Project Deliverable Report

Deliverable Nr 2.2 – Semantic meta-model, integration and transformations v1

Work Package
WP2 Combination of Creativity Techniques

Task
T2.2 Semantic Metamodel Design, T2.3 Transformation Methods Definition, T2.4 Integration Methods Definition

Date of delivery
Contractual: 31-12-2008      Actual: 13-02-2009

Code name
D2.2           Version: 1.0      Draft □ Final ☑

Type of deliverable
Report

Security (distribution level)
PU

Contributors
Authors (Partner)
Peter Dolog, Yujian Lin (Aalborg University), Roger Dols (Morpheus), Klaus Schmid, Per Pascal Grube (University of Hildesheim), Rani Pinchuk (Space Applications Services)

Contact Person
Peter Dolog (Aalborg University)

WP/Task responsible
WP2

EC Project Officer
Mr. Christian Wilk

Abstract (for dissemination)
This report introduces a topic maps based metamodel for creativity techniques, creativity process, and idea maps as results from creativity process. It proposes a graph based and hierarchical graph based transformation of idea maps for combination and integration of results of different creativity sessions. It further suggests a service composition model as an integration model based on service oriented architecture which integrates various creativity process supporting tools as services.

Keywords List
Creativity Techniques, Creative Process, Creativity Tools, Ideas, Innovation, Graph-Based Transformation, Metamodeling, Idea Maps, Service Composition Model, Combination of Creativity Techniques
# TABLE OF CONTENTS

1. **Introduction** .................................................................................................................. 8

2. **An Example of Creativity Support Tool** ........................................................................ 11

3. **Metamodel for Idea Map** ............................................................................................... 13
   3.1 **Mind Mapping** ........................................................................................................ 13
   3.2 **Statement Based Creativity Techniques** ................................................................. 14
   3.3 **The Metamodel for the Creativity Techniques** ......................................................... 15
      3.3.1 The Process ........................................................................................................ 15
      3.3.2 The ideatrigger-part ......................................................................................... 17
      3.3.3 Problem-Idea-Solution .................................................................................. 18
      3.3.4 The Relation between the Problem and the Creative Process ......................... 19
      3.3.5 Participants in the Creative Process ................................................................ 19
      3.4 **The Relation between Knowledge, Idea and Solution** ....................................... 20

4. **Graph Based Transformation** ....................................................................................... 21
   4.1 **An example of Ideation** .......................................................................................... 22
   4.2 **Idea Map and Tree Structure** ................................................................................ 24
   4.3 **Graph Transformation** .......................................................................................... 27
      4.3.1 **Graph and Graph Transformation** ............................................................... 27
      4.3.2 **Rule Based Transformation Over Idea Map** ................................................ 28
   4.4 **Hierarchical Graph Transformation over Idea Map** ............................................ 30
      4.4.1 **Rule Schema Add** ....................................................................................... 31
      4.4.2 **Rule Schema Remove** ................................................................................ 32
   4.5 **Examples of Transformation** .................................................................................. 33

5. **Related Work on Transformations** .............................................................................. 39

6. **Integration of Creativity Techniques** ............................................................................ 40
   6.1 **Integration of creativity techniques** ....................................................................... 40
   6.2 **SCA-based Integration** .......................................................................................... 41
      6.2.1 **SCA perspective** .......................................................................................... 41
      6.2.2 **Service-based creativity techniques integration** .......................................... 42

7. **IdeaTrigger: Combination of Creativity Techniques** ................................................... 44
   7.1 **Idea Generation with Two Different Modes** .......................................................... 44
   7.2 **Idea Organization** .................................................................................................. 45
1 INTRODUCTION

Creativity support tools are those extending the human capabilities to make discoveries or inventions from early stages of gathering information, hypothesis generation, and initial production, through later stages of refinement, validation, and dissemination (Shneiderman, 2007). Similar to this, idSpace project deliverable D2.1 reviews creativity techniques. These are used to support or extend human creativity and innovation in different stages. Some creativity techniques guide the creativity session by means of explicitly suggesting different steps. Some creativity techniques simply provide a set of questions to trigger ideas.

One of the goals of the idSpace project is to explore and understand whether combination of creativity techniques situated in pedagogical scenarios with context awareness could give some advantages.

By combination of creativity techniques we mean a dependent use of creativity techniques as a selection of those identified in D2.1. We define “dependent use of creativity techniques” as a concurrent or a sequential use of the creativity techniques where the results gathered in one of them are considered and connected to the results gathered from another one. We call such dependent use of creativity techniques also “combination of creativity techniques”. This deliverable (D2.2) provides a step towards technological support for such a combination.

Of course this could be quite ambitious and beyond the limitations of the idSpace project, as this might mean that we need to combine different alternatives for problem solutions as provided for example in SCAMPER, combine different views over problems described in very different models such as chemical models or genomes and so on. Therefore, we have reduced in the first step our experimentation to the so called statement based creativity techniques and their combination on results. Please note, that combination can also mean a combination of the same creativity techniques where a previous session results are connected and combined with an ongoing creativity session results.

We define statement based creativity techniques as those which provide a set of questions as a guidance for creativity sessions. In our definition, the questions are statements which can connect the ideas generated before to newly created ideas.

This reduction of the combination problem actually allows us to define an idea conceptual model as a graph which connects ideas as nodes by edges labeled with statements. As we will see later, this provides us with the following advantages:

- It has a simple meta-model very closely related to the topic maps or other similar semantic based knowledge representation
- It allows for a well defined graph based transformations when dealing with different scenarios for combination of creativity techniques
- It allows for **integration of various other tools** such as editors and chats on top of such a conceptual representation

- It allows for **preservation of ideas** as generated historically together with a connection to the other ideas as well as views over ideas provided by the integrated toolset

Such a solution, on yet simple creativity techniques, also falls into the hypothesis given by the idSpace project proposal, where we envisioned a model based transformation based on semantic **metamodel with a central “Idea” entity**. The creativity session, whether it is idea generation or idea elaboration, may span over a long period of time; therefore, the preservation of such kind of conceptual structure as the idea map can not only help technically but also provide guidance through the preserved ideas.

As mentioned in the project proposal, idSpace platform is expected to support interoperability in the conceptual level. Therefore, for each creativity technique, there should be a conceptual model and specific model language. That way, for a specific innovative problem, different results will be produced by using different creativity techniques. If they were represented by the creativity-technique-specific conceptual model languages, the transformation among these results would be expected. In other words, different solution by different creativity techniques should understand each other. However, this expectation turns out too ambitious by far.

D2.1 identified that there is a large number of creativity techniques and therefore it is not feasible to analyze them all within the lifespan of the idSpace project. Another complexity in conceptual modeling for creativity is that the creativity techniques dealing with innovative problems have great deal of uncertainty in what will be the outcome of the process. It depends on how the people involved in the creativity sessions are creative. Furthermore, the creativity is usually higher when people involved in creative problem solving are not too restricted by a prescribed process. What the creativity techniques can do in this case is to provide some principles based on experience for searching the solution and provide some specific guidance which of course can be adapted according to what the creative teams need.

Requirements gathered in WP5 refer to the kind of transformation we are dealing with as well. In D5.1, there are two user requirements related to transformation. One is that the system shall offer different innovation rooms (UR-2.2), which means there will be a merging of different sub idea maps; the other one is that ideas need to be transformed to new products or product features, which means that there will be a replacement operation on a certain idea map with an already existed sub idea map. These operations are covered by the graph transformation in Chapter 4 in this document (D2.2). We will study the coverage of the further requirements as they emerge from the users.

In the remainder of this document we first provide an introduction example of a tool which points to the problems we are dealing with in this deliverable (Chapter 2). Then we present the metamodel we have defined for the purpose of this project (Chapter 3). We discuss the graph transformation and other related work on transformation in Chap-
Chapter 4 and Chapter 5 respectively. Integration is discussed in Chapter 6. Finally the prototype of IdeaTrigger is introduced in Chapter 7.
2 AN EXAMPLE OF CREATIVITY SUPPORT TOOL

Let us provide an example of creativity support tool which could benefit from an idea map conceptual model and transformation before going into details. Imagine a standard creativity session which is organized locally with several team members by utilizing white or black boards and other tools such as post-its. In any session, there is a problem to start from and a team more or less structurally generates, explores, elaborates, and validates ideas. Team members often discuss and draw some sketches, make their personal notes, collaboratively contribute to the drawings. After the creativity session is finished or after several sessions the team is asked to produce final result to be presented to other organization members and stakeholders. Therefore, a consolidation is necessary. Furthermore, the other team members, who have not participated in the idea generation, would need something to guide them through the generated ideas to learn and understand them effectively. Therefore, it is also vitally important to provide a user friendly view on the idea map representation.

**Figure 2-1** IdeaTrigger Toolbox with Tabs of Idea Generation, Editing, Evaluation, Transformation and Chatting Room

Figure 2-1 presents a sketch of an example close to the one discussed above. It provides an editor for idea representation and connections between them, editing tab for domains of concern, evaluation and other functions. As can be seen from Figure 2-1 intuitively, the idea map is a graph, and any new idea actually transforms the graph to a new one according to a predefined operation as we will see later in this document. It is also intuitively visible that different tabs and parts of the user interface represent different inte-
grated functionality. These functionalities can be seen as services and composed into an application by so called Service Component Architecture (SCA).

As discussed in the deliverable D2.1 and as pointed in the idSpace project proposal as well, the tool support for such a consolidation is limited even though it is clearly visible that such a tool is needed. It simply takes to redraw the sketches to its final electronic versions, extend them with additional material and descriptions and connect them to some conceptual entities which ease later exploration and retrieval. Furthermore, if the team members are geographically distributed, it gets even more difficult as the personal notes, black board sketches and extensions are not really connected to each other. Therefore, one would need at least a combination and integration of a collaborative real time editor for specific domain of concern in the creativity sessions (for example programming, biology, architecture, and so on), under lying representation of ideas and connections between them and some kind of personal notes exchange tool such as chatting room.
3 METAMODEL FOR IDEA MAP

When people are facing an innovative problem and want to find a solution which is supposed to be comprised of ideas, firstly they have to think out ideas (idea generation), they need to organize the ideas (idea organization), and they need to choose (evaluation). For example, the problem might be divided into small pieces; the problem might be considered from different perspectives; or available resources are collected to make a feasible solution. As described in the deliverable D2.1, these activities are dealt with by varieties of creativity techniques.

3.1 Mind Mapping

Almost all the creativity software is essentially implementing mind mapping creativity technique as pointed in D2.1. On the other hand, there are more than one hundred creativity techniques besides mind mapping, but none of them is electronically implemented as software.

Mind mapping is broadly used in learning, brainstorming, visual thinking, and problem solving by educators, engineers, psychologists, and others. A mind map is often created around a single word or a text, placed in the center, to which associated ideas, words and concepts are added, as shown in Figure 3-1. From Figure 3-1, we can find that (1) only ideas are kept in a mind map, no association is kept between two ideas; (2) a mind map is essentially constructed in a tree structure, i.e., usually there is no cycle in a mind map; (3) a mind map is a useful tool to visualize thinking and idea organization; and (4) other creativity techniques might be used for creating a mind map, although the mind map yet does not support the preservation of associations between ideas beyond simple connections.

---

1 The picture is adopted from http://learners.in.th/file/noomguru/MindMappingBig.jpg
3.2 Statement Based Creativity Techniques

What is an idea? There are varieties of definitions of idea. The definition by John Locke is the object of the understanding when a person “thinks”, and in the idSpace project context, when he thinks about an idea for a problem solution. Analyzing idea generation from the psychological perspective leads to the conclusion that we generate ideas if we are questioned by some questions (usually triggered from or by certain problem), or reminded by some hints, and these questions or hints are treated as associations between ideas and the problem. We can also generate ideas by free thinking from personal experience or sudden enlightenment, which is still a kind of association. Therefore in the context of a specific problem there is definitely some association between the problem and the ideas as well as between ideas themselves. This kind of association is labeled by a statement suggested by a selected creativity technique in our IdeaTrigger toolbox.

---

2 The picture is adopted from http://learners.in.th/file/noomguru/MindMappingBig.jpg

idSpace -2008-216199
When going into greater details of certain specific creativity techniques, such as SCAMMPER or 5W1H recommended in D2.1, we find that they are about principles to guide people’s thinking, helping people to bear something in mind. Directed by these principles or hints, more and more ideas are generated. The association between the ideas therefore can be described by these principles or hints, i.e., the creativity techniques can be electronically implemented by formulating the principles to statements, and this is what we call in IdeaTrigger toolbox the **statement based creativity techniques**.

### 3.3 The Metamodel for the Creativity Techniques

The picture below shows the semantic metamodel for D2.2 of idSpace. It is the basis for the ontology which will be incorporated into version 2 of the idSpace-platform.

This model will be explained by cutting the model into smaller pieces, addressing the underlying themes of the model.

![Fig. 3-2 Metamodel for the creative process](image)

#### 3.3.1 The Process

The topic Creative Project is the most generic topic, representing the entire process of solving a problem which requires innovation.

A Creative Project can have one or more Creative Activities as identified in the Boden-model, being:

- Exploration
- Combination
• Transformation
• Evaluation.

A Creative Activity can have or more several Phases identified in the Wallas-model, being:
• Problem Recognition
• Preparation
• Incubation
• Verification/Elaboration.

Each Phase uses one ore more Sub Processes identified in the IPC-model, being:
• Problem Recognition
• Search for Knowledge
• Activation of Knowledge
• Restructuring of Knowledge
• Evaluation

Each Sub Process uses one ore more Operators identified in the IPC-model, being:
• Problem recognition
• Questions
• Experiments
• Association
• Mental Simulation
• Forgetting
• Concept Formation
• Adaptation
• Transfer / Induction
• Inference / Reformulation
• Reindexing
• Evaluation
3.3.2 The ideatrigger-part

This part of the metamodel shows the core of the ideatrigger process. This represents the actual creative process. A certain Sub Process may use a specific Operator which is supported by one or more Creativity Techniques to choose from. This technique forms the basis, along with the problem statements, for a set of questions and/or statements to be used. These questions/statements will trigger the generation of ideas. Ideas may also trigger new ideas. Furthermore, through the Trigger-Association, a question/statement can trigger for one idea leading to another.
3.3.3 Problem-Idea-Solution

Ultimately, a solution will be formulated, based on the ideas brought forward in the creative sessions. This solution will then be associated with the problem it solves. In a later session this solution could also be associated with other problems it may solve.

A solution may have a number of features, based on the ideas that lead up to the solution.

Starting point of any creative process will be a problem that has to be solved. This problem will consist of a number of problem statements, which formulate different aspects of the problem. The ratio for this division is that a problem may appear to consist of several sub-problems which have to be addressed separately. This division into problem statements may be made up front or it may be the outcome of a problem recognition activity.

A problem statement can be a direct trigger for an idea.
3.3.4 The Relation between the Problem and the Creative Process

Starting point of the creative project will be a problem that has to be solved. The implementation of the phases of this creative project will be based on the problem statements, because these may need separate attention.

3.3.5 Participants in the Creative Process

Participants in the creative process will be linked to operators, in which they can fulfill one or more roles. All the participants in a specific operator form a team. So the team composition depends on the specific operator it addresses. When addressing a different operator, this may lead to a change in the team composition.
3.4 The Relation between Knowledge, Idea and Solution

Ideas and solutions can be linked to knowledge sources. It can work in two directions. Firstly, a knowledge source can be the basis for an idea to solve a problem. Secondly, a solution can be documented, in any form, and serve as a knowledge source for new ideas in future creativity sessions. Any knowledge source should be linked to at least one knowledge type, being (at an instance level):

- Problem Knowledge
- Domain Knowledge
- Solution Knowledge

![Diagram](image)

**Fig. 3-8 Part of the metamodel: the Relation between Knowledge, Idea and Solution**
4 GRAPH BASED TRANSFORMATION

Transformation can be considered in different levels, as shown in Figure 4-1. At the metamodel level, concepts from topic maps can be for example used to build idea map for idea organization. As shown in the meta-model, ideas are connected (Trigger association) with statements which provide a kind of semantics. The semantics are actually provided by specific creativity techniques.

![Diagram of different levels of transformation](image)

**Fig. 4-1 Different levels of transformation**

Basic operations for idea map at the metamodel level are: (1) add an idea; (2) delete an idea; (3) merge two idea maps; (4) subtract one sub idea map from the parent idea map and (5) replace one sub idea map with another one.

At the data structure level, the idea map is established as a directed graph. Vertices are used to stand for ideas. For visualization purpose, edges are labeled with statements, describing the associations between ideas. From the graph perspective, the basic operations are mapped to the following operations: (1) add one vertex; (2) delete one vertex; (3) merge two graphs; (4) subtract one sub-graph from the parent graph; and (5) replace one sub-graph with another one. Actually these five operations are transformation we deal with in this document.

Now we can make use of the handy techniques of algebraic graph transformation (Loewe et al., 1997). Algebraic graph transformation considers graphs as members of a set where transformation rules define operations over the set forming together algebraic structures. The transformations are essentially a kind of matching and replacing operation, matching the host graph with rules and replacing the matching sub-graph in the host graph with another graph according the rules. For an arbitrary connected graph which has N vertices, more than $2^N$ rules can be designed which makes the computation very difficult. Therefore, we reduce our experimentation in this document just to trees.

We reduce existing idea map graph to a tree by not considering certain in edges. This way, by removing some edges which make cycles, we obtain a spanning tree which can be represented in a unified child sibling structure. In this document the rules are designed based on the unified tree structure.
In following sections we begin with an example of ideation in section 4.2; in section 4.3, idea map and the unified tree structure is introduced; in section 4.4 and 4.5, graph transformation is introduced and rules are designed; and the transformation example is applied to merge two idea maps in Section 4.6.

4.1 An example of Ideation

We show how to generate idea maps during ideation. We apply IdeaTrigger (a prototype shown already on the sketch in Figure 2-1) on itself:

Firstly, statement is used to generate ideas, and the process is shown in Table 4-1.

<table>
<thead>
<tr>
<th>ID</th>
<th>statement</th>
<th>Name</th>
<th>Creativity technique</th>
<th>pID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Define the problem</td>
<td>IdeaTrigger</td>
<td>Problem Statement</td>
<td>root</td>
</tr>
<tr>
<td>2</td>
<td>What is it for</td>
<td>Dealing with innovative problem</td>
<td>Purposing</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Identify key themes</td>
<td>Idea generation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Identify key themes</td>
<td>Idea organization</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Identify key themes</td>
<td>Idea navigation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Identify key themes</td>
<td>Idea evaluation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Identify key themes</td>
<td>Transformation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>How it works</td>
<td>Generate ideas in reactive mode</td>
<td>5W1H</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>How it works</td>
<td>Generate ideas in active mode</td>
<td>5W1H</td>
<td>3</td>
</tr>
</tbody>
</table>

We have a corresponding representation of idea map referring to Table 4-1, as shown in Figure 4-2.
Now somebody comes with some ideas over idea “4: idea organization” in a separate ideation, the process is shown in Table 4-2.

<table>
<thead>
<tr>
<th>ID</th>
<th>Statement</th>
<th>Name</th>
<th>Creativity technique</th>
<th>pID</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>How does it look like</td>
<td>Topic map</td>
<td>5W1H</td>
<td>4</td>
</tr>
<tr>
<td>4.2</td>
<td>How to store them</td>
<td>Tree structure</td>
<td>5W1H</td>
<td>4</td>
</tr>
<tr>
<td>4.3</td>
<td>What can you refer to</td>
<td>Mind mapping</td>
<td>5W1H</td>
<td>4</td>
</tr>
</tbody>
</table>

The corresponding idea map referring to Table 4-2 is shown in Fig 4-3.
Imagine there are many such kinds of separate ideation for the existed ideas; and after that a merge operation is required. One possibility is to merge them manually, which is time consuming and error-prone. Another possibility is to leave this job to computer, by defining rules for these operations.

We can see from Figure 4-2 and Figure 4-3, idea map is composed of rectangles with text filled in and arrows with labels. The rectangles filled with text stands for ideas, and arrows with labels show the process of ideation and association between ideas. If we abstract idea map ignoring the contents of text and labels, then we will get a directed graph.

4.2 Idea Map and Tree Structure

The previous example shows that the idea map is visualized as a directed graph whose vertices represent ideas and edges are labeled with statements. the problem of sub-graph matching is known to be NP-complete\(^3\). For any connected graph, we can always find a spanning tree, which makes such operations as storing, adding and deleting computationally more convenient. Furthermore, mind mapping is a broadly used creativity technique which organizes ideas in a tree. Therefore, there is a high probability that the idea maps resulted from creativity sessions will also be trees. Nevertheless, if the source of the transformation is a graph, we will generate a spanning tree out of it by removing some edges to eliminate circular connections. To keep however the information about what was removed, the information on target ideas of the removed associations will be stored as references in the source idea nodes that they can be later reconstructed and perhaps modified by humans.

Here we clarify between the visualization of the idea map and the data structure of the idea map. The idea map of Figure 4-2 and Figure 4-3 can be abstracted as a directed graph, as shown in Figure 4-4 and Figure 4-5. For simplicity, the labels of edges are not drawn.

---

Dealing with innovative problem

(3) Idea generation

(4) Idea organization

(5) Idea navigation

(6) Idea evaluation

(7) Transformation

(8) Generate ideas in active mode

(9) Generate ideas in reactive mode

Fig. 4-4 Directed graph representation of idea map Figure 4-2

With the spanning tree, a unified tree structure, i.e., child-sibling structure is used for the transformation introduced in the following sub-sections. Figure 4-6 and Figure 4-7 show the unified tree structure of Figure 4-4 and Figure 4-5.

idSpace -2008-216199 25
In Figure 4-6 and Figure 4-7, \( \Box \) is defined as empty node as well as end node: the child node of the left most leaf node is the beginning node; and the sibling node of the root node is the rear node. For tree has a recursive structure, the end nodes of sub-tree are also defined recursively.

This preparation enables us to proceed with rule-based graph transformation for idea map. The transformation of graphs by rules has been applied in several areas of computer science. In order to cope with complex applications, it is necessary to structure not only the graphs that are subject of transformations but also the sets of rules.
4.3 Graph Transformation

4.3.1 Graph and Graph Transformation

Let us first introduce some terms which we will use later.

Graph

A graph is a system consisting of two finite sets, set of vertices and set of edges; and the edges can be labeled.

Injective mapping

A mapping from set $S_1$ to set $S_2$ is injective, if every element in set $S_1$ has an image in set $S_2$, but there can be element in image set $S_2$ which has no source in $S_1$, as shown in figure 4-8.

![Fig. 4-8 Illustration of injective mapping](image)

Morphism

A morphism is a kind of mapping between two graphs, including vertex mapping and edge mapping. The mapping, establishes association between the vertex sets and edge sets respectively. A morphism is injective if both vertex mapping and edge mapping are injective. Figure 4-9 shows an illustration of injective morphism.

![Fig. 4-9 Illustration of injective morphism](image)
**Transformation Rule**

A transformation rule (rule for short) is a pair of morphisms which refer to three graphs, and one of them is injective and the other one is not required to be injective. The rule is designed for graph transformation.

**Graph Transformation**

Graph transformation is about the relationship between two graphs which associate with a rule. The rule transforms an original graph to a new graph, as shown in the following form (note that we are using double pushout approach for transformations):

\[
\begin{array}{ccc}
  L & \xrightarrow{\text{I}} & R \\
  \downarrow & & \downarrow \\
  G & \xrightarrow{\text{K}} & H
\end{array}
\]

In the above form, morphisms among graph \( L, I, R \) constitute a rule where graph \( I \) is called interface graph, graph \( L \) is the left-hand side of the rule and graph \( R \) is the right-hand side of the rule; the morphism from \( I \) to \( L \) is injective, which means graph \( I \) is a kind of sub-graph of graph \( L \) in the mapping sense; graph \( G \) is the original graph; graph \( H \) is the new graph after transformation by the rule; the morphism from \( L \) to \( G \) is also injective, which means \( G \) has a similar sub-graph structure of \( L \); graph \( K \) is a temporary graph during the transformation where \( K = G - L + I \) and \( H = K + (R - I) \).

### 4.3.2 Rule Based Transformation Over Idea Map

Idea map is used to organize ideas during the ideation, so that idea map has to support the following operations:

1) Merge : merge two idea maps
2) Enter : enter one idea to the idea map
3) Delete : delete one idea from the idea map
4) Subtract : subtract one sub idea map from the idea map
5) Replace : replace one sub idea map with another idea map

Before we head into the detail of the above operations, we should bear in mind the following things: (1) under which situation the operations will be executed; (2) how these operations will be executed; and (3) what is the result after the operation.

With the unified tree structure, we design rules for adding one node to idea map. When adding one node to an idea map, there are two kinds of possibilities: (1) add a child; and (2) add a sibling. If one node has no children then the coming node will be added as child; otherwise the coming node will be added as a sibling of the node’s children node.

In figure 4-10, we show a rule schema that **adds a node as child** to the idea map and a transformation with this rule. In the interface graph, the node has no child node, so the coming node will be added as a child. The digits \( 1,2,3,4,5,6 \) and the letters \( a, b \) indicate...
the morphisms from the interface to the left-hand side and right-hand side. The bottom part is an application with the rule, identifies the empty node. The rule adds the node 4 to node 1 as its child by identifying the beginning node and end node respectively.

In figure 4-11, we show a rule schema that adds a node as sibling to the idea map and a transformation with this rule. In the interface graph, the node 1 has already had one child node, so the coming node will be added as a sibling of its child node. The digits 1,2,3,4,5,6,7 and the letters a, b indicate the morphisms from the interface to the left-hand side and right-hand side. The rule adds the node 5 to node 2 as its sibling by identifying the beginning node and end node respectively.
4.4 Hierarchical Graph Transformation over Idea Map

In section 4.4.2, we can find that if we have to implement some complicated abstract sub-structures and the transformation would be very difficult. For example, if we want to merge two idea maps and one of them is already a mature solution, as shown in Figure 4-12, then it might destroy the unified structure of the idea map. To overcome this limitation, we borrow hierarchical graphs with an arbitrarily deep hierarchical structure from (Drewes et al., 2002). This is achieved by means of special edges called frames which may contain hierarchical graphs again. Moreover, it is useful to allow some frames to contain variables instead of graphs.

![Hierarchical Graph Transformation Diagram](image)

**Hierarchical Graph**

Comparing with graph, a **hierarchical graph** has a frame set more. A frame is a special edge with a mapping from the frame to its contents which can be either a hierarchical graph or a variable.

**Hierarchical Graphs Transformation with variables**

Rules for graph transformation are turned to rule schemata with variables and hierarchical graphs are transformed by applying instances of these rule schemata. A rule schema transforms the original hierarchical graph to a new hierarchical graph if there is an instance rule of the rule schema, as shown in the following form.

\[
\begin{align*}
L(a) & \rightarrow I \\
L & \rightarrow K \\
R(a) & \rightarrow H
\end{align*}
\]

In this form, the hierarchical graph \( L, I, R \) constitute a rule schema. This rule schema transforms the hierarchical graph \( G \) to hierarchical graph \( H \), and \( L(a), I, R(a) \) constitute a rule instance of the rule schema. The hierarchical graph \( K \) is a temporary graph where \( K = G - L + I \) and \( H = K + (R - I) \).
With hierarchical graph transformation with variables, the five operations mentioned in section 4.2.2 can be satisfied with two basic rule schemata. Operations of Merge and Enter can be done by the rule schema Add; operations of Delete and Subtract can be implemented by the rule schema Remove; and operation Replace can be combined by first Remove then Add.

4.4.1 Rule Schema Add

1. Add a sibling

In Figure 4-13, a rule schema that adds a framed item as sibling is shown together with a transformation example with this rule. The upper part is the rule schema. The item frame contains a variable $X$ regardless of its contents. The item frame contains a variable $X$ which makes it possible to duplicate the item idea map and to move one copy into the existed idea map. The variable $X$ in the item frame could be a single idea; the variable could be a sub-graph; and the variable could be another idea map. The bottom part is a transformation with the rule schema.

Fig. 4-13 Rule Schema Add sibling and an application

2. Add a child

In Figure 4-14, a rule schema that adds a framed item as child is shown and together with transformation with this rule.
4.4.2 Rule Schema Remove

1. Remove a sibling

In Figure 4-15, a rule schema that removes a sibling framed item is shown and together with transformation with this rule. The sibling framed item contains a variable X so that it can be removed entirely.

2. Remove a child

In Figure 4-16, a rule schema that removes a child framed item is shown and together with transformation with this rule.
4.5 Examples of Transformation

In this section, we show the merge operation of the two idea maps of the ideation example in Section 4.1, as shown in Figure 4-2 and 4-3. We show them together in Figure 4-17.

Fig. 4-17 The two idea maps need merging
Now the operation should be replacing the “(4) Idea organization” with the four-idea-sub-idea map. The four-idea sub-idea map is treated as a sub-tree from the hierarchical perspective, as shown in Fig 4-18.

Fig. 4-18 Hierarchical perspective of the sub idea map

“Replace” operation combines with first “Remove” and then “Add”.

Firstly, we apply the rule schemata Removesibling “4” over the graph of Figure 4-12, as shown in Fig 4-19

Then the rule schema Addsibling is applied to graph H in Fig 4-20.
Fig. 4-20 The rule schema Addsibling is applied to H in Fig 4-15 and the H in this Figure is the result.

Now we have the result graph by two step of hierarchical transformation, the graph H in Figure 4-20 is the final graph, as shown in Figure 4-21. The text representation is given in Table 4-3.
Fig. 4-21 The unified structure of the final graph
Fig. 4-22 The final graph

<table>
<thead>
<tr>
<th>ID</th>
<th>statement</th>
<th>name</th>
<th>Creativity technique</th>
<th>pID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Define the problem</td>
<td>IdeaTrigger</td>
<td>Problem Statement</td>
<td>root</td>
</tr>
<tr>
<td>2</td>
<td>What is it for</td>
<td>Dealing with innovative problem</td>
<td>Purposing</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Identify key themes</td>
<td>Idea generation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Identify key themes</td>
<td>Idea organization</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Identify key themes</td>
<td>Idea navigation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Identify key themes</td>
<td>Idea evaluation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>---</td>
<td>------------------</td>
<td>----------------------------------</td>
<td>-------------</td>
<td>----</td>
</tr>
<tr>
<td>7</td>
<td>Identify key themes</td>
<td>Transformation</td>
<td>Context Map</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>How it works</td>
<td>Generate ideas in reactive mode</td>
<td>5W1H</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>How it works</td>
<td>Generate ideas in active mode</td>
<td>5W1H</td>
<td>3</td>
</tr>
<tr>
<td>4.1</td>
<td>How does it look like</td>
<td>Topic map</td>
<td>5W1H</td>
<td>4</td>
</tr>
<tr>
<td>4.2</td>
<td>How to store them</td>
<td>Directed graph</td>
<td>5W1H</td>
<td>4</td>
</tr>
<tr>
<td>4.3</td>
<td>What can you refer to</td>
<td>Mind mapping</td>
<td>5W1H</td>
<td>4</td>
</tr>
</tbody>
</table>
5 RELATED WORK ON TRANSFORMATIONS

Modeling is one of the key activities for successful human problem solving. It is, in particular, a precondition for dealing with complex problems. As a consequence, various forms of modeling are used heavily in all kinds of natural and engineering sciences. The value of models stems from various uses: explanation, prediction, or as a way to model fundamental (causal) relationships in reality (Hesse & Mayr, 2008). What is the most appropriate form of a model depends on the specific situation and goal. A model always relies on an abstraction, i.e., the omission of details. Whether specific details can be safely omitted, depends on the specific goal that must be achieved.

In this deliverable, we aim to represent creative ideas and perform transformations among them. Thus, of particular interest to us are forms of model transformation and rewriting. Approaches of this kind are used in the area of symbolic reasoning, especially for theorem proving. A very important form of transformation is term rewriting (Baader & Nipkow, 1999), which goes back to the seminal work by Knuth and Bendix (Knuth & Bendix, 1970). In this approach an equational theory is successively transformed into a different representation of the same theory, which allows to more easily make deductions.

The rewriting of models also plays an important role in the context of software engineering, especially model-based software development and the accompanying transformation techniques. An extensive overview of different approaches to model-based development is given in (Czarnecki & Helsen, 2006).

As Metzger describes (Metzger, 2005), in software engineering a model must be seen relative to its formalism, which in turn is defined by abstract and concrete syntax, their static and dynamic semantics. Based on this differentiation Metzger also distinguishes different types of model transformation that may occur if the system, the model or the formalism is changed. A typical approach for performing model transformation, is to use the formalism of graph transformation as a basis as described by Grünske et al. (Grünske et al., 2005). An overview on various approaches to graph transformation was given by Ehrig et al. (Ehrig & Taentzer, 1996) and more recently in (Taentzer et al. 2005).

Graphs as a means of knowledge representation (and with them graph transformation) have an important shortcomings. Essentially, they are flat, they do not allow for a means of structuring and modularization of the information. In order to overcome this limitation, the notion of hierarchical graphs was introduced, which allow for the encapsulation of sub-graphs within an edge (cf. Drewes et al., 2002).
6 INTEGRATION OF CREATIVITY TECHNIQUES

As a main task of the combination of creativity techniques, IdeaTrigger deals with two kinds of integration. On one hand, the combination of creativity techniques itself is a kind of integration, combining various creativity techniques which are represented as a list of statements based on the principles of creativity techniques.

On the other hand, integration can be treated as a mash-up (Merrill, 2006) of a variety of different services from the implementation perspective. IdeaTrigger covers a variety of functionalities based on our storyboard, i.e., idea generation, editing, evaluation, transformation, chatting-room, member list maintenance and so on. Each functionality can be implemented as a service, which can be easily plugged in and plugged out.

6.1 Integration of creativity techniques

Creativity techniques that are used to generate ideas can be regarded as a list of questions. These questions are used to trigger new ideas from existing ones. Users of such creativity techniques are provided questions, and ideas are generated when they answer the questions. And associations are built between the ideas according to the questions.

Creativity technique \( CT_1 \) is mapped to question set \( S_1 = \{s_{11}, s_{12}, \ldots, s_{1n}\} \). Creativity techniques \( CT_2 \) is mapped to question set \( S_2 \), and so on. For idSpace platform, it should provide combination of several kinds of creativity techniques \( CT_1 \ldots CT_m \). Users might generate new associations which cannot be mapped to the questions that the idSpace platform already offer, then he/she can add these questions to the question set. As set operation is a mature mathematic model for implementation, it will be easy to do union operation of Set.

Fig. 6-1 Integration of different creativity techniques

As shown in Figure 6-1, idSpace platform only provides \( m \) kinds of creativity techniques, represented in the blue blocks; however, some users might find other questions might be useful to generate ideas, for example, as far as a special domain or a concrete problem are concerned. IdSpace should provide the functionality that the new questions can be easily added to (plug-in) the existing question list. Actually, what the creativity techniques are called does not matter. What the idSpace cares is that these techniques (question set) can be implemented and can trig new ideas in practice.
6.2 SCA-based Integration

Generally, an application can be considered as a set of software components working together. These components might run inside the same operating system process, in different processes on the same machine, or across the web. Service Component Architecture (SCA) (Chappell, 2007) defines a general approach which deals with how to create components and how to combine these components into complete applications.

6.2.1 SCA perspective

SCA is an executable model for assembly of services into business solutions. For IdeaTrigger, SCA has the following advantages: firstly components integrate with other components without needing to know how other components are implemented and can be easily invoked either synchronously or asynchronously; second, components can easily be replaced by other components. It also provides the flexibility to create solutions using existing services (bottom up) or model the business and develops the solution (top down).

Before heading into the details, we firstly introduce some terminologies. The basic building block for SCA is component implementation. A component implementation includes for attributes: (1) service which describes the function the components provides; (2) reference which describes the dependencies the component has in order to function; (3) property which defines configuration parameters controlling the behavior of the program logic; and (4) intent policy which describes assumptions on how the component will behave, as shown in Figure 6.2.

![Illustration of a Component](attachment:image)

Fig. 6-2 Illustration of a Component

Individual components can be used alone, or they can be grouped together in a composite. A composite is a type of component whose implementation is not code but an aggregation of one or more components cooperating to provide service as a whole. A composite is typically described in an associated configuration file, the name of which ends in .composite. This file describes the components this composite contains and specify how they relate to one another.
As shown in the storyboard, IdeaTrigger covers several main functionalities, such as idea generation, editing, navigation, idea organization transformation chatting-room and so on. From the perspective of SCA, theses functionalities can be dealt with as services. Each function is formulated as a service, implemented as a component; several similar services are grouped to a composite.

6.2.2 Service-based creativity techniques integration

In order to make the creativity techniques more easily for plug-in and plug-out, each creativity technique is implemented as a service. The idea generation component provides idea generation service, and it calls specific creativity technique services in order to generate ideas with some specific creativity technique. As shown in Figure 6.3, such components as 5W1H, SCAMMPER, and Mind mapping implement the counterpart creativity techniques, and provides 5W1H, SCAMMPER, and Mind mapping services. The idea generation component provides idea generation service and has 5W1H, SCAMMPER, and Mind mapping references, which means the idea generation component depends on the services provided by 5W1H, SCAMMPER, and Mind mapping components.

![Fig. 6-3 Idea generation composite, composed by different services](image)

The assembly is defined in .composite file through Service Component Definition Language (SCDL). Here the idea_generationcomposite would define that idea generation component references the other three components. Here the SCDL configuration would look like this, as shown in Figure 6-4.
<composite name="IdeaGeneration">
  <component name="IdeaGeneration">
    <property name="mode">reactive</property>
    ...
    <service name="IdeaGeneration">
      ...
    </service>
    <reference name="fiveWoneH" target="fiveWoneH">
      ...
    </reference>
    <reference name="Scammper" target="SCAMMPER">
      ...
    </reference>
    <reference name="mindmapping" target="Mindmapping">
      ...
    </reference>
  </component>
  <component name="fiveWoneH">
    ...
    <service name="fiveWoneH">
      ...
    </service>
  </component>
  <component name="SCAMMPER">
    ...
    <service name="Scammper">
      ...
    </service>
  </component>
  <component name="Mindmapping">
    ...
    <service name="mindmapping">
      ...
    </service>
  </component>
</composite>

Fig. 6-4 The ideageneration.composite file of Figure 3
7 IDEATRIGGER: COMBINATION OF CREATIVITY TECHNIQUES

WP2 deals with the combination of creativity techniques; IdeaTrigger is a web-based application which implements the combination of creativity techniques. The functionalities it covers include idea generation, idea organization, idea navigation, idea evaluation and transformation, all of which use idea entity as the central point of reference. As a web-based tool for collaborative work, it supports both asynchronous and synchronous cooperation among geographically distributed working groups. For better communication it also supports visualization of ideas based on a modified mind-mapping technology and an online chatting section.

7.1 Idea Generation with Two Different Modes

When directed thinking happens, some statements are used to stimulate ideas, as shown in Figure 7.1. When thinking freely, ideas are generated by a sudden enlightenment, as shown in Fig 7.2. However, there is definitely some relationship behind the sudden enlightenment.

Traditionally, creativity techniques are socially used. People are sitting around, talking about their ideas, exchanging ideas and evaluating ideas. There might be some principles and guidance for this kind of brainstorming, but much more attention is paid to ideas themselves instead of the relationship between ideas. That can be a reason why current creativity software is essentially mind-mapping tool, only for idea organization. From Figure 7-1 and Figure 7-2, we can see that relationship between ideas can be summarized as statement, and this statement can be some kind of trigger for new ideas. That means if some creativity techniques can be designed as a list of statements or questions, then we can implement these creativity techniques electronically by keeping the predefined statements or questions somewhere. When brainstorming begins, the predefined statements will come up in certain sequence so that people will keep them in mind. However, the predefined statements cannot cover all clues for stimulating new ideas; therefore, we classify the two kinds of thinking as two working modes in IdeaTrigger,
reactive mode and active mode. In reactive mode, people generate ideas guided by the predefined statement; in active mode, people generate ideas as they like but they are required to complement the association by adding the statement which has stimulated them.

### 7.2 Idea Organization

The creativity technique of Mind mapping provides a good graphical approach for idea organization. While using mind mapping, the central node is the beginning of the brainstorming session. Different ideas are radiated from the central node.

IdeaTrigger also organizes ideas basing on mind-mapping techniques, but with small changes because association between ideas will be stored as well. IdeaTrigger not only generate ideas but also keep the relationship between ideas. Figure 7-3 illustrates the modified mind maps of IdeaTrigger.

![Fig. 7-2 The modified mind maps of IdeaTrigger](image)

As shown in Figure 7-3, the nodes of ellipse shape are ideas. Q1, Q2 and Q3 are different questions which are predefined according to the principles of certain creativity techniques. The different instances of the same question, for example, oq1, oq2, oq3 and oq4, means that the predefined Q1 might be incomplete and is modified in the practical situation. There might be another kind of situation; people come out new ideas before questions are asked when getting the first sight of some idea, for example from C to C1, then the association S:s1 is established after and this statement can be added to the predefined question set if necessary.

By demonstration, we have an idea map tree as shown in Figure 7.4; the idea map begins at the root node, and radiates with children nodes. Because of the recursive structure of tree, each node can be treated as a beginning point to start a new session and end with a sub-tree.
7.3 Navigation, Evaluation and Transformation

The idea organization with a tree structure provides a visualization of the idea map. Not all the ideas generated are useful or relative to the problem. With the first-hand ideas, more operations are required, i.e., idea evaluation. During this phase, ideas will be marked as useful or useless, relative or irrelevant, as well as more associations might be established among ideas which have not existed during the idea generation phase.

Evaluation is more dependent on the subjective opinion; it is difficult to set up an objective standard so that an automatic operation can be carried out. IdeaTrigger will implement a voting mechanism, so that members can democratically decide what will be kept and what will be removed.

In order to make an efficient evaluation, the navigation mechanism is of great importance. The idea map tree is in essence a directed graph; therefore the navigation is in essence a kind of transversal of the graph.

During the evaluation, ideas are selected and the idea map tree might become a graph with circles or become several separated parts or sub-trees are required to be incorporated as part of the whole solution of the problem. Transformation based on hierarchical graph transformation deals with these issues.
There is another kind transformation which IdeaTrigger supports. After evaluation, all the ideas are considered as useful. Ideas are designed to have many attributes, some of which are about the person information. With the profile information and context information, automatic classification will be carried out based on these attributes such as person profile.

### 7.4 Collaborative editing

As a web-based application, IdeaTrigger is expected to support synchronous collaborative editing and drawing among geographically distributed group. In order to achieve this goal, a real time collaborative editor is incorporated.

### 7.5 SCA based implementation

Service Component Architecture (SCA) is an executable model for assembly of services into business solutions. For IdeaTrigger, SCA has the following advantages: firstly components integrate with other components without needing to know how other components are implemented and can be easily invoked either synchronously or asynchronously; second, components can easily be replaced by other components. It also provides the flexibility to create solutions using existing services (bottom up) or model the business and develops the solution (top down).

From the SCA perspective, such functionalities as idea generation, idea organization, idea navigation, evaluation and transformation can be considered as components providing respective services. Take idea generation for an example, as shown in Figure 7-4.

![Fig. 7-4 Transformation composite, composed by different services](image)

Components such as 5W1H, SCAMMPER, and Mind mapping implement the counterpart creativity techniques, and provide 5W1H, SCAMMPER, and Mind mapping services. The idea generation component provides idea generation service and has 5W1H, SCAMMPER, and Mind mapping references, which means the idea genera-
tion component depends on the services provided by 5W1H, SCAMMPER, and Mind mapping components.
8 CONCLUSIONS AND FURTHER WORK

In this deliverable, we have proposed a metamodel for the ideas generated from creativity sessions. We have further proposed a graph based transformation for electronic addition and merging of idea maps. Service composition model and transformation are further utilized for integration and combination of creativity techniques.

Graph transformation is essentially a matching-and-replacing operation on the graph. We define five basic operations on the idea map, and design rules respectively. Due to both the complexity of sub-graph matching and semantics of the idea map, the tree structure is selected, thus making use of its unified child-sibling structure. In order to make sub-graph operation straightforward without considering its inner structure, hierarchical graph and variables are introduced as well, which helps to classify the five basic operations into two categories.

There might be other approaches to deal with transformations from other perspectives, which might consider from the model level directly. Although mind maps are basically represented as tree structures, the idea trigger does not guarantee that there will not be a cycle. What would happen if there were cycles in the idea map? There might be many possible solutions, for example, during the creativity session people are demanded to avoid cycles. When a cycle would appear, people were required to duplicate the idea to avoid. However, duplication operation will generate much redundant information.

A better solution is to make use of hierarchical graph as mentioned in Chapter 4.4. The sub-graph which has cycles is considered as a whole. During transformation, the whole part is assigned to a variable, ignoring its inner structure.

Semantic information from topic maps can be a further direction to explore with regard to transformations. For example, even two sub-idea-maps with different labels has the same structure, there are completely different idea maps. Taking Add operation for an example, the Add points has to be firstly determined, then the Add rule can be applied.

Service oriented architectures and other means for service integration and composition will be further studied for the purpose of the next deliverable.
9 REFERENCES
http://www.davidchappell.com/articles/Introducing_SCA.pdf
Appendix A. DESCRIPTION OF THE MODEL

In this document we have introduced a metamodel for the idea maps. This model was adopted in this example to a simpler structure to reduce complexity and to show especially how it resembles the creativity techniques. The base classes for the entire structure are Node and Edge, which are used to represent the graph itself. Those classes are then specialized to represent the basic structure used for the idea map. The class diagram was then again extended showing the representation of the statements for the SCAMPER creativity technique. Using inheritance to store the specific information of each statement still allows us utilize the basic concepts of graph transformations without the need of redefining the operations for each creativity technique.
In the following, those classes are used to present a creativity session which utilizes the SCAMPER technique to create ideas how the ideatrigger could look like.

The example shows the final graph of this creativity session to define the needs for the ideatrigger using SCAMPER. In the following a step by step description is given.

1. The first step this the definition of the problem for this session. In this case it is to design ideatrigger. The definition of the problem is always the first step in a creativity session.

2. The first idea created is DrawingSheet. It uses the Combine statement of the SCAMPER technique. The idea is to combine several DrawingSheets into one application.

3. Triggered by the Combine statement, the idea is created to combine the drawingsheets with a mechanism that always fits them to the screen size of the device showing the application.

4. Triggered by maybe the external interrupt like a phone call, the user uses the Eliminate Statement and thinks that an application to share and interact with other people would be good. As a result the idea TelefonWithoutDialpad is added.
5. Looking at his/her desk, the idea is created by the PutToDifferentUse trigger to add PostIt notes as a part of the application, as they allow to quickly add information at different places.

6. Thinking further, the Substitute trigger generates the idea to replace the opaque paper with something transparent to allow reading the information below the PostIt.

7. As having too many PostIts is impractical. The idea is created to use a tree to sort them and replace the leaves by PostIts.

8. The Reverse Statement triggers the idea to turn the tree upside down, as computer scientists usually draw trees with the root on top.

9. In a last step, the Combine trigger tell the user that it would be by far more practical then different applications and he creates the idea to combine all three things into one nice application.