAPLAN_URL>/api/data
```

Basic authentication

New since release 3.3

You can use **HTTP Basic Authentication** to access the RESTful iteraplan API.

HTTP Basic Authentication will work for URLs like <ITERAPLAN_URL>/api/**.

An "Authorization" header line must be provided in the HTTP-request, as specified in RFC 2617 (HTTP Authentication: Basic and Digest Access Authentication):

Authorization: Basic <base64 encoding of user:pass>

For example, the Authorization header line for a user "system" with password "password" it would be:

Authorization: Basic c3lzdGVtOnBhc3N3b3Jk

If the Authorization header is missing, then the server will consider the client to be a normal web browser and perform a redirect to the login form to perform form-based authentication, just like in the normal iteraplan web application.

Migration from iteraplan 3.0 and 3.1

iteraplan 3.0 and 3.1 provide a similar integration API for data export and import. This API was also known as REST API informally.

To migrate an existing script from the 3.0/3.1API to the 3.2 REST API, follow these steps:

1. Keep the authorization step unchanged.
2. Change the URI for the import step.
3. Change the method from POST to GET and change the URI for the export step.

The Excel file format is not changed from version 3.1.

In addition to updating the access method, consider using the JSON format instead of Excel.

You can find the documentation for the previous versions in the iteraplan wiki.

Import strategies

You may select an import strategy (as described here) passing the name of the mode as query parameter "mode" in the URL, as follows:

`ITERAPLAN_URL>/api/data?mode=<mode>`

The following values for <mode> are currently supported:

- additive
- CUD

Notes

- The <ITERAPLAN_URL> variable must include the protocol, hostname, port, and web-application context, for example: `http://localhost:8080/iteraplan` .
- The -v argument is optional, it merely displays verbose curl output which is helpful for debugging purposes.
- curl's documentation can be found here: http://curl.haxx.se/docs/manpage.html

Only for release <= 3.2.x:

- The same security mechanisms apply to the export and import HTTP requests made with curl like when iteraplan is used with a web browser; a login needs to be performed which stores session information into a cookie. This cookie then gets passed along on subsequent requests, allowing access to the secured areas of the application.
- The HTTP client needs to accept cookies sent in combination with a redirect response. For a Java-based client, for example Apache Commons HttpClient. The following clients might lead to authentication problems: Mathematica, Perl, Java SE URLConnection.

REST API Resource Structure

The REST API provides read and write access on resources. The GET method for read access and the JSON format (see below) are always supported.

The full model ("Data") resource is also available in Excel format and for write access via POST. The structure of the full model in Excel is defined in Excel EA Data Format. The "Building Block List" resource also allows POST to create new building block. The content must be JSON data representing a new building block, following the same structure as the result of a GET request on a single building block. The "Building Block" resource also allows PUT to update a single building block and DELETE to delete it. The content must be JSON data in the same structure as GET result. When POSTing or PUTting a building block resource, one may leave out irrelevant fields ('query', 'id', time or user of last modification) or fields one does not want to update or initialize.

Overview

The REST API has these resources:

Resource	Relative URI	Description	URI Example	Possible verbs	Supported Formats
Metamodel	api/metamodel	Building block types with attributes and relations and user-defined enumerations	`http://www.iteraplan.de/iteraplan_ee_demo/api/metamodel`	GET	json
Data	api/data	All building blocks with attributes and relations	`http://www.iteraplan.de/iteraplan_ee_demo/api/data`	GET POST (Excel only)	json, xlsx, xls
Building Block List	api/data/<building block type>	All building blocks of the given type.	`http://www.iteraplan.de/iteraplan_ee_demo/api/data/InformationSystem`	GET POST (for single BB, json only)	json, xlsx
Query	api/data/<query>	Building blocks found by the given iteraQl query	`http://www.iteraplan.de/iteraplan_ee_demo/api/data/InformationSystem[@vendor="IBM"]`	GET	json, xlsx
Building Block	api/data/<building block type>/<id>	Single building block with given type and id.	`http://www.iteraplan.de/iteraplan_ee_demo/api/data/InformationSystem/42`	GET DELETE (json only) PUT (json only)	json, xlsx

The base of the relative URI is the start URI of the iteraplan instance, for example http://www.iteraplan.de/iteraplan_ee_demo.

Technically, the Building Block List is a special case of Query with a trivial iteraQl query that contains only a building block type name. So both resources have the same structure.

Example JSON format

The following is an example result of a Building Block query for a single Building Block. Use the same format for PUT and POST in your payload. DELETE does not need a payload.

```
{
  "query": "Project[@id\u003d\"98\"]",
  "result": [
    {
      "id": [
        "98"
      ],
      "lastModificationUser": [
        "XmiImport"
      ],
      "description": [
        "Introduction of mobile TAN due to security reasons."
      ],
      "name": [
        "mTAN"
      ],
      "lastModificationTime": [
        "2013-12-10T11:59:34.000+0100"
      ],
      "runtimePeriod": [],
      "position": [
        5
      ],
      "Strategic drivers": [
        "excellence"
      ],
      "Strategic value": [
        8.00
      ],
      "Costs": [
        10.00
      ],
      "Value added": [
        8.00
      ],
      "Accountability": [
        "max"
      ],
      "informationSystemReleases": [
        {
          "id": "216",
          "name": "Electronic banking # 2.3",
          "elementURI":
"http://localhost:8080/iteraplan/api/data/InformationSystem/216"
        }
      ],
      "children": [],
      "parent": [],
      "elementURI": "http://localhost:8080/iteraplan/api/data/Project/98"
    }
  ]
}
```


Please note:

- With POST / PUT the "query", "id" and "elementURI" elements are ignored.
- The relevant ID in PUT is in the URI only, not in the query neither in the 'id' field.

- To unset a value use an empty array, e.g.
  "Costs": [ ],
- Elements not present in PUT will be ignored, i.e. they stay unchanged in iteraplan
- Elements not present in POST will be not initialized, i.e. they will have no value.
- When initializing or changing a relationship end, only the 'id' of the related elements are relevant.
- Stand-alone Relationship types (e.g. BusinessMapping) have to be changed directly via POST/PUT. They can not be changed indirectly by POST/PUT on one of their relationship ends.
  For example: To remove a BusinessProcess from an existing BusinessMapping, use PUT on the BusinessMapping. Using PUT on the BusinessProcess (leaving out the reference to the BusinessMapping) will have no effect on the BusinessMapping.
- The 'name' and 'URI' fields of a related element in the value of a relationship end are ignored.

Metamodel Resource

The metamodel contains all building block types with attributes and relations and the user-defined enumerations.

Name of Resource

api/metamodel relative to start URI.

Fixed, no parameters

Structure of Resource

Metamodel := JSON-Array of (Enumeration OR UniversalType)

Enumeration

Enumeration := JSON-Object with fields

"type": "EnumerationExpression",

"persistentName": String, internal technical name. For user-defined enumerations, the persistent name start with de.iteratec.iteraplan.model.attribute.EnumAT. Otherwise, the enumeration is predefined.

"name": String, user-defined name of the enumeration type, for example Complexity

"description": String, empty for predefined, empty for user-defined attributes, future extension for user-defined attributes

"abbreviation": empty String, for future extension

"literals": JSON-Array of EnumLiteral

EnumLiteral := JSON-Object with fields

"persistentName": String, internal technical name.

"name": String, name in current locale",

"description": String, description in current locale, empty for predefined enumerations

"color": String, default color for visualizations, format rgb(rrr,ggg,bbb)

"index": integer, unique within enumeration, defining the order of the literal values.

UniversalType

UniversalType := JSON-Array with fields

"type": "<SubstantialTypeExpression" or "RelationshipTypeExpression" for a stand-alone or associative building block, respectively

"persistentName": String, internal, technical name

"name": String, name in current locale

"description": description in current locale

"abbreviation": abbreviation in current locale,

"features": JSON-Array of Feature

Feature := JSON-Object with fields

"type": name of value type or buidling block type for a attribute/property or association/relationship endpoint, resp.

(Possible ocurrences for property types: string/richtext/integer/decimal/date/date_time/boolean/duration)

"persistentName": String, internal, technical name

"name": String, name in current locale

"description": description in current locale

"mandatory": boolean

"multiple": boolean

Note that the multiplicity of a feature is expressed by boolean flags mandatory and multiple, but not as numerical upper and lower bounds.

Data Resource

The data contains the full model, that is all building blocks. It is organized by building block type.

Name of Resource

api/data relative to start URI.

Fixed, no parameters

Structure of Resource

Data := JSON-Array of QueryResult , one QueryResult for each building block type.

QueryResult := JSON-Object with fields

"query": String, query part of the resource name

"result": JSON-Array with BuildingBlock

BuildingBlock := JSON-Object with multiple BuildingBlockFields, depending on type, and additional one URIField

BuildingBlockField := FeatureName: Values

URIField := "elementURI": String, URI of the Building Block resource

Note: a BuildingBlockField is a name-value pair in a JSON Object, with a literal colon.

FeatureName := persistent name of feature, that is of either property or relationship end

Values = JSON-Array of either Value or RelatedElement, can be empty

Note that both single-valued and multi-valued features are represented as arrays of objects.

Value

Value := one of

NumericValue := JSON number, with decimal seperator ".", in any locale

DateValue := ISO-8601 format, modification timestamp with resolution millisecond, regular date with resolution day of month.

BooleanValue := JSON true or false, not as a String

EnumerationLiteralValue := internal, technical non-localized name.

Note: Colors are only part of the EnumerationLiteral, that is part of the metamodel.

Related Element

RelatedElement := JSON Object with fields

"id": String, internal ID of building block

"name": String, user-defined name of building block. Field is omitted if building block type has no name.

"elementURI": String, URI of building block resource

Query Resource and *Building Block List* Resource

A Query resource contains all building blocks or all pairs of building blocks that are the result of an iteraQL query. The set of building blocks representing a relationship is called a binding set.

A Building Block List resource contains all building blocks of a given type.

A Building Block List resource is a special case of a Query resource, where the Query is a plain Building Block Type name. Both resources have the same structure.

Name of Resource

api/data/<building block type>

api/data/<iteraQL query>

The iteraQL query does not end in a semicolon. This is intentionally different from the iteraQL query console, where the semicolon is required.

Names in a iteraQL are case sensitive.

Special characters must be encoded in an URI. More specific, the slash character is written as "%252f".

Step by step:

1. The character "/" is URI-encoded as "%2f"
2. The character "%" is URI-encoded as "%25"
3. Combined: "/" is twice URI-encoded to "%252f"

iteraQL queries that use operators and result in synthetic types are not valid in resource names. For example, objectify, nullify and power are not allowed.

More information on iteraQL is defined in iteraQl How To

Structure of Resource

Query := QueryResult or BindingSet

BuildingBlockList := QueryResult

Structure of QueryResult is defined above.

BindingSet := JSON Array of BindingPairs

BindingPair := JSON Array with fields

"from": RelatedElement

"to": RelatedElement

Structure of RelatedElement is defined above

Building Block Resource

A Building Block resource is a single and uniquely identified building block with all attributes and related elements.

Name of Resource

Type based: api/data/<building block type>/<id>

Query based: api/data/<iteraQL query>/<id>

Structure of Resource

Structure of BuildingBlock see above.

Compatibility to older versions

For compatibility reasons, the pre-3.0 versions of Excel and XMI Export/Import will still be offered in the 3.0 releases of iteraplan.

They can be accessed with the following URLs:

- Excel: <iteraplan-URL>/excelimport/init.do
- XMI: <iteraplan-URL>/xmideserialization/init.do

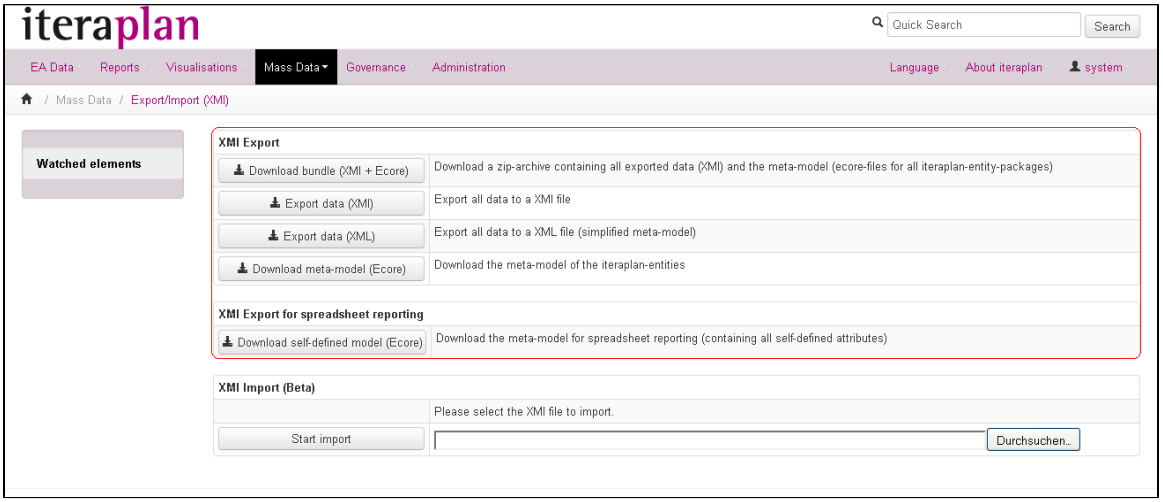
Please replace <iteraplan-URL> with the URL you access iteraplan with, for example: www.myserver.de/iteraplan

Import and Export Based on XMI Format

This section describes the functionality to save, create, modify and import data using XML Metadata Interchange (XMI), a XML standard depending on meta models, the so called Ecore Meta Models. iteraplan provides downloads of its meta models (ecore-files) for different use cases which will be described in more details in the next section.

Export

As you can see iteraplan offers four different files for download on the XMI export page:



XMI export - options to download different files

Download Bundle:
The zip archive ("iteraplanXMIExport.zip") consists of a set of five files:
A XMI file containing all persisted data (**Building Blocks**, **Attributes**, **Users**, **Roles**, etc.) named "iteraplanSerialization.xmi" and four Ecore files representing the meta model of the iteraplan entity-classes. These files are only necessary to open the XMI file in a graphical XMI editor e.g. the "Sample Reflective Ecore Model Editor" which is available for free as an extension of eclipse.

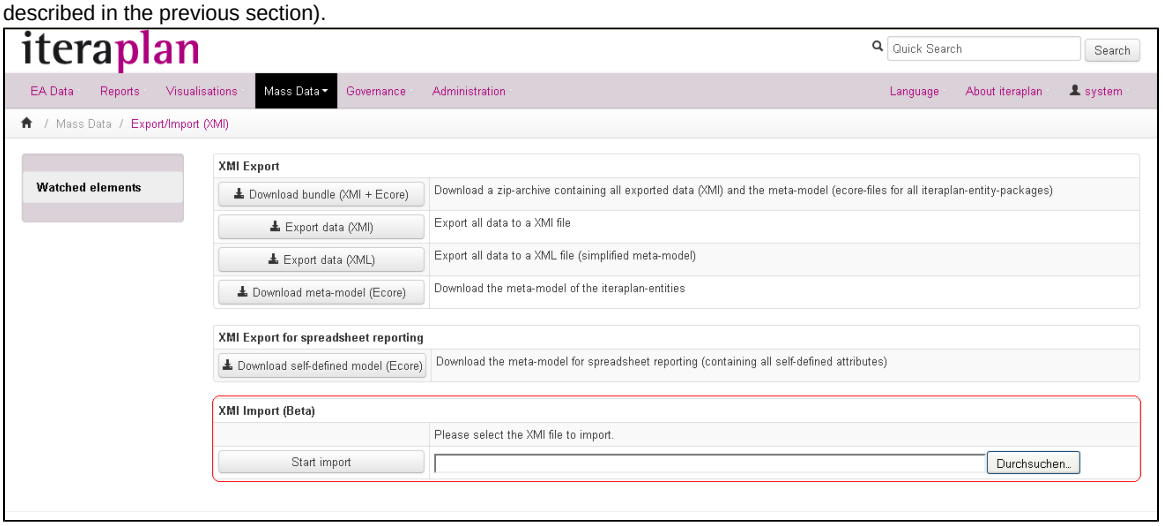
Export Data:
The XMI file "iteraplanSerialization.xmi" has the same content as the XMI file contained in the zip archive: XML data for all persisted objects.

Download meta-model:
To display the whole information-model of iteraplan, you can download the Ecore file "iteraplanModel.ecore" and open it with the previously mentioned "Sample Reflective Ecore Model Editor" in eclipse or any other XML editor of your choice.

Download self-defined model
This Ecore file represents the meta model for the XML export which is available at **Reports/Spreadsheet Reports**. Use this file to open the exported XML file in a graphical editor.

Import

iteraplan also offers the possibility to import XMI files - as long as they match the iteraplan meta models (Ecore files of iteraplanXMIExport.zip. described in the previous section).



XMI Import
For the import you just have to select a valid XMI file and start the import. During the import process, iteraplan searches for objects of the same class in the database which have the same IDs as the objects defined in the XMI file. If such an object already exists, the import will overwrite all attributes of the persisted object with the values defined in the XMI file. If no such object can be found, a new object will be created and persisted if this would not lead to any database inconsistencies. If you for example try to import a new **Information System** and its name is set to the same value as the name of an already persisted one, the import will not start and the conflict between the XMI file and the persisted object is displayed to the user. Generally there are two ways to solve such conflicts: You can set the ID of the XMI instance to the same value as the ID of the

persisted object. This will lead to an update of the persisted object in the database. That means of course that no new object will be created during the retried import process. The other way is to change the value of the unique attribute (name and release version in this case) in the XMI file. This will lead to the creation of a new object which gets persisted when the user retries to import the changed XMI file.

The content of a valid XMI file schould look as follows:

Headers

```
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:de.iteratec.iteraplan.model="platform:/resource/iteraplan/XMI/model.eco
re"
xmlns:de.iteratec.iteraplan.model.attribute="platform:/resource/iteraplan/XMI
/attribute.ecore"
xmlns:de.iteratec.iteraplan.model.files="platform:/resource/iteraplan/XMI/fil
es.ecore"
xmlns:de.iteratec.iteraplan.model.user="platform:/resource/iteraplan/XMI/user
.ecore">
```

It is strongly recommended, that the imported XMI files contain these headers. Otherwise the import will fail!

Content

Runtime Periods must be the first type of elements in the xmi file.

Caution: Do not change the position or the order of *RuntimePeriods*. As they do not have IDs, they are referenced by their position in the XMI file! When addin Runtime Periods, add them at the bottom of the runtime period section.

```
<de.iteratec.iteraplan.model:RuntimePeriod
start="2007-01-01T00:00:00.000+0100" end="2009-01-01T00:00:00.000+0100"/>

...

<de.iteratec.iteraplan.model:BusinessProcess
buildingBlockType="BuildingBlockType_5"
description="businessProcess.virtualElement" children="BusinessProcess_41"/>

<de.iteratec.iteraplan.model:BusinessProcess
buildingBlockType="BuildingBlockType_5" id="BusinessProcess_41" 
description="" name="Management processes" parent="BusinessProcess_3"/>

<de.iteratec.iteraplan.model:BuildingBlockType availableForAttributes="true"
typeOfBuildingBlock="businessProcess.singular"/>

...
```

Each line of the XMI file defines a single Object, its attributes and its references as they are defined in the corresponding ecore files.

The content of the XMI file can be edited or enlarged by additional Objects. Thereby it is strongly recommended, that every referenced Object is defined in the XMI file as well. In the example above, the content defines two **Buisiness Processes**. To complete their definition, you have to assert that the referenced **BuildingBlockType** (BuildingBlockType_5) is also defined in the XMI file.

To alleviate the modifying of XMI files it is useful to use an just exported XMI file as a basis for the further handling as the complexity of the development of an XMI file from scratch can increase rapidly.

End

```
</xmi:XMI>
```

Common Errors

Q: When importing the following error message appears: **Details: Unresolved reference 'BuildingBlockType_33'. (iteraplanSerialization.xmi, 8, 210)**

This usually means that you modified the xmi file and did not provide all required references. In this case, the element described on line 8, refers to "BuildingBlockType_33" which itself is not present within your .xmi file. Make sure to provide all referenced items in the xmi file to successfully import it.
Q: When importing the following error message appears: Details: Package with uri 'files.ecore' not found. (iteraplanSerialization.xmi, 59, 396)

This message might indicate that you are trying to import an XMI-Export that was generated with a different version of iteraplan. For example if you try to import a File that was created with iteraplan 2.6 into iteraplan 2.7, the following elements and attribute combinations might cause trouble:

- <de.iteratec.iteraplan.model.files:SavedQueryFile... /> - the format of the saved queries has changed fundamentally. To Import the xmi file, remove these objects.

- <de.iteratec.iteraplan.model:InfrastructureElement ... technicalComponentReleases=".." /> the connection between infrastructure elements and technical components has been changed. Remove any relations between these two types of objects before commencing the import.
- <de.iteratec.iteraplan.model:TechnicalComponentRelease ... infrastructureElements="..." /> the connection between infrastructure elements and technical components has been changed. Remove any relations between these two types of objects before commencing the import.

Use-Cases

Backup and Restore:

Every exported XMI file of iteraplan is a backup of all actual persisted objects. That means that every XMI file can be used to restore the database state of the time the XMI file was created.

If there are objects in the database at the time of the import, the following applies:

- Objects in the database will be overwritten with older versions of themselves (if existent) from the backup.
- New objects added to the database since the backup will stay in the database.
- Objects deleted from the database since the backup was made will be restored.

Editing data:

You can edit the content of XMI files by using a XML editor. Importing this edited XMI file will update all affected persisted objects by overriding the instances with the same ID.

Please only change IDs in order to solve conflicts between XMI data and already persisted objects. Otherwise it could be possible that the import leads to undesirable changes. When changing an ID inside the XMI file, you have to assert that the change is made for every occurrence or the import may fail.

If an object is referenced by another one, you have to edit the ID in this reference as well!

Creating new data:

Creating XMI files with a XML editor is an easy way to create new **Building Blocks**, **Attributes**, **Users** and objects of all other classes defined in the iteraplan meta model ("iteraplanModel.ecore", section 7.2.1). Before starting the import-process of this file, you have to assert that every recommended attribute (an attribute is recommended if its definition in the Ecore file defines a "lowerBound" of "1") is set for all created instances.

When creating data, be sure to leave the id attribute blank (Note that no other attribute may be left blank). If the id attribute were set in the XMI, iteraplan would update existing objects matching those IDs rather than creating new ones. If objects in your XMI specify IDs not found in the database, iteraplan ignores the specified IDs and creates the objects just the same.

During the import, iteraplan always assigns the next unused ID to each created object. Whenever objects are created in this process, you will be prompted to download an updated XMI containing only the same objects from your original XMI, but with their IDs filled in from what is now in the database. This XMI allows you to update these objects in the future instead of creating them anew.

Import and Export (Excel)

iteraplan supports data exchange with other management applications by means of Microsoft Excel files (97-2003 file format). iteraplan can import Landscape Data, attributes and object related permissions. You can also use the Export/ Import functionality to perform bulk updates on data, (export-import round-trip).

An Excel Export can be obtained by generating a Spreadsheet Reports or Successor Reports.

Using Export and Import

iteraplan allows to export data in XML and Excel files, and to import data from XML and Excel files.

For more details on typical workflows, see here: How to use Import and Export

Formats

Starting with iteraplan version 3.0.0, there is a new Excel file format for exports and imports. The old Excel file format is still supported in version 3.0

Only Excel 97-2003 files in one of the supported iteraplan formats can be imported. These are the types of supported formats:
Excel EA Data Format

Allows importing of a complete EAM data set, or parts thereof. This format can be used to transfer data between iteraplan instances, or it can be used to for bulk updates of parts of the data. Using the data export/import is described in How to use Import and Export, the data format is described in Excel EA Data Format.
Landscape Data

Allows importing of Building Blocks, building block Attributes, and building block Relations.
Attributes Import

Allows importing of attributes and attribute values.
Object related permissions import

Allows importing of Objected Related Permissions.

> A Template for each import functionality is available by clicking **Download template** in the **Excel Import** section, under the respective headings.

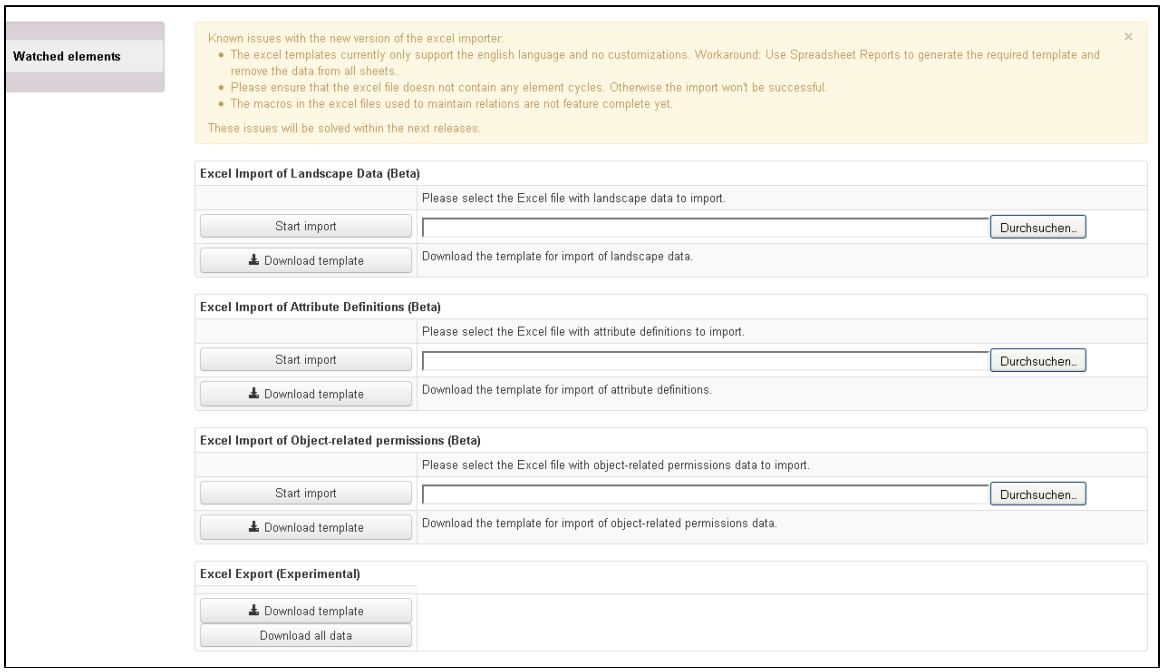
Landscape Data

This Import functionality allows for the importing of Building Blocks, Attribute Values, and Relations.

Preparing the Import Data

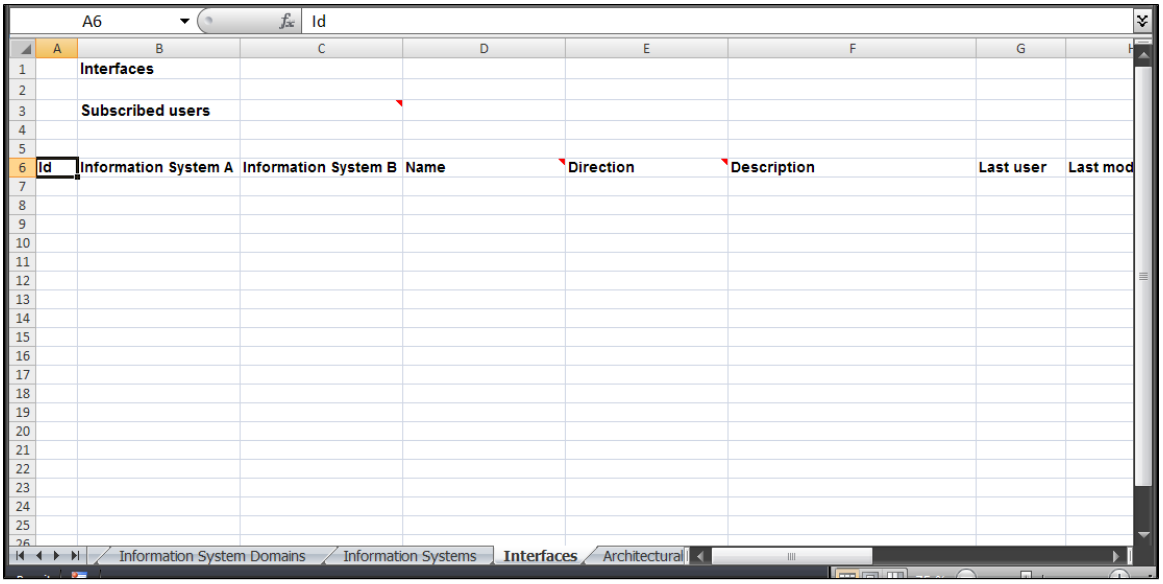
There are two possibilities to create a suitable Excel import file.

1. Create a Spreadsheet Report or Successor Report in iteraplan and choose Excel as output format. The resulting Excel is already in a table format that is suitable for importing. This option can be used to update existing data in iteraplan, as well as to import new data.
2. Download the empty import template file from iteraplan and fill in your data into the predefined table structure. This template file is available in iteraplan on the **Import (Excel)** dialog, in section *Excel Import of Landscape Data*. Click the button labelled *Download template* to initiate the download. The following figure shows a screenshot of that dialog. This option is best if you want to import new data. Only this files comes with additional with additional explanations in header row cell comments.



Data Format

Once you have downloaded any of the files, open it in Microsoft Excel (version 97 or newer). You should see a screen like the following (using Microsoft Excel 2007):



There is a separate worksheet for each building block type that iteraplan supports. None of the supported worksheets needs to be present, they are all optional. Sheet order is inconsequential and sheets are identified by name. That mean you must change worksheet names if you want their contents to be imported. You can add your own worksheets, as every sheet with an unsupported name will be ignored (a warning message is written to the import log).

Each Sheet contains a header row, starting with "Id", and going on to enumerate other data fields for that particular Building Block Type, such as its description, various Relations, and Attributes. The header row consists of a fixed and a user-defined column set. Both parts of the header row are separated by a column containing a hashmark #. That means that you **should not change** any of the columns to the left of the hashmark, and that you can and **should change** (or add to) the columns to the right of the hashmark. These columns will cause no changes. You can add or remove rows before the header as you like; the import routines will detect the position of the header row. The column "Id" must always be in Excel column A in order to be found.

Below the header row, each following row represents a single building block, including its relations and attributes. Inserting empty rows is discouraged.

There is one sheet called *ExcelTemplateSheet* which contains Locale information: A 2-letter language code (e.g. en or de) representing the language of the workbook structure is stored here. This language determines in what language the iteraplan expects Sheet and Header names during import. Do not change this value, as it will prevent iteraplan from locating your data. The other fields on that worksheet have no effect at the moment.

Rules and Conventions for the Structure of Import Data

Insert vs. Update

- If an ID is specified and a building block of that type with that ID can be found, that building block is updated. In case the name of the building block was changed and would result in a duplicate name for the respective building block type, iteraplan will still import and update the existing building block with the information given. In case the relations were left out in the Excel file were not maintained, iteraplan will apply the previously existing relations, e.g. between business processes and information systems, to the updated entry. ??

  Note for developers: review behaviour for duplicate names and missing relation values.

- If either the ID is blank, or it cannot be found, that building block is inserted with a new ID. The new ID is generated by iteraplan and cannot be influenced. In case a building block of same type with same name exists already, iteraplan will ignore the element during import and leaves the already existing building block unchanged.

Building block-specific properties

- Information systems and technical components have a column **Status** for which only pre-defined values are accepted during import. The Locale (language) setting (see above) of the import file determines which values are accepted. For English content, the standard values are *Current*, *Planned*, *Target* and *Inactive*. Note that installation-specific customizations can imply different values.
- Information systems, technical components and projects have columns for runtime periods. The values for start and end dates must be

formatted with a Excel-built-in date format; otherwise they cannot be imported by iteraplan and a warning will be to the import log.

- Values in the two columns **Last user** and **Last modification** are always ignored. They are always maintained by iteraplan and cannot be influenced directly.
- Interfaces: the name column can be deleted, but not Information System A and Information System B.
- Business Mappings: no column can be deleted

Relationships

- All updated building blocks have their relationships overwritten by the Excel import. If you want to keep the relationships during an update, do not delete them from the Excel file after it has been exported. Yet, this rule applies only to those building blocks that have not changed their name. If a building block has changed its name, its relationships remain intact, even if they are removed from the Excel file.

> Note for developers: review whether described exception is correct

- Related building blocks may be specified with their hierarchical as well as their non-hierarchical names. Because the non-hierarchical names (in conjunction with a release name, if available) are always unique, they are used for identification. In fact, the path through the hierarchy is removed from hierarchical names before they are processed.
- If a building blocks has relations to more than one element of the same type, use the semicolon character and a blank to separate the related elements.
- Relationships are created in the last phase of the import process, after all building blocks from an Excel file have been inserted. This requires that the names used in relations shall be updated in the Excel file if the name of the involved building blocks had been changed in the Excel file. Otherwise, the relation cannot be created and iteraplan will apply existing relations to the updated or new building block.
- In order to create a relationship between two building blocks, it suffices to specify the relationship on one side. The other side(s) may omit the relationship, or the respective worksheet may even be excluded from the import file. For example, a relationship between two information systems is still created in iteraplan, even if the relationship is defined at one information system only.

> Always refer to Building Blocks by their (potentially) new names from the Excel file when specifying relationships.

- In hierarchical names, a three-characters string *space-colon-space* (:) is used to separate hierarchy levels from each other. Specifying the full hierarchical name is optional for all relation columns, except for the Hierarchy column (obviously).

Business Mappings

- Business Mappings are only imported from their own sheet. The sheets for Information Systems, Business Processes, Business Units, and Products have a column for business mappings as well, but it is ignored (a warning is written to the import log). Those columns are included solely for export purposes.

> **Important**: Unlike other sheets, the "Id" column in the Business Mappings sheet is the ID of the associated Information System, and is ignored during the import process. Therefore, to update a Business Mapping's Attributes, the Business Mapping is identified by its relations.

Interfaces

- Interfaces are only imported from their own worksheet, and not from the Information Systems worksheet. This behavior enables the ability to define multiple interfaces between Information Systems. For interfaces, the name is optional and can be left empty.
- Cell in the interfaces *direction* column may have one of the following values:

<-	flow from B to A
->	flow from A to B
<->	bidirectional flow
-	undirected flow

Cells must be explicitly **formatted as text cells** or values must be prepended by an apostrophe, so that Excel treats them as text.

- The flow direction of transported business objects is represented in a similar way. The name of the business objects must be preceeded by one of the four above direction indicators and a space. As for other relations as well, a semicolon and a space are used to separate one business object with direction from the next business objects and its direction.

Partial-Excel-Import

Sometimes you don't need all columns for your changes. You can leave some columns out and they will be ignored. That means the import behaves like the columns are present with the previous content. But some columns are not allowed to delete:

- The column containing the id
- The column containing the name
- Note that Business Mappings do not have a name column
- For Interfaces, "Information System A"- and "Information System B"- columns are needed, but you can leave out the name-column here

Attributes

- Attribute values are only imported for attributes that already exist in iteraplan. Attribute values for attributes that do not exist in iteraplan are silently discarded during the import process. Invalid attribute values are reset to the default attribute value.
- Some attribute value types demand that Excel cells have a suitable data type. Numbers must be formatted with a Excel-built-in number format. Dates must be formatted with a Excel-built-in date format. Enumeration values, Responsibles, and Text values must have a cell format which can be parsed as text.
- Values for enumeration attributes and responsibility attributes are always validated against the respective attribute definition in iteraplan. Only enumerated values will be imported; all other values will be ignored and a warning is written to the import log.
- Attribute names (when they are part of the header row, appearing to the right of the hashmark) can be stated together with their attribute group, enclosed in brackets.

> ⚠ The current implementation cuts off all text after the first opening round bracket in the column heading; consequently, avoid round brackets in attribute names.

Miscellaneous

- The name of a building block, as well as its properties, may contain Unicode characters, e.g. Latin letters, Chinese symbols, Cyrillic letters, etc.
- Subscriptions can be also set by an Import. To separate the subscribers, use the semi-colon (;) mark as separator. If done so, all existing subscribers will be replaced by the ones specified in the Excel file. If a subscriber is not known to iteraplan, a new user account for that subscriber is created. To remove all subscriptions from the building block, clear the subscriber field in the Excel file.
- Excel formulas: iteraplan can deal with most standard Excel formulas. However, an import will always fail if the file uses *any* macro-imple mented functions or references cells in other Excel files. Workaround: Replace worksheet contents with the *formula results* (Paste special / Values) and remove any worksheets with unsupported names.

> ⚠ The import will process all building block types which the user has create and update permissions for. No read permission is necessary to update or insert building blocks via import.

Starting the Import Process

To import data from your prepared Excel file, navigate to the *Import (Excel)* dialog, found in the *Administration* menu group. The figure above shows a screenshot of that dialog. There you can select the Excel file from your local file system. To start the import process, press the button labeled **Start import**. Be sure to enter the file in the right section of the dialog, i.e. here the Excel Import of Landscape Data.

After you clicked the button, your browser will upload the Excel file to the iteraplan server and processing starts. The entire import process is wrapped in a single database transaction, so that either the complete file gets imported (except for syntactical errors) or nothing gets changed. The latter can happen if an unexpected error occurs and iteraplan has to abort the import to avoid inconsistent data.

As soon as the import process succeeded, the browser will update the dialog page and display a log of the import where you can see which warning or errors were encountered.

> ⚠ iteraplan's Excel import is not yet fully optimized and offers only inferior performance in the current beta implementation. The number of attribute columns on each sheet as well as the number of rows directly influences the duration of the import process. If possible, try to import a huge chunk of data in several increments, so that you get feedback earlier.

After the import process completed, be sure to inspect the log for any warnings or errors that might require corrections. Some data from your file may have been skipped because of inconsistencies.

Attributes Import

This type of import enables you to import and attributes and attribute values.

Preparing the Import Data

Data Format

To download the template sample file, please follow the instructions described here.

There is a separate worksheet for each attribute type that iteraplan supports. None of the supported worksheets needs to be present, they are all optional. Sheet order is inconsequential and sheets are identified by name. That means you must change worksheet names if you want their contents to be imported. You can add your own worksheets, as every sheet with an unsupported name will be ignored (a warning message is written to the import log).

Each Sheet contains a header row with two columns: "Attribute Name" and "Old Attribute Name". The "Attribute Name" has always to be present. The "Old Attribute Name" optional. This column is necessary to rename the attribute. Then follows more optional columns for updating the attributes. For columns where "YES" or "NO" are the only possible values, other or empty values are ignored and the column is automatically set to "NO" per default. The column "Activated Building Block Types" has no default value, incorrect entries are ignored. All existing activated Building Block types for the attribute are deleted in the beginning of the import, only all types stated in the Excel sheet are valid after the import.

For **Enumeration Attributes**, the columns Right to "Attribute Values" are for adding or updating enumeration values. The first value is in the same line as the attribute is. The next value comes in the rows below. In this rows the columns left to "Attribute Values" have to be empty. There is also the optional column: ?"Old Attribute Values". Use this to rename values.

Here is an example for adding one attribute "EnumAt1" with three values: "testvalue1", testvalue2" and "testvalue3". The existing attribute "enum", will be renamed to "EnumAt2", will get a new value "testvalue1" and the existing value "value" will be renamed to "testvalue2".

	A	B	C	D	E	F	G	H	I	J
1	**Attribute Name**	**Old Attribute Name**	**Description**	**Attribute Group**	**Mandatory Attribute**	**Multiple Values**	**Activated Building Block Types**	**Attribute Values**	**Old Attribute Values**	**Attribute Value description**
2	EnumAt1		EnumAt1 description		YES	NO	BusinessProcess;BusinessUnit	testvalue1		testvalue1 description
3								testvalue2		testvalue2 description
4								testvalue3		testvalue3 description
5	EnumAt2	enum	EnumAt2 description		NO	YES	BusinessProcess	testvalue1		testvalue1 description
6								testvalue2	value	testvalue2 description
7										

Due to formatting problems with Excel, one should be careful while filling out the column "User defined ranges" for **Numeric Attributes**. These cells must be prepended by an apostrophe, so that Excel treats them as text. Otherwise an error occurs because Excel has problems with the ";" between the numbers.

There is one sheet called *ExcelTemplateSheet* which contains Locale information: A 2-letter language code (e.g. en or de) representing the language of the workbook structure is stored here. This language determines in what language the iteraplan expects Sheet and Header names during import. Do not change this value, as it will prevent iteraplan from locating your data. The other fields on that worksheet have no effect at the moment.

Rules and Conventions for the Structure of Import Data

For the import you just have to select a valid Excel file, that contains attributes, and start the import. During the import process, iteraplan searches for the attributes with the same names in the database. If such attribute already exist, the attribute will be updated. Otherwise the attribute will be created. Also other warnings and errors will be shown, indicating which data was imported and saved, and which were skipped, because of inconsistent data.

For importing attributes the following rules apply:

- If the **attribute name** is empty, the row will be ignored, except the row is for adding **enumeration values**.
- The **attribute name** identifies the attribute when **old attribute name** is empty.
- If **old attribute name** is not empty, this one identifies the attribute. The attribute will be renamed (if possible) to the content of **attribute name**.
- All cells which are empty, the column does not exists, or the content is not applicable to the column, will be set to a default value.
- The same rules are applied for **enumeration values**.
- **Responsibility values: *If a *user** does not exists, a new user will be created. If a **user group** does not exists, you only find a warning is the log.
- **Numeric attributes:** The **lower bound** has to be smaller than or equals to the **upper bound** field.
- **Numeric attributes:** You can only define **user defined ranges** if **range uniform distributed** is set to **NO**.

Starting the Import Process

To import data from your prepared Excel file, navigate to the *Import (Excel)* dialog, found in the *Administration* menu group. The figure above shows a screenshot of that dialog. There you can select the Excel file from your local file system. To start the import process, press the button labeled **Start import**. Be sure to enter the file in the right section of the dialog, i.e. here the Excel Import of Object related permissions.

After you clicked the button, your browser will upload the Excel file to the iteraplan server and processing starts. The entire import process is

wrapped in a single database transaction, so that either the complete file gets imported (except for syntactical errors) or nothing is changed. The latter can happen if an unexpected error occurs and iteraplan has to abort the import to avoid inconsistent data.

As soon as the import process succeeded, the browser will update the dialog page and display a log of the import where you can see which warning or errors were encountered.

Object related permissions import

This type of import enables you to import and assign specific rights to *users* for editing particular instances of building blocks (see Object related permissions).

Preparing the Import Data

Data Format

To download the template sample file, please follow the instructions described here.

There is a separate worksheet for each building block type that iteraplan supports. None of the supported worksheets needs to be present, they are all optional. Sheet order is inconsequential and sheets are identified by name. That mean you must change worksheet names if you want their contents to be imported. You can add your own worksheets, as every sheet with an unsupported name will be ignored (a warning message is written to the import log).

Each Sheet contains a header row with three columns: "Id", "Name" and "User". The "Id" column represents the Building Block ID, the "Name" column can be used to identify a Building Block by its name and the "User" column contains a list of semicolon-delimited list of users, which should get exclusive read/write permissions on this element. The "Id" and "Name" columns are optional in the sense that as long as one of them is filled out the other can be left empty or entirely omitted. Below the header row, each following row represents a single building block.

There is one sheet called *ExcelTemplateSheet* which contains Locale information: A 2-letter language code (e.g. en or de) representing the language of the workbook structure is stored here. This language determines in what language the iteraplan expects Sheet and Header names during import. Do not change this value, as it will prevent iteraplan from locating your data. The other fields on that worksheet have no effect at the moment.

Rules and Conventions for the Structure of Import Data

For the import you just have to select a valid Excel file, that contains object related permissions, and start the import. During the import process, iteraplan uses the provided ids (and/or names) to search for the building blocks in the database. If such building blocks already exist, the rights will be assigned for the specified users. Non existing users will be created and persisted. In case the building blocks do not exist, a log message will be shown. Also other warnings and errors will be shown, indicating which data was imported and saved, and which were skipped, because of inconsistent data.

For importing object-related permissions the following rules apply:

- If a building block name is given in the Excel file, iteraplan will use it to perform a search for the element in the database. If in addition an id is specified, the id of the building block found will be tested against the provided id. If only an id is used, iteraplan will try to find a building block with the same id and type in the database.
- If a matching building block cannot be identified in the database, existing permissions will not be changed, and a warning message will be written to the import log.
- If the building block is found, existing permissions will be replaced with the ones given in the Excel file.
- If a user name from the Excel file does not exist in iteraplan's user list, the user will be created and the permissions will be set accordingly.
- Newly created users will not be able to login to iteraplan, because they are not created in the iteraplan account management system (e.g. iTurm, LDAP, etc.). To enable those users to login, user accounts for those users have to be created.

Starting the Import Process

To import data from your prepared Excel file, navigate to the *Import (Excel)* dialog, found in the *Administration* menu group. The figure above shows a screenshot of that dialog. There you can select the Excel file from your local file system. To start the import process, press the button labeled **Start import**. Be sure to enter the file in the right section of the dialog, i.e. here the Excel Import of Object related permissions.

After you clicked the button, your browser will upload the Excel file to the iteraplan server and processing starts. The entire import process is wrapped in a single database transaction, so that either the complete file gets imported (except for syntactical errors) or nothing is changed. The latter can happen if an unexpected error occurs and iteraplan has to abort the import to avoid inconsistent data.

As soon as the import process succeeded, the browser will update the dialog page and display a log of the import where you can see which warning or errors were encountered.

Users, Roles and Permissions

iteraplan provides a system of access rights with which permissions can be defined and restricted at the levels of both roles and objects.

> ⚠ **Assignment of roles to users**
> The assignments of roles to users is **not** managed in iteraplan! Your organisation's identity management system must take care of this. However, for a quick start you may use iTURM for managing role assignments, a very simple user and role management application.

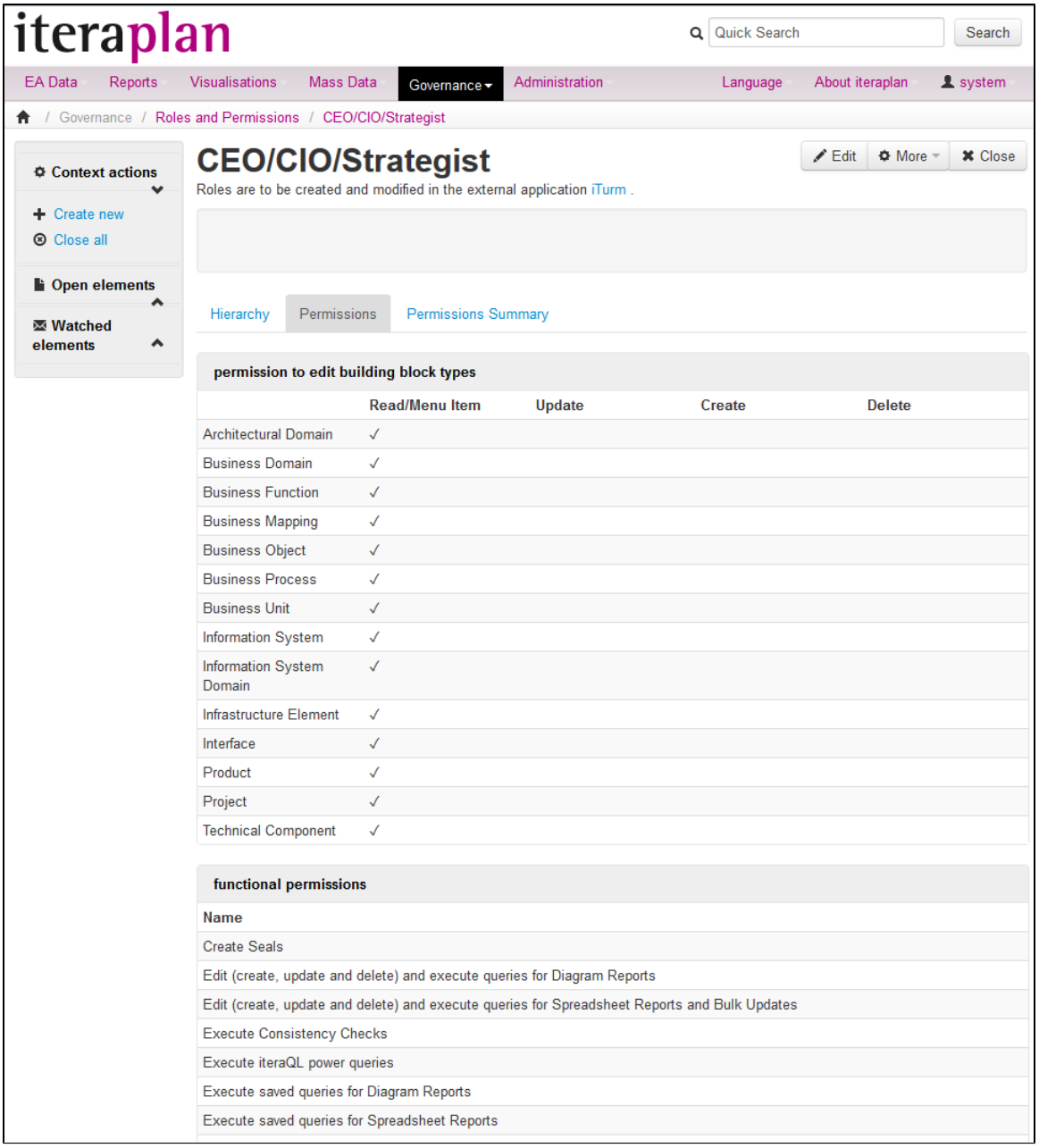
The Table below surveys the permissions which can be assigned to users, user groups and roles.

Table: Working with permissions: users, roles and user groups

User	Role	User group
Object-specific write permissions		Object-specific write permissions
	Functional permissions	
	Type-specific permissions	
	Read and write permissions for attribute groups	

Permissions summary

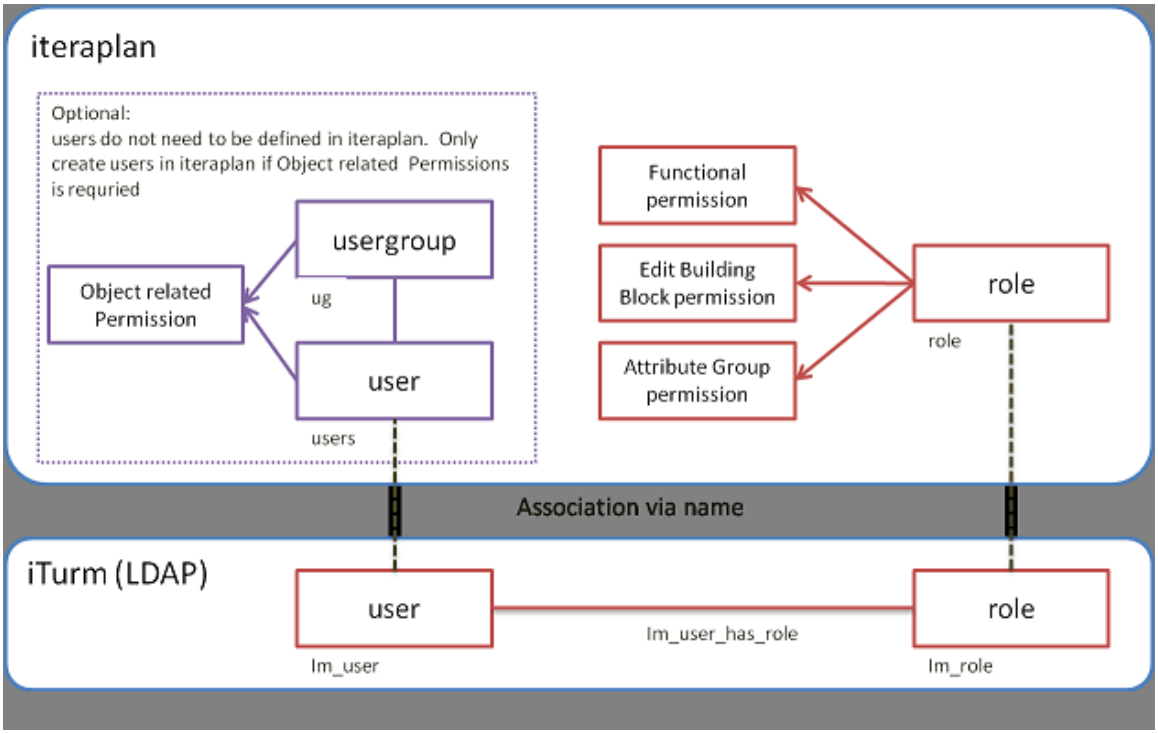
The User Management, User Group Management, and Roles and Permissions pages each have a tab titled **Permissions Summary**. This tab lists the permissions assigned to the role either directly or indirectly (e.g. via subordinate roles), giving you an overview of assigned access rights. The term **aggregated** with a permission type indicates that the list also includes all permissions of subordinate users or roles. The permissions summary is a read-only list. To modify permissions, you must go via the pages for editing building blocks.



Permissions summary for a role

Permissions Metamodel

The assignments of roles to users is not managed in iteraplan! You should use your company's identity management system, typically an LDAP directory service, or iTURM. The following graph shows how users and roles are related in iteraplan and iTurm. The names outside the boxes are the database table names.



Relation between users and roles in iteraplan

Key points of iteraplan and Identity Management (like iTurm or LDAP)

- The roles in iteraplan must match the roles/groups in your identity management system (iTurm or LDAP), which means theirs names must be equal.
- The roles (LDAP: groups) in the identity management system are used to map users and roles.
- The roles in iteraplan are used to assign specific permissions.
- Uses should be created in the identity management system with their password (LDAP: with password and email address).
- When a user logs into iteraplan, the iteraplan user entry will be created (LDAP: and/or updated) automatically.
- The only static and unmodifiable role is "iteraplan_Supervisor" for the iteraplan superuser. Users with this role assigned (directly or by hierarchical role) have all privileges in iteraplan, independent of the confiuration in the role entry.
- A superordinate role is "composed" of the subordinate roles, it gets all their privileges. That means, you could create you own superuser role in iteraplan by configuring "iteraplan_Supervisor" as subordinate role.

Permissions for Users and User Groups

Users and user groups can be assigned write access rights for specific objects – i.e. it is possible to assign individual, exclusive write permissions for particular instances of building blocks. Object-specific permissions mean the object can only be modified or deleted by the user or users belonging to the group to which the object-specific permission is assigned. Other users do *not* have the right to edit this object.

The assignment of object-specific write permissions can be made on either of two pages. The first is the **Permissions** tab which exists for each building block (see Permissions for individual building blocks). You can select instances of the building block individually on this tab, and assign write permissions for each instance to specific users or user groups.

The second way to assign object-specific write permissions is using the menu command **Object related Permissions**. This is explained in the following.

This section and its child pages also explain how to create, modify and delete users and user groups, and which menu commands you use for these operations.

Object related permissions

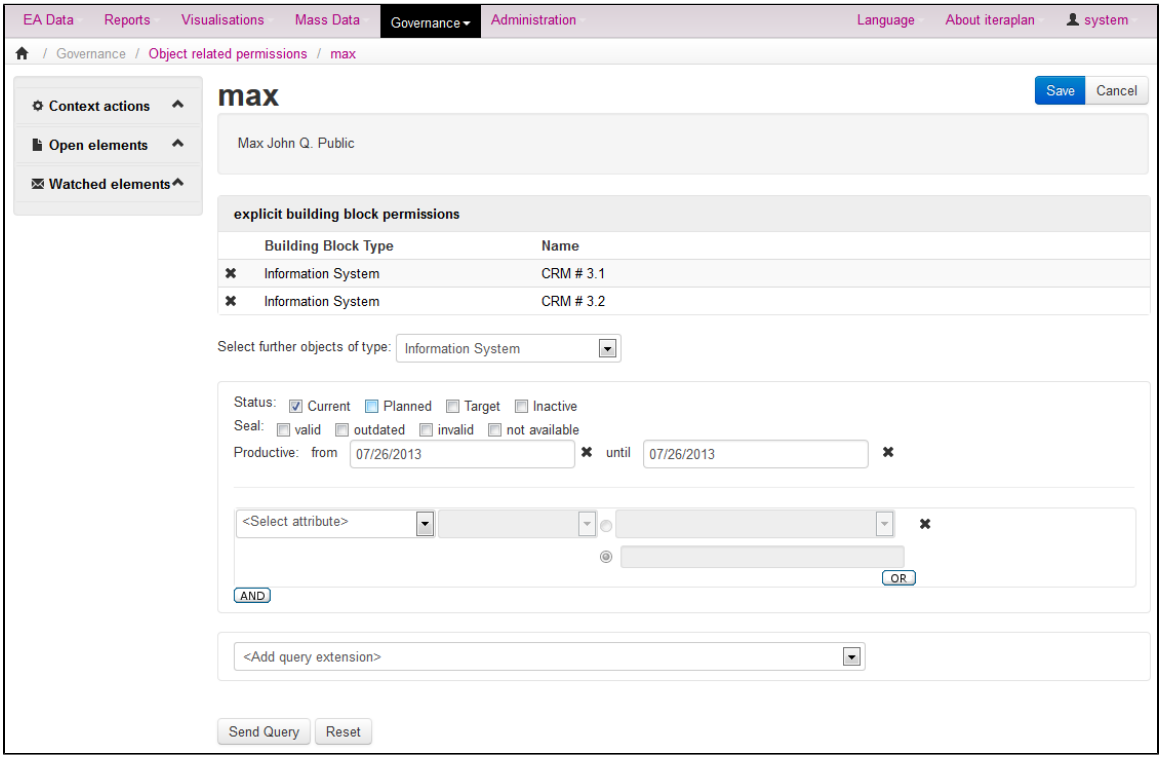
The **Object related permissions** page enables you to assign specific rights to users or user groups for editing particular instances of building

blocks. Only the designated user or user group will be authorised to edit the selected instance of the building block, irrespective of the user's role. These explicit building block permissions therefore always represent a restriction of permissions to a specific group or user.

If explicit permission is assigned to a user group for editing a particular instance of a building block, and a user in the group does not have write access to building blocks of this type, he or she will not be authorised to edit this specific instance of the block – despite the explicit permission. Write access for the building block type is a precondition for users or groups to be granted explicit permissions in this way.

> **superuser permissions**
> A superuser, most notably the user named "system" in the demo instances, does have write access to an object, even if the access is restricted via object related permissions.
> Superuser permissions are exclusively linked to the default role *iteraplan_Supervisor*.

Building blocks are selected by means of a query form which – similar to spreadsheet reports – enables you to locate blocks by specifying properties.

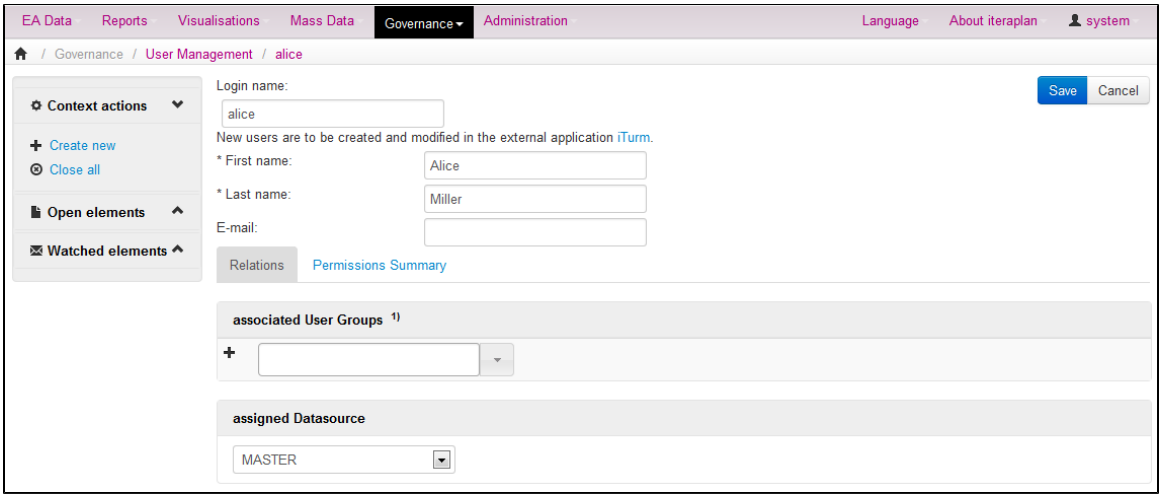


Assigning object-specific permissions to users and user groups
Working in Edit mode, you first select the building block type, e.g. information system. Click **Send query** to execute the query and display the list of results. You can select individual elements in the list of results by activating the check box next to their names. When you have completed your selection, click the button **Add explicit permissions for the selected objects**. This assigns the permissions to the user or user group which you previously selected.

User management (menu command)

The **User Management** command in the **Users, Roles and Permissions** section of the menu enables you to create new users and modify the properties of existing ones. User management is using the same transaction concept as editing Building Blocks, see here.

Each user has a *Login name*, *First name* and *Last name*, an *e-mail address* and can belong to one or more *User Groups*. Changes to the user's group affiliations take effect the next time he or she logs on.



Managing users

As well as an affiliation to a group, a user can also be assigned a datasource (via **Assigned datasource**) on which he or she works.

The mere step of creating users in iteraplan does not automatically permit them to log in to iteraplan. User and role management as such must be performed by the associated identity management system. To make user management a bit easier, iteraplan creates a user entry automatically when a user has been authorised by the identity management system and logs in to iteraplan for the first time.

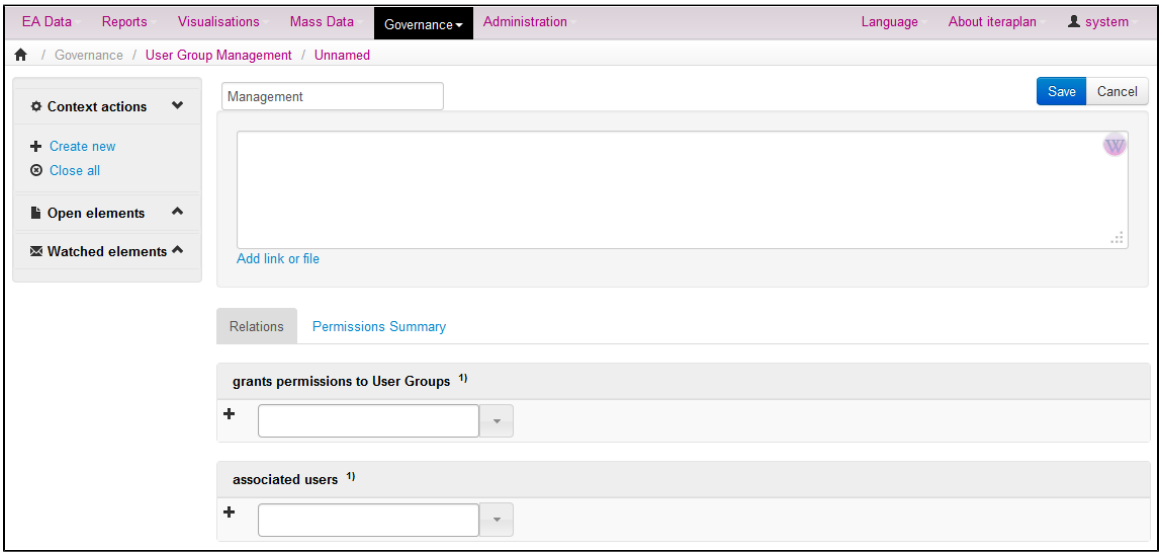
The following steps are therefore necessary to enable a user to access iteraplan:

1. Create the user in the associated identity management system
2. Optional, only if you want to specify more user details upfront: Create the user in iteraplan with the exact same login name as assigned in the identity management system, and enter e-mail address, datasource allocation etc.

User group management (menu command)

Users can be organised into groups. Each user group has a *Name* and a *Description*. Groups can be assigned object-specific permissions. To do this, either go via the menu command **Object related permissions** (see Object related permissions) or via the **Permissions** tab of the building block you are editing (see Creating and Editing Landscape Data).

A user group can grant its permissions to other user groups. For example, if group A grants its permissions to user group B, group B automatically has all permissions of group A. The page for managing user groups also provides functions for assigning users to the group.



Managing user groups

Role Permissions

There are three types of permissions for roles, determining how users with these roles will be able to work on building blocks. There are *functional permissions*, *permissions to edit (update, create, delete) building block types* and *explicit permissions for attribute groups*. These permissions are explained in the following sections.

Functional permissions

Basic access rights are assigned by means of *functional permissions*. These permissions can be assigned to individual roles on the **Permissions** tab of the page displayed by the **Roles and Permissions** menu command. These permissions control the visibility of menu commands and screens, as well as the access to specific functions.

Functional permissions cannot be extended, so there is no menu command for modifying them.

iteraplan works with the following functional permissions:

Functional Permission	**Description** ***Affected Feature or Area of User Interface, Format: Menu Entry / Tab***
Access iteraplan via REST API	Permission to use the REST API of iteraplan to its full extent. Note that no other permissions are required, neither for iteraQL queries nor for Excel export/import. REST API, UI n/a
Add and remove template files	Permission to add or remove Excel templates for spreadsheet reports, Visio templates for information flow diagrams and to create dashboard templates. Users without this permission can still use existing templates where applicable. *Administration / Document Templates*
Change the data source	Permission to select a different datasource for iteraplan for all or single users. See ?Other Configuration Options and Notes for more information about master and scenario datasources and ?User management to find out how to set the datasource user-specific. *Administration / Configuration / Assigned Datasource* *Governance / User Management / <user> / Assigned Datasource*
Configure attribute groups	Permission to add, edit or delete ?attribute groups and to control edit rights. Be aware that users without this permission can still change the attribute group of attributes individually if they are allowed to **Configure attributes**. *Administration / Attribute Groups*
Configure attributes	Permission to add, edit or delete ?attributes and ?date intervals. Be aware that users without this permission can still edit attribute values. *Administration / Attributes* *Administration / Date Intervals*
Create and execute queries for Diagram Reports	This is the second of three permission levels for diagram report queries. The users must have the first permission level for this permission to take effect. See **Execute saved queries for Diagram Reports** for the previous level. See **Edit (create, update and delete) and execute queries for Diagram Reports** for the next level. In addition to the first level this permission allows the user to create and configure their own queries. *Visualizations / <all diagram types>*

Create and execute queries for Spreadsheet Reports and Bulk Updates	This is the second of three permission levels for spreadsheet report and bulk update queries. The users must have the first permission level for this permission to take effect. See **Execute saved queries for Spreadsheet Reports** for the previous level. See **Edit (create, update and delete) and execute queries for Spreadsheet Reports and Bulk Updates** for the next level. In addition to the first level this permission allows the user to create and configure their own queries. In bulk updates, this permission only affects your possibilities to select elements you want to update, not the kind of updates you can perform on them. *Reports / Spreadsheet Reports* *Mass Data / Bulk Updates*
Create Seals	Permission to create ?seals. *EA Data / Information Systems / <one IS> / More / Create New Seal*
Download audit log file	Permission to download the audit log file (audit logging is not activated by default)
Edit (create, update and delete) and execute queries for Diagram Reports	This is the third of three permission levels for diagram report queries. The users must have at least the first permission level for this permission to take effect. See **Create and execute queries for Diagram Reports** for the previous level. In addition to the second level, this permission allows the user to save queries and delete saves queries. *Visualizations / <all diagram types> / Save query ...*
Edit (create, update and delete) and execute queries for Spreadsheet Reports and Bulk Updates	This is the third of three permission levels for spreadsheet report and bulk update queries. The users must have at least the first permission level for this permission to take effect. See **Create and execute queries for Spreadsheet Reports and Bulk Updates** for the previous level. In addition to the second level, this permission allows the user to save queries and delete saves queries. *Reports / Spreadsheet Reports / Save query ...* *Mass Data / Bulk Updates / Save query ...*
Edit configuration of iteraplan	Permission to edit the ?configuration of iteraplan. *Administration / Configuration*
Edit roles and grant permissions	Permission to manage ?roles and permissions. Be aware that users without this permission can still edit explicit permissions for building blocks (see ?object related permissions) *Governance / Roles and Permissions*
Execute Consistency Checks	Permission to execute ?consistency checks. *Reports / Consistency Checks* *Governance / Consistency Checks*

Execute iteraQL power queries	Permission to use the query console to execute ?iteraQL power queries. Note that this permission does not control iteraQL queries via the REST API. *Reports / Query Console*
Execute saved queries for Diagram Reports	This is the first of three permission levels for diagram report queries. See **Create and execute queries for Diagram Reports** for the next level. It allows the user to view and execute the saved queries for all diagram types. Users without this permission can only view dashboards and cannot see other links in the top menu. *Visualizations / <all diagram types>*
Execute saved queries for Spreadsheet Reports	This is the first of three permission levels for spreadsheet report queries. See **Create and execute queries for Spreadsheet Reports and Bulk Updates** for the next level. It allows the user to execute saved queries. Users without this permission cannot see the menu link. *Reports / Spreadsheet Reports*
Execute Successor Reports	Permission to execute ?successor reports. *Reports / Successor Reports*
Execute Supporting Queries	Permission to execute ?supporting queries. *Governance / Supporting Queries*
Grant object-related permissions per Building Block Grant object-related permissions per user	Permission to grant object related permissions per building block or per user. *EA Data / <building block type> / Permissions* *Governance / Object related permissions / <user>*
Manage users Manage user groups	Permission to manage ?users / user groups. *Governance / User (Group) Management*
Run Bulk Updates	Permission to run ?bulk updates. *Mass Data / Bulk Updates*
Run export (XMI) Run import (XMI) Run import (Excel)	Permission to run ?import and export. Note that this does not control the import/export via the REST API *Mass Data / Export/Import*
Run search on Building Blocks	Permission to run search on building blocks. This affects the global search as well as the search boxes for all building blocks.
Subscribe to Building Blocks and view all personal subscriptions	Permission to ?watch element changes and view your personal subscriptions in the left menu (watched elements).
View all subscribers of a Building Block	Permission to view watchers of a building block (see toolbar).

View Dashboard	Permission to view the Dashboard visualization. Watch out no to confuse dashboards with custom dashboards! *Visualisations / Dashboard*
View History	Permission to view a building block's history *EA Data / <building block type> / History*
View overview of all Building Blocks	Permission to view the overview of all building blocks. *EA Data / Overview*

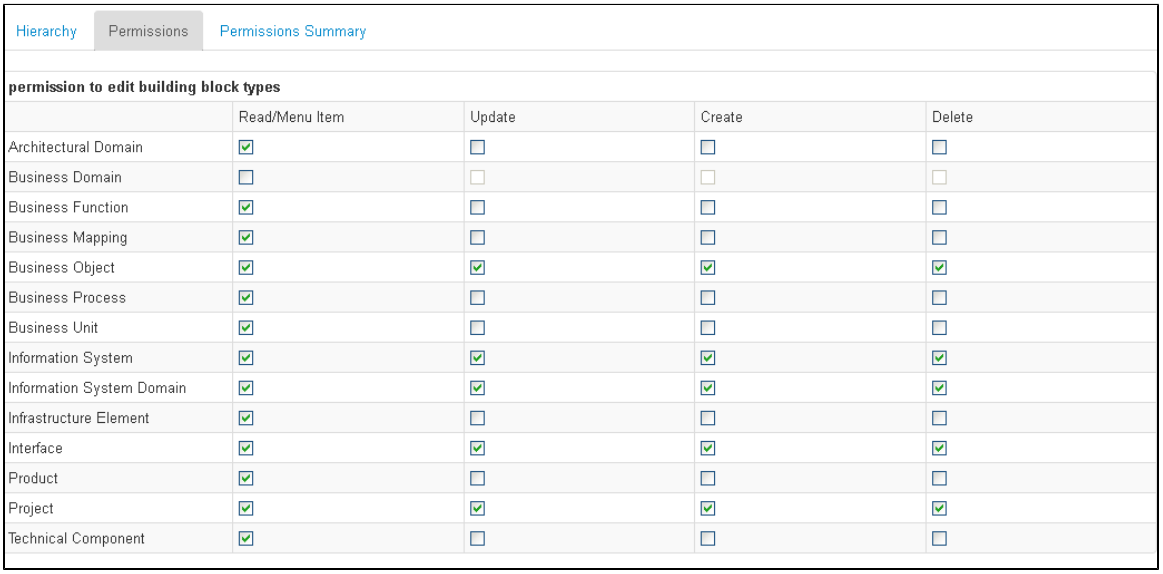
⚠ The functional permissions for building block types can be found in the first column of the matrix on the page Permission to edit building block types. But they are still functional permissions.

Permission to view and edit EA data

The visibility of menu commands – and thus also the different building block types – is controlled by functional permissions. The right to read, create, update and delete building block instances is determined by permissions to edit building block types. There are 11 building block types, and permissions for each can be assigned to roles. For example, the role *Manager* can be granted update permissions for information systems and projects. Access can also be assigned to multiple roles. These settings are made on the **Permissions** tab of the page displayed with the **Roles and Permissions** menu command.

There are four types of permission types:

- **Read/Menu Item:** The menu item for this building block type is available, and users having a role with this permission, can view all information's about building blocks of this type. This permission is also required to grant the write permissions. Note that the menu permissions are also functional permissions.
- **Update:** Users with this permission are able to edit building blocks of the respective type.
- **Create:** Users with this permission are able to create new building blocks of the respective type. The *Copy* and *New release* actions are also protected by this permission.
- **Delete:** Users with this permission are able to delete building blocks of the respective type.



Read and write access for attribute groups

It is possible to assign particular roles the permission to view and edit certain attributes, preventing confidential information (the values of attributes in the same attribute group) from being viewed (and modified) by every user. Confidential data can be modelled as attributes such as *Costs* and *Value added*, which can be organised into the attribute group *Strategy contribution*.

Use the section *Explicit permissions for attribute groups* to determine the visibility and write access rights for attributes of the selected group. You can indicate in each case whether permission is read-only or write access.

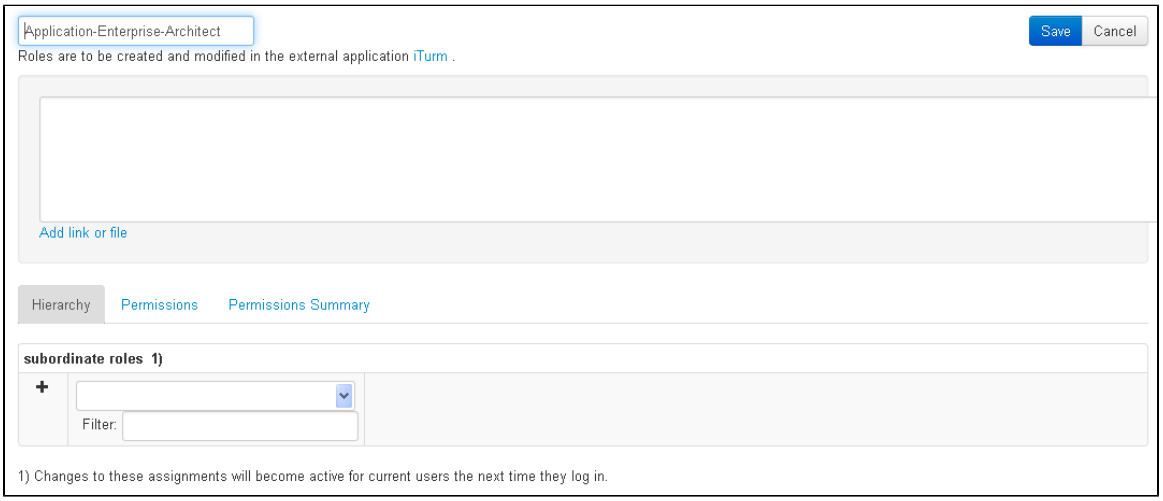
Once a role has been assigned permission for attribute groups, all users who do not have this role are excluded from the permission. (You assign permissions to roles on the **Permissions** tab of the page displayed with the **Roles and Permissions** menu command.).

Roles and permissions (menu command)

Use the **Roles and Permissions** menu command to set and modify the properties of a role. Each role always has a *Name* and a *Description*.

Hierarchy tab

You can structure roles by assigning them *subordinate roles*. Subordinate roles transfer all their permissions to their superordinate roles. Each role can have multiple superordinate roles and multiple subordinate roles. The transfer of permissions always takes place from bottom to top. Let us assume user *Smith*, for example, has the role *admin*. He or she also has all the permissions of the role *manager* in addition to the *admin* permi ssions, because *manager* is a subordinate role of *admin*.



Editing roles

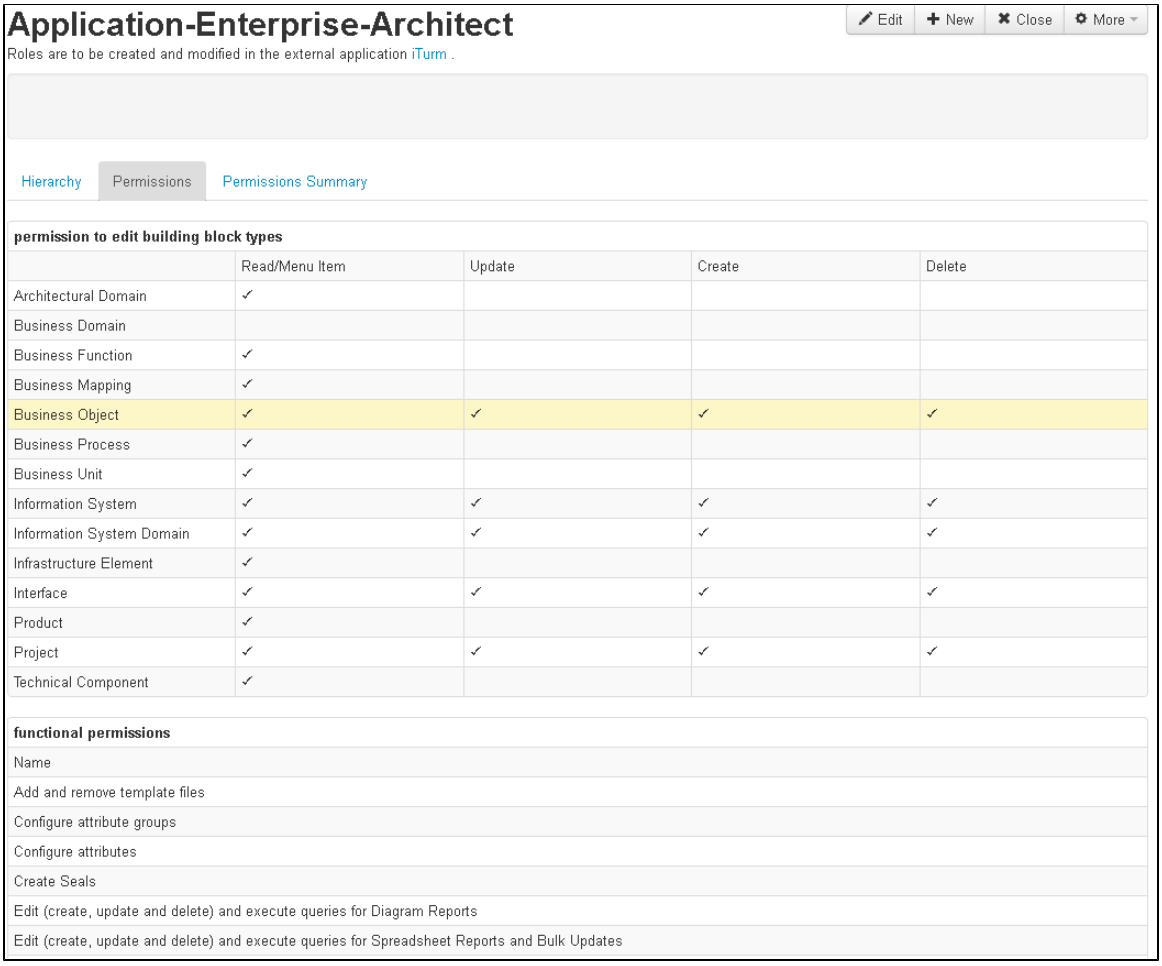
Permissions tab

Use the **Permissions** tab to specify permissions for the selected role. You can assign three types of permissions on this tab:

- Functional permissions;
- Permission to edit building block types;
- Explicit permissions for attribute groups.

Functional permissions determine which menu commands will be visible for the role in question. By assigning a role the permission to edit building blocks of a specific type, you enable users with this role to modify building blocks of the types specified. For instance, the figure above indicates that the role *CEO/CIO strategist* has permission to edit building blocks of the types business object, business process and product. Users with the designated role are authorised to create, modify and delete building blocks of these types.

Explicit permissions for attribute groups determines visibility and write access for the attributes in the specified attribute groups (View or View/Edit permission).



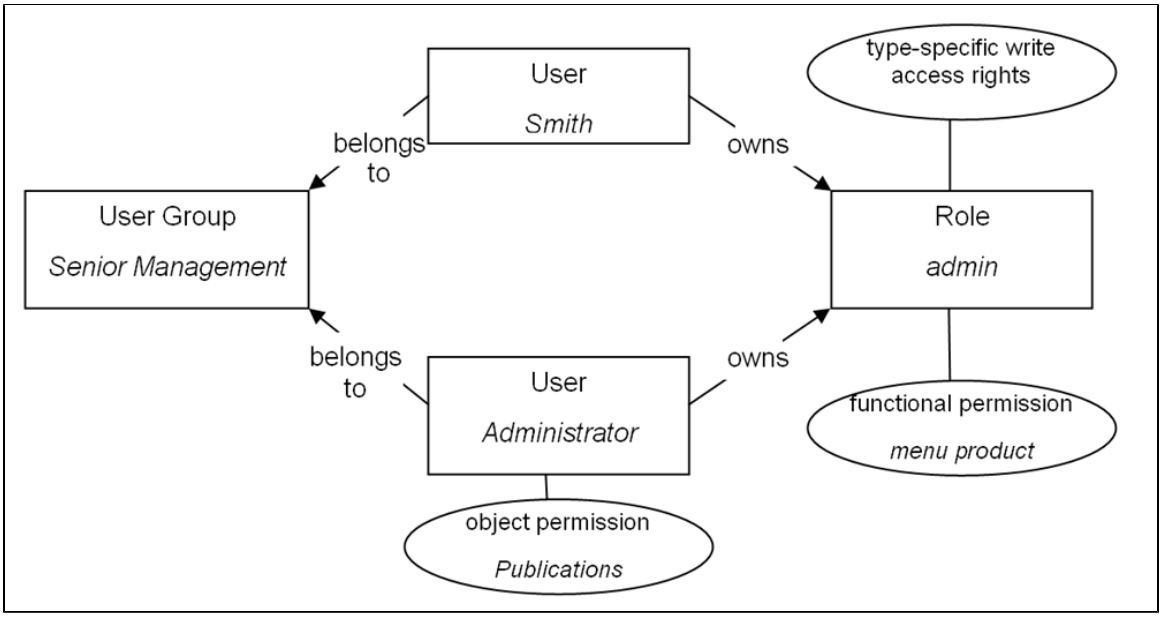
Managing role permissions

Permissions Summary tab

This page (see Users, Roles and Permissions) summarises all the permissions assigned to a role, also showing you the permissions which the role receives by way of its subordinate roles.

Example of permissions system

The two users *Administrator* and *Smith* in the next example both have the role *admin* and are members of the *Senior Management* user group. Access rights assigned to the *admin* role include the functional permissions menu *'Products'* and write access for products. Hence the administrator and Smith can edit every product excepting those for which object-specific write permission has been assigned to other users or groups. Let's now assume object-specific write permission is assigned to the *Administrator* role for the product *Publications.* This means *Smith* will no longer be able to edit the *Publications* products – even though he or she has type-specific write access rights for products through the *admin* ro le – because the *Administrator* now has (exclusive) object-specific write access.



Example illustrating interdependencies between users, user groups and roles with permissions

Typically Used Roles (Reference)

iteraplan comes with only two default roles *iteraplan_Supervisor* and *MainUser* in a fresh installation. *iteraplan_Supervisor* is a pre-defined superuser role and cannot be deleted. *MainUser* is a sample role with rather broad permissions, and can be used as a starting point.

For your reference, this page documents typically used roles, and permissions for these roles. These roles each model a key stakeholder in IT landscape management. Please feel free to adapt the suggested role configurations to your specific situation.

Table 1: Default roles

Name of role	Application Manager	IT-Architect	Application-Enterprise-Architect	Business-Enterprise-Architect	CEO/CIO/Strategist	Administrator iteraplan
Description of roles	Documents current and planned landscape	Blueprints, analyses and evaluates, plans & directs technical landscaping	Analyses & evaluates, plans & directs IS landscaping	Analyses & evaluates, plans & directs business landscaping (processes and information)	Standards, appropriate indicators, views to match information requirements	Administration of users, groups, roles and permissions for iteraplan

Table 2: Functional permissions assigned to default roles

Functional permission (visibility of menu commands)	Application Manager	IT-Architect	Application-Enterprise-Architect	Business-Enterprise-Architect	CEO/CIO/Strategist	Administrator iteraplan
Execute iteraQL power queries	X	X	X	X	X	
Download audit log file						X
Change the data source						X

Create and execute queries for Diagram Reports	X			X		
Create and execute queries for Spreadsheet Reports and Bulk Updates	X			X		
Create Seals	X	X	X	X	X	X
Edit (create, update and delete) and execute queries for Diagram Reports		X	X		X	
Edit (create, update and delete) and execute queries for Spreadsheet Reports and Bulk Updates		X	X		X	
Grant object-related permissions per Building Block						X
Run import (XMI)						X
Configure attribute groups		X	X	X		
Configure attributes		X	X	X		
Run Bulk Updates	X	X	X	X	X	
Edit configuration of iteraplan	X	X	X	X		X
Execute Consistency Checks	X	X	X	X	X	
View Dashboard		X	X	X	X	
Execute saved queries for Diagram Reports	X	X	X	X	X	
Run export (XMI)	X	X	X	X		X
Run import (Excel)	X	X	X	X		X

Grant object-related permissions per user						X
View overview of all Building Blocks	X	X	X	X	X	
Edit roles and grant permissions						X
Run search on Building Blocks	X	X	X	X	X	
Execute saved queries for Spreadsheet Reports	X	X	X	X	X	
Subscribe to Building Blocks and view all personal subscriptions	X	X	X	X	X	
Execute Successor Reports	X	X	X	X	X	
Execute Supporting Queries						X
Manage user groups						X
Manage users						X
View all subscribers of a Building Block	X	X	X	X	X	X
View History						X
Add and remove template files		X	X	X		X

Table 3: Permissions for building block types assigned to the default roles

Types of access permission for building blocks of type...	Application Manager	IT-Architect	Application-Enterprise-Architect	Business-Enterprise-Architect	CEO/CIO/Strategist	Administrator iteraplan
Business Domains					R	
Business Processes	R	R	R	C R U D	R	
Business Units	R	R	R	C R U D	R	

Products	R	R	R	C R U D	R	
Business Functions	R	R	R	C R U D	R	
Business Objects	C R U D	C R U D	C R U D	C R U D	R	
Business Mappings	R	R	R	R	R	
Information System Domains	C R U DX	R	C R U D	R	R	
Information Systems	C R U D	R	C R U D	R	R	
Interfaces	C R U D	R	C R U D	R	R	
Architectural Domains	R	C R U D	R	R	R	
Technical Components	R	C R U D	R	R	R	
Infrastructure Elements	R	C R U DX	R	R	R	
Projects	R	C R U D	C R U D	C R U D	R	

Abbreviation key: **C** - Create element, **R** - access to menu and read permission on elements, **U** - update elements, **D** - delete elements

iTURM

iTURM is a very simple application for managing assignments of roles to users, see also Users, Roles and Permissions.

Installation and configuration are described in the Installation Guide: Installing iteraplan and iTURM config files

To add or edit users, roles and assignments of roles to users, you need to log in to iTURM with a user who has the role iteraplan_Supervisor. In a freshly installed and initialized iTURM, a default supervisor user is already present. The address to access iTURM usually is:

```
http://<iteraplan-server>/iturm
```

Please note that users with the role iteraplan_Supervisor are also able to log into iteraplan as superuser, see Typically Used Roles (Reference).

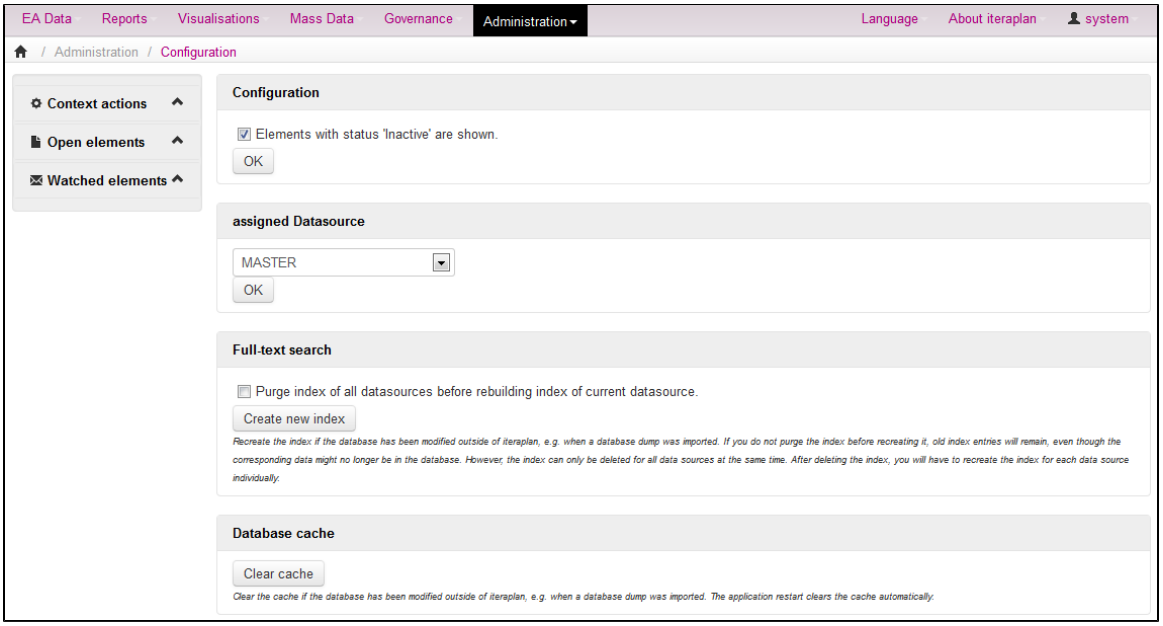
Changing the password of a user, however, only requires the knowledge of the username and the current password, see also Change Password.

Administration - Misc

The menu commands **Attribute Groups and Attributes** are discussed in (see Creating and Editing Landscape Data). The section below describes other commands, such as **Configuration, Clear Session**, and outlines the **Change password** procedure.

Configuration

The Configuration page provides options with which every user can choose whether to display or to hide elements with the status *Inactive*. With the default setting, elements in *Inactive* status are included in information displays. Any change to this setting is valid only for one session; if you wish to exclude inactive elements from the display, you have to make the change explicitly each time you log on.



Configuration page

Scenarios

iteraplan provides a concept of scenarios to support various use cases, as it is described in working with scenarios.

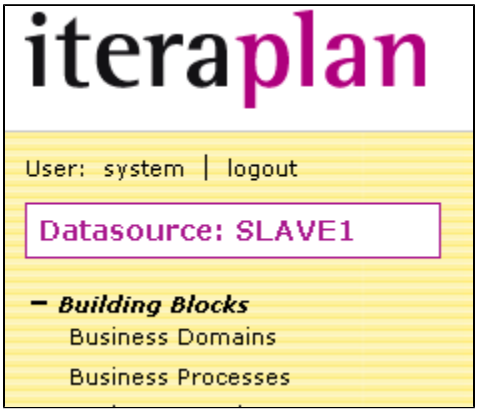
Technical management of scenarios and initial population with data is the task of the system administrator who manages iteraplan operation. He or she can copy the entire productive data to a scenario on a given day so as to provide planners with an appropriate baseline of data for their work.

There is as yet no function for automated transfer of productive data to a scenario and vice versa.

To start working with iteraplan scenarios you first have to configure the corresponding data sources. After successful configuration you can start creating your different scenarios. To work with a single scenario, you must first select it. Provided you have the necessary permissions, you can open the scenario directly in iteraplan via the menu command **Configuration**.
The scenarios preconfigured by the system administrator are selected from a drop-down list of data sources (assigned Datasources). The productive database is always the one designated MASTER. The preconfigured scenarios appear under the names assigned by the system administrator: "slave1" and "slave2".

It is possible to switch scenarios within the same iteraplan session. A message is displayed prompting you to confirm; you can then immediately begin work on the scenario data.

When you are working on a scenario (i.e. not in the MASTER data source), this is indicated at the top of the iteraplan menu.



Line showing the scenario which is currently selected (database connection)

Full-text search index generation

The global search functionality (see Global search) in iteraplan relies on a search index to retrieve search results quickly. In some rare cases, it

might happen that the search index does not reflect your actual building block data, leading to incorrect search results. The configuration page allows you to trigger re-creation of the search index by clicking the button *Create new index* (iteraplanDocumentation303:see screenshot above). The search index will always be stored in the location that has been specified by the administrator during installation. This is typically in a subdirectory of the Tomcat installation directory.

Should you suspect that results of the global search are incomplete, outdated or otherwise incorrect, please use the index re-creation function on the configuration page to create a clean and current index.

Please note that the **process of index creation may take up to five minutes** and should not be interrupted. You will not see any progress in your browser windows within that time, but processing does take place on the server.

Database cache

To improve the performance of the iteraplan application, the database caches are used (via Hibernate and Ehcache). If the database has been modified outside of iteraplan, e.g. when a database dump was imported, the database and cache will be out of synchronisation. If this occurs, the cache must be cleared or the iteraplan application must be restarted.

Working with scenarios

The concept of scenarios in iteraplan is quite powerfull because it provides you with the possibility to work with different data sources using one single installation of iteraplan. In combination with our role- and user-based access you can use this concept for different purposes as it is explained below.

Strategic Development of your Enterprise Architecture

In course of strategic planning of your Enterprise Architecture it is reasonable to create different possible future state scenarios which can enable you and your colleagues to make the right decision towards the desired architecture. This can be done in iteraplan by creating as many separate future state scenarios as required.

After making decision, it is required to define the concrete steps towards the decided architecture, so called planned scenarios. This can be done either in the main data source together with data of your current architecture or in a separate scenario which gives you the possibility to "play around" with your planned scenario without having an effect on you current data.

Scenarios and federated approach

In connection with the user-based access to the defined scenarios it is also possible to use iteraplan to support federated approach. You can define separate sources to keep data of single subsidiaries using a central installation of iteraplan. In addition you can grant access to certain users who can switch between the sources and have a look at the development of the corresponding data. This activity is even better supported by our new dashboard which provides a quick overview of the data.

Sandbox

Of course there is another possibility how you can use the scenario concept – as a kind of sandbox. You can create a datasource for testing purposes. It can be used by your colleagues either to become acquainted with the tool or the subject of EAM or just for testing purposes.

Access Control
While using multiple scenarios you can define the access to these scenarios as follows. For the existing users you can define which data source is visible for which user. This is a 1:1-relation. Additionally you can define which role has the permission to switch between different data sources.

Please be aware that only one authorization concept is available for all defined sources, i.e. only roles and permissions defined in the original data souce (= MASTER data source) are relevant. Changes concerning roles/permissions made in other data sources are ignored.

Chance to contribute
If you are using the scenario concept for another purpose please feel free to share this with other community members.

Clear Session

Users can clear their current session with the **Clear Session** command, which is available in the user menu (top right). It makes possible to reset the application to a consistent state following problems or errors. All changes to building block data made by the *user executing the session clearance* will be discarded.

The clearance is performed only for the user executing the command, *not* for other users.

Change Password

Integrated identity management

If iteraplan is integrated into your company's single identity management system, please contact the relevant person or unit in your company in order to change your password.

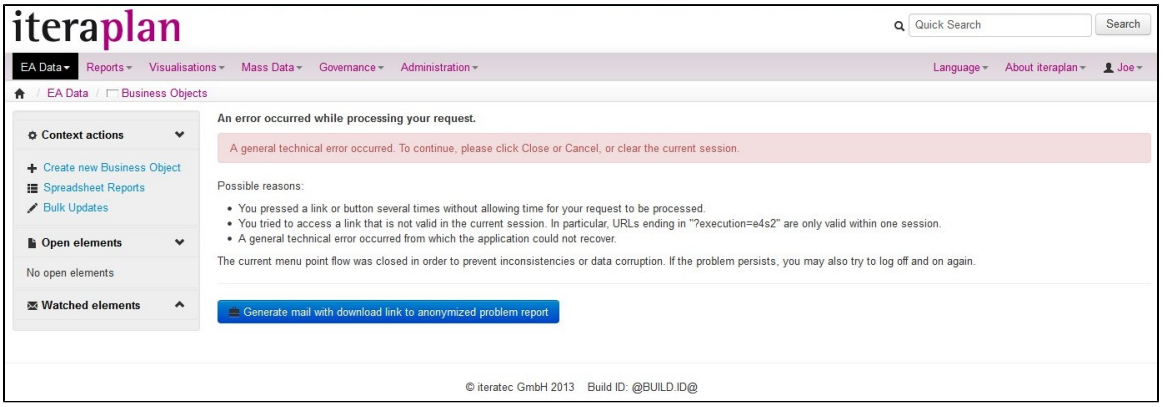
iTURM

Otherwise, please use the iTURM application provided for this purpose. The address is usually: http://<iteraplan-server>/iturm/password

Problem Reports

When a user gets to the iteraplan error page, all information about the reason is available technically. Therefore iteraplan has established a mechanism to gather this information and make it accessible to iteraplan administrators (= someone who has superuser privileges).

You will find a new button on the error page (see screenshot). Clicking on that button creates a notification email in your local mail client, which you can send to your administrator. In the background, the anonymized problem report is created in-memory. The administrator can download it as ZIP from the link inside the generated mail.



Only administrators with superuser privileges can download and read these problem reports. The maximum amount of reports in memory is limited, so that it won't cause any memory issues.

Please note, that all reports will be lost, when the web application is shut down. In this case, all links to problem reports becomes invalid, and a new report has to be created, if the problem still persists.

Inside the ZIP, you will find text files for several technical aspects of the iteraplan web application like "stacktrace", "request" and so on. Login names, passwords and other sensitive information is either excluded or masked.

The administrator's email address in the generated mail can be configured during installation, or changed later in the properties file "iteraplan_local.properties", property "admin.email".

Problem reports are:

Helpful ... because all information that may be needed for the analysis of the problem is gathered at one place when the error occurs.

Transparent ... because the administrator has full insight to the report, which consists of a ZIP that contains several text files.

Confidential ... because nothing will be sent automatically. iteraplan users and administrators have full control about the workflow.

Secure ... because only administrators are allowed to view the report information.

Voluntary ... because the feature can be switched off by setting the property "problem.reports.enabled" in the file "iteraplan_local.properties" to "false".

Release Notes

iteraplan 3.3.1 Final

Fixed bugs (vs. 3.3.0):

- Fixed an issue which occurred, when trying to apply a filter on Relationship-End with IteraQL.
- Fixed an issue preventing successful imports if there were values missing for mandatory attributes in Business Mappings.
- Update of multivalued attributes in interfaces is now possible.
- Fixed an issue when using the query filter on building block overview pages.
- Fixed an issue causing errors with Export/Import and Power Queries, if attributes had invalid names.
- Fixed issue which allowed the creation of attributes and attribute values with invalid or conflicting names.
- Fixed an issue preventing the re-import of a partial export using enumeration attribute values in its filter predicate
- Fixed an issue preventing iteraplan to run correctly when installed in Tomcat's ROOT context.
- Fixed an issue which lead to a permanent opened coloring dialog for Nesting Cluster Diagram.
- Fixed misleading error message during file downloads.
- Fixed issue allowing the creation of building blocks with duplicate names when using certain database configurations.
- Fixed an issue which sometimes prevented successful iteraplan installation using console (headless) mode.
- Fixed an excel import issue with elements of type Isr2BoAssociation, Tcr2IeAssociation and InformationFlow which prevented their creation or deletion.

Improvements/new features (vs. 3.3.0):

- Added a default color to enumeration attribute values added by the Excel/XMI import.
- Improved error messages and error handling in some import cases (both Excel and XMI).
- Response status codes in iteraplan's REST API are now more meaningful.
- The output format "CSV" is now supported for direct list export in Tabular Reporting.

iteraplan 3.3.0 Final

=> Annotated Release Notes 3.3 (PDF)

New features (vs. 3.2):

- Major overhaul of iteraplan's export & import capabilities
  - partial export/import: possibility to export and import only a specific subset of all data
  - Different strategies for import: Additive (Create/part. Update) & Overwrite (Create/Update/Delete)
  - REST interface: Write access to single building block elements (Create/Update/Delete)
  - HTTP Basic Authentication for REST-API access
- New direct Excel export: WYSIWYG-export of simple-list query results or spreadsheet reports
- Report type independent overview for all saved queries
- Added Kerberos-based Single Sign On in combination with Microsoft IIS
- Tested for Oracle 12 compatibility
- Backwards 3.x compatibility with Excel import improved
- LDAP-based authentication: given and last names are synchronized at login time
- Improved IE10 support
- Improved error reporting capabilities

Fixed bugs (vs. 3.2):

- Fixed issue with displaced text on attribute description popups
- Fixed issue with missing connection lines in information flow diagrams
- Fixed a bug causing an error page during configuration of spreadsheet reports. It was related to certain business mapping queries.
- Fixed a bug preventing the deletion of infrastructure elements in certain cases
- Fixed issue with notifications not being sent for certain status changes
- Fixed issue with switching the language on the query console page
- Fixed issue with changing enum attribute values using Internet Explorer
- Fixed issue with deleting an infrastructure element
- Various minor improvements and fixes

iteraplan 3.3.M2

This is a milestone release. Do not use in production environments.

New features (vs. 3.3.M1):

- Major overhaul of iteraplan's export & import capabilities
  - partial export/import: possibility to export and import only a specific subset of all data
  - Different strategies for import: Additive & Overwrite (Create/Update/Delete)
  - REST interface: Write access to single building block elements (Create/Update/Delete)
  - HTTP Basic Authentication for REST-API access
- Tested for Oracle 12 compatibility
- Backwards 3.x compatibility with Excel import improved

iteraplan 3.3.M1

This is a milestone release. Do not use in production environments.

New features (vs. 3.2):

- Added Kerberos-based Single Sign On in combination with Microsoft IIS
- LDAP-based authentication: given and last names are synchronized at login time
- New "netto" Excel export: WYSIWYG-export of simple-list query results or spreadsheet reports to an Excel file
- Report-type-independent overview for all saved queries
- Improved IE10 support
- Improved error reporting capabilities

Fixed bugs (vs. 3.2):

- Fixed issue with displaced text on attribute description popups
- Fixed issue with missing connection lines in information flow diagrams
- Fixed a bug causing an error page during configuration of spreadsheet reports. It was related to certain business mapping queries.
- Fixed a bug preventing the deletion of infrastructure elements in certain cases
- Fixed issue with notifications not being sent for certain status changes

iteraplan 3.2.0 Final

New features, changes and fixed bugs (vs. 3.2.RC1):

- Smaller improvements of REST-interface JSON-structure
- Edit backward-compatibility of CE Visio template
- Improvement of sample data (date intervals and dashboard example)
- Fixes regarding XMI-import/export
- Fix to avoid SQL Exception on Custom-Dashboard initialization
- Performance-improvement of Nesting-Cluster Diagram-Generation
- Enabled defective attribute-group-menu for MS SQL Server use
- Fix to enable last modification properties for instances of relationship types
- Validation of support for MS SQL Server as backing database
- Finalization of documentation

iteraplan 3.2.RC1

New features and changes (vs 3.1):

- Improved robustness of Excel and XMI import
- Performance improvements of Excel and XMI import
- Nesting Cluster Diagram:
  - An option was added to show inner elements without relation to any of the outer elements in a separate block.
  - Coloring of displayed elements according to their enumeration attribute values was enabled

        - Color legends were added
        - Filtering of inner and outer elements was enabled
    - Introduction of context visualisation "Neighborhood Diagram" for Information Systems
    - Possibility of more than one dashboard template for each building block type was added
    - Improvement of the history functionality by enabling following information to be included:
        - Changes to successors and predecessors of Information Systems and Technical Components
        - Changes to the assignments for Enumeration and Responsibility attribute types
        - Changes to Business Mappings

In final testing phase (vs 3.1):

- Introduction of a REST-interface for requesting landscape data
- Addition of support for MSSQL Server as backing database.

Fixed bugs (vs 3.1):

- Problems when assigning several Infrastructure Elements to the "uses" relation of another Infrastructure Element where fixed
- Creating Nesting Cluster Diagrams without contents now works with PDF as well
- Issue with enumeration attributes without values during Export resolved
- Some issues in the display of custom dashboards (wrong numbers, colors, layout) were fixed
- Portfolio Diagrams don't change orientation depending on their output format anymore
- Several issues with the Visio output format of diagrams were fixed
- An error when using the query extension "Properties of assigned Interfaces" of Information Systems was fixed
- Other minor fixes

iteraplan 3.1.0 Final

New features, changes and fixed bugs (vs. 3.1.RC1):

- Improved start script in Community edition to find the Java runtime more reliably from the Windows registry
- Fixes to the Excel import, which corrupted the hierarchy information in the database in 3.1RC1
    - Fixed NullPointerException in Excel import when wrong cell formats are set
    - newly added Building Blocks can now directly be used in a new relationship
    - reduced the file size of generated Excel template files
- Various minor improvements to the new Custom Dashboards
- Masterplan diagram:
    - Added an option to colour date interval bars according to their defined colour
    - Unified labels of the time span bars
- The default state of switch "Elements with status 'Inactive' are shown." is now configurable server-side
- Various improvements to the user documentation

iteraplan 3.1.RC1

New Features and Changes (vs. 3.1.M2):

- Master plan diagram:
    - Visio export support for new masterplan configuration options was added
    - A management UI for date interval definitions has been added (expected minor changes before the 3.1 final release)
    - Saving and loading of queries with new masterplan configuration options works now
- Custom-defined dashboards:
    - Administrators can embed saved diagram queries into dashboard templates. Height or width of the resulting diagrams can be influenced as well.
    - Users can create their individual dashboards from administrator-defined dashboard templates, by choosing which set of building blocks shall be fed into the dashboard's diagrams
    - A rich-text editor for dashboard templates was added, which helps insert Wiki syntax for saved diagram queries.
- Portfolio diagram: When both axes show discrete attributes, circles are positioned into portfolio tiles without overlapping each other. Under these circumstances, exact positioning of circles provides no value at all, as they overlap in almost all cases.
- Tree View for hierarchical types:
    - Hierarchies can be reordered by dragging and dropping elements or entire sub-trees to a new position in the hierarchy
- Building block overview lists can now also show status (applies to Information Systems and Technical Components)
- Query Console results for iteraQl queries can be transferred to graphical report configurations, where applicable. This brings the advanced query power of iteraQl to iteraplan's graphical reports.

Fixed Bugs (vs. 3.1.M2):

- Excel import: imported Interfaces were often incorrectly mapped to existing interfaces, mixing up data for interfaces: Fixed
- Excel export: Interfaces without a name were missing in full model Excel exports; fixed
- Bulk updates: Some relation dropdown were always empty: fixed to show available elements for settings relations
- Spreadsheet reports: Fixed date format for date attribute columns in Simple list results
- Spreadsheet reports: Fixed incorrect date format validation when specifying operands for date attribute comparison. It was always expected in ISO date format, not the currently active locale's date format.
- Corrupted layout on definitions page was fixed

iteraplan 3.1.M2

New Features and Changes (vs. 3.1.M1):

- Master plan diagram:
  - All Building Block types can be used in the diagram, also in combination with the new time-span reporting capability
  - Indirectly related elements can be included in the diagram as well, to show constellations like Product, supporting Information Systems, and underlying Technical Components
  - Restructured configuration UI for better usability with new features
  - Some new functionality is not yet available in Visio exports, but in all other export formats (this is still work in progress and we will further improve it)
- Custom-defined Dashboards: Use Wiki syntax to define dashboard templates for each building block type. Users are then able to create individual dashboard instances where they can chose the base set of building blocks that the embedded diagrams should reflect. (this is still work in progress and we will further improve it)
- Tree View for hierarchical types: In addition to flat list views, hierarchies can now be viewed as trees
- Export/Import: XLSX Excel file format support further improved for import and export
- Deployment: iteraplan bundles include Tomcat 7 now, Java 7 is supported
- Deployment: Compatibility with Java 7; Java 6 support is unaffected and still the required minimum
- Deployment: Now all settings in iteraplan.properties can be overridden by iteraplan_local.properties, making it easier to maintain site-specific settings across version updates

Fixed Bugs (vs. 3.1.M1):

- Fixed caching parameters for lists of building blocks. This led to missing/ outdated entries in dropdowns when setting relations between building blocks
- Composite Diagrams: Write permissions for at least one building block were required; fixed to require read permissions only
- Removal of values from Enumeration or Responsibility attributes caused an internal error; fixed
- Wiki syntax in multi-line text attributes was not parsed properly; fixed
- Excel Import: Improved checking for reserved characters in names
- Special characters in names of saved queries were not correctly displayed in dropdown choices "filter by saved query" on building block overview pages; fixed

iteraplan 3.1.M1

New Features and Changes (vs. 3.0.4):

- Search saved queries: The reporting user interface offers a full-text search on names and descriptions in a list of saved queries
- Master plan diagram: Additional time spans can be reported for each building block; they must be captured in date attributes (this is still work in progress and we will further improve it)
- Export/Import: XLSX Excel file format is supported for import and export (default in Excel 2007/2010/2013, a converter for older Excel versions is available from Microsoft)
- Export/Import: Exported templates include dropdown lists to assist entering relations between building blocks
- Metamodel: Infrastructure Elements have a usage relation
- Improved support for Active Directory Domain forests when using LDAP authentication
- Deployment: Compatibility with Tomcat 7; Tomcat 6 support is unchanged

Fixed Bugs (vs. 3.0.4):

- Improved an insufficient log message in XMI import
- Information Flow Diagram: Diagram title and content can no longer overlap in Visio Exports
- Excel Import: Improved normalization of names with releases, avoiding "Not found with that name" errors
- Excel Import: Numbers in text-formatted cells can now be imported into text attributes

iteraplan 3.0.4

New improvements and fixed bugs (vs. 3.0.3):

- [ITERAPLAN-573] - Copying a Responsibility attribute does not copy its assigned values
- [ITERAPLAN-885] - Mail address validation in user management fails on dashes in domain names
- [ITERAPLAN-886] - When using the French UI, some Javascript code is broken
- [ITERAPLAN-888] - Enumeration attribute values of an element can't be changed if another element has the same value assigned and hibernate second level cache is deactivated
- [ITERAPLAN-897] - Excel Import wrongly classifies addition of new enumeration literals as unapplicable change
- [ITERAPLAN-899] - Underscores in Interface names are not shown properly in Interface section of Information Systems/Technical Components
- [ITERAPLAN-907] - Excel Export: hierarchical names and formatting incorrect in rows > 308
- [ITERAPLAN-925] - Tooltip picture does not appear when running iteraplan without an explicit context root
- [ITERAPLAN-987] - Wiki syntax for text attributes is not interpreted properly
- [ITERAPLAN-988] - XMI import doesn't respect relations defined in the XMI file
- [ITERAPLAN-1000] - Importer causes non-critical database corruption when changing hierarchical relationships of already existing elements
- [ITERAPLAN-1004] - Importer won't accept a number when it's expecting a string in attribute values
- [ITERAPLAN-1010] - Attribute Detail popups cannot be closed in Chrome
- [ITERAPLAN-1026] - Excel Template Export of a metamodel with enumeration properties with many and/or long literals can result in an Error
- [ITERAPLAN-1033] - ModelLoader throws exception when running against oracle databases with many building blocks (affects Excel import)
- [ITERAPLAN-1194] - Ecore import can't handle string values where the first character was escaped during export

New Feature (vs. 3.0.3)

- [ITERAPLAN-1092] - Excel Import via REST

iteraplan 3.0.3

New features, improvements and fixed bugs (vs. 3.0.2):

- [ITERAPLAN-785] - Excel-Import: check for duplicate names not always working for Information Systems and Technical Components
- [ITERAPLAN-819] - Excel Import is only possible with supervisor user
- [ITERAPLAN-826] - Massupdate cannot edit text attributes (freely defined ones)
- [ITERAPLAN-827] - Select-Dropdowns always add first element after clicking on them
- [ITERAPLAN-829] - PrintView: Missing images - e.g. Interface-Direction
- [ITERAPLAN-823] - Improve MetaModelMatcher and MetaModelDiffer to make imports more robust

iteraplan 3.0.2

New features, improvements and fixed bugs (vs. 3.0.1):

- Attributes on relation between Technical Component and Infrastructure Element
- UI: Show attribute details on-click not mouse-over
- UI: Show active datasource in breadcrumbs
- Fixed: IE8 cannot download saved Excel reports over HTTPS connections

iteraplan 3.0.1

New features and fixed bugs (vs. 3.0):

- Improved layout and content page for exported Excel sheets.
- Improved web UI for Excel Import.
- Detailed user documentation for new Excel export.
- Warning when showing business mappings fixed.
- Bugfixes in diagram configurations.
- Bugfixes in subscriptions.
- Several minor bugfixes concerning navigation, form usability, security and UI.

iteraplan 3.0.0 Final

New features, changes and fixed bugs (vs. 3.0.RC2):

- Support for various devices and resolutions (like smartphones/tablets) with responsive design
- Improved Print View with all tab sections expanded
- Several minor bugfixes regarding GUI, export, bulk updates and more

iteraplan 3.0.RC2

New features, changes and fixed bugs (vs. 3.0.RC1):

- New Excel-Export and Import, removed old Excel-Import
- Improved Email Subscriptions (Watched Elements)
- Improved example data and documentation, reflecting new features
- Several bugs regarding GUI, export, query console and more

iteraplan 3.0.RC1

New features and changes (vs. 3.0.M2):

- Improvements on new UI - based on Twitter Bootstrap, HTML5/CSS
    - Show subscriptions in Context Menu
    - Allow refresh in reporting and close all for open elements of one type
    - Keep collapse state of open elements and watched elements
    - Breadcrumb navigation showing hierarchy
    - Dashboard integrated in visualisation overview
    - Replaced Dojo with JQuery as standard JavaScript framework
    - Many further improvements with less tables, new tooltips, help, ...
- Great enhancements of InformationFlow Diagram (thx. to Sponsorings)
    - Upload Visio template to retain layout in InformationFlow Diagram
    - Filtering BusinessObjects in InformationFlow Diagram
    - New option to show attributes of relation BusinessObject - InformationSystem
- Nesting Cluster Graphic: Colouring based on number attributes
- Upload Excel templates for spreadsheet reports
- Filter attributes on relation BusinessObject - InformationSystem
- Add user-chosen columns to building block type overview pages
- Edit/Copy/Delete actions directly accessible in element overview page
- Reordered hierarchy and relation tabs to reflect same schema across all EA data elements
- Changed QueryConsole - new iteraQL-Syntax
- Improved example data, including attribute CRUD on relation BusinessObject - InformationSystem
- Improved documentation, with new screenshots, reflecting new UI

Fixed Bugs (vs. 3.0.M2):

- Several other minor bugs regarding GUI, export and more

iteraplan 3.0.M2

New features and changes (vs. 3.0.M1):

- New UI - based on Twitter Bootstrap, HTML5/CSS
    - cleaner design, nice Icons for most actions
    - better structured menu, showing Visualisations in TopMenu
    - default actions highlighted
- Finer Permission Control - Distinguish between Create, Delete and Update
- Filtering Interfaces in InformationFlow Diagram
- Sorting of columns in Element Overview/Search pages
- Theming - Define colours of attributes as default colour schema
- Colour range for numerical attributes

- Improved permission names
- Enhanced History functionality (EE only)

Fixed Bugs (vs. 3.0.M1):

- Attribute order gets updated after save
- Updated libraries and refactorings
- several other minor bugs regarding GUI, export and more

iteraplan 3.0.M1

New features and changes (vs. 2.9.1):

- Show/edit self relations of InformationSystems in both directions
- The association from Information System Release to Business Object is attributable
- New overview page showing top elements for all building blocks
- Drop-down boxes for relations are combined with search filters and offer better performance
- Landscape diagram: new options to fine-tune aggregation of hierarchical elements
- Nesting Cluster Diagram: hierarchy can be unrolled and relations with attributes can be visualized
- Information flow diagram: Line captions can show more than one aspect
- Enhanced History functionality (EE only)
- Added localizations for Swedish (contributed by Peter Svensson)
- Several GUI Improvements

Fixed Bugs (vs. 2.9.1):

- Filtering Information System based on their Interface direction returned wrong results
- Bulk update: "Add link or file" button works in all lines now
- Attribute groups for Business Mappings could not be expanded / collapsed
- XMI Export lacks values of date attributes
- Printing with the non-default theme showed a broken page
- Bad server address in url of direct link to visualization was fixed
- several other minor bugs regarding GUI, export and more

iteraplan 2.9.1

Bugs fixed which were part of release 2.9.0:

- ITERAPLAN-140 - Error-page when selecting an invalid ExcelWorkbook as template for excel export
- ITERAPLAN-306 - Several Context Graphic trigger TechnicalException (EntityNotFound)
- ITERAPLAN-307 - Dashboard not shown in the menu navigation if an user has functional permissions only for dashboard
- ITERAPLAN-308 - NullPointerException when deleting an attribute within a new created ATG
- ITERAPLAN-323 - Mouseover image partially overlayed by other page content
- ITERAPLAN-327 - Tabular reportings Xmi-Export: the availableForInterfaces attribute is not exported
- ITERAPLAN-364 - Clear error messages in Excel Import page on new load
- ITERAPLAN-365 - Page number not reset after reload
- ITERAPLAN-366 - Untrimmed values for responsibility attributes cause an error after Excel import
- ITERAPLAN-369 - Improve GUI validation for attribute type input fields
- ITERAPLAN-382 - Inserted additional columns in Master plan graphic shown as black cells
- ITERAPLAN-386 - Nesting Cluster Diagram and Permissions
- ITERAPLAN-399 - Landscape Diagram: NullPointerException when not using a names legend
- ITERAPLAN-412 - Direct execution of saved queries fails to download with IE + HTTPS

Improvements over release 2.9.0

- ITERAPLAN-187 - Load Attribute Type descriptions using Ajax
- ITERAPLAN-271 - Retain expand/collapse state of attribute groups in building block edit mode
- ITERAPLAN-294 - Excel Import for ATs: cell references in error and warning messages
- ITERAPLAN-295 - Excel import for Object-Related-Permissions: cell reference in error and warning messages
- ITERAPLAN-298 - Improve page load speed by reducing the resources count
- ITERAPLAN-309 - The name of the doesObjectExist() method is misleading
- ITERAPLAN-336 - Spreadsheet reports: Use field "suitable for interfaces" in filter criteria and as optional spreadsheet column
- ITERAPLAN-338 - Improve performance of list filter fields when thousands of elements are in the list

- ITERAPLAN-357 - Performance improvement in Building block sorting by hierarchical name
- ITERAPLAN-361 - Improve Nesting Cluster Diagram Output
- ITERAPLAN-362 - QR Code in diagram reports should be made larger
- ITERAPLAN-371 - Name as optional identificator for object related permission import
- ITERAPLAN-372 - Use semicolon as delimiter in object related permission import
- ITERAPLAN-374 - Nesting Cluster-Diagram: The initializing of the Candidates takes too long
- ITERAPLAN-380 - Refactoring of ColorDimensionOptionsBean to use a map instead of two lists
- ITERAPLAN-381 - Nesting Cluster Diagram: allow simple switching of inner and outer element types
- ITERAPLAN-384 - Space before colon when looking at attributes of building blocks
- ITERAPLAN-388 - Nesting Cluster Diagram: improve robustness
- ITERAPLAN-403 - Include MetaInformation of Attributes in Excel-Export
- ITERAPLAN-340 - Masterplan: Add self relations of Informationsystems
- ITERAPLAN-332 - Refactoring of the iteraQl Loader
- ITERAPLAN-387 - Documentation for filtering with seal is missing

iteraplan 2.9.0

New Features and Changes

- New diagram types:
  - Nesting Cluster Diagram
  - Bar Chart
  - Pie Chart
  - Composite Bar and Pie Charts
- Information flow diagram (Enterprise Edition only)
  - Visio exports include macros for impact analysis
  - Visio macros for copying diagram layout
- Landscape Diagram - new options & improved readability
- Masterplan Diagram - additional self-defined columns available
- PowerQueries (Enterprise Edition only) - Multiple hops on the meta-model & Traversing hierarchies
- Generating a Bar-Code on diagrams
- Performance improvements
- Management of Business Mappings in matrix style
- Quality Seal on Information Systems
- Custom Excel templates for exports
- Improved Excel import (Performance, Subscriptions, AttributeDef.)
- Improved XMI Import
- Unified copy & new release function
- Successor/Predecessor reports for Technical Components
- User Interface: New Action Icons on Building Block pages
- Possibility to clear the Hibernate second-level cache
- Further improvements based on community feedback, contributions and sponsoring

Fixed Bugs

- see detailed Milestone Releases for all Bugfixes

iteraplan 2.9.0

New features and changes (vs. 2.9.RC1):

- Management of Business Mappings in matrix form was improved further
- Dashboard: Improved visual alignment of the diagrams and tables
- iTURM allows administrator users to reset other users' passwords
- Building block dialogues: attribute groups do no longer show "Edit" links for users without the required permission
- Information Flow diagrams: Improved layouting performance when many sub-IS are to be displayed
- Full text search boxes handle special characters more gracefully, giving better warning/error messages

Fixed Bugs (vs. 2.9.RC1):

- User and role management: Several bugs have been fixed
- XML export for spreadsheet reports: attribute type information was not exported: Fixed

- Nesting Cluster Diagram robustness was improved
- Attribute type group management: Sometimes permission changes were not saved: Fixed
- Dashboard: Permissions on attributes were ignored: Fixed
- Some JavaScripts were not properly loaded during first page load if advanced authentication mechanisms are in use: Fixed
- Visualization tab in building block details works more reliably on older browsers
- Role permissions were not completely respected when change notifications were sent out: Fixed
- Permissions were enforced too strictly when loading saved reports involving business mappings: Fixed
- Various Performance improvements and bug fixes

iteraplan 2.9.RC1

New Features and Changes

- Information flow diagrams: Visio exports include macros for impact analysis (Enterprise Edition only)
- Information flow diagrams: Visio macros for copying diagram layouts (Enterprise Edition only)
- Landscape diagrams: Improved readability for Visio diagrams (Axis elements on level 3 were made higher, to give room for longer names)
- Performance improvements
    - Performance improvements for SVG-based graphics. With large numbers of building blocks, graphics generation is multiple times faster
    - JavaScript performance for editing business mappings has been improved
    - Various other improvements
- Excel Import (Enterprise Edition only):
    - Improved detection of interface flow direction
    - Attributes of all types can now be created via Excel Import (using a dedicated workbook)
    - Importing workbooks with macro formulas or formula references to other files is now possible; cached formula results are used
    - Warnings and error message now include coordinates of the cell which caused the problem
    - Modifications via Excel Import now cause notification mails to be sent to subscribers
- iteraQL Power Queries: The query knowledge base is automatically kept in sync with the database (Enterprise Edition only)
- Management of Business Mappings in matrix form was added as prototype implementation
- Saved queries with result format Excel can now also be loaded for bulk updates
- Full-text search: Error messages and documentation have been improved with respect to special characters
- Improved layout of graphical report type overview page
- Export format select box in Spreadsheet reports was widened, to improve readability in Internet Explorer
- Dashboard: The list of available attributes is now sorted alphabetically

Fixed Bugs

- Configuration option "Elements with Status 'Inactive' are shown" does filter inactive technical components
- Syntactical errors in XMI/Ecore export after repeated exports. Fixed
- XMI Export: Some attributes values were serialized incorrectly: Fixed
- Several bugs in Excel Import were fixed
- References to User Groups could not be created in permission management functions: Fixed
- Attribute groups containing at least one attribute could not be deleted: Fixed
- Attributes could not be copied in some constellations: Fixed
- Managing Attribute Groups led to internal error in some cases: Fixed
- Responsibility attributes: it was possible to assign an already assigned user again, leading to an error message: Fixed
- Number attributes could not be edited when their ranges were not uniformly distributed (user-defined ranges): Fixed
- Changing a number attribute value for a copied building block changed it for the original building block as well: Fixed
- Deleting more than one value at a time in enumeration and responsibility values caused an internal error: Fixed
- Deleting business mappings from a Business Process, a Product, or a Business Unit left zombie entries in the database: Fixed
- Spreadsheet reports: After choosing an post-processing option, some export formats were no longer available: Fixed
- Bulk updates: Insufficient permission checking on building block types: Fixed
- Hot-keys for Copy and Save did not work as expected: Fixed
- Landscape diagram: Business Process names were not rendered for SVG/PDF output: Fixed
- After subscribing to a building block, the list of subscribers was not updated visually: Fixed

iteraplan 2.9.M3

New Features and Changes:

- New Diagram type: Nesting Cluster Diagram
- Performance improvements (Read, Edit, Excel-Import)
- PowerQueries using our new Query-Console, enabling multiple hops on the meta-model as well as traversing hierarchies.
- Landscape Diagram - Readability improvements & new Options
    - new Option to not scale content to fit into page
    - optional content spanning of the Landscape Diagram
    - Changed default behaviour and description of Landscape Diagram option - Strict BusinessMapping
    - Non-hierarchical names on landscape diagram (PDF) axis labels
    - Long Colour legends moved to second page
- Improved Excel Import
    - Performance improvements
    - Handling exceptions during the import of a row without error page, but log warnings and skip of the erroneous row
    - Partial Import possible (Massupdate via Excel)
- Quality Seal on InformationSystems
- User Interface: New Action Icons on Building Block Pages
- Unified Copy: InformationSystems and TechnicalComponents copy now in one step
- Added possibility to clear the Hibernate second-level cache
- Interface Name is no longer marked as mandatory
- Pie-chart in Dashboard shows also empty values

Fixed Bugs:

- Switching to second page showed unfiltered results on overview page, even though a filter was specified
- Changesets for subscription emails not correct
- Wrong Results with BusinessMapping as QueryExtension in TabularReporting
- Error after deleting an element of an enumeration attribute
- Landscape Diagram: remove empty columns/rows not working properly
- Excel-Import: Date attributes can not be imported
- Error after deleting two elements of an enumeration attribute
- IS release (copy) function duplicates all enumeration attribute values
- Saved query "Information systems affected by projects but without accountability" loaded with errors
- Cluster Diagram with Attributes criterion could not be loaded
- Cannot open the Attribute Groups page under some circumstances
- Error based on Corrupted Database (Hierarchy&Positions) - Enhance robustness
- Notification emails: Number-attributes mistakenly recognized as changed

iteraplan 2.9.M2

New Features and Changes:

- New diagram types: Bar charts, pie charts, composite diagrams bar and pie charts
- Enhanced filtering for Interfaces, based on interface direction
- Enhanced filtering: Products and Business Units can be filtered with respect to linked Information Systems
- Reporting enhancements: Landscape Diagram with new options
- Excel exports offer more than one generation template
- Successor/Predecessor reports for Technical Components, in addition to those for Information Systems
- Additional self-defined columns in masterplan diagrams available
- Improved Browser compatibility of Dashboard page
- Improved XMI import
- Improved Excel import
- Compatibility between Scenario functionality and database caching
- Performance improvements
- Many Improvements, bug fixes, updated libraries, ... thanks to all reporters!

iteraplan 2.8.1

Bugs fixed which were part of release 2.8.0:

- ITERAPLAN-8 Info flow diagram generation (Visio) fails with some interfaces
- ITERAPLAN-11 Permission-bug when creating reports without permission to view information systems
- ITERAPLAN-12 Non-hierarchical names on landscape diagram (PDF) axis labels

- ITERAPLAN-13 Error at filtering of Infrastructure Elements with query
- ITERAPLAN-14 Print views are not CSS-styled
- ITERAPLAN-18 IndexOutOfBoundsException at iteraplan Dashboard
- ITERAPLAN-25 CSS resources are not UTF8-encoded but JAWR expects UTF8
- ITERAPLAN-29 Reset-button does not reset the GUI
- ITERAPLAN-30 Bulk delete of technical components
- ITERAPLAN-32 The copying and creating of new ISR releases happens in more than one transaction / fails partly
- ITERAPLAN-39 NullPointerException while rendering business mapping view extension of assigned Technical Components
- ITERAPLAN-41 Error 500 in context graphics Masterplan with Technical Components
- ITERAPLAN-67 Certain names of saved queries prevent the "directly execute saved query"-menu of the diagram reports start page from showing
- ITERAPLAN-69 Spreadsheet reports: Output formats not working correctly
- ITERAPLAN-73 Hibernate query exceeds Oracle SQL limitations for IN clauses
- ITERAPLAN-74 Hibernate/EhCache should respect active datasource and not mix up database contents
- ITERAPLAN-76 Bulk Delete of hierarchical elements fails: LazyInitException
- ITERAPLAN-78 Excel import of attributes for interfaces does not work properly
- ITERAPLAN-81 Enum Attribute detail description texts are not properly escaped for JavaScript
- ITERAPLAN-82 Enum attribute value descriptions cannot be edited
- ITERAPLAN-92 Error page when loading saved landscape diagrams under special circumstances
- ITERAPLAN-94 Deleting users in iteraplan fails
- ITERAPLAN-98 Improve saving of the BusinessMappings --> enhanced validation to prevent DB corruption
- ITERAPLAN-105 Criteria Filtering on overview pages applies too strict date ranges and omits legitimate search results
- ITERAPLAN-119 Respect user's permissions when rendering Dashboard views
- ITERAPLAN-122 Permission denied: cannot execute a saved cluster diagram query defined by me
- ITERAPLAN-123 Error page when switching back and forth between configuration steps of graphical reports, in case the query returns a high number of elements
- ITERAPLAN-131 Opening information flow diagram results in error
- ITERAPLAN-143 Spreadsheet report ignore attribute group permissions
- ITERAPLAN-161 Correct default behaviour and wrong/unclear description of LandscapeDiagram option - Strict BusinessMapping
- ITERAPLAN-171 New release / Copy Information System fails with error page, but semi-initialized copy is created

Milestone releases

Please note, that releases marked as a milestone are not fully-blown iteraplan releases. We might build milestones from time to time to demonstrate certain improvements and new features on the way to an regular iteraplan release. Not all features are completed in a milestone release, so we welcome your feedback especially for milestones.

Milestones aren't intended for production use. Migrating from or to a milestone release isn't covered by any iteraplan support contract.

Release Notes for previous versions of iteraplan

- Release Notes for iteraplan 2.8
- Release Notes for iteraplan 2.7
- Release Notes for iteraplan 2.6
- Release Notes for iteraplan 2.5
- Release Notes for iteraplan 2.3
- Release Notes for iteraplan 2.2
- Release Notes for iteraplan 2.1
- Release Notes for iteraplan 2.0

Added a default color to enum values, that were added by the Excel/XMI import.

FAQs

This page provides useful pointers and answers to frequently asked questions.

How to maintain users and roles in iteraplan and iTURM?

iTURM is an authentication tool which can be used to map users and roles. Please refer to the illustration on this page: Background

As a minimum requirement you need:

- A **equal role** in both applications (iteraplan and iTURM).
- A **user** in **iTURM**, who is associated with such a role.

After that, the user can log in to iteraplan with his password. The iteraplan user instance will be created at login time automatically.

Optionally, the iteraplan user instance can be created manually to previously define custom properties. In this case, the user's login name must match the respective iTURM user.

⚠ Currently there is no replication/synchronization mechanism for roles between iteraplan and iTURM, or for additional user information such as first name, last name or email-address.

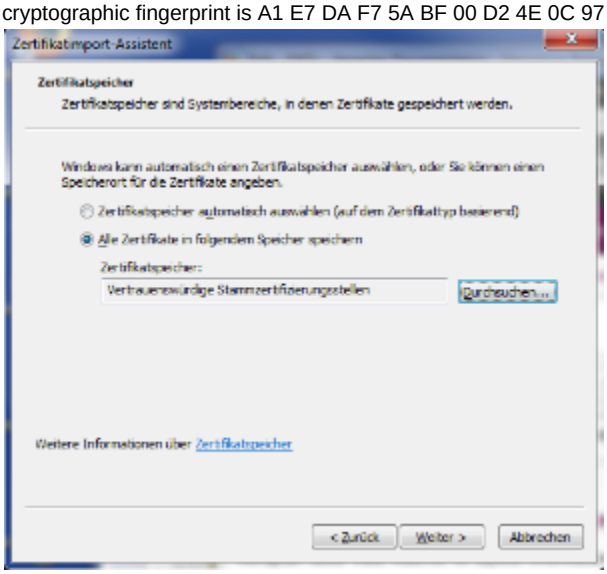
Why are no connectors shown in an information flow diagram (Visio format)?

iteraplan uses Visio macros to layout information flow diagrams in Microsoft Visio. By default, the macro security level is too high for that macro to be executed. Depending on your version of Visio, you see a macro warning or macros are silently disabled. Microsoft *Visio Viewer* is not capable of executing macros at all, i.e. it cannot be used to display information flow diagrams.

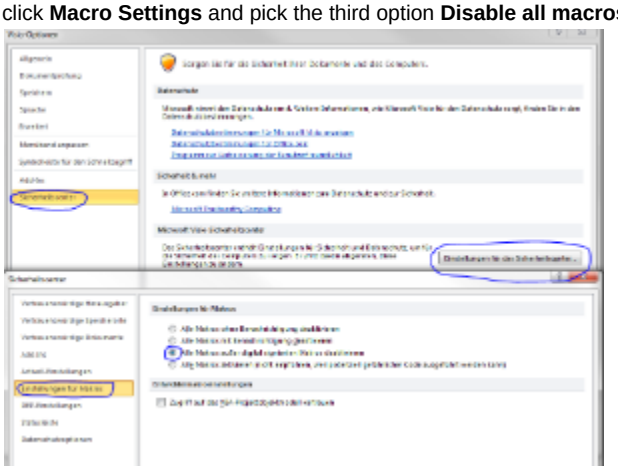
To display the diagram in Microsoft Visio correctly, you need to adjust your macro security settings once, and make sure that Visio trusts iteraplan's digital signature on the Visio macros. Here is a step-by-step procedure to do so (following Visio 2010's menu structures):

1. Download the iteraplan certificate from the graphical report overview page (button *Download Visio certificate* at the bottom of the page). You may have to right-click the button and select *Save Link Target As*.
2. Find the downloaded file in Windows Explorer and right-click it. Choose the to import the operation *Install certificate*. On the second page, the import wizard allows to select the certificate store to import into. Click *Browse* and select *Trusted Root Certification Authorities*. Follow the wizard to complete the import procedure. Then repeat the same process, but this time select the certificate store Trusted Publishers.

Visio needs the certificate in both stores. The certificate is issued by *iteratec GmbH* to *iteratec GmbH* and is valid till 2099-01-01, its cryptographic fingerprint is A1 E7 DA F7 5A BF 00 D2 4E 0C 97 F4 F5 D2 40 C4 A8 05 7C 9A.



3. Open Visio and open the Options dialog (File > Options). Click *Trust Center* in the left-hand pane and open *Trust Center Settings.* Then click **Macro Settings** and pick the third option **Disable all macros except digitally signed macros**.



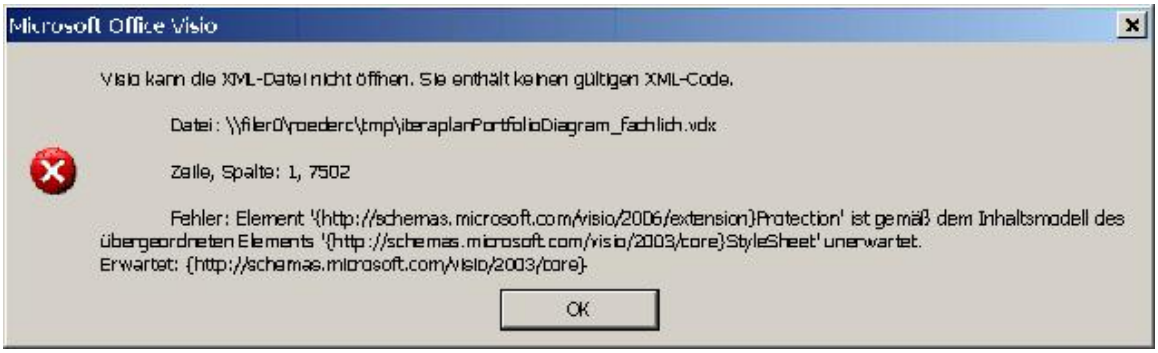
4. When you open the next Visio report from iteraplan, all information flow diagram should be displayed correctly and without further warnings.

Please note that this procedure may vary, depending on the Visio version and Windows version you are using.

Also, verify that you are running Microsoft Visio with the **latest service pack** (currently Service Pack 3 for Visio 2003, Service Pack 2 for Visio 2007, or Service Pack 1 for Visio 2010).

Why can't I open any of the Visio diagrams generated by iteraplan?

Most probably your Visio version is too old. You must have at least Visio 2003 with Service Pack 3 running. Generally speaking, make sure that you have the latest service pack for your version of MS Visio installed. If your Visio Version is too old or you do not have the latest Service Pack installed, you see an error message similar the shown below.



An error message appears if your Visio version is too old to open generated Visio diagrams.

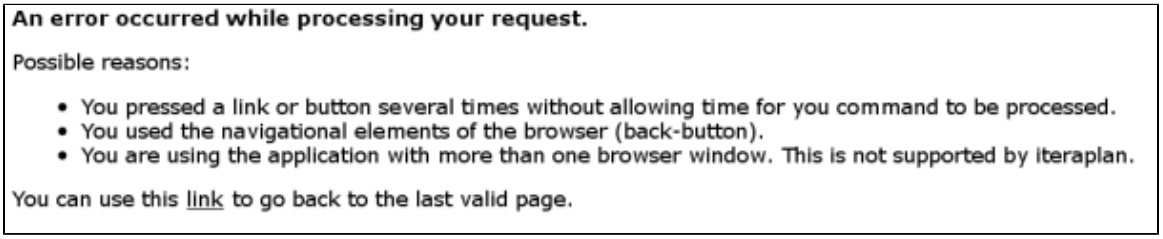
Make sure that you are running Microsoft Visio with the **latest service pack** (currently Service Pack 3 for MS Visio 2003 , or Service Pack 2 for MS Visio 2007).

Browser problems

Being a web application, iteraplan is subject to certain constraints. Therefore, be sure to observe the following points:

- Do not abort queries in progress by clicking **Cancel** in your browser;
- Do not click links or buttons several times in succession without waiting for the response;
- Some links are only valid within your current session. If you logged out in the meantime, you may need to repeat some steps that were not saved during the last session.

Failure to observe these points can lead to this error message.



General error message

In addition, using Internet Explorer Version 6 or later with a very high resolution (>96 dpi) can lead to poor-quality screen presentation. This is because Internet Explorer scales the diagrams when your screen has high resolution.
To resolve this problem, make the following entry in the registry to disable scaling for Internet Explorer:

[iteraplanDocumentation303:HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main]

"UseHR"=dword:00000000

Concurrent use

When iteraplan is used concurrently by multiple users, problems can arise in the event users make incompatible changes. The following error messages are issued:

- This element cannot be saved successfully anymore. This may be due to the following reasons:# The element has been modified by another user.

1. Attributes have been modified by another user or in the dialog Attributes.
2. Explicit permissions have been modified by another user or in the dialog Object-related Permissions.
3. Associated elements have been deleted by another user or in the corresponding dialog.
   To continue, please click cancel or clear the session

- A chosen entity was not found in the database. It is possible that it was deleted by another user. To continue, please click cancel or clear the session. This transaction cannot be successfully completed anymore.
- The chosen entity was not found. Another user may have deleted it in the meantime or it may be not visible due to a filter. Please select another building block.

In the event you see one of these messages, please follow the instructions as indicated. Failure to do so can cause data inconsistency. If in

doubt, you can return the application to a consistent state using **Clear Session**.

How to run iteraplan on an Ubuntu Server (Version 9.10 and higher)

Standard installation of Tomcat 6 on ubuntu activates Java Security per default. In this case iteraplan can write its log files only within Tomcat's directories without running into access problems. If you require iteraplan's log files outside of Tomcat's directories, please de-activate Java Security.

Why are the links of a report in Microsoft Office formats not working?

Due to untypical behaviour of MS Office applications, it is offhand not possible to link directly to a building block, although the hyperlink is correct. Instead of forwarding the link directly to a browser, Microsoft Office applications (Excel, Visio) seem to communicate with the iteraplan server first. The server redirects the call to the login-page (due to the fact that the Office application has no valid session) which is then forwarded by the Office application to the browser. After login only the standard overview is presented. Due to the same communication issue the hyperlinks work in iteraplan CE (where no login is required), but only after the second click. The only solution seems to be a change to a Registry Key which forces Office application to omit this internal communication.

Attention: this change concerns the behaviour of **all** Microsoft Office products running on your system. The required procedure is (Source ---> http://support.microsoft.com/kb/218153/) :

1. Quit any programs that are running.
2. Click **Start**, and then click **Run**. Type **regedit** in the **Open** box, and then click **OK**.
3. In Registry Editor, browse to the following subkey:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\9.0\Common\Internet
Note: This registry key is the same for all versions that listed in the "Applies To" section. This registry key will not be different for the different versions.

1. Make sure the **Internet** subkey is selected. On the **Edit** menu, point to **New**, and then click **DWORD Value**. Add the following registry value:

Value Name: ForceShellExecute

1. Double-click **ForceShellExecute**, and then set the **Value data** to **1**. Click **OK**.
2. On the **Registry** menu, click **Exit**.

Alternatively, download and run this .reg file to apply the above changes to your registry.

Moving or scaling a diagram in Visio format

When editing a diagram export in Microsoft Visio format, the user might want to move or scale the entire content of a graphic. In the case of the Landscape and Masterplan diagrams, this is not directly possible, since the intuitive selection does not include the background grid, used to highlight the diagram cells. To select the entire content in those two diagrams, either use the *Ctrl + A* keyboard shortcut, or make sure to include the bottom left corner of the page in your selection scope.

Why is my generated PDF diagram empty?

When using Microsoft Windows and the PDF Reader provided from Adobe, it may sometimes happen that a generated PDF diagram seems empty. This is because the generated diagram is larger than the page size supported by Acrobat Reader (approximately 5 meters). Using a different PDF reader, for example the Foxit Reader, should allow the user to see the contents of the diagram. For Mac OS and Ubuntu Linux the built-in PDF readers are known to have no problems with large diagrams.

Feedback

Feedback iteraplan 3.3

For all registered Confluence users: You can add your feedback on this page, simply add a comment. Alternatively, you can send us an email to contact@iteraplan.de. Or open a new discussion topic in our EAM & iteraplan community: Forum - iteraplan (login required)

Think you found a bug in iteraplan 3.3? Use our issue tracker.

Known Bugs

iteraplan 3.3

Excel-Export or iteraQL queries return only out-of-date results or general error pages under certain circumstances (fixed in 3.4)

One possible reason for this are user-defined attributes named like internal technical attributes, for example "ID" or "name". Other problematic names are those of the building block types in any localization, or when two attribute values of the same enumeration attribute have, case-insensitively, the same names, for example "VALUE" and "value".

Attempting to load the data model for the export and iteraQL when such names are present, currently can cause the whole loading process to get stuck and never be attempted any more. This means the data status for export and iteraQL will be frozen. If this error happens on the first load, no data will be there at all, thus causing the mentioned error pages.

A workaround is to rename problematic attributes or attribute values and then restart the tomcat-server iteraplan is running on.

iteraplan cannot be deployed with Tomcat ROOT context (fixed in 3.4)

Due to a bug in a third party library (JAWR), the iteraplan web application cannot be deployed with ROOT context to the Tomcat servlet engine.

Workaround: Install with a different deployment context like "iteraplan". This bug is fixed in the third party project and hence in iteraplan.

iteraplan 3.2

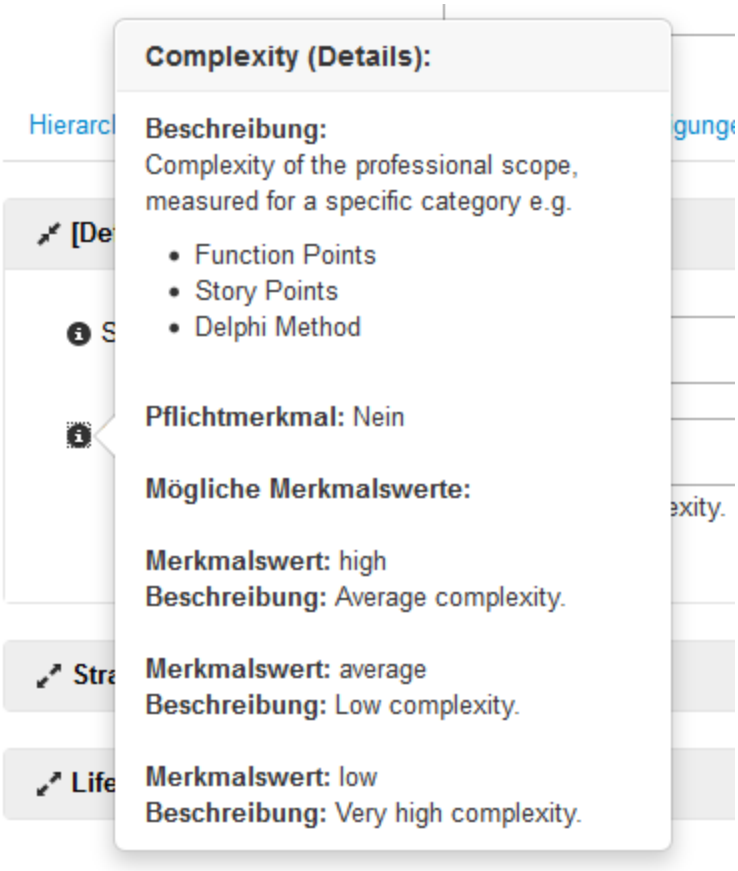
Changing enum attribute values doesn't work properly for IE9, IE10 and IE11 (fixed in 3.3)

Changing attribute value assignments inside the "Attributes" section on an element's detail page will most likely not work properly, if the browser is Internet Explorer Version 9 or higher. Internet Explorer 8 and other browsers don't have this problem.

The workaround is to configure "Internet Explorer 8" as Document Mode in Internet Explorer 9/10/11, or to use another browser.

Displaced enumeration attribute value descriptions in information popups (fixed in 3.3)

Only in edit mode: When a user clicks on the information icon near an enumeration attribute on a building block detail page, the description strings for the attribute values are displaced.

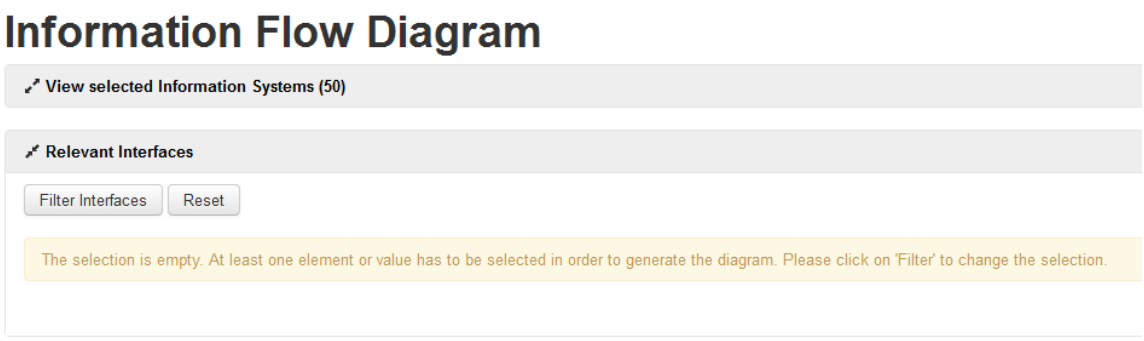


It might be an option to remove the attribute value descriptions, until this issue is fixed.

No lines visible in Information Flow Diagram (fixed in 3.3)

There is a bug where the default filtering of interfaces in Information Flow Diagrams results in no interfaces being selected. Affected are newly created Information Flow Diagrams, as well as some saved ones if they don't have an explicit interface filter. Resetting the interfaces filter by clicking "Reset" has the same effect.

The second page of the configuration of affected diagrams looks as follows:



A workaround is to click "Filter Interfaces" and execute a query without conditions to get all interfaces, then to select them all and use these interfaces in the diagram. If a diagram formerly without filter is re-saved after this workaround was applied, newly created interfaces will automatically be selected in future executions of the diagram.

Filter with no selected elements cannot be saved and loaded as part of a Nesting Cluster Diagram configuration

When a user defines a filter for the outer or inner elements of a Nesting Cluster Diagram, and deselects all elements, the resulting diagram can be

used to display special cases of that diagram: Without outer elements, only inner elements without connections are displayed; without inner elements, only the tree structure of the outer elements is displayed. These diagrams can be configured and generated.

However, if such a configuration is saved and loaded, or executed directly, the empty selection becomes a full selection, and the diagrams are generated differently.

As a temporary solution, the user can load the query, edit the filter, de-select all elements with a single click, and generate the diagram.

Known issue with square brackets in user-defined names on MS SQL Server (fixed in 3.3)

When iteraplan uses MS SQL Server, users cannot use names with square brackets. This is due to a missing escaping in a library (Hibernate) of non-standard extension in MS SQL. Elements with such invalid names are not found, and this results in various problems.

As a temporary solution, users can follow a naming rule.

See Microsoft SQL Server (additional package) for details.

iteraplan 3.1

Diagrams not displayed on Dashboard (note: not Custom Dashboard)

That particular dashboard relies on javascript which is not compatible with more recent Internet Explorer versions, while Firefox is happy with it. (ITERAPLAN-1938)

Workaround: Use a different browser or use IE8 compatibility in your Internet Explorer (browser mode and document mode).

Changes on Successors of Information System Releases and Technical Component Releases are neither send as email notifications nor displayed in the history (fixed in 3.2)

Due to a bug in a library iteraplan uses for the detection of changes applied to a Building Block it is currently impossible to detect changes to the Successors of Information System Releases or Technical Component Releases.

Information System Masterplan Diagram does not show related Business Functions (fixed in 3.2)

Due to a bug in the Masterplan Diagram generation logic related Business Functions are not evaluated in an Information System Masterplan Diagram. This results in a Masterplan Diagram in which no related Business Functions are shown.

iteraplan 3.0

This section lists Known Issues for Release 3.0 and possible workarounds

iteraplan menus don't open up in Internet Explorer 10 (fixed in 3.1)

This is a known issue with a recent windows-update for IE10. We have created a patched version of our layout. In order to apply the patch to your iteraplan-3.0.4 installation please execute the following steps:

1. shutdown tomcat
2. replace the file jsp/layouts/standard.jsp of your iteraplan installation with the attached one (if you want to keep a backup of the existing file, please store it outside of the tomcat installation)
3. remove the tomcat work directory of iteraplan (usually located at <tomcat_base>/work/Catalina/localhost/iteraplan)
4. start tomcat

Add "File address" in description field doesn't work in modern browsers

Adding a "File address" into a Building Block's description field in the "Add link or file" section doesn't work as expected. Due to the security behavior of modern browsers, the full file path can't be inserted by a file input field anymore.

Version 3.0.4

Excel Import: When importing Interfaces without assigned Business Objects, data is mis-assigned

(fixed in 3.1)

1. The interface direction may be interchanged with another interface
2. An interface might be related to a different set of endpoints
3. The changes identified in the import process are not plausible

The only available workaround is to assign a "dummy" business object to each Interface, and to make sure that each Interface has a name. A proper solution is being worked on.

Excel Import: Excel Importer fails to move a child element to the parent level (fixed in 3.1)

The available workaround is to change hierarchies through the webinterface. A proper solution is being worked on.

Information Flow: Configuring Information Flow (fixed in 3.1)

- Configuring and saving an information flow starting with the visualization context of an information system does not preserver the corresponding information systems selection. When executing the saved visualization, all information systems are used to generate the output.
- Saving the configuration of an Information Flow Diagram does not save the selected Visio template. The template has to be selected again when loading the saved configuration. Therefore you need the permission to execute saved visualizations and permission to create visualization configuration.

Diagram Reports: Information Flow Diagram content overlapping (fixed in 3.1)

In the Information Flow Diagram, the title and content sometimes overlap in Visio Export. This happens especially when only a few information systems are displayed. No workaround available.

Composite Diagrams: Write permissons required (fixed in 3.1)

The Composite Diagrams require write permission to execute. If you have only read permissions for all building block types, you cannot execute composite diagram reports currently. You need at least one update permission for an arbitrary building block type.

Diagram Configuration & Spreadsheet Reports: Query Extension Fails (fixed in 3.2)

In diagram configurations or spreadsheet reports, the query extension "Properties of assigned Interfaces" fails if no criterion for the interfaces is configured. A solution here is to remove the restriction criterion.

UI-Webinterface: Wiki Syntax in Spreadsheet Reports (fixed in 3.1)

Currently in the spreadsheet reports, the wiki syntax is not parsed properly for text attributes. Workaround: Accept seeing Wiki syntax in such columns.

UI-Webinterface: Wrong displayed Umlauts in Saved Queries (fixed in 3.1)

On the building block overview page, Umlauts or other non-latin characters in saved queries are not correctly displayed. An escaped character will be shown instead.

Remove of Enumeration or Responsibility attributes leads to an error page (fixed in 3.1)

If you navigate to the attribute page and remove one or more values from the Enumeration or Responsibility attributes and try to save the changes, an error occurs.

Excel Import: Blank as first Character (fixed in 3.1)

You cannot import a excel sheet when there is a blank as first Character in the Release Number of an information system, otherwise the importer fails to import the data. Workaround: Remove the extra blank.

Excel Import: Number when String is expected (fixed in 3.1)

The Excel Importer won't accept a number when it's expecting a string, and the import fails. Workaround: Add a non-numeric character in the Excel cell, e.g. an underscore.

Excel Import: Enumeration Literals (fixed in 3.1)

If you add enumeration literals in the Excel template and import it, this will lead to an error if an attribute group has a restricted visibility.

Excel Import: Name formatting (fixed in 3.1)

When an Excel file contains, for example, an InformationSystemRelease named "IS#1.0" instead of the standard formatting "IS # 1.0" and that ISR is referenced in a relation sheet with the same formatted name "IS#1.0", the importer does not find the InformationSystemRelease the relation points to, and reports an error when the import is attempted. Workaround: apply consistent naming your complete Excel file and surround the hash sign with spaces.

Version 3.0.3

Import (Excel and XMI): Issues when importing interfaces and information flows to update already existing data

When there are interfaces which do not transport any business objects, trying to update those interfaces and related information flows by means of the importer can lead to problems. In this case there is the possibility of changes being displayed which are no actual changes, or even unintended data changes happening (like interfaces suddenly connecting information systems previously connected by another interface).
A possible workaround is to create a "dummy" business object and assign it to all interfaces which normally wouldn't have a business object assigned, before attempting any updates with the importer.

> ⚠ **Remove sheets to avoid problems**
> If you want to update data not related to interfaces or information flows, it's strongly recommended to remove the **Interface**, **InformationFlow** and **TC-IF** sheets from your excel file before importing.

Version 3.0.2

Excel-Import: duplicated names are not always detected (fixed in 3.0.3)

- Names of Technical Components and Information Systems are denoted in the format "name # version" in the Excel file. If you have, for example, two Technical Components named "tech # 1.0" and "tech#1.0" both will be created as a technical component with the name "tech" and version "1.0" in iteraplan. Since they are written differently in the excel file, they are not recognized as duplicates, thus two identical elements will be created in iteraplan, which will cause problems when analyzing or further editing data. A workaround to avoid this is to always use the default format of "name # version" with one space between "#" and name, respectively version, and using only "name" for elements without version. The alternative form "name #" is discouraged.
- While iteraplan treats names as case-insensitive in respect to duplicity (in most database setups), the importer does not. This means if your Excel file contains the Business Objects "account" and "Account", the importer will attempt to create both objects. This most likely will result in an error page.

Import only possible as supervisor-user (fixed in 3.0.3)

Due to a flaw in permission handling, it is only possible to perform imports (Excel and XMI) successfully, when being logged in as a user with supervisor role. Attempting to import with other users results in an error page.

Unexpected results when using a partial Excel-Import, meaning import of an excel file with partially removed sheets and/or columns (fixed in 3.0.3)

If a large portion of the excel files sheets and/or columns are removed before attempting an import, the importer might have trouble to interpret the remaining data correctly. This can result in error messages about unapplicable changes which are not intended by the user anyway.

Version 3.0.0

Special characters in names of saved queries for diagrams can break the menu (fixed in 3.0.1)

Especially the single quote character ' can cause unexpected behaviour of the visualization sub-menus.

List of watched elements in the context menu is not always correct (fixed in 3.0.1)

- After clicking "watch", the newly subscribed element doesn't show up in the list. The page needs to be refreshed for it to show.
- Watched building block types aren't shown in the list, only individual building blocks.
- For very long lists of watched elements scrolling might not be possible. This depends on the browser.

iteraplan 2.9

This section lists Known Issues for Release 2.9.x and possible workarounds

Version 2.9.1

Broken print-view with non-default UI theme

Using the iteraplan default theme works.

Version 2.9.0

Excel Import: untrimmed values in Excel cells (fixed in 2.9.1)

If your Excel file contains cells with several assigned values at ones, like for multi value assignment attributes or relations, please make sure the values are trimmed. This can lead to data inconsistency problems otherwise.
Examples:
"alice ; bob; max " or "joe ;alice" are bad
"alice;bob;max" is good

NullPointerException when not using a names legend in Landscape Diagrams (fixed in 2.9.1)

Under certain circumstances, when there are many available colors defined, not using a names legend when creating a landscape diagram can result in an error page. As a workaround, enable generation of a names legend

iteraplan 2.8

This section lists Known Issues for Release 2.8.x and possible workarounds.

2nd-Level Database Cache is incompatible with Scenario functionality/ multiple data sources

Starting with release 2.8.0, iteraplan comes with an integrated 2nd-Level Database Cache which improves the system's performance in many situations. However, iteraplan's Scenario Functionality is not yet compatible with that cache implementation. As soon as one (or more) user(s) switches to a secondary data source, many operations will fail randomly and result in a general error message being shown the user. This is because objects from different data sources will get mixed up in the cache and cache sanity checks will fail. The system fails safely, i.e. your data is not at risk to be corrupted and modified inadvertently.

The current solution is to disable the 2nd-Level Database Cache when additional scenario databases/ data sources are configured. To disable the cache, edit the file `$TOMCAT_HOME/webapps/iteraplan/WEB-INF/classes/iteraplan.properties` and make sure that following two options are set to `false`:

```
hibernate.cache.usesecondlevelcache=false
hibernate.cache.usequerycache=false
```

These settings apply after the application is restarted.

SVG Information Flow Diagram takes too long to generate

Since the generation of an SVG Information Flow Diagram employs non-deterministic techniques, the generation process can, sometimes, continue for a long time. In many cases, iteraplan will detect this and avoid long generation times by applying simpler routing algorithms, which produce a less appealing result.

iteraplan 2.7

This section lists Known Issues for Release 2.7.x and possible workarounds.

- None so far -

iteraplan 2.6

This section lists Known Issues for Release 2.6.x and possible workarounds.

Filter for InformationSystem B

When using the Filter for InformationSystem B, after clearing the filter box not the full list of elements is restored in the selection box. (Known, Fixed in next Release).

Sorting enumeration attribute values with Internet Explorer 7

When sorting values of enum attributes the already defined assignments between buildingblocks and enum attribute values might get mixed up! It's highly recommended to use a modern browser or a more recent version of Internet Explorer. Alternatively apply the attached fix for iteraplan 2.6.2 (by extracting it to $TOMCAT_HOME/webapps/iteraplan/) to assure sorting with IE7 works correctly.