Mining for Behavioural Information in Creative Processes

Ph.D. Thesis

Sascha Westendorf

This thesis is submitted in partial fulfilment of the
requirements for the degree of Doctor of Philosophy

Software Technology Research Laboratory
De Montfort University
March 2011

Publications

H. Zedan, A. Cau, K. Buss, S. Westendorf, S. Thomas, and A. Hugill, Mapping human creativity, In Proceedings of the 12th Serbian Mathematical Congress, Novi Sad, 2008.

H. Zedan, S. Westendorf, K. Buss, The effect of collaboration and co-creation on the creative processes, In Proceedings of Second International Conference of Creativity and Innovation in Software Engineering, Ravda (Nessebar), Bulgaria, 2009.

K. Buss, S. Westendorf, H. Zedan, Mining for behavioural knowledge and information in the creative processes, In Proceedings of Second International Conference of Creativity and Innovation in Software Engineering, Ravda (Nessebar), Bulgaria, 2009.

Declaration

I declare that the work described in this thesis is original work undertaken by me between April 2007 and September 2010 for the degree of Doctor of Philosophy, at the Software Technology Research Laboratory (STRL), De Montfort University, United Kingdom.

Acknowledgements

I would like to express my deepest gratitude to my supervisors Prof. Hussein Zedan, Prof. Sue Thomas and Prof. Stephen Brown for their advice, experienced guidance and encouragement during the last years. It was surely one of the most challenging and important periods of my life. I would also like to thank De Montfort University (DMU), the Institute of Creative Technologies (IOCT) and especially Prof. Andrew Hugill for the financial support.

Furthermore, I would like to thank all colleagues of the Software Technology Research Laboratory (STRL) at De Montfort University (DMU) for their valuable discussions and suggestions. Especially Keno Buss, with whom I worked together for the last years, Peer Bartels, Stefan Natelberg and Matthias Ladkau.

I am particularly grateful to my parents for their invaluable love and support, which gave me the courage to continue my studies - I could not have done without them.

Abstract

Creativity is a topic of high interest in a variety of domains; many innovations, discoveries and developments are the result of creative ideas. A prerequisite for the identification of creativity is an artefact, which needs to be evaluated by the domain that receives it. The involved actions during the creation of this artefact represent the creative process of a creator and include essential information about the involved creativity. Structuring and analysing this data are important steps for a better understanding of its nature. A domain independent framework, which allows to represent the mentioned structures and provides a set of sound mathematical rules for its study is used as a formal underpinning for the presented approach.

Each sequence of actions, which is included in a creative process describes a behaviour. It contains a rich set of information, like the particular order or duration of the creation steps and can be analysed to gain insight into the process. This data might then eventually be used for the creativity support. The creative process itself is usually non-linear, as previous stages or ideas can be revisited by the creator. It instead describes a complex structure with multiple branches, which is called a creativity map. This map builds the essential preliminaries for this thesis. The proposed research presents an approach for the recording of creative processes and construction of creativity maps.

Especially if an extensive observation of the creative process is undertaken, it is possible that a creativity map grows large. It might then contain information, which are irrelevant or even disturbing for the current view. This can include particular subsets or sequences of actions as well as insignificant time periods or other items that are related to the particular process. It would be beneficial for the support of an aim oriented analysis to conceal this information at least temporarily. The proposed approach therefore introduces the concept of a Partial Creativity Map (PCM), which allows to hide subsets of the originally recorded behaviours. A creativity map can then be modified with respect to the requirements of the analysis.

Each behaviour in a creativity map possesses a particular frequency, which allows for a detailed insight into the creator's preferences as well as common or uncommon activities. This range of frequency related information can be useful for the creativity support, for instance in situations where the creator is "stuck". Behaviours that were performed frequently in similar situations might be helpful in this case. The proposed research introduces a classification of frequent behaviours and explains its use for the description of their dynamics, which enable to reason about temporal properties of behaviours.

An initial version of the De Montfort Creativity Assistant (DMCA) is implemented for the prototype tool support of this research. Its aim is the construction of an open and extendible framework that allows to study the collected data and support creative processes. It is designed as a pluggable system which provides a convenient environment for the creation, sharing and communication of artefacts. Clear and domain-independent interfaces provide the required structures for a seamless integration of new components. The included tools and particularly the De Montfort Creativity Mapper (DMCM) are capable of observing, constructing and modifying the creative process and the corresponding creativity map.

The presented approach is evaluated with three case studies. They illustrate the recording of creative processes, construction of creativity maps and information hiding and extraction strategies. The thesis is finally summarised, limitations are explained and suggestions for future directions are presented.

Table of Contents

List of Figures

List of Tables

List of Acronyms

BDA Behaviour Description Automaton

BDL Behaviour Description Language

CMCE Creativity Map Construction Engine

CMR Creativity Map Repository

CPS Creative Problem Solving

DFA Deterministic Finite Automaton

DMCA De Montfort Creativity Assistant

DMCE De Montfort Creative Environment

DMCM De Montfort Creativity Mapper

DMU De Montfort University

EBNF Extended Backus-Naur Form

ERM Entity-Relationship Model

GUI Graphical User Interface

IME Information Mining Engine

IOCT Institute of Creative Technologies

JDK Java Development Kit

JRE Java Runtime Environment

KDD Knowledge Discovery in Databases

NLP Natural Language Processing

PCM Partial Creativity Map

STRL Software Technology Research Laboratory

UML Unified Modeling Language

WWW World Wide Web

ϵ-NFA Non-deterministic Finite Automaton with ϵ-transitions

Chapter 1

Introduction

Objectives

- Explain the motivation and aim of the proposed research.
- Formulate the research questions.
- Present the scope of the thesis.
- Emphasise original contributions.
- Illustrate a brief overview of the thesis organisation.

1.1 Motivation and Aim of Research

The high interest in creativity resulted in a large number of models for creativity and creative processes. Most of them lead to specifications of subconscious activities, without the ability to be manipulated actively. They describe the creative process at an abstract level, mainly happening inside the head of the creators, resulting in some kind of creative outcome. These models often try to map different roles and specify their interactions. They are usually unable to represent detailed information about the individually performed steps and therefore difficult to utilise for computational creativity assistance. For an active support of the creative process, it is necessary to analyse the steps that are performed

during the creation of an artefact, as they include valuable information about the creator as well as the creative process and are essential for a better understanding of creativity.

The mentioned activities represent the behaviours of a creator and the observable influences on the creative process. In order to reason about creativity and study the steps of the creative process, it is necessary to introduce a model that maps this process into a structured form. A sound set of rules and laws is needed to support its analysis. The previously mentioned approaches do not specify any structures, which satisfy these requirements. Especially the absence of a mathematically underpinned model of the creative process was a major reason that motivated this research. The presented approach, which is used for the mapping of the process, is based on a generalised transition system [15]. It allows for the capturing and representation of behavioural information about the creator.

The model enables a fine grained representation of the artefact creation. Especially for large projects, it is possible that the creative process becomes complex and difficult to handle. Some of the performed steps might be irrelevant for the analysis. The concealment of these activities would benefit the extraction of useful information. Particularly the preprocessing phase needs to allow for the modification of the creative process in order to enable an aim oriented analysis. As a solution, the thesis presents a behaviour based information hiding approach for the creative process, which allows to specify and hide any of its parts.

Another issue especially for large creative processes is the extraction of frequency related information. It enables an insight into the relationship between actions and the personal preferences of the creator. For this reason, a behaviour extraction approach will be introduced, which concentrates on the range of frequent information that is contained in the creative process. Not only the most frequent behaviours, but also common, uncommon and rarely used ones are useful for the analysis. They enable to describe the dynamics of the creative process and can be used for tracking purposes.

The described model additionally allows the mentioned approaches to be integrated into software tools for the observation and study of the creative process. This enables the computational analysis of the captured data in order to extract valuable information about the involved creativity. Results might then be used to create assisting feedback and support the creation. In order to process this material, well designed data, information and knowledge representations are needed. They build the foundation for creativity mining. The presented research aims at the discovery of possibilities, which allow for a computational analysis and support of the creative process.

1.2 Research Methodology

The presented thesis belongs to the research area of software engineering. This field particularly refers to the development and contribution of knowledge in the form of new algorithms, methods, models, frameworks and theories that are evaluated through a number of case studies. The presented research aims to develop techniques that support the analysis of creative processes. A generalised transition system as the formal mathematical underpinning is the foundation for the presented methods and tools. It allows to reason about creative processes and their modifications by referring to a set of defined rules and laws. These rules are used for the introduction of information hiding and extraction approaches. The presented research was realised as follows.

Identification of the Problem, Research Question and Hypotheses

The first stage of the research started with the problem identification. Related literature was studied to gain sufficient background knowledge for the understanding of this problem in its full scope. The presented research crosses different domains, which made it necessary to study related research projects and literature about creativity and creative processes as well as data and information mining. Most helpful for the first part were the models of Boden [11], Csikszentmihalyi [22] and Wallas [97]. A useful overview of the data mining process and its relation to knowledge discovery is presented in [32].

Construction of a Model and Implementation of the Approach

An essential requirement was a sound model for the mapping of creative processes, which allows for their analysis. It was used as a formal underpinning for the development of the information hiding and extraction approach. Both techniques are based on the profound set of rules and laws. Models for a Partial Creativity Map (PCM) together with frequency and frequent behaviour categories were developed for their realisation. A prototype software has been developed to demonstrate and evaluate the need and practical applicability of the proposed approach.

Evaluation of Hypotheses

The presented hypotheses were evaluated with the help of three different case studies. It was demonstrated that the conducted approach is applicable and produces reliable results. Each case study focussed on particular criteria. The first one tackled the construction process of creativity maps, the second one demonstrated the information hiding approach and the third one illustrated the extraction of frequent information and the tracking of three different behaviours to emphasise their dynamics.

Interpretation and Future Directions

The results of the evaluation have been interpreted and conclusions were drawn. Limitations of the presented approach were identified and future research directions in this field were formulated. Both affected the presented models as well as the demonstrated prototype tool support.

1.3 Research Questions

The motivations and aims described before explain the problems in current research of creativity and creativity support which the proposed research addresses. The following research questions are answered by this thesis.

1. Is it possible to record the creative process computationally?
2. How can the creative process be used to mine for behavioural information?
3. Can (temporarily) irrelevant information of the creative process be hidden?
4. What kind of frequency related information can be extracted from the creative process and how can it be used to describe the dynamism of behaviours?
5. How can the proposed research be used to implement initial tools for computational creativity support?

Chapter 9 presents a summarised evaluation and refers to the particular locations of this thesis, where the questions are answered in their full scope.

1.4 Research Hypotheses

1. A model for the representation of the creative process together with a formal underpinning will allow for the analysis of behaviours. This is beneficial for computational creativity support and the design of tools that provide assistance during the creative process.
2. When capturing the creative process, it is necessary to record as many steps as possible for a detailed representation. The creative process can grow very large and

the analysis will become time consuming. Techniques to hide irrelevant parts of it are needed to enable an aim oriented analysis.

3. The frequency of behaviours can be utilised for the representations of preferences, common or uncommon activities of a creator. This information is useful for the further study and can help to build behavioural patterns. Techniques for frequent behaviour extraction will support the analysis and allow to gather essential information about the creative process.

4. The creative process is dynamic and information will continuously change over time. The representation of this dynamism will enable the tracking of behaviours for one or more creative processes, which can be utilised for the computational creativity support.

1.5 Scope of the Thesis

This thesis will propose an approach to modify creative processes and extract information for the support of their analysis. A technique for information hiding based on behaviours will be introduced. Together with this, a frequency metric and an information extraction approach will be presented, which enable the mining for frequency related information and its dynamism. The scope of this thesis includes:

1. Description of a dynamic construction procedure for creativity maps.
2. Definition of data, information and knowledge and identification of behaviours.
3. Introduction of Partial Creativity Maps (PCMs).
4. Specification of a formal language for the description of behaviours.
5. Explanation of a strategy to hide behaviours from creativity maps in order to reduce their size and support the analysis.
6. Specification of frequency and frequent behaviour categories, which allow for the distinction and extraction of frequent behaviours from creativity maps and the tracking of their dynamics.
7. Implementation of the De Montfort Creativity Assistant (DMCA) and De Montfort Creativity Mapper (DMCM) as tool support to enable collaboration and the recording, visualisation and modification of the creative process.

8. Evaluation of the approach based on three case studies. Each of them presents a different scenario.

1.6 Original Contributions

The original contributions of this thesis are the following.

1. Specification of data, information, knowledge and behaviours in the creativity mapping model. These components are essential for the information hiding and extraction approaches.
2. Specification of a formal language for the description of behaviours. This enables a convenient definition of behaviours for the information hiding process.
3. Introduction of information hiding, restriction, revealing and restrictive revealing operations, which work seamlessly together with the formal language for the description of behaviours. Collapsing and pruning approaches for further size reduction of creative processes are discussed.
4. Specification of frequency and a corresponding classification of behaviours. This is used for the introduction of frequent behaviour categories that represent the dynamism of behaviours and are utilised for their tracking. A technique for the extraction of frequency related information, which allows for an individual adjustment based on three frequency thresholds is presented.
5. Development of a prototype tool for the creation and sharing of artefacts and the recording, visualisation and modification of the creative process to demonstrate the applicability of the described approach.

1.7 Organisation of Thesis

Chapter 1 - Introduction

The motivation and aim for the proposed research are described in this chapter. This includes the research methodology, research question, research hypothesis, scope of the thesis, original contributions and the outline.

Chapter 2 - Background and Related Research

This chapter discusses the related research and background information. It reviews models for creativity and creative processes, discusses creativity enhancement techniques, addresses computational creativity support and illustrates data mining approaches, which are all related to the presented research.

Chapter 3 - Preliminaries

The creativity mapping model is introduced in this chapter as preliminaries for this research. Creativity maps are discussed informally, their ability to represent collaboration is explained and the hopping phenomenon is presented.

Chapter 4 - Creativity Maps and Behaviours

This chapter explains the construction process of creativity maps. The three entities data, information and knowledge are furthermore defined with respect to creative processes and their relationship is explained. A hierarchical classification of creativity maps is discussed and behaviours, which are a major component of the proposed research, are introduced.

Chapter 5 - Information Hiding

The information hiding approach, which is used to dismiss irrelevant behaviours of creativity maps is explained in this chapter. This includes the introduction of Partial Creativity Maps (PCMs), which represent only a subset of the original information, the specification of a formal language, namely Behaviour Description Language (BDL), for a convenient expression of behaviours and the definition of essential operations for information hiding and revealing.

Chapter 6 - Frequent Information Extraction

This chapter discusses the extraction of frequent information. It introduces four frequency categories, which represent preferred as well as common and uncommon behaviours. The dynamism of the creative process is expressed with additional frequent behaviour categories. It is illustrated how the information hiding and extraction approaches can be combined to build one overall step.

Chapter 7 - Tool Support

The prototype tool support for the proposed approach is presented in this chapter. Initial versions of the De Montfort Creativity Assistant (DMCA), the collaboration framework De Montfort Creative Environment (DMCE) and the Collaborative Editor are described. The prototype of the De Montfort Creativity Mapper (DMCM), which allows for the recording and study of the creative process is introduced.

Chapter 8 - Case Studies

Three case studies are presented in this chapter to demonstrate the applicability of the proposed approach.

Chapter 9 - Conclusion and Future Work

This chapter summarises the thesis. It evaluates the presented research, illustrates limitations and suggests directions for future studies.

Chapter 2

Background and Related Research

Objectives

- Discuss different models for creativity and the creative process.
- Present common techniques for the enhancement of creativity.
- Review approaches for computational creativity.
- Discuss related research in the area of computational creativity support.
- Explain data and sequence mining techniques.

2.1 Introduction

This chapter presents an overview of the background and related research in the fields of creativity, creativity support and data mining. It explains and reviews some very popular models for creativity and creative processes. The area of computational creativity is described shortly and it is shown how the previously mentioned models are transformed into processable representations. An overview of computational creativity support, which is a relatively new area of interest and research, is presented. Furthermore, some popular creativity support techniques that are already used in businesses and other organisations are explained. Data mining techniques are presented in the last part of this chapter and it is described how patterns are recognised and frequent information can be identified.

2.2 Creativity and Creative Processes

Creativity has been a topic of high interest for many years [7][100] and a lot of models for the creative process and particularly the steps during the construction of creative ideas were introduced. They try to identify general sequences of, in many cases, abstract steps, which result in a creative outcome. These steps are often difficult to observe and complicate the evaluation of creativity, which is usually influenced by subjective factors. A mathematician might describe a new solution for a differential equation as creative, whereas an artist thinks the Mona Lisa is a creative painting. More importantly, the domain that receives the artefact instead of the creator decides about its creativity. This domain or its experts provide sufficient knowledge to determine novelty and usefulness.

The book *Creativity* [96], which was edited by Vernon and published in 1970 includes a good overview of some major contributions to the field of creativity. It contains a very early model of the creative process from Graham Wallas [97]. A review of current research in creativity is provided in the *Handbook of Creativity* [92] that was published in 1999. It covers several aspects and collects influencing models, like the Conceptual Spaces from Margaret Boden [11] or the Systems View of Creativity from Mihaly Csikszentmihalyi [21]. These two models and the previously mentioned one from Wallas are reviewed in this section.

2.2.1 Wallas' Stage Model

Graham Wallas presented one of the first models of creativity [97]. He analysed a talk from the German physicists Hermann von Helmholtz and identified four distinct stages, namely *Preparation, Incubation, Illumination* and *Verification*. In [97], Wallace firstly described his model in the following way.

> The first in time I shall call Preparation, the stage during which the problem was 'investigated ... in all directions'; the second is the stage during which he was not consciously thinking about the problem, which I shall call Incubation; the third, consisting of the appearance of the 'happy idea' together with the psychological events which immediately preceded and accompanied that appearance, I shall call Illumination. And I shall add a fourth stage, of Verification, which Helmholtz does not here mention.

The problem is investigated in all directions during the preparation stage. Task specific information as well as a wide overview are gathered to receive a large variety of impressions. Wallas mentions that an educated person is able to focus on ideas. This allows for a better thinking process, especially when the attention is directed to smaller sub-elements of a problem.

The mind is unconsciously thinking about the problem during the incubation stage. This time can be spent either with contemplating on other problems or simply with relaxing. Wallas points out that of course the first way uses the time more efficiently. However, it is sometimes necessary to relax and let nothing else interfere with the unconscious process of the mind.

The illumination phase describes the stage, where a new idea arises. It is illustrated as a successful train of associations that may be preceded by an unsuccessful one. This stage cannot be influenced by direct effort or will, as Wallas points out.

> [...] the evidence seems to show that both the unsuccessful trains of association, which might have led to the 'flash' of success, and the final and successful train are normally either unconscious, or take place (with the 'risings' and 'fallings' of consciousness as success seems to approach or retire), in that periphery or 'fringe' of consciousness which surrounds our 'focal' consciousness as the sun's 'corona' surrounds the disk of full luminosity.

Wallas furthermore distinguishes a specific phase called the intimation that precedes the rising of a creative idea. It is the moment, where possible associations are made and which indicates that a conscious 'flash' of success is coming. If the brain is able to identify associations, the intimation automatically creates the new and creative idea that is then recognised consciously.

The last phase in Wallas' model describes the conscious verification of the idea, for instance by using mathematical or logical rules. Wallas describes this stage as being very similar to the first one, which was introduced as preparation. For example, the same set of rules, which was used during the preparation is also utilised to verify an idea. A good and deep preparation is therefore necessary for a successful evaluation and verification process.

The creative process as specified in Wallas' model describes creativity as an unconscious process. It is first of all difficult to recognise and determine when the actual creative idea arises and furthermore hardly possible to actively influence the process. This complicates

the retrieval of information about the events during the illumination, which seems to be essential for the understanding of human creativity. It also restricts the usability of Wallas' model for the support of creativity and creative processes, as a detailed understanding of the performed actions in each of the phases is essential.

2.2.2 Boden's P-/H- Creativity and Conceptual Spaces

Margaret Boden, a Professor of Philosophy and Psychology, starts to describe creativity in [11] as a mysterious phenomenon, which cannot be explained or identified at all.

> Creativity is a puzzle, a paradox, some say a mystery. Inventors, scientists, and artists rarely know how their original ideas arise. They mention intuition, but cannot say how it works. Most psychologists cannot tell us much about it, either. What's more, many people assume that there will never be a scientific theory of creativity - for how could science possibly explain fundamental novelties? As if all this were not daunting enough, the apparent unpredictability of creativity seems to outlaw any systematic explanation, whether scientific or historical.

Boden firstly distinguishes between two senses of creativity. On the one hand the psychological creativity (P-Creativity), which involves the rising of novel ideas that are new to the person who created them. It implies a very personal understanding of creativity and it is irrelevant, if someone else had the same idea. On the other hand the historical creativity (H-Creativity), which involves ideas that are firstly P-creative, but necessarily no one else came up with them before. Where P-Creativity can be valued, although it might not always be regarded as worth having, H-Creativity cannot be methodically explored. A systematic explanation for H-creativity is impossible, as Boden describes in [11].

> [...] there can be no systematic explanation of H-creativity, no theory that explains all and only H-creative ideas. Certainly, there can be no psychological explanation of this historical category. But all H-creative ideas, by definition, are P-creative too. So a psychological explanation of P-creativity would include H-creative ideas as well.

This quote illustrates that the evaluation or measurement of creativity are impossible, due to the existence of various unrelated and unforeseeable factors. What is valued by one

group to be creative, will not necessarily be estimated in the same way by a different group or person. However, for the study of creativity, it is crucial to understand P-creativity, since it is not relevant who had the idea first, but what lead to its creation.

The specification of P- and H-creativity does not provide insight into the emergence of creativity. Therefore, Boden developed a model of three ways of creativity [12]. The first includes the creation of unfamiliar ideas by combination of familiar ones. An example is a journalist who compares a politician with some kind of animal. The combination procedure requires knowledge in a variety of areas, as the journalist must be informed about politics as well as animals. Every combination itself needs to be sensible, especially if presented to others, so that relationships between both concepts are evident even if combined randomly. The second and third way of creativity are related to conceptual spaces, which are specified as styles of thought and "any disciplined way of thinking that's familiar to (and valued by) a certain social group" [12]. A variety of thoughts is possible within a given space, of which only some were actually thought. An example is chess, where a finite, astronomically high number of possible moves exists, but probably not all of them were performed yet.

The second way of creativity is based on the exploration of these conceptual spaces. Creating a new and also unexpected idea is to a certain degree creative in an explorative sense. It reveals possibilities that would have stayed hidden otherwise. An example is a car journey on a motorway in an unknown country that reveals unexpected buildings whenever the road is left and small ways are followed. Boden mentions that the exploration of conceptual spaces often leads to new ideas and additionally shows their limitations. To overcome these limitations, it is necessary to change the conceptual spaces.

The third way of creativity is based on the transformation of conceptual spaces. With respect to the example above, the majority of the roads is fixed and cannot be changed. However, it is possible to modify the thinking styles. A new idea arises, if a preexisting style is changed in a specific way. By tweaking or complete transformation, creative ideas can develop which were previously impossible. Creating a new route on the motorway in mind is difficult, but may for example result in a journey to a surprisingly new country. For the transformation of a conceptual space, it is necessary to think in different ways than the previously possible and existing ones. This allows for thoughts that were literally inconceivable before [12].

Boden's different ways of creativity can be used to describe the creative process in an abstract fashion. Combining ideas or exploring and transforming spaces are activities on a

higher level, which need to be broken down to more personal actions, such as reading, writing or programming. This means that the model requires more specificity to support the study of creativity. The introduced P- and H-Creativity are useful for a description of the relationship between creator and domain. Especially P-Creativity seems to be important to understand the generation of creative ideas and artefacts. Analogies to these elements can also be identified in other models, such as the one from Mihaly Csikszentmihalyi [21]. The idea of conceptual spaces and their exploration or modification sometimes raises the questions, how computers themselves can be creative or how computational creativity can be produced. However, this is not relevant for the support of human creativity. It is more important to focus on a person's creative process and provide assistance.

2.2.3 Csikszentmihalyi's Systems View of Creativity

Mihaly Csikszentmihalyi [21] developed *The Systems View of Creativity*, which is based on the idea that creativity does not emerge from the individual alone, but from its interaction with the social-cultural environment. His model includes the three entities domain, field and individual and specifies their interplay as shown in Figure 2.1, according to [21].

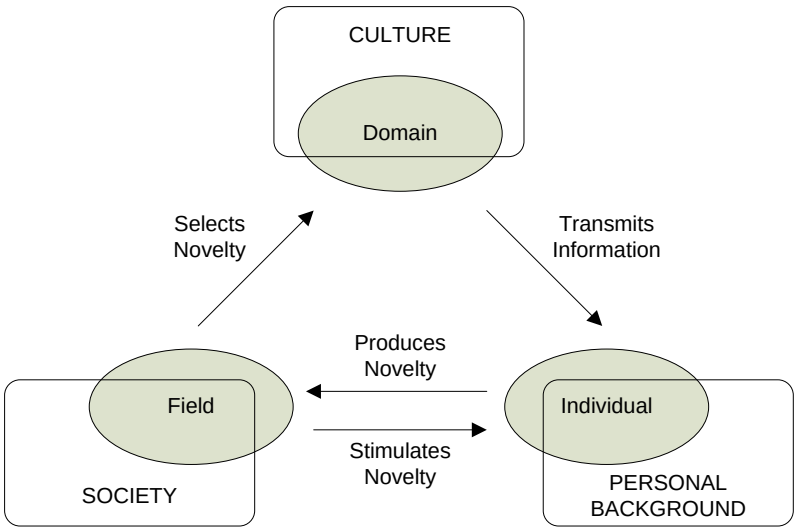


Figure 2.1: Csikszentmihalyi's Systems View of Creativity

Domain The domain specifies the part of an environment that the individual lives in. It is similar to a repository which contains all existing patterns and the knowledge of the environment. A domain is essential, as an artefact or product can only be determined as being new in relation to something existing or old. For instance, new experiments

in chemistry can only be creative, if it is possible to evaluate and compare them with respect to existing knowledge and previously performed experiments.

Field The field represents a group of experts with sufficient domain knowledge in order to determine if an idea or artefact should be included into the domain. Csikszentmihalyi describes the field as gatekeepers, which he narrows down to the social organisation of a domain, like journal editors or foundation directors. These people decide about the domain and its content. He mentions the example of Einstein's theory of relativity, where the field is represented by a couple of university professors, who decided that his idea is creative. As a result, the majority of people also considered Einstein to be (highly) creative, without necessarily understanding his theory.

Individual The individual specifies the person who generates a possibly creative idea or artefact. In contrast to the overall judgement that is performed by the field, the individual is only able to evaluate on a personal basis.

From Csikzentmihalyi's point of view, especially the interplay of these three entities represents the essential component of creativity. In his book *Creativity: Flow and the Psychology of Discovery and Invention* [22], he defines creativity in the following way.

> [...] Creativity is any act, idea, or product that changes an existing domain, or that transforms an existing domain into a new one. And the definition of a creative person is: someone whose thoughts or actions change a domain, or establish a new domain. It is important to remember, however, that a domain cannot be changed without the explicit or implicit consent of a field responsible for it.

According to Csikszentmihalyi's view, a typical process starts with the individual who uses information provided by the domain. The person transforms this information to create something novel and creative. If the transformation is estimated as valuable by the field, it will be included in the domain that is held by the culture. The manipulated domain and therefore extended knowledge repository then becomes the foundation for the next cycle. Instead of the particular elements themselves, the essential part of creativity in this model can be identified as the described interactions.

The domain entity is required to provide a sort of symbolic system, which enables the identification of improvement. If the current creation does not improve the domain, it is usually not considered to be creative. Especially if the system does not implement this

notion sufficiently, the judgement can become difficult. Furthermore, Csikszentmihalyi identifies a difference in the accessibility of domains and the simplicity of their modification. The reason for this is usually the protection of a domain by a group or class of people. It was for instance very difficult for Galileo to change the domain of astronomy, because of the church which did not accept his ideas. However, the accessibility of domains has improved in the last couple of years due to the development and growth of the internet. It allows to publish and distribute vast amounts of information that are in turn accessible for a large number of people.

The field entity represents an essential element in the systems view of creativity, which allows to infer that an objective estimation or measurement of creativity is impossible. The determination process will always be subjective, as it relates to the personal knowledge, experience and background of the judges in order to identify a creative artefact. The field is usually independent from the creator. However, if being a castaway on a desert island, the creator can only decide himself about the creativity and therefore becomes one of the judges. This does not necessarily mean, that the constructed artefact is determined to be creative by others, for example when returning from the island. Csikzentmihalyi's model specifies that creativity is not directly stored in the creation itself, but rather in its effect on others. He compares his view with the model of evolution, where organisms mutate and only the best ones are selected for the next generation.

Csikszentmihalyi's model of creativity focusses on the interaction of the three entities domain, field and individual. It describes the creative process of a person in an abstract way as the exchange of information between them. As mentioned before, the creation of a creative outcome changes the domain and simultaneously establishes a communication between the domain and individual. However, the steps which lead to the product or artefact are not further considered in the presented model. Especially their determination and analysis are important for the support of the creative process, as they allow for the creation of valuable and assisting feedback.

2.3 Creativity Enhancement

Apart from the previously presented models for creativity and the creative process, researchers are also interested in possibilities for the enhancement of creativity [72]. One of these strategies is brainstorming, which was developed by Osborn [74][75] in the 1950's. It describes a process that is especially used by groups in order to create a large number and variety of ideas for a particular topic. The main purpose of brainstorming is to generate

and express thoughts without any evaluation, as criticism probably reduces the amount of ideas quickly to only sensible ones. In contrast, brainstorming enables the expression of any thoughts, with the ulterior motive to stimulate other team members, who then create new, maybe peculiar ideas themselves. It is very popular and can sometimes be identified in other models, such as Creative Problem Solving (CPS). This section presents three creativity enhancement techniques.

2.3.1 Creative Problem Solving (CPS)

Brainstorming builds the foundation for the development of another technique, namely Creative Problem Solving (CPS) [75] [53] [54] that also aims for the creation of creative ideas. CPS consists of three major components, each including a individual set of steps [35]. They are depicted in Figure 2.2, according to version 4.0 that is specified in [53].

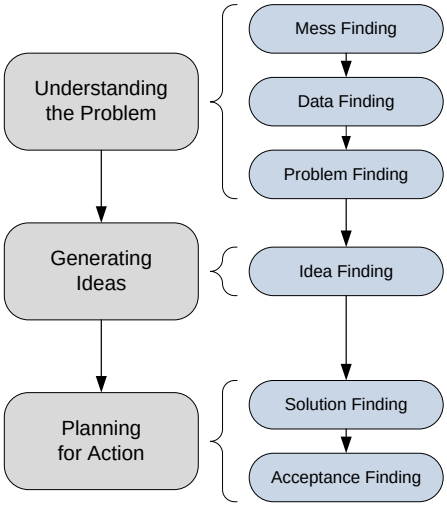


Figure 2.2: Creative Problem Solving

The first major component is *Understanding the Problem*. It is divided into the three steps *Mess Finding*, *Data Finding* and *Problem Finding*. *Mess Finding* is used for the identification of challenges. It is necessary to understand the opportunities and goals that should be achieved at the beginning. *Data Finding* identifies data that is related to and relevant for the current situation. Sources are examined from different viewpoints and the essential and most important facts are revealed. This is similar to a preprocessing step, which is performed in order to "hide" irrelevant data and focus on the relevant parts. *Problem Finding* explores the issues and generates possible problem statements. The most relevant and promising ones are extracted and further processed.

The second major component is *Generating Ideas*, which contains the single step *Idea Finding*. It generates a large set of ideas that tackle the previously specified problem. The brainstorming technique which was described above is one possibility for their creation. It is important that a large variety of ideas arises to offer a wide range of perspectives, which includes potential ideas for the solution of the problem.

The third major component is *Planning for Action*, which is divided into two distinct steps. The first step is specified as *Solution Finding* and describes the application of strategies and criteria that support the development of ideas and solutions. These strategies should support the discovery of the best solution that can be generated. The last step is specified as *Acceptance Finding* and is responsible for the generation of support and acceptance. It is necessary to understand and fully support the problem solution to provide an implementation in the best possible way. Any resistant needs to be broken.

CPS represents a probably common process that can be identified during the creation of ideas and solution. It is first of all necessary to read related literature and ideas, specify a problem, search for possible concepts and then define a solution based on the gained knowledge. A similar process can be identified in Wallas' four stages model of the creative process.

2.3.2 The Six Thinking Hats

Another technique that is used to enhance creativity is called the *Six Thinking Hats* and was developed by Edward de Bono [25]. This strategy assumes that a group of people is working on the solution of a problem. The thinking process is divided into six distinct categories, each represented by a single hat. Every team member "wears" one of these hats and similarly represents one category during the discussion. This creates different viewpoints of the problem and possibly enhances the efficient generation of new and useful ideas. The team can become more productive and generate better results. De Bono introduced the following six thinking hats.

White Hat The *white hat* focusses on the analytical thinking. The person who is "wearing" this hat is concentrating on the given facts and possibilities to achieve the goal. This thinking hat also considers missing data and its retrieval.

Red Hat The *red hat* is the "emotional" viewpoint. The person who is "wearing" it needs to focus on emotions, for example the feeling about the project or possible solutions. This thinking hat does not need to be rational or require a logical foundation.

Black Hat The *black hat* represents the critical point of view on the problem. It is a defensive position, which mainly tries to identify possible flaws, problems or weaknesses. This thinking hat intervenes and prevents the group from wrong decisions.

Yellow Hat The *yellow hat* represents a positive viewpoint. The person who is "wearing" it is rather optimistic and addresses the advantage of a solution. This thinking hat does not consider disadvantages or possible risks and tries to identify opportunities.

Green Hat The *green hat* represents creative thinking. It looks at the proposed ideas from different viewpoints and tries to modify and vary them for a new and possibly useful outcome. This thinking hat illustrates an associative thinking that combines existing ideas in order to identify new solutions.

Blue Hat The *blue hat* guides the whole idea finding strategy. The person who is "wearing" it organises, summarises and plans the discussion in order to manage the current process. It is probably worn by the moderator or leader of the group debate.

The six thinking hats are primarily interested in different viewpoints of an idea. Among other things, they reveal possible opportunities, flaws or boundaries. The comments of the team members stimulate and encourage other participants to create new ideas.

2.3.3 Walt Disney's Method

Robert Dilts studied the creative process of Walt Disney [29]. He realised that Walt Disney had three distinct phases in his creative process, in particular the *Dreamer*, *Realist* and *Critic* phase. They are specified in the following way.

Dreamer The *dreamer* phase represents a visionary understanding of the project and is responsible for the specification of possible aims. It does not criticise the collected ideas and is therefore similar to the yellow hat of de Bono's six thinking hats. The prohibition of any boundaries or restrictions enables this mode to think about strange and possibly non-logic thoughts and solutions. It supports the creation of creative ideas that can then be processed in the other two phases.

Realist The *realist* phase is responsible for the realisation of ideas. It tries to identify boundaries and steps that need to be performed in order to achieve the previously specified goals. These steps should be practical and most important executable under realistic circumstances. The realist phase considers existing ideas as well as solutions and identifies possible evaluations.

Critic The *critic* phase represents a pessimistic or critical point of view. It reveals flaws, communicates advancements that need to be considered and also addresses possible risks and chances of the proposed ideas.

It is not mandatory that these three phases or viewpoints are represented by the same person. Similar to the two previously described techniques for creativity enhancement, it is also possible that they are distributed to the members of a team.

The three presented techniques for the enhancement of creativity (Creative Problem Solving (CPS), Six Thinking Hats, Walt Disney's Method) are probably very popular, but still only a subset from a large variety of strategies. They usually try to distinguish between different viewpoints, which are then used for the stimulation of new ideas and the guidance of the thinking process. Their ultimate aim is the identification of creative problem solutions. Some of the models also share similar components, such as the yellow thinking hat (de Bono) and the dreamer phase (Dilts). However, none of them directly integrates evaluation techniques or any kind of guidance for the production of creative results, which might also be very difficult, as the strategies are not bound to any domain or field.

2.4 Computational Creativity

Computers are able to efficiently produce a variety of new outcomes by combining existing input. The result may be novel but not always creative with respect to Margaret Boden's first way of creativity [12]. Links between two concepts need to be sensible, which can only be evaluated with the help of additional knowledge. This in turn means that computers cannot generate creative outcomes in the described way without an awareness of the environment. Existing systems are to a certain degree able to explore and transform a conceptual space [101, 102], as it was previously specified in Boden's model. For instance, AARON [67] is a software that creates original artistic images. It is able to use a particular style for the drawing of distinct paintings, which is equivalent to the exploration of a conceptual space. New styles cannot be learned automatically and need to be translated into source code and integrated manually. An example of a self transforming program is Automated Mathematician [63], which discovers mathematical concepts and relationships. It analysis and transforms short LISP programs and links them, for example, by nesting their functions inside each other.

Saunder and Gero's framework for computational creativity [82] adapts Liu's Dual Generate-and-Test Model of Creativity [65] that itself is partly based on Csikszentmihalyi's systems view of creativity. The model from Liu includes two so called generate-and-test loops, one for the individual and one for the socio-cultural level. The individual level contains creative thinking, problem finding, solution generation and a creativity evaluation test. The socio-cultural level converts the interactions of Csikszentmihalyi's model into computationally processable elements. Figure 2.3 depicts Liu's Dual Generate-and-Test Model, according to [65].

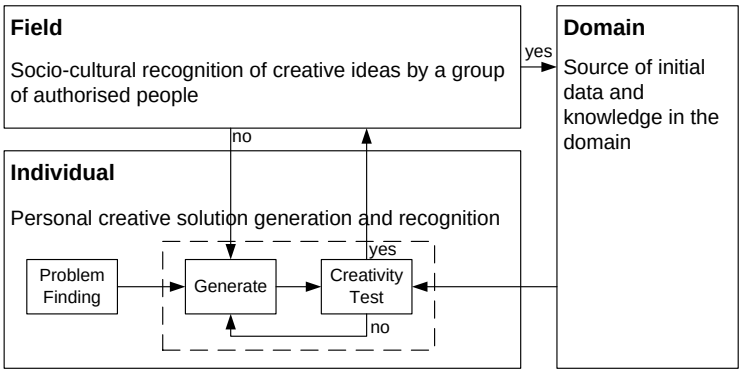


Figure 2.3: Liu's Dual Generate-and-Test Model of Creativity

It can be observed that the three entities *field, individual* and *domain* in Csikszentmihalyi's model are used for the description of an evaluation strategy. The individual starts with the problem finding and continues with the generation of a new solution, which is at first evaluated based on the personal creativity test. If the solution passes this test, it is transported to the field, which represents a group of authorised experts. A successful evaluation from the field leads to a communication of the solution to the domain. It is then accessible for all individuals. An unsuccessful evaluation will return the solution to the individual, who can then perform modifications.

The model of Saunder and Gero addresses the presented socio-cultural creativity test to map the behaviour of creative societies. Their main approach is the use and combination of several individual's creativity tests. The socio-cultural creativity test can be realised by permitting communication of evaluations between individuals. An example for this is the interaction of individuals A and B as depicted in Figure 2.4 (according to [82]). A forwards a piece that passed its creativity evaluation and is therefore considered to be creative to individual B. This process is illustrated with the arrow going from the "Creativity Test" of Individual A to Individual B (labelled with "yes"). B then evaluates this item itself and sends the results back to A. This is displayed with the arrow that is labelled with "yes" and going from B's "Creativity Test" to Individual A. B additionally adds the piece to the

domain if it is creative. This means that individual B is able to reward A and therefore influence its creativity test. This can for instance affect the creation of future artefacts by A. On the other hand, A can also influence the behaviour of B since the evaluation of creativity involves an evaluation of novelty. By a reduction of novelty, A is able to alter B's evaluation process and change its notion of creativity.

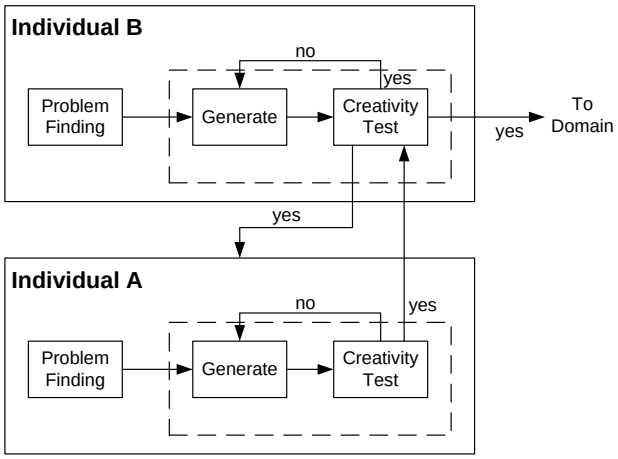


Figure 2.4: Communication of Evaluations between Individuals

Each individual in Saunder's and Gero's model is implemented as an agent [83][37] that is able to communicate with other agents and the domain. This interaction represents socio-cultural behaviour in a creative society as the collective of individual evaluations. The group of agents is similar to the field that was introduced in Csikszentmihalyi's systems view of creativity. Saunder's and Gero's system is of dynamic nature and the notion of creativity can change, if the domain is modified or new individuals join or leave the socio-cultural network.

The models for computational creativity that were presented can be utilised for the construction of computer programs which produce creative artefacts. However, this is a very difficult task and especially the judgement about creativity by the field in terms of novelty and usefulness can cause problems. For the support of the human creative process, it is furthermore irrelevant that a computational system is able to construct these outcomes itself. It should rather be able to analyse the creative process and assist the user during the creation of an artefact.

2.5 Computational Creativity Support

Creativity support [88] and especially creativity support tools [87] are research topics that gained much interest during the last years. New technologies and increasing computational power together with the growth of the internet and social communities opened a research area with high potential and risks. Shneiderman et al. [31] mentions that it is necessary to study creativity for the production of reliable outcomes in order to enhance an individual's creativity as well as a group's creative potential. However, there is also much scepsis, justified by the believe that creativity is beyond every scientific process or innate.

Shneiderman et al. [31] developed a set of design principles and patterns for the development of creativity support tools. They focus on "composition tools", which are computer systems for the generation, modification, interaction and sharing of media like texts, images or music. The principles are vague and can be stated more precisely or extended for particular cases. They concentrate on easy exploration, rapid experimentation and fortuitous combinations to support creative thinking and innovation. The following 12 principles have been specified.

1. Support exploration
2. Low threshold, high ceiling, and wide walls
3. Support many paths and many styles
4. Support collaboration
5. Support open interchange
6. Make it as simple as possible - and maybe even simpler
7. Choose black boxes carefully
8. Invent things that you would want to use yourself
9. Balance user suggestions with observations and participatory processes
10. Iterate, iterate - then iterate again
11. Design for designers
12. Evaluate your tools

The illustrated principles cover a variety of thoughts. The first one is to try many different alternatives through exploration, as the outcome cannot be known in advance. According to the second point, tools need to support an easy entry and the possibility for investigation together with the ability to work on sophisticated projects. They should provide an immediate confidence that the project can succeed and should also be easily understandable. Collaboration is very important and is mentioned in the fourth and fifth principle. Creative work can be performed in interdisciplinary teams and tools should enable a contribution of knowledge from different fields. Systems should be as transparent as possible (seventh principle). The identification of the lowest manipulable elements needs to be defined in advance to guarantee a clear and convenient artefact creation. User interaction needs to be integrated thoughtful and balanced. Feedback is important on the one hand, but too much integration and suggestions will hinder the development process. The last three principles point out that the process is always unfinished and improvements are possible at any stage. Creativity support tools should be flexible and allow others to design, create and invent. It is also necessary to evaluate the tools themselves for improvements and the development of future features.

The design principles are vague and leave room for interpretation. A particular and essential problem of interdisciplinary collaboration that needs to be addressed is the synchronisation of artefacts across distances. Synchronous or asynchronous communication are two examples. Evaluation mechanisms for creativity support tools should be defined during the design process. They might result in improvements and the development of future features. Socio-technical environments for the enhancement of creativity are other important factors that need to be considered.

Collaboratively created projects ought to be saved during their creation and should be retraceable to support the study of creative processes. A version control system can be considered for this task. As the user represents the central role in a creative process, tools must offer facilities to react on feedback and process it adequately. An environment that seamlessly integrates multidisciplinary tools would satisfy at least some of the design principles mentioned above.

The issue of collaboration is addressed by Fischer [34]. He introduces two types of communities, namely Communities of Practice (CoP) and Communities of Interest (CoI). A CoP is a homogeneous community that includes collaborators who work together for the accomplishment of a similar task. Examples are software developers, architects or research groups. The internal vocabulary for their communication is problem and area specific, which allows for accurate problem specifications on the one hand but represents a high

boundary for outsiders on the other hand. A CoI is defined as the stakeholders of different CoP, who work together in order to solve a particular problem of common concern. An example is a team of software designers, programmers and marketing specialists, all interested in the development of a software product. A CoI has multiple centres of knowledge and more potential to be innovative and creative than CoPs. It provides a shared understanding of the problem, but lacks of a consistent communication, due to several vocabularies.

The communication problems of collaborators can be overcome by boundary objects. They represent externalisations of ideas that are used to communicate and facilitate shared understandings across spatial, temporal, conceptual, or technological gaps [34]. Boundary objects are used to establish a shared context for communication and help to identify breakdowns. Examples are documents, rules or unspoken norms. They are evolving, develop audibility over time and are conversation elements for the communication of knowledge instead of simple knowledge containers. Fischer justifies their importance by explaining that boundaries are the places where knowledge is being produced, unorthodox and innovative solutions are identified and the unexpected is converted into the expected. If they should be used intuitively, it is necessary to keep them clear and structured to also support a convenient construction.

The presented approaches for computational creativity support leave room for interpretation. Especially Shneiderman's design principles need to be implemented according to the domain requirements. However, it is probably impossible to define a fixed set of rules for the construction of creativity support tools which guarantees beneficial outcomes. It might also be necessary to integrate feedback into the development process and let the users decide about missing features. This enables a steady modification and adaptation to the personal needs and might support the successive creation of a convenient environment. The Communities of Practice (CoP) and Communities of Interest (CoI) illustrated that collaboration might be an important factor for the enhancement of creativity. It is essential to establish a communication between the different team members and allow them to collaboratively modify an artefact or boundary object. However, the influence of collaboration on creativity is still difficult to determine.

2.6 Data Mining

Creativity and creative processes provide a large variety of data and information that can be studied. The identification of patterns and frequent information is important for

a better understanding of these processes. It enables the analyser to provide useful information for the assistance of creativity. The area of Data Mining [89][66][58] describes techniques that realise the previously mentioned tasks and can be adapted for creative processes. They were mainly developed to analyse large amounts of data, which require profound techniques for the extraction of subjectively important information. One example is the Quest project, which was started by IBM in 1996 [4] in order to build a system for data-intensive decision support. It should accelerate and improve the decision process. As the data is usually stored in databases, Data Mining is often used synonymously with the terms Knowledge Discovery in Databases (KDD) or Knowledge Mining [45]. In [84] James G. Shanaham specifies knowledge discovery in the following way.

> Knowledge discovery is commonly viewed as the non-trivial general process of discovering valid, novel, understandable, and ultimately useful knowledge about an application domain from observation data and background knowledge, in which the discovered knowledge is implicit or previously unknown.

The field of knowledge discovery has a long history and it developed from techniques based on clean and statistical data to techniques for exploitation of multidisciplinary alternatives, which are able to deal with imperfect data. A major part of this change is grounded on the perception that humans are mandatory to solve practical real world problems. These complex tasks are for example image detection, Natural Language Processing (NLP) or the creation of an artefact in general, where uncertainty and background knowledge are key elements in the process of pattern recognition.

Data mining was developed from studies in the areas of computing, marketing and statistics [39]. For the field of computing, it is placed in the area of machine learning [109][50], as it has a lot in common with pattern recognition. Machine learning includes the detection of patterns in order to insert and categorise data as well as derive previously unknown information. The essential idea of statistics is the analysis of data for the extraction of numerical information. Statistical methods are usually developed in close relation to the data that is going to to be studied. Even if this enhances the efficiency on the one hand, it makes the process static and limits the ability to adapt to other data sets on the other hand. Data mining utilises parts of the described techniques and develops solutions to overcome their weaknesses.

Most of the main data mining techniques were developed in computer science rather than in statistics. They differ in their parameters and their methodologies of the data exploration. Classification is a technique that tries to recognize new patterns or insert

them into predefined categories. Common patterns of a specific class can be tested against the new data set to reveal previously unknown information and knowledge. Another technique is the mining for association rules, which is based on the representation of knowledge as rules. It explores data for the identification of links between items like it is commonly used in market basket analysis to reveal relationships between products in customer purchases. Clustering techniques [56][18] are useful for data visualisation and explorative data analysis. They build clusters of similar data sets and try to reveal previously unknown similarities [78]. The human explorer is considered to be a major part of this technique. Other techniques try to predict values of certain variables by exploring patterns in the data sets. These patterns are then analysed with the help of mathematical strategies in order to derive new values. The process is mainly used for numerical data sets.

2.6.1 Data Mining Process

The data mining process starts with data and ends with previously unknown patterns and knowledge. It can be broken down into a number of steps that are performed to realise this conversion process. Figure 2.5 depicts 5 distinct activities, according to [32]. Raw data is selected and pre-processed, before it is further transformed and analysed to reveal patterns that can finally be interpreted for the creation of new knowledge.

Selection The vast amount of raw data needs to be preselected for the succeeding steps in order to decrease its quantity. It is necessary to define project aims, to be able to select data based on features, observations or background knowledge and possibly highlight data deficiencies. Data understanding and background knowledge are crucial requirements for the selection phase. Irrelevant data sets can be eliminated, if factors like quality, nature and relation to the previously defined aim were specified. Data selection is mostly automated and its result represents a subset of the overall data set that fulfils necessary requirements and supports following analysis steps.

Preprocessing The preselected data is verified during the preprocessing step to identify inappropriate values and necessary modifications. The verification may discover missing data as a result of non or wrongly measured values or instrument malfunctions. Missing values can be completed for example by human input, averaged values or fuzzy set values. It is necessary to chose this techniques with respect to the application and project aims. The result of the data preprocessing step is a structured and complete data set.

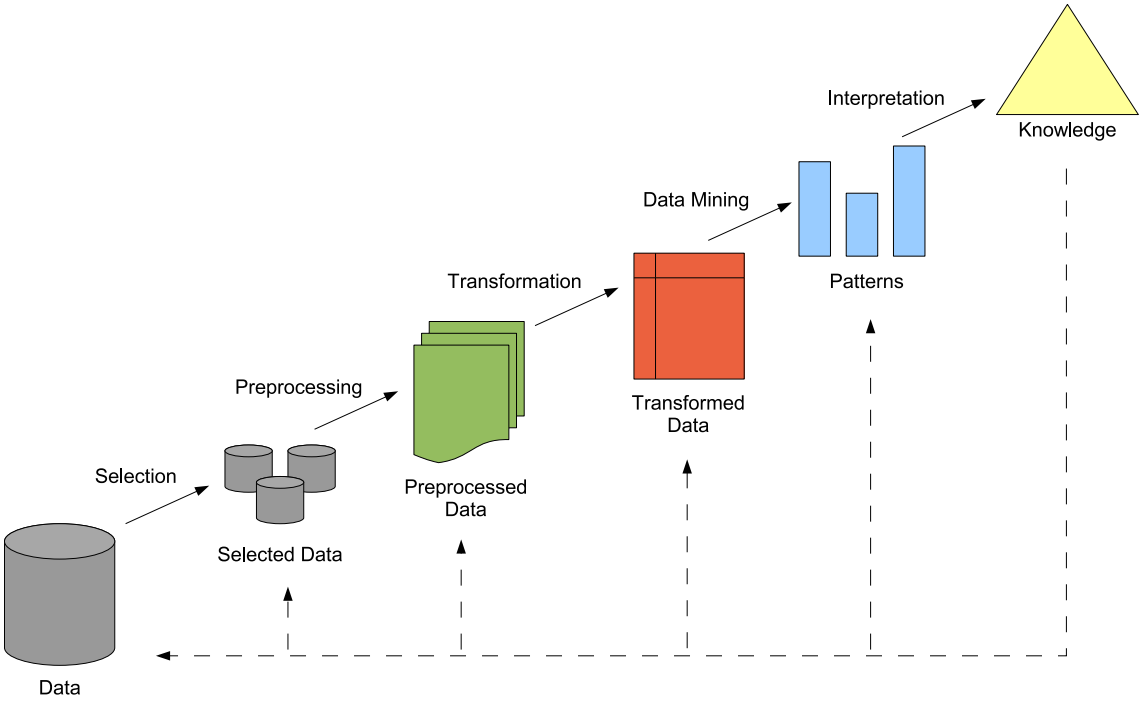


Figure 2.5: Data Mining Process

Transformation The transformation step is executed after the preprocessing to create a descriptive data model that allows for computer based processing. This model needs to suit to the application. Furthermore, there are general requirements every model needs to satisfy. It should be parsimonious to decrease the computational costs and consider only necessary data values. When the data set is decreased, it is necessary to keep the content as similar as possible in order to to not distort possible results of an analysis. Data transformation is an important step, as succeeding algorithms may only be capable of handling normalised data.

Data Mining The Data Mining step is responsible for the recognition and extraction of patterns from the transformed data set. A data mining technique (e.g. clustering) fitting best to the application requirements needs to be chosen. This is of course one of the first steps, since it affects the model construction and the initial steps of the data mining process. One possibility for the implementation of pattern recognition is a visual data analysis, which allows for the extraction of structural information. An evaluation process for the identified results is needed for the development of further enhancements and to guide the knowledge discovery process. This ensures a dependable validation of the revealed patterns and supports the learning process.

Interpretation The interpretation is the final step of the data mining process. The results and extracted patterns are interpreted to create new knowledge. Human interaction is probably required to produce reliable results, as fixed strategies for automatic data interpretation might be difficult to specify. This phase can therefore be influenced by subjective decisions of the observers.

The arrows in Figure 2.5 illustrate that the results of a single step can be affected by every other step of the overall process. For instance, if the accuracy of the outcome is not sufficient, it might be traced back to a partly ill defined model or a non fitting data selection algorithm, which can then be corrected in the transformation phase or the selection step, respectively.

The data mining process finishes with the creation of knowledge, as depicted at the top right corner in Figure 2.5. This knowledge can be stored in different forms and especially the area of knowledge representation [8, 24] tries to build models that transform domain specific knowledge into a computer processable format. Many representation of knowledge have been developed but only few have proven to be advantageous and it quickly emerged that a generally applicable model cannot be designed. A lot of the knowledge representations, for example in the fields of Artificial Intelligence (AI) or Natural Language Processing (NLP) are based on the assumption that a problem can be mapped onto entities and their relations. One example is the words of a language, which are linked to build sentences or even larger structures.

Other knowledge representations include rule-based systems, semantic nets, or ontologies. One possibility for the description of knowledge in a rule based manner is the use of logical operators (AND, OR). Terms are linked with these operators to express a formula that can be processed with the help of computer systems. An example is the field of market basket analysis, where baskets of customers are analysed for the extraction of knowledge about product relationships and buying habits. Certain products might be bought together frequently, like bread and cheese. To generate this information, it is necessary to analyse a variety of market baskets. The following syntax illustrates a general structure for the representation of knowledge as rules.

```
IF <condition> THEN <action>

IF <premise> THEN <conclusion>
```

It is illustrated that rules can be expressed in the form of if - then statements. If the *condition* is satisfied, then the *action* is executed. Similarly, if the *premise* is complied, the *conclusion* can be drawn. For the market basket analysis described above this means that if a costumer buys bread *(premise)*, this person also buys cheese *(conclusion)*.

The production rules are a very common way for the description of knowledge in the data mining process [3]. In knowledge based systems, inference engines are used to interpret these rules and execute them depending on the information that is supplied [90]. One method is forward chaining, which takes facts as input and draws conclusion based on satisfied conditions. The corresponding actions are then performed. Another method is backward chaining, which works in the opposite direction. It starts with actions and tries to identify conditions based on additional user input. Both methods do not explore new conditions or actions, they instead try to validate information.

One advantage of the knowledge representation as production rules is the possibility to split a knowledge base into several independent pieces. Rules are modular and describe a relatively small amount of knowledge. They can be stored separately and then be composed into chains if necessary. It is easily possible to extend the knowledge base by adding new rules or reducing its complexity by removing irrelevant ones. Rule based systems are usually used for the validation of input values, but not to derive new conclusions. For instance, they might help a medical doctor in diagnosing a particular disease, but only if the symptoms are covered in the knowledge base.

The described knowledge representation illustrate an overview of a very popular model. Similar to the rule-based approach, the creative process or a creative behaviour can be represented as a chain of actions. However, its structure is far more complex and therefore needs a further elaborated model for its representation. It might also be necessary to consider and maybe reuse existing information.

2.6.2 Mining Sequential Data

Sequential data mining is a sub-area of data mining, which describes techniques for the identification of frequent patterns in sequential data sets. It is for example widely used in biology in order to search for specific sequences in gene and protein structures, like the DNA. Sequence mining is relevant for the proposed research, as the creative process of a creator also contains sequences of the actions, which are called behaviours. Extracting frequency related information from this process is a main part of this thesis.

An initial and influencing work in the area of frequent sequence mining was introduced by Agrawal and Skrikant [5][6]. They described the idea of mining large amounts of sequential data, in particular transactions. One example is the analysis of traversal patterns that were recorded during the visits of web pages [17]. The problem of frequent sequence mining as explained by Agrawal and Skrikant and others [110] can be specified in the following general way. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. An itemset X is a set of items of I. A sequence $s =< t_1, t_2, \ldots, t_n >$ is a ordered set of transactions, where each transaction t_i is an itemset. The number of items contained in a sequence s defines the length of it. The length is denoted by $|s|$. Given two sequences $s_1 = \langle a_1, a_2, \ldots, a_n \rangle$ and $s_2 = \langle b_1, b_2, \ldots, b_l \rangle$, then s_1 contains s_2 if there exist integers $j_1, j_2, \ldots, j_l$, so that $l \leq j_1 < j_2 < \cdots < j_l \leq n$ and $b_1 \subseteq a_{j_1}, b_2 \subseteq a_{j_2}, \ldots, b_l \subseteq a_{j_l}$. A sequence is maximal if it is not a sub-sequence of any other sequence.

Given a set D of sequences, the *support count* of a sequence s is defined as the number of sequences in D that contain s. The *support count* is denoted by δ_D^s. The fraction of sequences in D that contain s is called the *support* of s. If the support of s is higher than or equal to a user defined threshold, then s is frequent. The aim is to find all maximal frequent sequences in D.

As the problem definition illustrates, the aim of frequent sequence mining is the identification of high frequent patterns in a sequence database. Every sequence which contains a pattern at least a single time is considered. A number of approaches that solve this issue have been developed. A strategy that was introduced by Agrawal and Skrikant is the apriori approach [6]. It describes one of the standard techniques, which is realised as a bottom-up approach, starting with a set of frequent items (itemset) of length 1 and steadily extending the length in each step. Non-frequent items are dismissed after each step. The apriori algorithm terminates, if the itemset cannot be extended any more. Several improvements of this approach, especially to decrease the number of database scans have been introduced [30].

Another approach that tries to overcome the described problems is the frequent pattern tree [46]. It does not construct the large sets of sequences, which occur during the apriori approach and also stores the frequent patterns in a compressed tree structure. This saves space and allows for an efficient pattern mining technique. The tree saves identical prefixes of frequent patterns only once and lets behaviours branch afterwards. The construction of the frequent pattern tree needs two database scans. The first one identifies the set of frequent items and the second one inserts them into the tree. Additional post-processing in the form of tree mining is required for the extraction of all maximal frequent patterns,

like described in [46]. An advantage of the frequent pattern tree is the reduced number of database scans compared to the apriori strategy. Suffix trees [43] are also sometimes used for data mining and searching [111]. They represent another tree structure that is able to store sequences in a compressed form.

The described approaches for frequent sequences mining are only capable of identifying the most frequent sequences. These are the ones with a frequency higher than the predefined threshold. In contrast to this, the frequent information extraction approach that is presented in this research is interested in a large variety of frequent information. This also includes less frequent behaviours, or even the least frequent ones. However, the described techniques are highly adapted for the extraction of the most frequent sequences. Especially the simplified understanding of frequency is not applicable for the creative process that is explained in this research. The succeeding chapters describe very detailed that a creativity map usually represents a complex structure and also differs from the previously mentioned data source, namely a sequence database. Even if it contains creative behaviours in the form of sequences, they are not stored separately and may instead share common prefixes. The described general techniques are therefore not usable as such.

2.6.3 Similarity of Data

In order to identify patterns, it is essential to specify a metric that is used for the comparison of data. With respect to the creative process, it needs to allow for the identification of similar stages that the creator went through. This enables to search for behaviours and construct branches inside the process. It might also help to predict behaviours of unknown, but similar objects. The creative process itself is domain independent, because every person is able to be creative. A similarity metric needs to consider this as a requirement and must not be bound to a single field.

One of the most common measurements for similarity is the geometric metric, as exemplified by the Multidimensional Scaling (MDS) [85][86]. Each feature of an object describes one dimension in a feature space, so that the whole object is represented by exactly one point. It is assumed that points which are not far from each other are more similar than distant ones. The most common way to calculate the distance d between two objects A and B is the Euclidean distance, which is specified as $d(A, B) = \sqrt{(a_i - b_i)^2}$.

Another similarity measurement is the *Feature Contrast Model* [94]. It assumes that objects can be represented based on their features. Tversky distinguishes between features

that both objects share and ones that only one of them possesses. Common features increase and individual ones decrease the similarity. This is formalised as $S(A, B) = \alpha * f(A \cap B) - \beta * f(A - B) - \gamma * f(B - A)$. In this definition, $A \cap B$ represents the common features of the objects A and B, $A - B$ describes the unique features of object A and $B - A$ are the unique features of object B. The variables α, β, γ are non negative parameters that determine the weight of these components. They provide adaptability and can be freely chosen, depending on the context.

Transformation Similarity [44] is another model, which defines the similarity of two objects A and B as the number of transformation steps it takes to get from A to B. The particular transformations needs to be defined in advance. It is important to specify them according to the objects and required study, as they can have a major influence on the model.

The similarity measurements described above present a small overview of some approaches. It was mentioned before that the similarity of objects in the creative process needs to be adaptable to the domain of interest. Especially for the construction of creativity maps, it is necessary to identify similar stages by comparing data that was recorded during the creative process. This data varies between domains, creators and projects. A writer needs a different metric for the comparison of documents than a software engineer or musician. It can even vary for a single domain or creator. The analyser should be able to specify the metric that is used for the comparison, to guarantee an adaptable similarity determination. This can also include one of the previously described measurements.

2.7 Summary

This chapter reviewed, summarised and commented on related work in the fields of creativity, creativity support and data mining. It explained the problem of identifying a generally accepted definition of creativity. However, it was also emphasised that creativity is usually related to new and useful ideas or other outcomes. Three popular models for the description of creativity and the creative process were reviewed. Similarities between them were shown, like Boden's H-creativity, which is to some extend related to the field component in Csikszentmihalyi's Systems View of Creativity. It was pointed out that these models consider creativity as a kind of undirected and unconscious component. A number of techniques for the practical enhancement of creativity were discussed. They mainly try to identify different viewpoints of a problem, which are then used for the stimulation of new ideas during group discussions. The techniques are therefore mainly applicable for groups or organisations. A number of examples from the field of computational creativity were

explained and it was shown how some of the previously presented models are mapped into a computationally processable form. However, it was also mentioned that computational creativity tries to build programs which are creative themselves. This is not the main aim of the proposed research. The chapter furthermore reviewed some of the latest research in the area of computational creativity support and creativity support tools. A variety of requirements for their construction, which were developed during a workshop from Ben Shneiderman have been discussed. It was illustrated that all of them leave room for individual interpretation. The last part of the chapter covered strategies and techniques in the field of data mining. The overall process as well as each individual step were described and some main approaches in the field of frequent sequence mining were discussed. It was explained that these techniques are only interested in the most frequent sequences and also use a rather simplified definition of frequency. The importance of similarity for the identification of patterns was emphasised together with a number of metrics for its determination. It was explained that the creative process, as described in this thesis, requires a domain independent and exchangeable similarity metric in order to identify the stages that a creator revisited.

Chapter 3

Preliminaries

Objectives

- Introduce three axioms of creativity that describe the fundamental understanding of creativity and the creative process.
- Present an informal description of the creativity mapping model and creativity maps.
- Describe the hopping phenomenon for creative processes.
- Explain the ability to represent collaboration with creativity maps.

3.1 Introduction

This chapter explains the preliminaries of the research. It introduces three axioms of creativity, which describe the understanding of creativity and are essential for the introduction of the creativity mapping model. These premises emphasise that a product as well as a domain are indispensable for the identification of creativity. The model is then explained informally, specifying a creation zone that records the observed creative process and explaining all of its components. The hopping phenomenon and the ability to represent collaboration are presented towards the end of this chapter.

3.2 Axioms of Creativity

The perception of creativity in the presented research is based on three premises, which build the necessary foundation for the creativity mapping model that is introduced in this chapter. According to [107], these axioms are specified as follows.

A0. Creativity is identified only by its product.

The creativity that is involved in the creation of an artefact, for instance a painting, piece of software or written document can only be captured by the artefact itself. Without this artefact, it is impossible to decide about the creativity of its creator. Particularly the observation of its construction allows to study the effects of creativity. A concrete and tangible artefact is therefore a necessary precondition for the identification of creativity.

A1. The value of creativity is determined only by the society that receives it.

The value of an artefact can only be determined by the people who belong to the same domain as the creation. These are the experts with sufficient background knowledge, who are able to decide about it and give feedback to the creator. The committee of the Nobel Prize for example decide about the most significant developments and decide that an innovation A in physics involved more creative thinking than an innovation B or any other discovery. This might be the case, because the area of research is very new, the received artefact seems to be a milestone in the current research or as a result of other reasons. However, despite the used metric, it is always the society that receives the artefact, which determines its value. This society shows similarities to the field component in Csikszentmihalyi's Systems View of Creativity [21].

A2. Creativity is an emergent phenomenon.

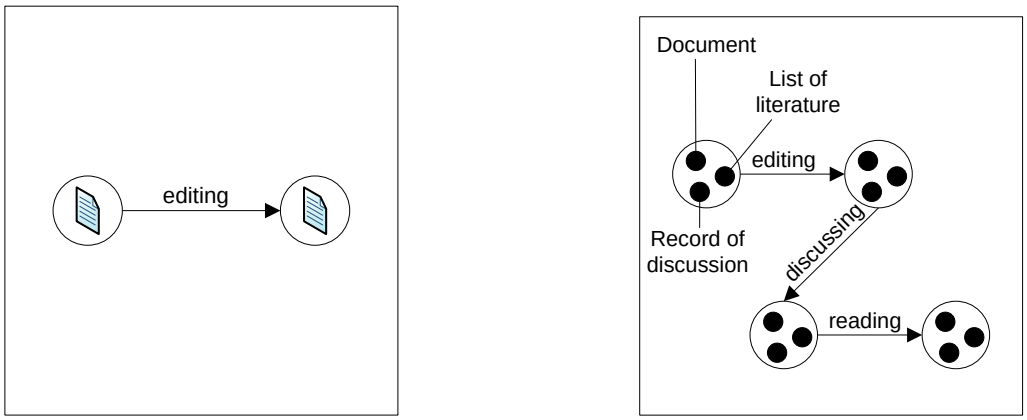
The creative process describes a complex system, which is composed of many interconnected parts (creation steps). The emergence of a new state and a new creative behaviour occurs abruptly as a specific threshold is passed. For instance, altering the brushes will let a painter use another sequence of colours. This is unexpected and the observed emergent behaviour can lead to an insight, resulting in a creative painting. However, emergence cannot be planned or predicted and emergent behaviours occur abruptly, spontaneously and suddenly.

3.3 Creativity Mapping Model

The creativity mapping model that will be introduced focusses on the description and mapping of the creative process. This process is perceived as the steps that are involved in the construction of an artefact, which describes the essential part for the determination of creativity, as mentioned before. It is therefore necessary for the model to record and map all actions that were performed in the creative process. A *generalised transition system* is used for its representation. This transition system differs from the traditional one [38], which only uses a fixed relation between states with a set of actions to label transitions. The system for the creative process is a generalisation and enables multiple relations between states, each associated with its own set of actions. An example is the time transition, which always runs in parallel with any other relation. This section introduces the model and its components in an informal way and describes its utilisation for the mapping of creative processes [107][15]. It is important to note that the model itself does not evaluate the creativity of an artefact, as this is assumed to be realised by the domain that receives it.

The origin is a zone of creation, where the observed actions (e.g. editing, thinking, etc.) and behaviours of a creator are recorded. It is of course only possible to capture observable actions. The zone is represented as a rectangle and recorded activities are mapped into states and transitions. A state can informally be described as an entity that saves all information with respect to the creation process of an artefact, including its current version. A set of states can then be used as a history for the tracking of changes. Transitions represent the actions of a creator [26] and connect two states to illustrate the modification that happened between them. The meaning of states and transitions for the mapping of the creative process can be illustrated with an example of a writer who creates a story. This writer will continuously work on the document (the artefact) and edit it (e.g. inserting, deleting, changing the format) for a certain time period. Whenever this happens, the document is modified and a new version is created. To capture these actions and changes, the artefact is encoded inside the states. The editing action itself is transformed into a directed transition that connects two states and changes the document from one version to the other. Figure 3.1(a) depicts the described scenario. However, editing is not the only action that the writer performs. Meetings with colleagues and discussions about the story or other topics and also brainstorming sessions [74] take place, which then create new ideas for the plot and point out sections for revision. Each state now additionally stores the information about this discussions and whenever a meeting happens, a new discussion transition will be added to the creation zone. The writer also performs a

reading action, together with the information about the read literature being stored inside the states. Figure 3.1(b) depicts a creative process with the three previously mentioned actions (editing, discussing, reading). The dots inside each state represent the different types of information, in particular the document, record of discussion and list of read literature.



(a) Creative Process with one Editing Action

(b) Creative Process with different Actions

Figure 3.1: Examples of Creative Processes

The different types of information, which are encoded inside the state (artefact, discussion, literature) are the *viewpoints* of the creative process and represent its different perspectives [107]. For instance, the artefact viewpoint stores the current document, a list of read literature is saved in the literature viewpoint and the discussion viewpoint contains records of discussions. Each viewpoint has a set of *actions* that are performed by a creator. For instance, a writer performs an *editing* action that belongs to the *artefact* viewpoint. An action in turn modifies the viewpoint that it is part of. These dynamics and changes of the different viewpoints describe the creative process. The model furthermore distinguishes between external and internal viewpoints. External viewpoints are part of the creative process' environment [77] and may include social, political or psychological forces. A deadline or the restriction of materials are only two examples [103]. Internal viewpoints in contrast are directly involved in the creative process, like the described artefact viewpoint. To better distinguish different internal viewpoints, their transitions can be coloured accordingly, as depicted in Figure 3.2. It illustrates the three viewpoints artefact, discussion and literature coloured in red, green and blue, respectively. If the name of the viewpoint is relevant, it can be additionally displayed underneath a transition.

The figure shows the creative process as a linear sequence of actions. However, a creator probably returns to a previous state during the creation of an artefact. For example, a

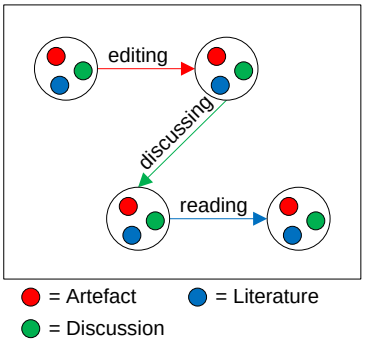
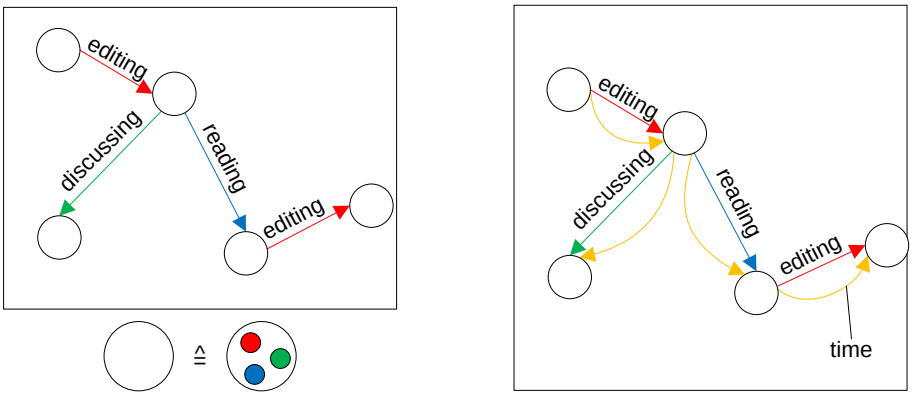


Figure 3.2: Creative Process with Coloured Viewpoints

writer who is dissatisfied with the newly written paragraph can revert to an older version of the document. When this happens, the current state is left and the creative process continues from the state that the creator moved to. A new branch is created in the previously sequential structure of the creative process. As this scenario probably occurs many times, the structure changes to a *Creativity Map*. A simple example is depicted in Figure 3.3(a). The viewpoints (coloured dots) are hidden in this figure to simplify the illustration of the states. It can be observed that a creativity map does not possess a distinct final state, because every state may be considered by the creator to be final in the creation life cycle of an artefact. The creativity map in Figure 3.3(a) contains a number of possible final states. In contrast, each map has exactly one start or root state where the creative process begins. Additionally to the presented transitions, it can be inferred that all actions consume time, which is represented by time transitions (orange) as depicted in Figure 3.3(b). These are additional transitions, which represent the amount of time that was spent for the activity. They are usually not displayed in order to keep the creativity map clear. The tool support, which is discussed in Chapter 7, shows that time transitions are realised with timestamps, which are stored in the creativity map states. Each of them illustrates the moment, when the action of a state's outgoing transition begins. The consumed time of an activity is therefore described by the difference between the end and start state of the corresponding transition. If multiple outgoing transition exists, the state stores a timestamp for each of them.

A creativity map normally grows to a more complex structure than the example maps that are described above illustrate. Especially if the creative process is captured for a long time period, the creator is able to perform a larger quantity of actions. Projects that last for some weeks or months will ultimately lead to more data that is being captured. Each map allows for several types of analyses, such as clustering, classification or optimisation. By comparing different creativity maps of the same creator, it might also be possible



(a) Simple Creativity Map

(b) Simple Creativity Map with Time Transitions

Figure 3.3: Examples of Creativity Maps

to explore characteristic patterns. These can additionally be studied across domains to analyse how creativity crosses the boundaries of disciplines.

3.3.1 Hopping

The creator is not necessarily bound to a single creation zone and especially the interaction with other creators and collaboration allow for the alternation between different zones. This phenomenon of the creative process, which enlarges the creator's state space and accessible information is called *hopping* [107]. Interdisciplinary teams can cross boundaries by hopping between different creation zones. Their domains will be linked and the previously separated knowledge will be connected and maybe enhanced. Figure 3.4 depicts an example of the hopping phenomenon, which is represented with the help of dotted arrows.

The creator started in the creation zone at the left-hand side and performed two reading activities. He then "hopped" into the creation zone that is illustrated on the right-hand side, edited a document followed by a discussion with some colleagues. Another hopping back into the left zone followed. After a discussing action, the creator returned to a previous state and performed discussing and editing activities. He then "hopped" into the right zone and read an article. The result of this process are the two creativity maps, which are depicted above.

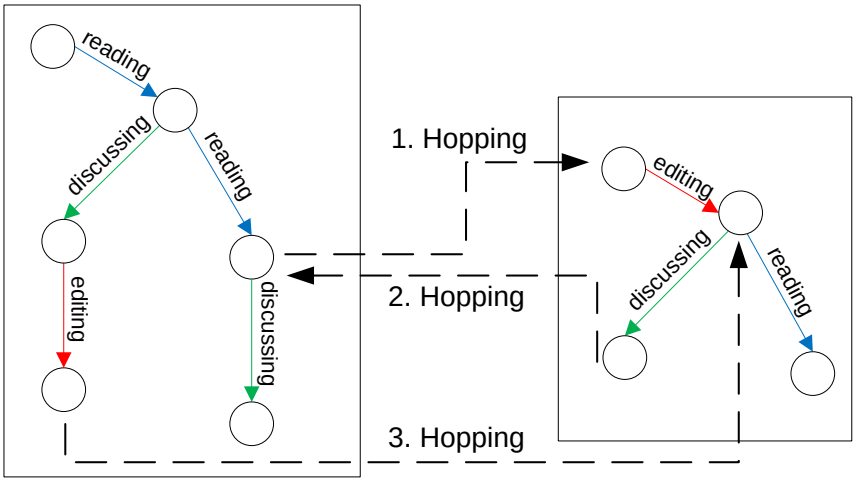


Figure 3.4: Hopping between Different Domains

3.3.2 Collaboration

The creativity mapping model is also capable of representing collaboration [108]. It is modelled with one creation zone that is divided into several sub zones. Each of these sub-zones is used for the recording of a collaborator's creative process, similar to the observation of a single process. As collaborators work on the same artefact, they share the state of the creation, which therefore needs to be kept consistent in all zones. Collaboration is not restricted to this single viewpoint and creators might share other information as well. Figure 3.5 depicts a collaboration between two creators, where the individual creation zones are separated by a dotted line.

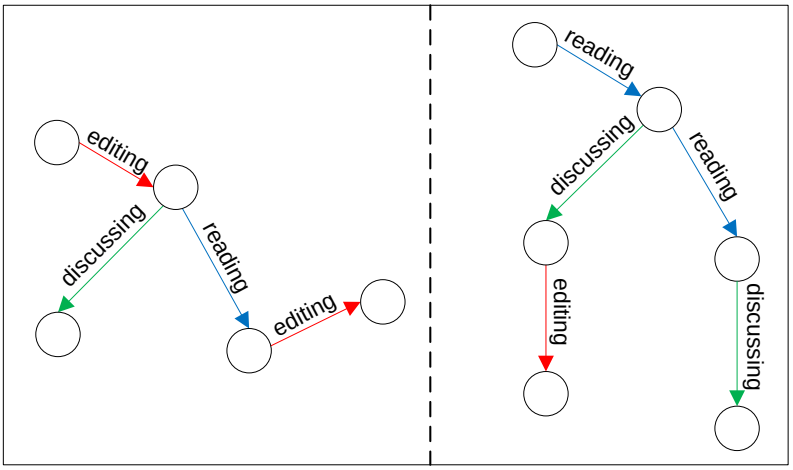


Figure 3.5: Collaboration

The figure illustrates two small creativity maps in each of the sub-zones. Collaborators are able to hop between them if necessary, as explained in the example above. This enlargement of a creator's state space includes additional information and knowledge that can be utilised for the following steps in the creative process. For instance, an interdisciplinary team of creators can share knowledge and support the creative process by enabling the hopping between domains.

3.4 Summary

This chapter introduced creativity maps as a model for the mapping of creative processes. Three axioms of creativity were introduced, which are fundamental for the understanding of creativity as used in the proposed research. The informal description of creativity maps explained the creation zone and the transition system that is used to record the performed actions of a creator. It was illustrated how collaboration and the phenomenon of hopping create possibilities for the interaction of several persons.

Chapter 4

Creativity Maps and Behaviours

Objectives

- Describe the creation process of creativity maps.
- Present a classification of creativity maps into a hierarchical structure.
- Specify data, information and knowledge with respect to creativity maps.
- Specify behaviours in creativity maps.
- Describe the unification and granularity of behaviours.

4.1 Introduction

This chapter explains the construction process of creativity maps and specifies behaviours. A similarity measurement for states that enable this construction is introduced. The role of data, information and knowledge is specified and the relationship between these three entities is explained. A knowledge creation process, which illustrates the steps that are performed for the conversion of data into information and knowledge will be presented. Furthermore, a classification of creativity maps into a hierarchical structure is introduced and it is explained how these levels can be combined in a customised way. A detailed description and specification of behaviours follows. Their unification and granularity are discussed towards the end of this chapter.

4.2 Creation of Creativity Maps

To capture the initial creative process, it is necessary to observe and record all activities that are performed by a creator during the construction of an artefact. Each of these actions modifies one particular viewpoint, like for example *editing* that changes the *artefact* viewpoint. The result of the observation is a sequence of chronologically ordered actions, which describes the initial creative process. However, this structure is probably an incorrect representation. The process is usually not linear, as the creator rethinks ideas and returns to old stages. Moving between states and continuing at different positions leads to several branches and the construction of the previously defined *creativity map*. For instance, a writer who dislikes a paragraph returns to a previous stage of the creative process where this text did not exist. The process then continues from this position and creates a new branch.

A creativity map was specified as a structure consisting of a set of states and a set of transitions. The states contain the different viewpoints of the creation. The artefact viewpoint for instance stores the versions of the created artefact. Possible other viewpoints are discussion or contemplation. As described above, a creator might jump inside the creative process and go back to an old position and continue from this state. However, the person does not always consciously return. A writer for example deletes a previously written paragraph and therefore unconsciously jumps back to an old version of the document. It is necessary to identify these similar states in the creative process to reveal all branches of the creativity map. A creative process otherwise remains a linear sequence of actions.

The identification process probably requires additional information about the previously described cases. It is for example hardly possibly to observe situations where the creator returned to old ideas. A combination of an automatic viewpoint comparison with user provided information will therefore help to build the creativity map that represents the creative process best.

The construction process of creativity maps can be divided into three steps. Firstly the capturing of the creative process, which is responsible for the recording and creation of the initial linear structure. Secondly the identification of similar states, which was mentioned above with the example of a writer. The third step is responsible for the construction of the creativity map structure by repositioning transitions. This section explains these three steps.

4.2.1 Capturing of the Creative Process

As mentioned in Section 3.2, creativity is identified only by its product. It is necessary to capture and save the steps that were involved in its creation to allow for the construction of creativity maps and an analysis of the creative process. However, recording these actions can become a difficult task which can be performed automatically [104] or with the interaction of the creator. The advantages and disadvantages of both approaches are discussed in this section.

Automatically capturing [71][70] the creative process of a creator is advantageous, because the person is able to continue with the creation of the artefact without the necessity to interact with the capturing system. One possible realisation of this automatic recording process is a simple camera that records the gestures of the creator. The recorded gestures can then be used for the derivation of the performed actions, like thinking, reading or editing. This approach [36] is probably advantageous for the creator, but difficulties occur with respect to the analysis of the captured material. The gesture recognition has to function very well and it is hardly possible to recognise all actions and distinguish them. A writer might read something on the screen but the system is unable to recognise the document which is being read. It is additionally impossible to record actions that are performed outside of the recording scope. For instance, a possible discussion with some colleagues cannot be observed. Gestures might also differ between creators. The capturing system needs to be adjusted for every person and training sessions might be necessary [68]. Although this describes only one approach of an automatic capturing process, it emphasises some general disadvantages with these kinds of systems. If the actions are not identified by the creator, it is necessary to derive them during post-processing of the captured material. This is a difficult and time consuming task which requires human assistance and knowledge.

An alternative approach is a user interactive capturing systems. The creator is required to report the currently performed actions to the system. When the action switches, the system needs to be informed about it. The advantage in contrast to the automatic capturing described above is the lack of extra post-processing, in particular the action derivation. Every creator can have a personal set of actions, which are configured in advance and then recorded by the system. This enables a flexible design without the need of additional training sessions for new users. These kinds of systems can also be realised very easily and without additional knowledge about the actions. The creator can for instance write the activities down with a text editor or use a simple program for the recording. Independent from the recording technique, the system depends on the interaction of the user,

who is responsible for the correct observation of the creative process. If a creator forgets to inform the system about the current action, it is simply not captured. To prevent this scenario, the user interface [55] needs to be simple and intuitive for a less distractive recording process.

The approach that is realised in the presented research is an user interactive capturing system. It is implemented as a toolbar, representing viewpoints and actions. When an activity (e.g. editing) is started, the creator needs to click the corresponding button and the system captures it. Every newly started action automatically stops the current one. Details of the user interface and its functionality are described in Chapter 7. Figure 4.1 depicts a small sequence of a creative process that was recorded with the tool. It illustrates 6 sequentially ordered transitions, in particular one review, three editing and two reading actions. This example is used in the following two steps to illustrate the transformation into a map structure.

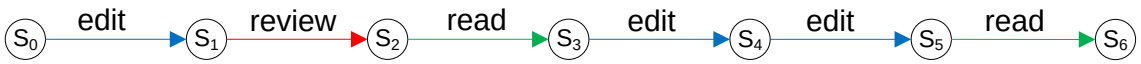


Figure 4.1: Captured Creative Process

4.2.2 Similar States in the Creative Process

As mentioned before, the states of creativity maps store the different viewpoints of the creation, which can be either observable or non-observable. It is possible to keep track of the observable viewpoints during the creative process, for instance the artefact viewpoint. The way a writer edits a document (i.e. insertion, deletion) or a musician edits a piece of music can be observed and modifications can be measured. In contrast to the observable viewpoints there are also non-observable ones, for example *knowledge*. It is hardly possible to observe if a creator has gained or maybe lost knowledge during the creative process. However, the observability of a viewpoint depends very much on its particular definition. If the knowledge viewpoint represents the amount of literature that was read, it can be described as a list of books or papers and therefore becomes observable.

Observable viewpoints are captured inside variables and each of their actions modifies one variable in a particular way. Non-observable viewpoints are static during the creative process, as they cannot be measured. Table 4.1 illustrates examples of observable and non-observable viewpoints.

Observable Viewpoints	**Non-Observable Viewpoints**
Artefact	Knowledge
Discussion	Mood
Concept	Personal Background
Model	Political Background
Evaluation	
Time	

Table 4.1: Observable and Non-Observable Viewpoints

Observable viewpoints are essential for the identification of similar states in the creative process. In order to measure the similarity [48] between them, a flexible and yet representative metric is needed. The similarity metric for documents in the writing domain is probably inappropriate to measure the similarity between two composition in the music domain. It can also be different for viewpoints of the same state. A metric that determines the similarity of two documents from the artefact viewpoint is different from the metric that is used for the comparison of two discussion viewpoints. It is therefore hardly possible to specify a similarity metric, which can be used in general. The analyser instead needs to define one or more viewpoints, which are used for the comparison of two states and a metric for each viewpoint to identify if two states are similar. This specification allows states to be compared with respect to multiple viewpoints, each being assigned a different metric. For example, the artefact viewpoint probably needs a different measurement than the discussion viewpoint.

The choice of a metric is essential for the following processes. For instance, if a writer changes a colon ":" into a dash "-" in a document, this has probably no major effect on the intention and the documents are still very similar. In contrast, if the word "alive" is replaced by "dead", this might change the intention in parts of the document. If the similarity metric M is based on the number of words that were inserted and deleted, then one word was deleted and one was inserted in both cases. Therefore the states in the examples are still equally similar. The importance of the metric becomes more obvious, if the same metric M is used for the comparison of source code. If only the term "true" is replaced by "false" in the condition of an if-statement, it can modify the whole behaviour of this program. However, based on the number of edited words, the two source code files are very similar. Another example is the detection of a "mostly undone" editing activity, in particular the deletion of a long paragraph and its replacement by a short sentence. The states prior and after this action might be similar, depending on the comparison process. Examples for similarity metrics of documents as described in [62] are binary similarity measurements based on the occurrence or absence of words. Other ones utilise the amount of keywords and their occurrences in a document. The source code of software

can be measured with the same metric or for instance the more popular metric Lines of Code (LoC) [59]. These are only some examples and it clearly depends on the purpose of the analysis to chose adequate measurements and viewpoints for the state comparison.

It is very important to note that additional information is (usually) required for the identification of similar states. For example, situations where the creator returned to an old idea are very difficult to observe and it is therefore necessary that the person provides this data. It can then be integrated into existing viewpoint or used as external data during the construction process. This allows the capturing system or analyser to use this additional information in combination with the previously discovered states. Only both the similarity understanding for the automatic discovery and the individual data allow for the construction of a creativity map that illustrates the creative process best. An automatic identification of similar states for the construction of a creativity map itself is limited.

The example that is depicted in Figure 4.2 illustrates the recorded values for the previously mentioned creative process. It shows the two observable viewpoints *Artefact (Art.)* and *Literature (Lit.)* and their modifications according to the performed actions. The first editing action modifies the artefact viewpoint from a_1 to a_2. After this, the review action leads to a change of the literature viewpoint from l_1 to l_2 and so on. Every action additionally consumes time and a time transition would usually occur in parallel with each of the depicted transitions. However, it is not displayed in any example of this thesis to allow for a clear representation of the creativity maps.

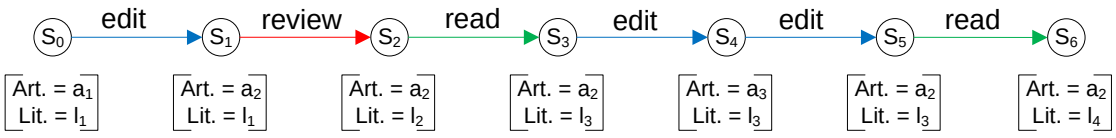


Figure 4.2: Recorded Sequence with Values

It is assumed that only the artefact viewpoint was chosen for the comparison of two states in the depicted example. The measurement that is used for the comparison of states defined them as similar, if they are identical, which means that they need to contain identical values. Table 4.2 illustrates the results of this step.

It is important to note that a sequence of identical values for a particular viewpoint, which does not contain any of the viewpoint's actions, is represented by its first state, as this is the

Artefact Viewpoint	States
a_1	S_0
a_2	S_1, S_5
a_3	S_4
a_4	S_6

Table 4.2: Similar States

first time, when the value appears. For instance, the artefact viewpoint in Figure 4.2 does not change between the states S_1, S_2 and S_3, as only actions of the literature viewpoint have been performed. All states are therefore represented by S_1. This is essential for the following step that constructs the creativity map structure.

4.2.3 Creativity Map Structure by Transition Repositioning

After similar states in the creative process are identified based on a proper comparison process, it is necessary to reposition their transitions in a second step for the construction of the creativity map structure. As described above, the similarity operation compares two states based on certain viewpoints, which means that similar states can still differ in one or more other viewpoints. They need to remain in the creativity map, as they can possibly be important for the further analysis. Each state contains a timestamp, which specifies the moment when the action of the incoming transition ends and the action of the outgoing transition starts. The sequence that is depicted in Figure 4.2 contains the timestamps $t(S_1), t(S_2), \ldots, t(S_6)$. They are ordered chronologically so that $t(S_1) < t(S_2) < \cdots < t(S_6)$. The timestamp $t(S_0)$ is the moment when $\overset{editing}{\longrightarrow}$ starts, $t(S_2)$ specifies the time when $\overset{editing}{\longrightarrow}$ ends and $\overset{review}{\longrightarrow}$ starts and $t(S_3)$ describes the moment when $\overset{review}{\longrightarrow}$ ends. If the creator returns to a previous state of the creative process, this means that the timestamp of this then similar state is lower than the one of the current state. For a set of similar states, the creator always returned to the one with the lowest timestamp. This means that each outgoing transition needs to change its source to this state. For the states that are illustrated in Table 4.2, this means that the outgoing transition of S_5 changes its source to S_1. The result is depicted in Figure 4.3.

It can be observed that the state S_5 is still present in the map, only without an outgoing transition. Whenever all outgoing transitions of a state are repositioned, it automatically becomes a final state of this branch. Every behaviour which contains the state ends at this position. It is possible that multiple branches start from a single state, which means that the creator went back to this state several times. As mentioned before, similar states can

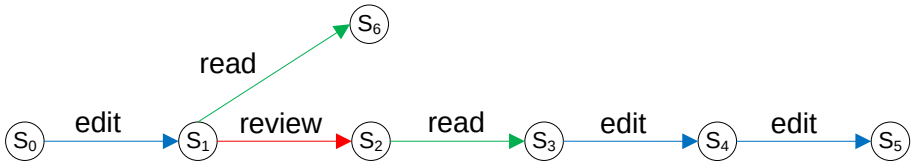


Figure 4.3: Transition Repositioning

still differ in one or more of the viewpoints that have not been used for the comparison. It is therefore necessary to add a reference to each of the original states to be able to track the reverting process.

The construction process of creativity maps finishes with the previously described transition repositioning action, resulting in a creativity map that includes all steps which were involved in the creation of an artefact. This map contains the different behaviours of a creator and consists of a rich set of information for further studies. A precise specification of behaviours is presented later in this chapter.

4.3 Classification of Creativity Maps

A corpus of creativity maps describes the set of all maps that were recorded and might be relevant for a later scenario. This can be for instance all creativity maps from a domain, creator, project or even only a single one. The analysis needs to specify which particular subset should be considered. It is also possible that maps from different domains or projects occur in the corpus. For example, 10 different writers, each creating 10 different books with one creativity map for each book create a total of 100 maps. They can then be analysed based on the particular book, creator, or any combination. These are different viewpoints on the data which allow for several types of analyses.

The previous example illustrated the ability of the corpus to represent different and distinct sets of data. Each of these sets is called a *category*, as for example writer or book. Based on them, it is possible to hide or extract information and compare maps. Categories [2] represent a hierarchy and can possibly be nested. The books are part of the project, which belongs to a creator. As each creativity map belongs to at least one category, each behaviour that is stored in this map automatically belongs to it as well. It is of course possible that similar behaviours from different creativity maps belong to different categories.

Categories allow to structure and preprocess the corpus of creativity maps, which enables the dismissal of irrelevant maps or the construction of subsets. There are in particular three different categories that are introduced in this section, namely *Project, Creator* and *Domain*. They build a hierarchy that organises the creativity maps into different sets and allows for their distinction based on simple properties. Each creativity map that is recorded belongs to a single project, creator and domain. It is possible that several maps belong to a project or that multiple creators participate in the same project as collaborators. Each creator furthermore belongs to at least one domain. Figure 4.4 depicts the relationship of the stated categories.

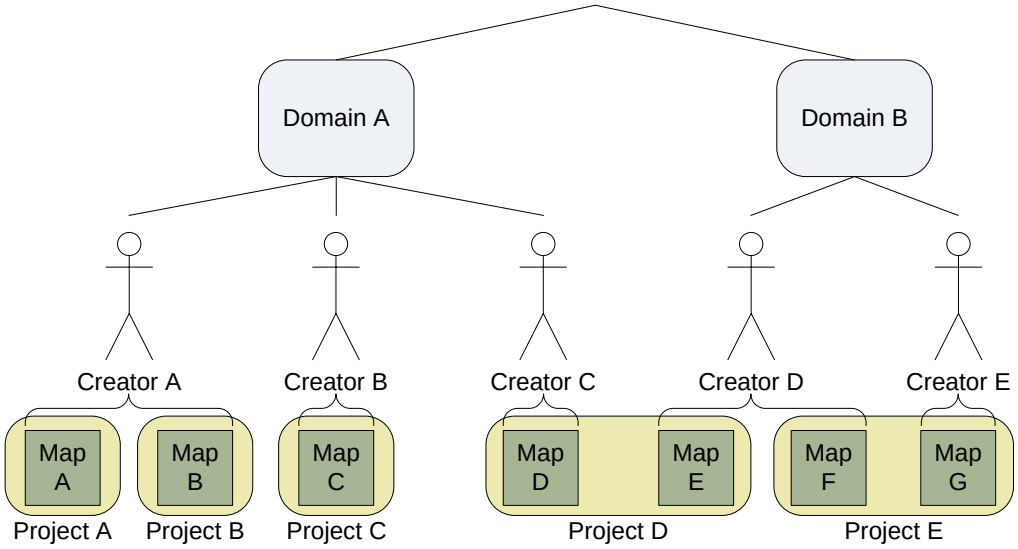


Figure 4.4: Creativity Map Categories

The figure shows the two domains A and B that are both linked to the root node, which represents all possible domains, creators and projects and therefore the whole corpus of creativity maps. It might be similar to all maps that are stored in a database, like it is discussed in Chapter 7. The first domain contains three creators, the second domain two. The first creator worked on two projects and created a creativity map for each of them. The second one constructed a single creativity map for one project. The third one worked on a project in collaboration with another creator from a different domain. Each of them created a single creativity map, which both belong to the same project. Another collaboration between the two creators of domain B can be detected.

The previously mentioned categories can be identified at each level of the hierarchy. Each domain, each creator and each project build a distinct category. The creativity map A for example belongs to Project A, Creator A and Domain A. The creativity maps D and

E both belong to the same project category, but different creators and different domains. Each creativity map can therefore be labelled with three distinct labels. One label D for the domain, one label C for the creator and one label P for the project. Map F, which is depicted in the previous figure can then be specified as [P = Project E, C = Creator D, D = Domain B]. This section describes each of the previously mentioned categories and explains their purpose for the analysis.

Project Category

Each artefact that is constructed by a creator represents one particular project. A writer for example edits a book, a software engineer implements an application or a painter creates a piece of art. It is assumed that one creativity map is constructed for every of these projects and every time a new project is started, a new creativity map will be recorded. If projects grow large, they can possibly be further divided into sub-parts, such as the chapters of a book or the different components of a piece of software.

Instead of arranging each of these sub-parts into a separate category, it is assumed that either the set of creativity maps can be represented by a single one or a project category is represented by multiple maps. The project category is the lowest component in the hierarchy. Information that is stored in this category is useful for the analysis and comparison of creative behaviours and the extraction of sequences that are frequent for one particular project or across a range of projects. This is relevant, if for instance the creative process for a particular book should be compared with the one for another, previously written book.

The results of this study are beneficial for a better understanding of project related behaviours, which can be utilised for the assistance of the creative process. It will be described in Section 4.5.3 that behaviours of creativity maps can be unified for a better analysis. This means that terms which belong to the same creator or domain need to be identified. The simplicity of this process depends on the granularity of the recorded actions (Section 4.5.2).

Creator Category

The previously described project category represents the lowest level of the hierarchy. As each project was created by at least one creator, it enables the definition of the next

level in the hierarchy to be the creator category. This allows for a simple distinction between different creators on the one hand and summarises projects into groups for one corresponding creator on the other hand. If all projects that were constructed by a single person are relevant for the analysis, it is possible to retrieve all recorded creativity maps by addressing the particular creator category.

Whenever a creator collaborated with other creators, they are all linked to an identical project. It then contains a separate creativity map for each of them. For example, if writers A and B both edited book X, they each possess their own creativity map which is stored separately in the corpus. A creator category, or even the whole level can be analysed for the retrieval of frequent behaviours to allow for the comparison of different persons. This might be helpful when situations are entered that were previously encountered at creative processes of different individuals. Patterns that were identified might be reused by other creators of the same domain or even across domains. Each creator belongs to one or more domains, which specifies the next category in the hierarchical structure.

Domain Category

The previously described creator level summarises all persons, who recorded their creative process. Each of them has at least created one project, which can also be shared across collaborators. Creators are grouped into distinct domains at the top most hierarchical level. The previously mentioned book example would for instance create a writing domain that summarises all writers. Possible other categories are software engineering, music or mathematics.

One creator can be present in more than a single domain. If the writer of the previous example also develops software, the person is present in at least two categories. A creator can be kept redundant in all domains or multiple domains can be linked to the same creator, which is similar to the relationship between projects and creators. The domain category represents the highest abstraction level. It might be possible to group similar domains to some kind of super-domain, such as a technical domain that includes computer science, electrical engineering and other similar areas. However, this is also possible by adapting the domain definition to the needs of the analysis. A technical domain can then substitute the set of domains it includes.

As the domain categories represent an abstract viewpoint on the particular creativity maps, they can be analysed for the extraction of frequent behaviours from a large data

set. This is useful for the identification of similarities between existing projects or creators and a domain. They can also be utilised for the extraction of abstract domain patterns that are shared between a number of creators and projects. These patterns, which probably need preprocessing in the form of behaviour unification (Section 4.5.3) may reveal useful and domain independent information about creative processes.

Customised Set

The three previously described categories enable a simple retrieval of predefined sets of creativity maps. They can be utilised for instance to compare two domains, creators or different constellations of projects. This distinction might be sufficient for many cases, but if the analyser is interested in a custom subset of creativity maps which is not directly described by any of the groups, it is necessary to create a personalised set. It contains any map without a particular link to creators, domains or projects. For instance, if some chapters of one book and some chapters of another book should be analysed and each book defines a project, these chapters build a customised set.

This set does not necessarily fit into the hierarchy, it furthermore allows to link maps of different hierarchical categories. It is not situated at a particular level in the creativity map corpus. The customised set is always adapted to the conditions of the analysis.

4.4 Data, Information and Knowledge in Creativity Maps

A distinction between the entities data, information and knowledge is essential for the understanding of the information mining approach that is presented in this thesis. Data has a very unspecific meaning that describes nearly everything. It is usually collected through measurements, experiments or just produced by theoretical considerations. It is raw, can either be structured or unstructured and has no meaning on itself. Interpretation of this data leads to information and knowledge. The role of data, information and knowledge [112] allows for the illustration of their linkage and ways to derive one from the other. This section explains the three entities with respect to creativity maps.

Data

The actions that are described by the creative process and every single transition in the creativity map specify data. Every time a new action is performed by the creator (e.g. editing, discussing), a new piece of data in the form of a transition is added to the creativity map. Data represents the atomic units that are used for the map construction. Each transition consists of an action label, a start state and an end state. The number of distinct transitions that are used in a creativity map is usually rather small, but can increase during the creative process. This also means that its structure can grow quite complex. However, from the data viewpoint, a creativity map is simply a large set of transitions that are ordered to build a particular structure.

Data itself has no meaning without additional information or semantics [1]. Only the information that the transitions in a creativity map represent the actions of a creator allow to analyse them for particular purposes. Otherwise, it would hardly be possible to distinguish the map from any other labelled transition system or graph. However, requirements of the representation or purpose of the creativity map are always specified in advance. The analysis of the relationship between atomic transitions allows to retrieve additional information that is useful for the further processing.

Information

Information is derived from the interaction of data in the creativity map. It is specified as sequences of transitions, which represent the behaviours of a creator. Each one contains information about the specific order in which a particular set of actions was performed. A creativity map consists of a large set of possible behaviours, even if the number of distinct actions is small. Information has specific features, such as the length that is similar to the number of transitions or the particular distribution of actions. These are only two examples and it is of course possible to define more features based on specific purposes of the information.

Since the creativity map describes a complex structure, it contains many different paths [28] (including transitions) and a rich amount of information. Every single branch contains a number of behaviours of different lengths. This information is the first result of the data analysis and might reveal behavioural patterns that were used by a creator. However, knowledge is required for its interpretation. For instance, the information that a particular creator uses a reading transition followed by an editing transition is useful, but a further

analysis to possibly reveal that this behaviour always leads to inappropriate results is necessary. External information is needed for the transformation of this information into knowledge.

Knowledge

Knowledge is created with the help of semantics that are added to information. The analysis of the behaviours in a creativity map allows to construct patterns of transitions for each creator or collaborator of a team. These patterns contain knowledge such as the amount of transitions from a particular viewpoint or their relative probability. It includes personal information about the creator and the particular set of behaviours. Once these patterns are revealed, they might be stored in a knowledge database [105], which can for instance be divided into patterns that were useful in terms of efficiency [13] and ones that were not useful. This allows to compare incoming information with this database and create feedback for the creativity support.

The extracted knowledge can be encapsulated and reused for further studies. Rich sets of data and information allow to construct large amounts of knowledge about the creator and the creation itself. They represent an interpretative environment that enables a linkage between knowledge components. This is for instance useful for the generalisation of patterns to domain patterns (e.g. writers, musicians or mathematicians).

Knowledge Creation Process

The three previously defined entities data, information and knowledge are closely linked to each other. The relationship between them, as described above, builds the knowledge creation process that is depicted in Figure 4.5.

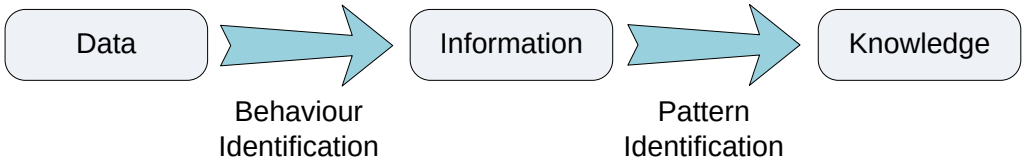


Figure 4.5: Knowledge Creation Process

Data has been specified as the transitions in the creativity map. It is analysed for the identification of sequences, which are known as behaviours. These behaviours describe information and enable the construction of knowledge in the form of behavioural patterns for a creator in a second step. The relationship between data, information and knowledge [80] is very often visualised as a pyramid [73], showing data at the bottom, followed by information and knowledge. It represents a similar understanding of the three elements to the explanation that is given above. The role of data, information and knowledge is important for the understanding and analysis of the creative process. Especially when techniques for information hiding and extraction are described, a specification of these entities becomes essential.

This thesis discusses the role of data and information in the creative processes and focusses on the first step of the creation process. It explains the linkage of transitions for the construction of creativity maps and introduces behaviours. The steps for the transformation of information into knowledge via the identification of behavioural patterns is discussed in a different thesis [14].

4.5 Behaviours in Creativity Maps

As mentioned before, creativity maps contain a variety of information about the creative process, which is represented as sequences of transitions. The order of these actions and the duration of each of them generate specific sets of sequences and represent the particular and personal preferences of the creator. A writer for instance might start to conceptualise a document at the beginning, followed by a discussion with others and an editing action. This is one particular sequence of this person. Another writer who chooses the same set of actions uses them in a different sequential order. However, it might be possible that both of them share some sequences. Conceptualising is probably more frequent at the beginning of the creation than towards the end. In contrast, the amount of discussions might stay constant over time or even increase as the process continues. This depends on the individual creator and also maybe the domain.

Each sequence of actions specifies one *creative behaviour*, or simply *behaviour*. It can be used for the construction of a behavioural pattern, which illustrates frequent sequences in the creativity map. A comparison of these patterns possibly leads to the identification of similarities between the creations of one or more creators. As the creative process is of dynamic nature, behaviours will probably develop and change over time.

4.5.1 Specification of a Behaviour

A creativity map contains one root state where the creation starts. It describes the moment when the observation of the creative process begins. Each branch in a creativity map contains one final state. It specifies the situation, where the creator stopped the creation or returned to a previous stage of the creative process (different state) and continued from there. A transition inside the creativity map connects two states, belongs to one viewpoint which is part of the set of viewpoints V and has one action label that is part of the set of actions A. These four components build a quadruple (r, s, a, v) consisting of:

- A start state (r)
- An end state (s)
- An action $(a \in A)$
- A viewpoint $(v \in V)$

Each action belongs to exactly one viewpoint. The operation $a(t_i)$ returns the action and $v(t_i)$ the viewpoint of the transition t_i. A transition t that connects the states r_0 and s_0, is labelled with the action *editing* and belongs to the viewpoint *artefact* is specified as $t = (r_0, s_0, editing, artefact)$. This allows for the specification of a behaviour b as a sequence of connected transitions $t_0 t_1 \dots t_{n-1}$ in the following way.

$$b \;=\; (r_0, s_0, a_0, v_0)(r_1, s_1, a_1, v_1) \dots (r_{n-1}, s_{n-1}, a_{n-1}, v_{n-1})$$

where $s_i = r_{i+1}$ for all $0 \leq i \leq n-2$ with $s_i, r_i \in \Sigma$.

The illustrated form might not always be a suitable representation of a behaviour. It is very often unnecessary or impossible to specify it in the complete way with all start and end states. The actions of a creator is probably the most important information in most of the cases and usually the required component for the further study. A more abstract definition, which only represents a sequence of labelled transitions is introduced for this reason.

A transition is stated as an arrow labelled with an action, e.g. $\overset{editing}{\longrightarrow}$. It is also possible but not necessary to additionally label the arrow with the corresponding viewpoint, which is presented at the bottom right corner. A behaviour can then be specified as a sequence

of arrows that are labelled with actions (a_i) and possibly viewpoints (v_i) in the following way.

$$b \; = \xrightarrow{a_0}_{v_0} \xrightarrow{a_1}_{v_1} \ldots \xrightarrow{a_{n-1}}_{v_{n-1}}$$

or simply

$$b \; = \xrightarrow{a_0} \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}}.$$

The latter case reduced a behaviour to its essential information and is used in this thesis. It is usually not necessary to specify the viewpoints additionally to the actions. However, if the former specification is needed in the following chapters, it is explicitly stated.

Length

The length of a behaviour $b \; = \xrightarrow{a_0} \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}}$ is determined by the number of its transitions, in particular

$$|b| \; = \; n.$$

This property is important for the analysis and can be used as a threshold for certain tasks. A determination of the relationship between transitions from the beginning and the end of a sequence might be especially difficult for long behaviours. Lengths thresholds enable the reduction of behaviours to a subset with a particular maximum or minimum length. The longest possible behaviours that can be identified in the creativity map start at the root state and end at one of the branches' final states. They illustrate a decomposed representation of the creativity map. Especially large behaviours can for example be used to identify different creation stages in the creative process. The rather small creativity map that was constructed for the transition repositioning example in Figure 4.3 contains the final states S_5 and S_6. The two longest behaviours b_1 and b_2 are specified in the following way.

1. $b_1 : \xrightarrow{edit} \xrightarrow{read}$
2. $b_2 : \xrightarrow{edit} \xrightarrow{review} \xrightarrow{read} \xrightarrow{edit} \xrightarrow{edit}$

4.5.2 Granularity

Each transition of a creativity map contains one action label, which as previously defined, belongs to one viewpoint. This label determines the activity that was performed by the creator during the creation of an artefact. Some of the actions might have been defined more abstract, like *reading* and others very specific, for instance *reading book X*. These examples differ in their level of detail, which specifies the *granularity* of a behaviour. The granularity is higher in the latter case and a behaviour is able to represent a range of granularities. The action set of the artefact viewpoint for instance can be described with a high granularity to enable a profound analysis of these behaviours, whereas the knowledge viewpoint might contain actions that are specified less detailed.

The granularity of actions depends very much on the recording process. If this process is able to observe the performed actions with a high level of detail, the behaviours automatically become more specific. The corresponding map grows more complex at the same time. If the observation is only able to determine an abstract action like the previously mentioned *reading*, the construction of such a detailed creativity map is not possible. A creator can also contribute or even remove details, which in turn influences the granularity. The recording process that is used in this approach, as it was described above enables the creator to define the actions and therefore the granularity on a personal basis. Chapter 7 explains the tools that are used for the recording in more detail. It shows how new actions can be added and existing ones can be changed or removed during the recording process, which enables a dynamic adjustment of the accuracy.

If the granularity of the recorded actions is high, the number of recorded behaviours grows. However, it is not always necessary to analyse the transitions at a high level of detail. Similar transition, like for instance *reading book X* and *reading book Y* can be collapsed into a single, more abstract *reading* action. This might then be sufficient for the current study. It is of course possible to revert this process by expanding the action again. As these operations are performed, the creativity map is able to represent different levels of detail for distinct actions or viewpoints. The granularity does not necessarily illustrate the quality of the recording process. Especially as large creativity maps can become difficult to handle, it should be chosen with caution.

The granularity of behaviours is not further discussed in this thesis. It is assumed that the actions of a creator are recorded with an appropriate level of detail. The creativity maps that are used in the examples of the succeeding chapters do usually not differentiate between very similar actions, like *reading book X* or *reading book Y*. It is instead

expected that these actions were either collapsed or the granularity does not allow for their distinction.

4.5.3 Unification

The mapping of creativity is not restricted to any particular domain, as creativity maps allow for the representation of creative processes from any field. They possibly differ between creators, groups or even domains as the set of actions is different. It is obvious that a writer performs different activities than a musician or a software engineer. Even in a single domain, members use different terms and actions to describe their creative process. One software engineer might use the term *programming*, whereas another one uses *coding*. Although these two actions are different in their description, they are very similar or even identical in their meaning.

The previous example illustrated the similarity of terms even if they are initially considered to be different. For the analysis of creativity maps and behaviours that origin from different creators, it might be necessary to identify these similarities for the creation of useful knowledge and behavioural patterns. The knowledge creation process illustrated a possible necessity for the integration of external knowledge to perform the transformation from information to knowledge. One major problem is the unification of different terms, which is especially important if external sources access creativity maps. Natural Language Processing (NLP) [64] provides a solution to this problems, mainly in the form of text analysis techniques. Additionally to this, databases of synonyms like Wordnet [69] for the English language can be used to detect similarities.

An unification of behaviours from different domains can in some cases be necessary as well. The previously mentioned synonyms might not be sufficient for this task, as a more abstract unification is needed. For instance, the action *editing* of a writer and the action *painting* of an artist can both be identified as activities that change the artefact. Even if the artefact differs in both domains, the relationship between these actions might be helpful for the identification of shared behavioural patterns. However, it was mentioned before that the analysis of creativity maps or behaviours is not part of this thesis and it will therefore not elaborate on the unification of these entities. This section should just emphasise possibilities for a broader study, if a sufficient unification of behaviours from either the same domain or distinct ones is supported.

4.6 Summary

This chapter presented the creation process of creativity maps, introduced their classification and defined behaviours. It was described how the initial sequential creative process is recorded and converted into a map structure. Similar states are identified in the first step, based on a process that compares viewpoints. It was emphasised that additional information about the situations when the creator returned to a previous stage is usually required to assist this step. The outgoing transitions of these similar states are then repositioned to construct a creativity map. The role of data, information and knowledge was defined and these three entities were identified with respect to creativity maps. A classification of creativity maps into a hierarchical structure was introduced. Behaviours were defined as sequences of transitions and different possibilities for its expression were illustrated.

Chapter 5

Information Hiding

Objectives

- Explain the observability of behaviours.
- Introduce Partial Creativity Maps (PCMs)
- Introduce hiding, restriction, revealing and restrictive revealing operations for the information hiding process.
- Introduce the Behaviour Description Language (BDL) for the description of behaviours.
- Describe techniques for information hiding with the help of the Behaviour Description Automaton (BDA).
- Explain collapsing and pruning techniques to reduce the size of PCMs.

5.1 Introduction

This chapter explains information hiding with respect to creativity maps. A distinction between observable and non-observable behaviours is illustrated, followed by the introduction of hidden transitions and Partial Creativity Maps (PCMs) as the result of the hiding process. The Behaviour Description Language (BDL) as a formal language for

the convenient specification of behaviour descriptions is specified. It seamlessly integrates with the hiding, restriction, revealing and restrictive revealing operations that are defined for the modification of creativity maps. The concept of a Behaviour Description Automaton (BDA) is illustrated and its use in the hiding process is demonstrated. Collapsing and pruning techniques for the size reduction of PCMs are introduced towards the end of this chapter.

5.2 Observability of Behaviours

Creativity maps usually grow with the recording time, as more actions are being performed and observed. For instance, if the creative process is observed for 30 days and 30 actions are preformed each day, the corresponding map contains 900 transitions. If this project would last a couple of month, the structure can easily grow to a couple of thousand transition. It is difficult to handle very large maps, in particular display or analyse them. They can additionally contain irrelevant information which might hinder the analysis. This section introduces a PCM as a solution to the described problems.

5.2.1 Observable and Non-Observable Behaviours

The previous chapter illustrated the differences between observable and non-observable viewpoints. In a similar fashion, it is possible to distinguish between observable and non-observable actions and similarly behaviours. Examples for the former ones might be *insert*, *delete* or *change font* with respect to the *artefact* viewpoint and the latter ones can contain *gain knowledge* or *forget* from the *knowledge* viewpoint. Observable actions are related to observable viewpoints and non-observable actions to non-observable viewpoints. Table 5.1 illustrates some examples.

Observable Actions	**Non-Observable Actions**
Editing	Gaining Knowledge
Discussing	Losing Knowledge
Conceptualising	Being Happy/Sad
Reading Through	Influences of the Creator's Background(s)
Comparing	
Painting	

Table 5.1: Observable and Non-Observable Actions

Some non-observable behaviours might be indirectly identifiable. If a writer increases the use of adjectives as for instance "scary" or "dark", this allows to infer that he or she is in a bad mood. However, a writer of crime stories might be in a good mood when the same development is observed. A software developer who starts to use a particular design pattern possibly had a discussion with a software architect. The source code in a later stage of the creative process can still contain new appearances of it. This allows to indirectly infer that knowledge about the use of design patterns was gained.

Non-observable behaviours can have long term influences on the creative process. A new design pattern needs to be learned and understood by a software developer once and can be used afterwards. A similar assumption can be made for the first example, when a writer changes the mood and becomes sad. Observable reactions can also be related to non-observable actions that were performed in the past. If a behaviour varies during the creative process, it might be a result of interfering non-observable activities. However, these assumptions need to be made very carefully, especially if they are going to be generalised for projects, creators or domains.

A creativity map only displays the observable viewpoints and behaviours. However, it is possible to convert them into their non-observable counterparts. Irrelevant behaviours can be hidden from a map, so that they are no longer recognisable afterwards.

5.2.2 Partial Creativity Map (PCM)

Information was previously defined as a sequence of transitions, called behaviour. This in turn means that the information hiding process aims at stashing irrelevant behaviours of a creativity map. It is a process of concealing actions [49] so that they are no longer recognisable for external observation. This enables to reduce the visual size [61] on the one hand and allows for a more focussed study on the other hand. Invisible actions are sometimes specified as silent actions [33], especially in process algebra.

It is important to note that hidden actions are still existent in the creativity map. If a particular behaviour with hidden transitions would be replayed, then all actions including the concealed ones are performed. However, the difference in this situation is that the hidden actions are not visible to external observers. It is almost as if they are not existent. Let for example the creativity map of a musician contain *singing* transitions, which are irrelevant for the current analysis and will therefore be hidden. External observers will not be able to hear the person singing when the behaviours are replayed afterwards, even

if this activity is still performed. This is similar to placing the musician into a soundproof box.

It is crucial for the analysis to distinguish between hidden and visible behaviours inside a creativity map. Hidden transitions therefore carry the unique label α and no other visible transition is allowed to be labelled identically. A creativity map that contains both visible and hidden behaviours is called a Partial Creativity Map (PCM). The transition set R of a PCM is extended with the hidden transition $\xrightarrow{\alpha}$, which leads to the following specification.

$PCM = (\Sigma\ ;\ R\ \cup\ \{\xrightarrow{\alpha}\})$

A PCM can for example be used as input for a frequent information extraction process, as it is described in Chapter 6, in order to produce only relevant patterns. It is furthermore not necessary for the analysis to distinguish between creativity maps and PCMs as both definitions are based on states and transitions. Hidden α transitions can simply be ignored when needed.

5.3 Description of Behaviours for Information Hiding

In order to hide irrelevant information, the analyser should be able to specify it in a convenient way, which allows to address any component of a behaviour, in particular the transition labels and the state values of the different viewpoints. A formal language for the description of behaviours, namely the Behaviour Description Language (BDL) is introduced for this reason. The aim is to provide a simple and yet powerful tool that allows for the convenient definition of (complex) behaviour descriptions without being necessarily familiar with it.

5.3.1 Behaviour Description Language (BDL)

As mentioned before, a behaviour is specified as a sequence of transitions. Each transition belongs to a viewpoint and is labelled with an action. Together with the start and end state of a transition, these are the major components the analyser should be able to express with the BDL. A choice between different transitions should be possible to allow for a simple specification of more than a single behaviour. The BDL contains all these options to conveniently specify a set of behaviours, which is called *behaviour description*.

The following Extended Backus-Naur Form (EBNF) [51] illustrates the syntax of the BDL. Square brackets [ ] delimit optional constructs, braces { } indicate zero or more repetitions of the enclosed construct, parentheses ( ) represent simple grouping of constructs and a vertical bar | expresses a choice of one from many.

```
BehaviourDescription  ::=  Behaviour { "|" Behaviour }
Behaviour             ::=  Factor { Factor }
Factor                ::=  ( Transition | SubDescription )
                           [ Quantifier ]
Quantifier            ::=  "[" N ".." ( M | "*" ) "]" | "*"
SubDescription        ::=  "(" BehaviourDescription ")"
Transition            ::=  "<" State "-" Edge "-" State ">"
Edge                  ::=  AnyEdge | MapEdge [ "\{" Exclusion "}" ]
MapEdge               ::=  "V=" Viewpoint | "A=" Action
Exclusion             ::=  MapEdge { "," MapEdge }
State                 ::=  AnyState | StateMarker
                           | [ StateMarker ]
                             ( StateCondition { "," StateCondition } )
StateCondition        ::=  Condition for one or more viewpoints,
                           specified by the analyser
StateMarker           ::=  Refers to the currently visited state of
                           the creativity map
Viewpoint             ::=  One viewpoint of the Creativity Map,
                           e.g. Artefact, Literature, Discussion, etc.
Action                ::=  One action of the creator, e.g. Editing,
                           Reading, Discussing, etc.
AnyEdge               ::=  "E" (Place holder for any edge of the
                           Creativity Map)
AnyState              ::=  "S" (Place holder for any state of the
                           Creativity Map)
N                     ::=  Natural Number >= 0
M                     ::=  Natural Number, so that M >= N
```

The BDL specifies a *BehaviourDescription* as a sequence of *Behaviours*, divided by a vertical bar "|", which is interpreted as a choice. A *Behaviour* is a sequence of one or more *Factors*. Each *Factor* is either a *Transition* or a *SubDescription*, followed by an optional *Quantifier*. A quantifier can either be a range of two values $[n..m]$ which refers

to minimum n and maximum m repetitions, or an open range $[n..*]$, which specifies minimum n repetitions and no maximum boundary or a star operator $*$, which refers to zero or more repetitions of the previous construct and is similar to $[0..*]$. A *SubDescription* is a *BehaviourDescription* construct surrounded by a pair of parentheses, which enables the grouping of transitions. Each *Transition* is described as an *Edge* consisting of a start and end *State*. Each *Edge* can either be described by an *AnyEdge*, which is simply a place holder for any edge of the map, or a specific *Viewpoint* or *Action* (Section 4.5.1), specified as *MapEdge*. This allows for the description of transitions that belong to a viewpoint or are labelled with particular actions. An Edge is followed by an optional *Exclusion*, which allows to specify one or more *MapEdges* that should not be included in the previously specified *MapEdge*. It allows for instance to exclude particular actions from a viewpoint. A *State* can be described in three different ways. First of all as an *AnyState*, which is a place holder for any state similar to the *AnyEdge* construct for edges. Secondly as a *StateMarker*, which saves a reference to the currently visited state and thirdly as a list of *StateConditions* that are seperated by "," with an optional preceding *StateMarker*. A condition for a viewpoint needs to be satisfied by the value that this viewpoint stores. A possible condition could specify that the word count of a document needs to be less than 500. Conditions are always referred to by their names, for example $C_1 = \textit{word count} < \textit{500}$. In this example, the word count refers to the number of words, which the document contains. It is one feature of the artefact.

The described syntax enables the specification of behaviours in a convenient way. Especially the combination of transition labels and state conditions allows for the representation of any behaviour of a creativity map. The place holder for transitions and states can be used to define patterns of behaviours. A good example of a behaviour description is the specification of all behaviours that start and end with an action which modifies the artefact. It is stored inside the artefact viewpoint of the state and all of its changes are recorded during the creative process. Each process contains actions that are directly and indirectly involved in this modifications. For example, an *editing* action changes the document that is observed during the creative process of a writer and a *discussion* action edits the record of discussions, but not directly the artefact itself. Although the artefact is not modified, it might happen at a later stage as a result of the previous action. A behaviour that changes the artefact indirectly is described as the sequence of actions between two direct modifications. Figure 5.1 depicts a creativity map with highlighted "artefact behaviours".

All the behaviours that are marked in Figure 5.1 can be described in a convenient way with the help of the BDL. The start and end transition of these behaviours need to belong to the artefact viewpoint and any quantity of arbitrary transitions, except for the ones of

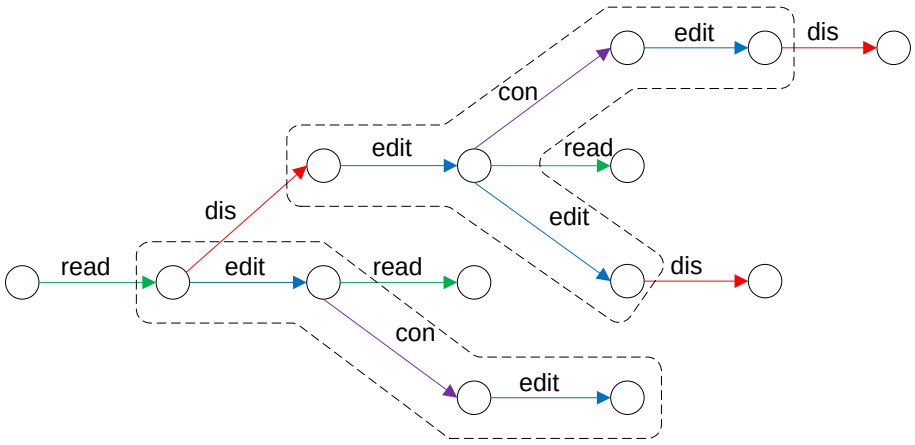


Figure 5.1: Creativity Map with Highlighted Artefact Behaviours

the artefact viewpoint, is allowed to occur between them. The behaviour description d_1 represents the corresponding expression in BDL.

```
d1 = <S-V=Artefact-S><S-E\{V=Artefact}-S>*<S-V=Artefact-S>}
```

The first and last elements `<S-V=Artefact-S>` describe the set of transitions that belong to the artefact viewpoint. It is not necessary that their states satisfy any conditions. The element in between `<S-E\{V=Artefact}-S>*` describes any transition that occurs in the creativity map and does not belong to the artefact viewpoint. The additional star operator `*` determines zero or more repetitions of it. The "artefact behaviour" can be further refined with the BDL. For example, if at least 2 and at most 5 transition should occur between two actions from the artefact viewpoint, this is specified in the following behaviour description.

```
d2 = <S-V=Artefact-S><S-E\{V=Artefact}-S>[2..5]<S-V=Artefact-S>
```

The quantifier `[2..5]` restricts the preceding construct `<S-E\{V=Artefact}-S>` to a quantity between 2 and 5. None of the marked behaviours in Figure 5.1 is represented by d_2. The following two section will explain the use of state conditions and state markers to allow for the integration of state information into the behaviour description.

5.3.2 State Conditions

Especially the state conditions are powerful possibilities, which enable the analyser to additionally consider the viewpoints and values that are stored inside the states. An

example especially for the creative process of a writer is a behaviour which starts with any transition that leads to a document version with less than 500 words, followed by a discussion or reading transition. In other words, the artefact viewpoint of the first transition's end state needs to store a document with less than 500 words. Exactly this constraint is represented by the condition C_1 = *word count of the document* < *500*. The following behaviour description d_2 expresses this example.

d_3 = <S-E-C_1>(<S-A=Discussion-S>|<S-A=Reading-S>)

The first element <S-E-C_1> describes any transition which ends with a state that satisfies the condition C_1. The second element (<S-A=Discussion-S>|<S-A=Reading-S>) is a *SubDescription* which describes a choice between a discussion and reading transition. No particular conditions are specified for this *SubDescription*, which means that any transition with the described labels that follows a state with word count < 500 is accepted.

One viewpoint which can be identified in any creativity map is time. It was mentioned before that each creativity map contains time transitions, which run in parallel with the other ones. Every state in the creativity map also stores a timestamp that describes the moment when the action of the incoming transition ends and the action of the outgoing transition starts. As this is similar in all maps, it can for example be used for the construction of time-dependent PCMs. If the analyser is interested in different creation stages, the creativity map can be divided into according PCMs. The behaviours at the beginning are possibly different to the behaviours that are performed towards the end. The time viewpoint therefore becomes a powerful information source for the behaviour hiding approach.

State conditions can also be used for example to detect similar situations during the creative process. If particular behaviours that result in a modification of the artefact were previously identified as disadvantageous or ineffective, they might be irrelevant. The artefact viewpoint would be used for the definition of a behaviour description, which can then be utilised for the concealment of these parts. This reduces the amount of information and enables the analysis to focus on the remaining behaviours. However, this is only one example for the use of state conditions and the described situations can also be interpreted differently.

5.3.3 State Marker

The previously described state conditions are only able to refer to the values of the actual state and compare them with predefined values. It might in some situations be necessary to allow conditions to refer to previous states, to creates a larger range of possibilities for their use. For example, a sequence of three transitions should be hidden from the creativity map of a writer. The document that is stored in the end state of the last transition should not contain more words than the document of the start state of the first transition. It is impossible to describe this scenario with the conditions only.

State markers are introduced in the BDL for this reason. Instead of satisfying a condition, they simply refer to a previous state so that it can be used later in the information hiding process. This allows to create conditioned sequences of transitions. The previously described example can then be realised in the following way. Let the condition C state that the word count of the document needs to be less than the one that M refers to. These two components can now be used to create the following behaviour description.

d_4 = <M-E-S><S-E-S><S-E-C>

It is also possible to create behaviour descriptions, which combine multiple markers. For instance, the previous example can be extended, so that the document of the end state of the last transition needs to contain more words than the document of the start state of the second transition. Let M_2 describe a second marker and let C_2 describe a second condition, which states that the word count of the document needs to be higher than the one that M_2 refers to. This then leads to the following behaviour description.

d_5 = <M-E-S><M_2-E-S><S-E-C,C_2>

The two examples illustrate that state markers extend the language with a useful facility to create conditions that relate to previous state values.

5.4 Operations for Information Hiding

To allow behaviour descriptions to be used in the information hiding process, the four distinct operations hiding, restriction, revealing and restrictive revealing will be introduced.

All of them work seamlessly together with the BDL and allow to either hide, keep or reveal the specified behaviours.

5.4.1 Hiding Operation

The hiding operation, which uses the hiding operator "\" is performed in order to conceal transitions of a creative process. It uses a creativity map in conjunction with a behaviour description that is specified in the BDL. Each described behaviour that can be identified in the creativity map will then be hidden by relabelling its transitions with α, which might result in a PCM. The hiding operation is specified in the following way.

```
<Creativity Map> \ { <Behaviour Description> } = <Creativity Map>'
```

It is possible that the specified behaviours are not present in the creativity map, or incremental hiding processes are performed, so that both maps `<Creativity Map>` and `<Creativity Map>'` can describe a PCM. The following example illustrates the use of the hiding operation. Let the behaviour description d in BDL describe all behaviours which start with a reading transition, followed by an editing transition and a transition from the artefact viewpoint, in particular `d = <S-A=Reading-S><S-A=Editing-S> <S-V=Artefact-S>`. Let the artefact viewpoint contain the two actions editing and correcting. Accordingly, d describes the following two behaviours $b_1 = \overset{reading}{\longrightarrow} \overset{editing}{\longrightarrow} \overset{editing}{\longrightarrow}$, $b_2 = \overset{reading}{\longrightarrow} \overset{editing}{\longrightarrow} \overset{correcting}{\longrightarrow}$. As mentioned before, a viewpoint is similar to a category, which allows the set of actions to be grouped into labelled sub-sets. For a creativity map M, the hiding operation

```
M \ { <S-A=Reading-S><S-A=Editing-S><S-V=Artefact-S> }
```

hides every occurrence of these two behaviours. An example is depicted in Figure 5.2, together with the PCM as a result of the hiding operation. The PCM, which is shown at the right-hand side illustrates that the transitions of the two behaviours b_1 and b_2 have been relabelled with α. There are in total one reading, one editing and one correcting transition that were relabelled.

The PCM might be further minimised by collapsing or pruning hidden transitions after the information hiding process. It can increase the clarity and remove irrelevant information from the view or even the whole map. These techniques are explained in Section 5.6.

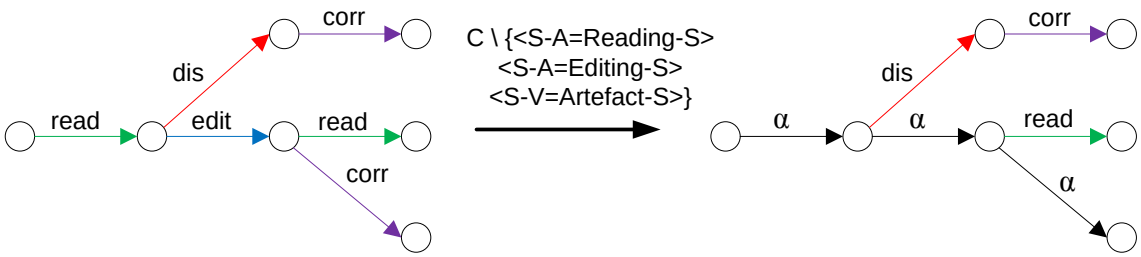


Figure 5.2: Information Hiding

5.4.2 Restriction Operation

The hiding operation enables the concealment of any behaviour that is specified with the BDL. However, depending on the number of different transitions that should be hidden, the behaviour description can become large or unclear and it is often easier and more convenient to describe the behaviours that should remain in a creativity map. For example, to hide all transitions but the ones that belong to the artefact viewpoint, it is necessary to specify all remaining ones in combination with the hiding operation in order to create the correct PCM. An operation that hides all transitions, but the ones of the artefact viewpoint would allow for a more convenient process. The restriction operation, which uses the restriction operator "/" is introduced for this reason. It is specified in the following way.

```
<Creativity Map> / { <Behaviour Description> } = <Creativity Map>'
```

For similar reasons that were mentioned before, any of the creativity maps `<Creativity Map>` and `<Creativity Map>'` can describe a PCM. The argument of the restriction operation is also a behaviour description expressed in the BDL, with the difference that behaviours which are not specified will be hidden. Given the identical behaviour description `d = <S-A=Reading-S><S-A=Editing-S><S-V=Artefact-S>` that was also used in the previous example. For a creativity map M, the restriction operation

```
M / { <S-A=Reading-S><S-A=Editing-S><S-V=Artefact-S> }
```

results in a PCM where all occurrences other than the ones specified in d are concealed. If the same creativity map as in the previous example is used, this operation would relabel all visible transition with α on the one hand and keep the original labels of all hidden behaviours on the other hand. Figure 5.3 depicts the result of this process. The original creativity map is shown on the left-hand side together with the PCM on the right-hand

side after the restriction operation has been performed.

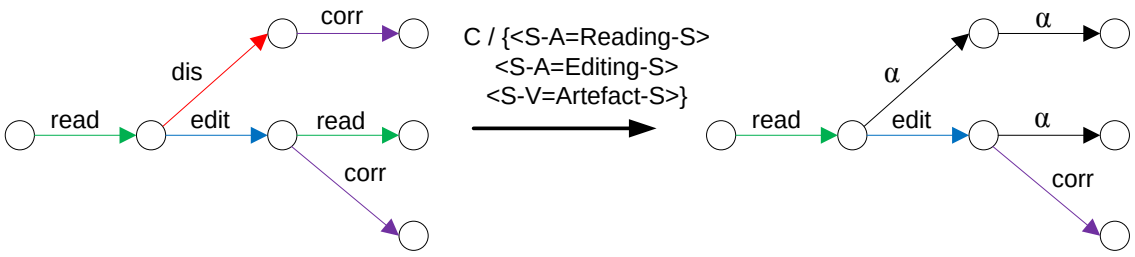


Figure 5.3: Information Restriction

There are in total three transitions which were hidden, in particular one discussing, one reading and one correcting action. To construct a similar PCM with the hiding operation, it is necessary to specify all behaviours that were relabelled, which in many cases results in a larger and unclear behaviour description. It is therefore very often more convenient to specify the description as previously explained and use the restriction operation.

For a creativity map M and a behaviour description d, the hiding operation `M \ { d }` creates a PCM that is equivalent to the restriction of this map `M / { B`$_{vis}$ `}` to all behaviours B_{vis} of M that are still visible after the hiding process.

5.4.3 Revealing Operation

Once a PCM with α transitions has been constructed, it can be used for further aim oriented studies. However, the original labels of these transitions might be required at some point and it is then necessary to reveal them again. This can be the case, if the PCM was only created temporarily and the original creativity map should be restored afterwards. To keep the revealing process flexible and enable the analyser to uncover only parts of the previously hidden transitions, a revealing operation, which uses the revealing operator "\˜" is introduced. It is specified in a similar way to the hiding and restriction operations that were defined above.

```
<Creativity Map> \~ { <Behaviour Description> } = <Creativity Map>'
```

It is again possible that any of the creativity maps `<Creativity Map>` and `<Creativity Map>'` describe a PCM. However, the revealing operation usually uses a PCM that contains α transitions instead of a regular creativity map. The argument is a behaviour description which enables the analyser to exactly specify the parts that should be revealed. It is

possible to reconstruct the original creativity map by using this operation in combination with the behaviour description `d = <S-E-S>*`, which specifies sequences of zero or more transition of any viewpoint or action. The result (`<CreativityMap>'`) of the revealing operation can therefore be a regular creativity map as well as a PCM.

The following example illustrates the use of the operation. Given the simple behaviour description `d = <S-A=Discussing-S><S-E-S>*`, which describes all behaviours that start with a discussing transitions followed by zero or more arbitrarily labelled transitions. To be able to reveal an α transition, it is of course necessary to access its original label. This can for example be realised with an additional variable in the concrete implementation of a creativity map. For the PCM that was constructed in the previous restriction example, the revealing operation

```
PCM \~ { <S-A=Discussing-S><S-E-S>* }
```

results in a map where one discussing and one correcting transition have been revealed, as depicted in Figure 5.4. The original PCM is shown on the left-hand side together with the result of the revealing operation on the right-hand side.

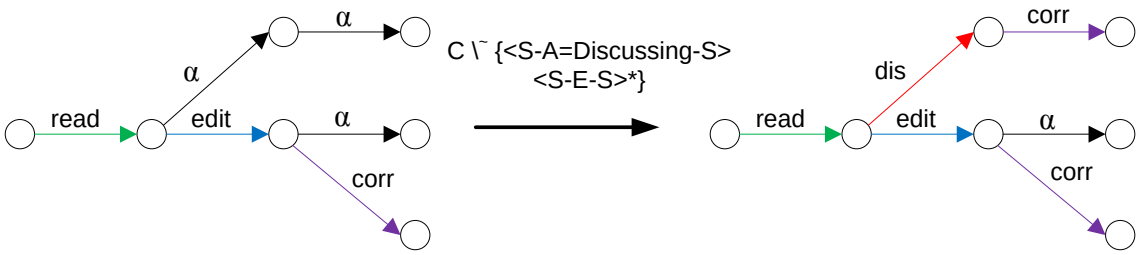


Figure 5.4: Information Revealing

Only two transitions were revealed in the resulting PCM. It was mentioned before that the original creativity map can be reconstructed by using the identical behaviour description in combination with the revealing operation. For example, the operation `PCM \~ { <S-A=Reading-S><S-A=Editing-S><S-V=Artefact-S> }` for the PCM on the right-hand side in Figure 5.2 would revert the process and result in the creativity map on the left-hand side in the same figure. This relationship can be specified in the following way.

```
For all M: (M \ { d }) \~ { d } = M
```

However, the statement is only true, if the original creativity map M did not contain hidden transitions prior to the first hiding process. Otherwise it is possible that these transitions are also revealed, which results in a different creativity map or PCM.

5.4.4 Restrictive Revealing Operation

The revealing operation that was introduced before, allows to unveil any behaviour that is specified by a behaviour description. Similar to the restriction operation that was introduced to limit the behaviours of a PCM, a restrictive revealing operation will be specified. For example, to reveal all behaviours but the ones of one particular viewpoint, it is necessary to specify the transitions of all other viewpoints in combination with the revealing operation. The restrictive revealing operation, which uses the restrictive revealing operator "/˜" simplifies this task. It is specified in the following way.

```
<Creativity Map> /˜ { <Behaviour Description> } = <Creativity Map>'
```

The syntax is very similar to the previous operations and any of the creativity maps `<Creativity Map>` and `<Creativity Map>'` can describe a PCM. However, it is sensible to use a PCM on the left side. The argument of the restrictive revealing operation is a behaviour description expressed in BDL. Any behaviour that is not specified by this description will be revealed. To enable a better comparability, the behaviour description `d = <S-A=Discussing-S><S-E-S>*` is chosen identically to the previous example. For the same PCM, the restrictive revealing operation

```
PCM /˜ { <S-A=Discussing-S><S-E-S>* }
```

results in a map where a single reading transition is uncovered, as depicted in Figure 5.5. The left-hand side shows the original PCM and the right-hand side illustrates the result of restrictive revealing operation.

Figure 5.5: Restricted Information Revealing

5.5 Behaviour Hiding

It was mentioned before that the information hiding process precedes the actual study and is used for the support of an aim oriented analysis. Two different roles exist in this process. On the one hand the creator, who is observed for the construction of creativity maps and on the other hand the analyser who is responsible for the construction of behaviour descriptions which are then used in combination with any of the hiding or revealing operations. Figure 5.6 depicts the structure of the information hiding process.

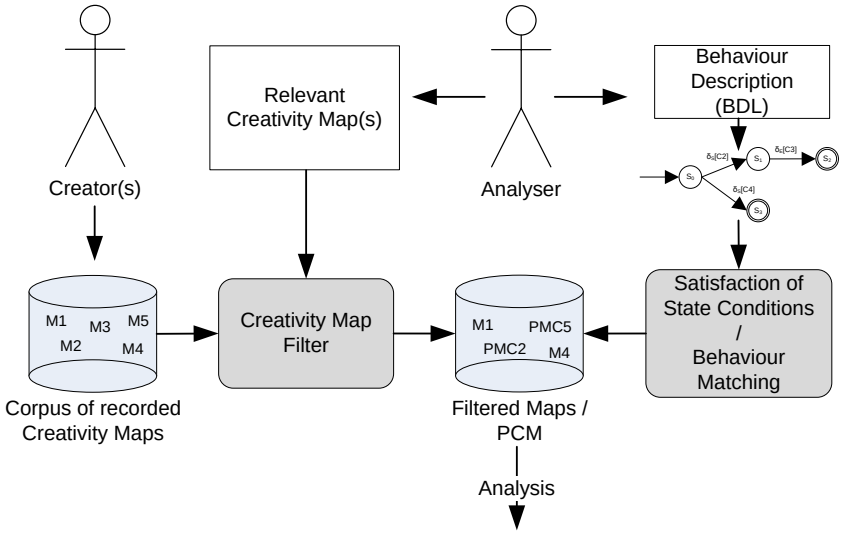


Figure 5.6: Information Hiding Process

The figure shows that one or more creator(s) record their creative processes, which then build the corpus of creativity maps. An analyser is now responsible for the concealment of irrelevant information with the aid of behaviour descriptions. It is necessary to convert this description into a processable form that can be used for pattern matching. One representation which satisfies these conditions and is used in the presented approach is an automaton [38]. It illustrates a structure similar to the creativity map and is therefore able to map any behaviour description in a convenient and reusable way. The automaton that is used in this thesis is called Behaviour Description Automaton (BDA). Specifications in the BDL are therefore transformed into BDAs which are then used in the information hiding process, as illustrated in the figure.

Creativity maps can additionally be filtered beforehand. The creativity map filter uses a pre-selection of maps, which can for instance be described with the categories that were explained in Section 4.3. This allows to reduce the corpus to those maps that are

relevant for the analysis, like the ones that belong to a particular project, creator, a whole domain or any combination that is specified as a customised set. This section describes the hiding process with the help of a BDA, including the specification and construction of the automaton from a behaviour description and the behaviour matching that includes the satisfaction of state conditions.

5.5.1 Behaviour Description Automaton (BDA)

The BDA needs to allow for the representation of any structure which can be described with the BDL, in particular actions, state conditions and markers. Actions are represented with regular transitions, whereas conditions and markers are transformed into special transitions, one for the start state $\delta_S(c)$ and one for the end state $\delta_E(c)$. Here, c represents a particular set of conditions, a marker or a mix of both, as the specification of a *State* in the BDL (Section 5.3.1) illustrates. The ANY transition that was described in Section 5.3.1 is also used in the BDA and every performed action of the creator is contained in the set A. The BDA in ϵ-NFA form is defined as a 5-tuple (S, Σ, T, s_I, F), containing:

1. A finite set of states (S)
2. A finite set of actions $(\Sigma \subseteq A)$
3. A transition function $(T : S \times (\Sigma \cup \{\epsilon, \delta_S(c), \delta_E(c), ANY\}) \rightarrow P(S))$
4. An initial state $(s_I \in S)$
5. A set of final states $(F \subseteq S)$.

A main disadvantage of the ϵ-NFA representation is the non-determinism of the transition function T. It allows to reach a set $P(S)$ of states for a single input symbol, which complicates the matching process of a behaviour and therefore the determination if it is accepted by the BDA. A ϵ-NFA also reduces the comparability of different behaviour description representations. Therefore, a DFA representation of the BDA is introduced, eliminating the previously mentioned disadvantages [60]. A BDA in DFA form is defined as a 5-tuple (S, Σ, T, s_I, F), containing:

1. A finite set of states (S)
2. A finite set of actions $(\Sigma \subseteq A)$

3. A transition function ($T : S \times (\Sigma \cup \{\delta_S(c), \delta_E(c), ANY\}) \rightarrow S$)
4. An initial state ($s_I \in S$)
5. A set of final states ($F \subseteq S$).

The definition of a behaviour description as an automaton allows for the specification of accepted behaviours, which describe the matched transitions of a creativity map. A behaviour has been defined as a sequence of transition in Section 4.5. Let $D = (S, \Sigma, T, s_I, F)$ be a BDA in DFA form and b be a particular behaviour. It is accepted, if there exists a representation of b in the form $t_0 \dots t_{n-1}$ where $t_i \in (\Sigma \cup \{\delta_S(c), \delta_E(c), ANY\})$ and a sequence of states $s_0, \dots, s_{n-1}$ where $s_i \in S$, so that:

1. $s_0 = s_I$
2. $s_i = T(s_{i-1}, t_{i-1}),\ i = 1, \dots, n-1$
3. $s_{n-1} \in F$.

This definition of an accepted behaviour is used for the information matching process, which will be described in this section.

5.5.2 Construction of the BDA

A common technique for the construction of an ϵ-NFA from regular expressions is the Thompson construction [93], which creates small ϵ-NFAs for simple expressions first and connects them successively. It specifies 5 distinct constructs. These are two basic ones for the ϵ- and action-transition together with one choice, one star and one sequence construct. An adaptation of this technique that allows for the conversion between behaviour description and BDA is described in the following. Necessary elements for this creation are transition, factor, behaviour, behaviour description and sub-description, which are identical to the components of the BDL that was specified in Section 5.3.1. Their conversion into ϵ-NFAs is illustrated and the successive construction process that finally results in the ϵ-NFA for the whole behaviour description is presented.

Preprocessing of Viewpoints, Exclusion Statement and Quantifiers

The BDL furthermore allows for the specification of edges in the form of viewpoints or place holders. As mentioned before, a viewpoint is a category that contains a set of actions.

However, the alphabet Σ of the automaton contains only actions, so that a viewpoint needs to be substituted by a choice between each of the actions that it contains. The first step of the BDA construction is the preprocessing of the Viewpoint and AnyEdge constructs that were used in the behaviour description. Given a viewpoint $v \in V$ with actions $a_0, \ldots, a_{n-1}$ where $a_i \in v$, the BDL construct

`<S-V=v-S>` is substituted by `<S-A=a`$_0$`|...|A=a`$_{n-1}$`-S>`.

The two expressions are equivalent and the automaton accepts any action a_i which belongs to the viewpoint v. The AnyEdge does not necessarily need to be substituted in the way described above. Instead, it is transformed into a transition of the automaton that is labelled with "ANY". Otherwise, it would need to be substituted with a choice of all actions that are part of the set A, which probably creates a more complicated construct. The Exclusion statement is substituted in a similar way, considering only the remaining actions. For instance, if the viewpoint *Artefact* with actions *Editing* and *Conceptualising* and the viewpoint *Literature* with actions *Reading* and *Comparing* are defined, the construct

`<S-E\{A=Editing,A=Comparing}-S>` is substituted by `<S-A=Concept.|A=Reading-S>`

and

`<S-V=Literature\{A=Reading}-S>` is substituted by `<S-A=Comparing-S>`.

The BDL allows for the specification of quantifiers, which represent different numbers of repetitions for constructs of the behaviour description. They need to be preprocessed as well to enable a simple construction process of the BDA. Given a Factor f, as defined in the BDL, the following cases are distinguished.

- $f[n..m]$ where $n = m$. This case is simply substituted by a sequence of n Factors f surrounded with parentheses. For example `<S-E-S>[2..2]` becomes

  `(<S-E-S><S-E-S>)`

- $f[n..m]$ where $m > n$. This case is substituted with choices between sequences of Factor f starting from length n and increasing until m, surrounded with parentheses. For example `<S-E-S>[2..4]` becomes
  `(<S-E-S><S-E-S>|<S-E-S><S-E-S><S-E-S>|<S-E-S><S-E-S><S-E-S><S-E-S>)`.

- $f[n..*]$ where $n = 0$. This case is simply substituted by $f*$.

- $f[n..*]$ where $n > 0$. This case is substituted by a sequence of n Factors f, followed by a Factor f with the * operator, all surrounded by parentheses. For example `<S-E-S>[2..*]` becomes `(<S-E-S><S-E-S><S-E-S>*)`

Transition

Transitions build the fundamental parts of behaviours. A transition consists of a start and end state, which are connected via an edge. An edge can either describe a single action or the choice between many actions, if a viewpoint has been substituted as a result of the previously described preprocessing step. Any edge with a single action is specified as `A=a`, where `a` describes the action label. The constructed ϵ-NFA of the specified edge is a simple transition also labelled with `a`, as depicted in Figure 5.7.

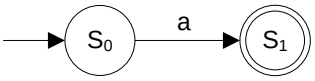


Figure 5.7: ϵ-NFA for an Edge

If a viewpoint has been preprocessed, it results in an edge with a choice between n actions, which is specified as `A=a`$_0$`|...|A=a`$_{n-1}$. This edge is created with a slightly adapted choice construct of the Thompson construction, as depicted in Figure 5.8.

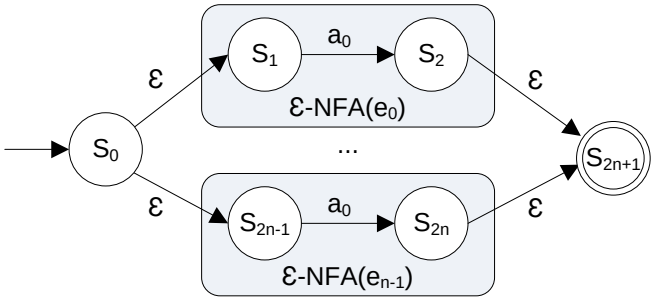


Figure 5.8: ϵ-NFA for an Edge with Several Actions

One new start state that is connected to all start states of the edges via ϵ transitions and one new final state that is connected with the final states of the edges via ϵ transitions were added. After the ϵ-NFA for the edge of a transition has been constructed, the start and end state of that transition need to be processed. Every state that is specified as a list of one or more state conditions with an optional preceding state marker is converted

into a δ-transition and concatenated with the edge transition afterwards. Conditions and markers for the start state are converted into a $\delta_S(c)$ and for the end state into a $\delta_E(c)$ transition. The ϵ-NFA for the transition `<C1-...-M,C2,C3>`, with a state condition C_1 for the start state and a state marker M together with list of conditions C_2, C_3 for the end state is depicted in Figure 5.9.

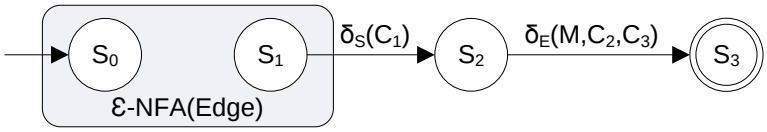


Figure 5.9: ϵ-NFA for a Transition with State Conditions

The $\delta_S(c)$ and $\delta_E(c)$ transitions are both appended to the edge transition in order to reduce the possibility of non-deterministic choices for the condition of the start state. For instance, the behaviour description `<S-A=Editing-S>|<C1-Reading-S>` specifies the choice between an editing transition without any conditions and a reading transition with a condition C_1 for the start state. The transition without a condition already includes C_1, which would lead to a non-deterministic choice when adding this condition in front of the ϵ-NFA for this particular edge.

Factor

A Factor is a choice between a Transition and a SubDescription followed by an optional `*` operator, which indicates zero or more repetitions of one of these components or an optional natural number, which specifies the exact number of repetitions. The construction of the ϵ-NFA for a transition was described above and the SubDescription ϵ-NFA will be discussed later. It is assumed that the automaton for any of these two elements e has been constructed. If the `*` operator is used, the construction process will create the corresponding ϵ-NFA for the expression `e*`, as depicted in Figure 5.10. A new initial and

final state are added to the element e, both connected to the according old states with ϵ transitions. Furthermore, the new states are connected with an ϵ transition as well as the old initial and final state of element e. This guarantees the possibility of zero or more repetitions.

Behaviour

A Behaviour is defined as a sequence of the previously described Factors. Each one is transformed into a corresponding ϵ-NFA as mentioned before. If more than a single Factor

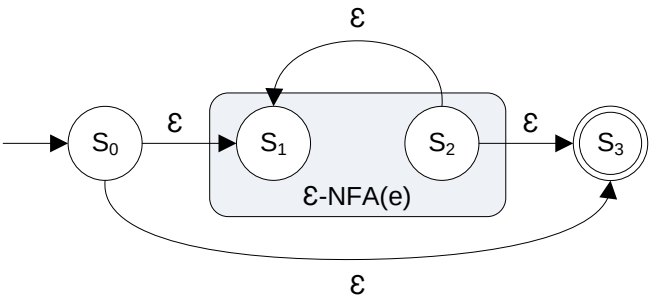


Figure 5.10: ϵ-NFA for a Factor with * operator

is used in the behaviour description, they will be concatenated with additional ϵ transitions to represent this sequence. Final states of a previous automaton are connected with initial states of the following one. The ϵ-NFA for the Behaviour $b = f_0\ f_1\ \ldots\ f_{n-1}$ with Factors f_i is created with the sequence element of the Thompson construction, as depicted in Figure 5.11

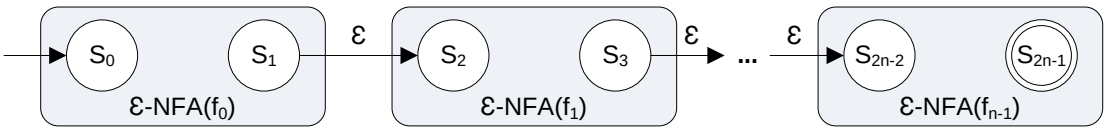


Figure 5.11: ϵ-NFA for a Behaviour

BehaviourDescription

A BehaviourDescription is specified as a choice between a number of Behaviour components. This is similar to the choice construct that was described above for several actions of the Transition element. One new start state that is connected to all start states of the Behaviours via ϵ transitions and one new final state that is connected with the final states of the Behaviours via ϵ transitions needs to be added. The ϵ-NFA for the BehaviourDescription $d = b_0|b_1|\ldots|b_{n-1}$ with Behaviours b_i is constructed similarly to the choice between a number of actions for the Transition construct, as depicted in Figure 5.8 above.

SubDescription

A SubDescription element allows for the encapsulation of BehaviourDescriptions. This is for instance important if a transition is followed by the choice between two other transitions, where surrounding parentheses `( ... )` can be used for the grouping of the choice construct. The ϵ-NFA for the SubDescription element is created similarly to the Be-

haviourDescription and is then integrated into the overall ϵ-NFA, according to the whole behaviour description. It is handled as one component that can for example be integrated into a sequence or choice with the help of ϵ transitions, as it was described before. This is similar to the other components that were explained above.

The previously described specifications allow for the construction of a BDA as ϵ-NFA from a behaviour description in BDL. Every BDA is created with this set of rules, beginning with small constructs like transitions and successively continuing to larger constructs such as factors, behaviours or sub-descriptions, until the behaviour description is fully modelled into an automaton. It was mentioned before that an ϵ-NFA is inconvenient for the behaviour matching, as its transition function T returns a set of states $P(S)$. This clearly complicates the pattern matching process, as it needs to check each of these paths in the automaton, which possibly requires backtracking whenever a successful or failed matching process finishes. For this reason, the ϵ-NFA needs to be converted to a DFA, which eliminates this disadvantage. The transition function of a DFA returns only a single state for one input symbol and therefore creates a deterministic choice. A common technique for the conversion between ϵ-NFA and DFA is the powerset construction. It maps a set of states that is part of the original ϵ-NFA onto a single state of the resulting DFA and therefore removes non-deterministic choices. The powerset construction is not further discussed in this thesis, as it can be used in its standard form [95] without any modifications.

5.5.3 Behaviour Matching

After the creation of the Behaviour Description Automaton (BDA), the creativity map will be traversed to search for matching behaviours. A behaviour is identified in the creativity map if it is accepted by the BDA, as defined in Section 5.5.1. Every accepted behaviour is marked and can be further processed based on whether the restriction, hiding, revealing or restrictive revealing operation is used. When the behaviour matching process traverses the creativity map, it needs to ensure that every transition is visited. Sequences of transitions, which describe the behaviours of a creator, should be visited until the final state first. Therefore, the traversal technique chosen for the behaviour matching is the depth-first search. It starts at the root state and stops, if the final state of a branch is reached in order to backtrack and traverse the next unvisited transition. A branch is therefore always traversed to its end first, before the next one is visited. Figure 5.12 depicts the depth-first search strategy.

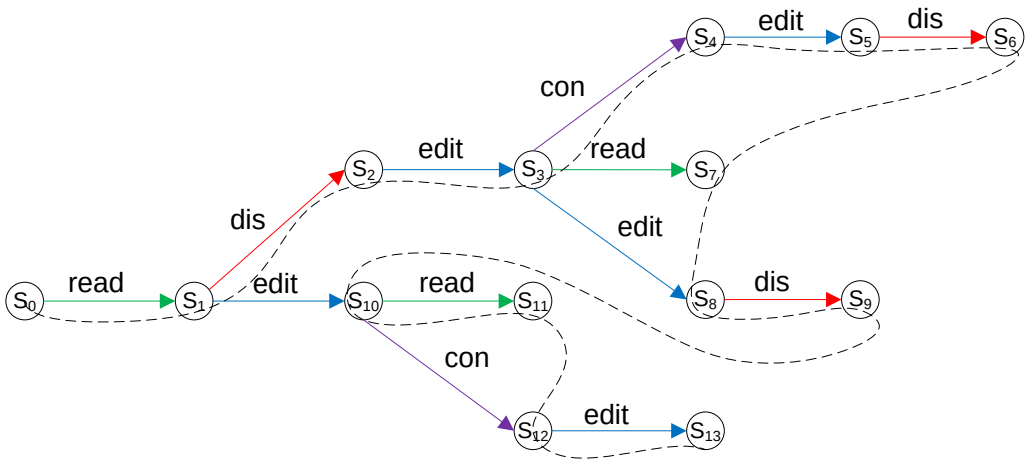


Figure 5.12: Depth First Traversal for the Behaviour Matching

As illustrated in the figure, the depth-first search starts at the root state S_0 and continues to visit the branches of the creativity map. After one branch is traversed, it tracks back to the starting state of the next branch. For example, after the state S_3 was visited and the traversal reaches S_6, which does not possess any outgoing transitions, the states S_6, S_5 and S_4 are backtracked until S_3. This state contains unvisited outgoing transitions, so that the traversal continues with S_7.

Behaviour Identification

The behaviour identification process splits into two depth-first traversals. The inner one is responsible for the behaviour matching. It checks each visited transition with the BDA. Every time a behaviour of the creativity map is accepted by the automaton, a matching behaviour of the behaviour description has been identified. The length of this accepted behaviour describes the number of transitions that need to be backtracked until the beginning of the matched sequence is reached. To identify the correct length, a counter will be incremented during the traversal process whenever a valid transition is visited and the BDA is able to move. If the automaton is unable to move, because the current transition of the creativity map cannot be processed, the depth first traversal for this branch is aborted and the backtracking starts. During this phase, the length is decreased by 1 with every step and the currently visited transition is marked as long as the length is still greater than 0. The beginning of the matching behaviour is reached when the length is equal to 0, which means that no further markings are necessary for this behaviour. Due

to the tree structure of the creativity map, behaviours can share a common prefix until the map branches. It is therefore necessary to remember the current state of the BDA at every branching node to be able to continue from this state when the next branch is visited. This means that whenever an unvisited branch is traversed, the BDA needs to be configured to the state that was active when the branching state was reached.

It was mentioned before that state conditions and state markers are mapped into δ-transitions. Whenever a state marker is reached during the traversal, its identifier is stored together with the current state label. This allows to retrieve the referred state from the creativity map, whenever it is needed at a later stage in the matching process. Conditions that are reached during the traversal can then either refer to a value that needs to be satisfied or to one or more markers. If a condition is not satisfied, the BDA is unable to move, which automatically leads to an abortion of the process. This is similar to a transition that is not accepted, as described above.

The previously described steps need to be performed starting from every state of the creativity map. This is realised with a second outer depth-first search that visits each state and then invokes the described steps. The information hiding process finishes, when all states have been visited by this second traversal.

The behaviour matching processes for the information revealing and restrictive revealing operations are executed in a very similar way, with the difference that they only consider hidden behaviours. Each sequence that is accepted by the BDA must therefore only contain α transitions. It was mentioned before that the original transition label is stored in an additional variable, so that it can be recalled if necessary. The traversal process itself can then easily be adapted by letting the BDA only move, if the original label is accepted and the transition is labelled with α. Any behaviour that satisfies these condition is marked, similar to the previously described technique.

Behaviour Processing

The last step of the behaviour matching process handles the marked behaviours with respect to the chosen hiding or revealing operation. This requires to traverse the transition of the creativity map a second time. The following four cases are distinguished.

- If the hiding operation has been used then all marked behaviours that are visited will be relabelled with α.

- If the restriction operation has been used then all transition that are not marked will be relabelled with α during the map traversal.
- If a revealing operation has been used then all marked transition will be relabelled with their original labels.
- If the restrictive revealing operation has been used then all hidden behaviours that are not marked will be relabelled with their original labels.

The final result can either be a creativity map or a PCM with a mix of visible and hidden transitions.

5.6 Minimising Partial Creativity Maps (PCMs)

The four operations described above are able to create a PCM that can contain high amounts of α-transitions. They can be hidden from the view for a compressed representation of the map structure which supports a better visibility and clearness. For the realisation of this procedure, it is necessary to identify all connected sequences of α transitions, also called α closures. An α closure for a state s describes all states and transitions that are reachable via sequences of α transitions. It is necessary to consider both components, as the revealing operations need to process the transitions and additionally access their original labels. A closure is not constraint to a sequential structure and can of course contain branching states. Once all of them were identified, they can be utilised to reduce the size of the creativity map, as described in the following collapsing/expanding and pruning approaches.

Collapsing / Expanding

Every α-closure that was identified in a PCM can be summarised into a single state, which then represents any collapsed sequence. This allows to reduce the visual size of a creativity map. An advantage of collapsing is that a map still contains all transitions, which enables the expansion of hidden parts if required. Figure 5.13 depicts an example of this collapsing process and illustrates how the visual size of a creativity map can be reduced. Especially for large creativity maps with many hidden transitions, the resulting map is very often much simpler to display. Relevant behaviours remain visible at the same time.

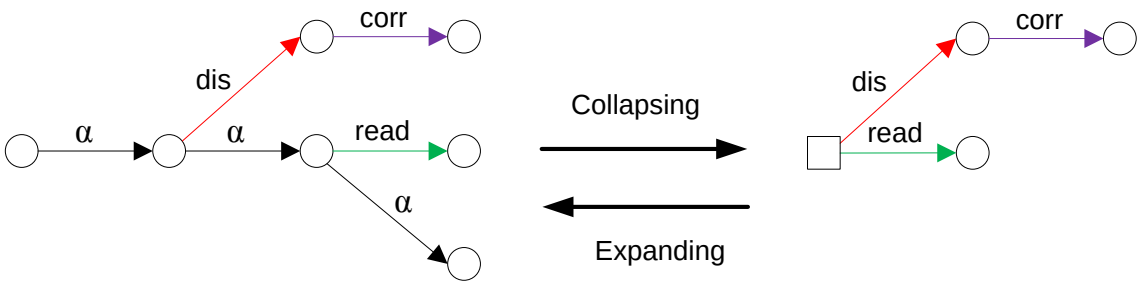


Figure 5.13: Collapsing Behaviours

The arrows in the figure illustrates the two directions of the collapsing and expanding operations. To distinguish states that contain collapsed closures from the regular map states, they are represented as squares. This highlights the existence of hidden transitions, which can become important for the analysis. Expanding a collapsed PCM will usually reconstruct the original PCM. However, it might not always be necessary to expand all transitions, like the revealing operations emphasised. This obviously depends on the requirements of the further processing.

The hidden α transitions are in many cases at least temporarily irrelevant. This means that the hiding process as described above and the collapsing of hidden transitions are very often performed straight after each other. Instead of displaying the creativity map with relabelled α transitions first, the collapsed PCM can directly be presented as the result. It was mentioned before that the collapsing operation only reduces the visual size of a PCM. Collapsed α closures are still part of the map, although they are possibly irrelevant for the analysis. If the hidden transitions are no longer needed, it is therefore disadvantageous to keep them in the map as they demand unnecessary storage space, especially if the size of the PCM is large. A pruning approach is introduced for this reason.

Pruning

A problem of the previously described collapsing approach is that the resulting PCM always has the same size as its original counterpart. Especially as hiding precedes the actual analysis and describes a preprocessing step, concealed information might not be relevant at all. The aim is therefore to reduce the map to only the visible states and transitions and remove any behaviour that is stored inside a collapsed state.

To minimise a creativity map, it is necessary to prune the identified α closures. This in turn means that these parts will be completely removed, with the disadvantage that they cannot be recalled or expanded later on. However, the advantage is that a map can be reduced in its physical size. This saves disc space and is beneficial for storage or copying activities. For instance, if several PCMs are created from one creativity map, pruning is able to prevent the storage of irrelevant and also redundant information. However, even if pruning produces an identical result as collapsing, whenever only observable behaviours are relevant, it is always important to remember that certain elements of a PCM have been removed. Especially the analysis might need to consider or even highlight this.

It was illustrated in the previous example that states which contain collapsed α closures are marked differently to distinguish them from the remaining ones. If the behaviours of these states are pruned, they will be marked differently as well. Pruning the map that is shown in Figure 5.13 produces the result which is depicted in Figure 5.14.

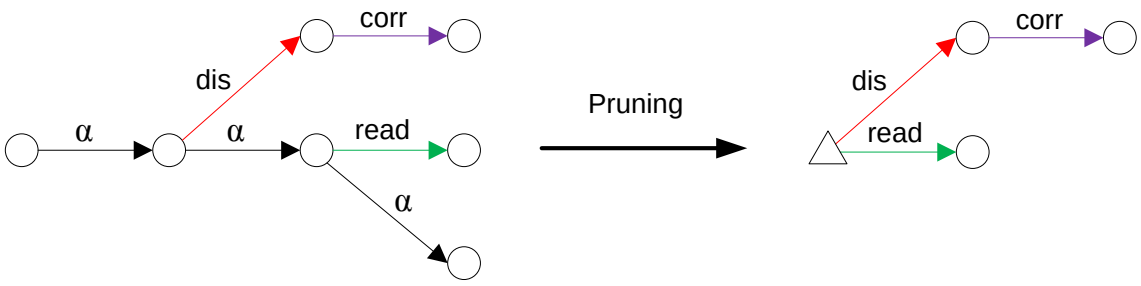


Figure 5.14: Pruning Behaviours

The illustrated PCM on the right-hand side looks very similar to the result of the collapsing process that is depicted above. However, the states which are represented as squares and contain hidden transitions in the previous example are now displayed as triangles. It emphasises the fact that actions were performed during this stage of the creative process, even if they are no longer part of the map. As mentioned before, this can be relevant for the analysis and allows to integrate additional and maybe useful information.

5.7 Summary

This chapter explained the information hiding approach. Observable and non-observable behaviours were distinguished and Partial Creativity Maps (PCMs), which contain a mixture of visible and invisible actions, were introduced. The Behaviour Description Language (BDL) has been developed and it was shown how this formal language enables a

convenient specification of behaviours in the form of behaviour descriptions. It also allows for the specification of state conditions, which is important whenever the value of a viewpoint should be considered. Hiding, restriction, revealing and restrictive revealing operations were introduced for the realisation of the approach. They all seamlessly integrate with the BDL and allow to hide or reveal any specific part of the creativity map. The conversion of a behaviour description into a Behaviour Description Automaton (BDA) was demonstrated and its utilisation for the hiding process was explained in detail. It was shown how the BDA is used to search for a set of input behaviours in a creativity map and how the backtracking phase of the depth-first traversal can be modified to mark any accepted behaviour. Collapsing and pruning techniques for the size reduction of creativity maps were presented towards the end of this chapter.

Chapter 6

Frequent Information Extraction

Objectives

- Specify frequency and the search space for frequent information.
- Present four frequency categories which are used for the analysis of the creative process.
- Introduce five frequent behaviour categories which describe the dynamism of the creative process.
- Describe the information extraction process and its combination with the information hiding process.
- Present the behaviour tracking process.

6.1 Introduction

This chapter explains frequent information extraction from creativity maps. It starts with the specification of frequency and the search space for frequent information. A model of four different frequency categories is introduced, which utilises three frequency thresholds for their distinction. It is then explained how these categories are relevant for the analysis of creative processes. Additionally to these categories, the dynamics of behaviours are specified with five distinct frequent behaviour categories. The importance of these categories is explained and their impact on the creativity support is presented.

6.2 Frequent Information in Creativity Maps

Frequent information is represented as frequent behaviours inside the creativity map, according to the specification of data, information and knowledge in Section 4.4. Creativity maps contain large amounts of this information and can grow quickly as mentioned before. However, each behaviour in a creativity map is built from a relatively small set of actions, which are freely chosen and can also change during the observation process. As this set is used in the creative process, some behaviours will occur more frequently than others. A writer for example starts to conceptualise a document first, followed by a discussion and an editing action, which is described by the behaviour $b = \overset{concept}{\longrightarrow} \overset{discuss}{\longrightarrow} \overset{edit}{\longrightarrow}$. This common pattern of a particular writer occurs frequently in the creative process and indicates that it is preferred over other behaviours.

Every pattern that is identified in one or more creativity maps has a particular frequency. The example behaviour described previously is a common sequence that can probably be identified often. A creation usually starts with a concept and discussions with colleagues, before the actual process begins. However, this is not mandatory and some writers listen to music before writing or other artists drink a glass of wine to gain some inspiration. Every creator has an own, personal set of frequent or even infrequent behaviours, which can be used for the assistance of the creative process. For example, if frequent behaviours can be categorised into "supporting" and "non-supporting" ones and stored in a database so that they are retrievable for comparison tasks, it is possible to utilise this information in order to assist the creator during the creation process. Hints that help in situations where the creator is "stuck" give a helping advice for solving current issues. Frequent information additionally describe the essential behaviours of the creativity map. Comparing creativity maps based on this core information allows to identify the dynamism of behaviours and can help to generate an useful overview for the construction of patterns.

Not only the most frequent behaviours are relevant for the analysis. It might be possible that a creator used a sequence of actions only rarely, but it describes a very efficient process. This still represents a personal behaviour that refers to the creator similar to a highly frequent sequence. Hints about this can help to increase the frequency and support the creative process. The difference between frequencies allows to describe the dynamics of behaviours. For instance, a preferred behaviour disappears, others emerge or previously infrequent ones are suddenly used commonly. Reasons for this might include a change of the work environment, collaboration, deadlines or other even external circumstances [81]. It is therefore essential to consider the dynamics of creative behaviours in parts of a single creativity map or also across a corpus of maps. This allows to observe behaviour

developments, which might be helpful to evaluate the creativity support. Results can possibly be analysed with respect to frequency alternations.

Once frequent behaviours were extracted from creativity maps, they can also be sorted or filtered based on the purpose of the analysis and post-processing. This can include particular actions, viewpoints, sequential orders or other properties of interest. Especially the previously described information hiding is able to conceal irrelevant information in a preprocessing phase. Information extraction can then be used as a major behaviour mining step that is built on top of it.

6.2.1 Specification of Frequency

The complexity and structure of a creativity map are essential properties that need to be considered for the frequency measurement. This includes its size as well as shared prefixes of two or more branches. For the calculation of frequency, it is therefore crucial to compare the occurrence rates of behaviours with similar length, as these have an equal chance of occurring. For example, longer behaviours are less probable, but also occur less often in a creativity map. In contrast, small sequences of actions occur more frequently. A particular behaviour of length n is therefore compared to the quantity of all behaviours of length n. This described metric normalises the frequency and adapts to the size of the creativity map. The following elements define the frequency of a behaviour.

- The *behaviour count c(b)* specifies the quantity of a behaviour b in a creativity map. Any occurrence of b is counted. The value might be higher for larger creativity maps, as more occurrences are possible. However, it is not mandatory, as large maps also contain behaviours that occur rarely.
- The *occurrence set* B_n contains the occurrences of every length n behaviour that can be identified in a creativity map. For example the set B_3 describes the occurrences of any behaviour b where $|b| = 3$. $|B_n|$ is the size of B_n, in particular the number of occurrences of length n.
- The *frequency f(b)* describes the relative frequency of a behaviour b in a creativity map. As mentioned before, the frequency of a behaviour needs to consider the size of the creativity map for a meaningful result. It is determined as the relation between the behaviour count $c(b)$ of a behaviour b and the amount of all occurrences of the same length $|b|$.

$$f(b) = \frac{c(b)}{|B_{|b|}|}$$

The proposed frequency metric implies that a behaviour needs to occur more often in a large creativity map than in a small one in order to gain an identical frequency. Common approaches in data mining and especially in frequent sequence mining [76] use a more simplistic definition of frequency [106], which disregards this information. Different requirements are usually the reason for this. These approaches are only interested in the number of sequences, which contain a certain pattern, without considering the quantity of this pattern in a single sequence. Hence, only a distinction between sequences without any occurrences and those with one or more occurrences is made. However, as creativity maps represent complex structures, it is necessary to ensure that the frequency is determined in a suitable way. The use of frequency thresholds for the further classification of frequent behaviours is discussed in Section 6.3.

6.2.2 Search Space for Frequent Information

The search space for frequent information specifies all behaviours that are relevant for the frequent information extraction process. It should be kept as small as possible to not consider irrelevant information which might distort the results. However, a size reduction during the extraction process is difficult, as a variety of frequencies is relevant for the analysis. It is also impossible to define a relationship between the frequencies of behaviours and sub-behaviours [57], because of the previously defined frequency metric. For instance, a length 2 behaviour that occurs 10 times in a creativity map has a different frequency than a behaviour of length 3 with identical behaviour count in the same creativity map, because the sets B_2 and B_3 contain different amounts of occurrences. It is therefore always necessary to determine the total number of occurrences B_n to be able to calculate the frequency of a particular behaviour of length n.

The original search space for frequent information can become very large and might contain at least temporarily irrelevant behaviours. It was described in detail in the previous chapter how this information in the form of viewpoints and actions can be hidden from a creativity map. This hiding process can be used prior to the frequent information extraction in order to reduce the amount of behaviours. It is additionally possible to further restrict the search space to behaviours of a particular length. Very long information, like for instance sequences with 10 or more transitions are probably irrelevant, as they are

difficult to handle and analyse. On the other hand, it is also possible that small behaviours of length 1 and 2 are not relevant. They are usually very frequent and hardly usable for the extraction of valuable information. The search space that is used for the frequent information extraction process is therefore restricted and specified by a number of conditions.

- The behaviour description that is used for the information hiding. It allows for the reduction of the search space to a particular and relevant subset of behaviours.
- The minimum length condition l_{min}, which specifies the minimum length of a behaviour. A behaviour is only extracted, if its length is equal to or greater than this threshold.
- The maximum length condition l_{max} which specifies the maximum length of a behaviour. A behaviour is only extracted, if its length is equal to or less than this threshold.

These restrictions allow the analyser to reduce and adapt the search space to the personal needs. Irrelevant behaviours can be hidden first and the remaining ones can then be further limited. A behaviour b is only extracted if $l_{min} \leq |b| \leq l_{max}$. The search space additionally constitutes only those creativity maps of the corpus, which are interesting for the analysis. They can be filtered in a preceding step, for instance by using the hierarchical categories that were described in Section 4.3. It enables to extract valuable information about a creator, certain projects or possibly a whole domain. This also saves storage space that would be needed otherwise to keep all maps in the corpus. It is always possible to further reduce or also enlarge the search space during the information extraction. Any of the previously defined conditions can be changed, which then leads to a modification of the corpus.

6.3 Frequency Categories

The frequency of a behaviour has been defined as the relationship between the behaviour count and the amount of all occurrences of similar length. It was mentioned before that not only the most frequent behaviours are relevant, but also uncommon sequences might be important for the analysis. This information can be used for a classification of behaviours, which illustrates the variety of frequencies that is present in a creativity map.

The four frequency categories *Preferred, Common, Uncommon* and *Zero* are introduced for this reason. They are depicted in Figure 6.1, each coloured differently. A distinction between them is realised with three thresholds. The *maximum frequency threshold* f_{max}, which describes the upper boundary, the *minimum frequency threshold* f_{min}, which is the lower boundary and the *zero threshold* f_{zero}, which specifies the lowest boundary. The four categories were chosen for the frequency mapping according to the semantics. For example, a behaviour that occurs very often is *preferred* by the creator and one that can be identified only rarely is *uncommon*. This allows each category to represent a particular frequency range and therefore the creator's personal behaviour preferences.

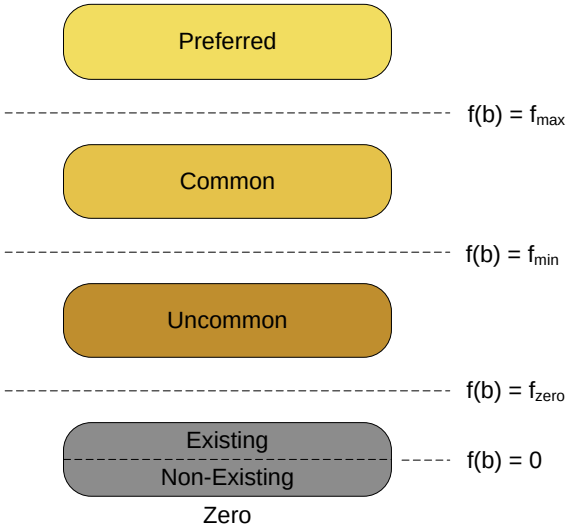


Figure 6.1: Frequency Categories

The categories build a hierarchy with the less frequent behaviours at the bottom and increasing frequency towards the top. The dotted lines represent the three previously mentioned frequency thresholds that are used for their distinction. The provided variability allows for an adaptation to the particular situation, search space and creativity maps. It might for instance be necessary to configure the thresholds differently for specific projects, creators or domains. Sufficient background knowledge about the creative processes needs to be provided by the analyser for their initialisation. This section explains the four illustrated categories in detail together with their purposes for the analysis and creativity support.

6.3.1 Preferred

The previously defined maximum frequency threshold f_{max} separates preferred behaviours from the remaining ones. This includes all behaviours with a frequency higher than or equal to f_{max}, which allows to specify the set of preferred behaviours B_{Pr} in the following way.

$B_{Pr} = \{b | f(b) \geq f_{max}\}$

Preferred behaviours specify sequences of actions that are performed more frequently than most of the other ones in a creativity map. For example, a behaviour such as $\overset{implementing}{\longrightarrow}$ $\overset{testing}{\longrightarrow}$ might be used often in the software development domain and can therefore be identified with a high frequency. However, this is not mandatory and cannot be generalised so easily, as some software developers might reject to use this particular pattern. Reasons for this can include different development models or given work flows that prohibit these activities. Personally preferred sequences can behave similarly, as they probably not share a high quantity among different creators. This is important to be considered in the study of the creativity maps.

The quantity of short behaviours is probably rather high in the preferred frequency category. Especially behaviours of length 1 or 2 have a high chance of repetition, which allows them to easily gain a higher frequency. However, as the number of short behaviours in a creativity map is higher than the amount of long behaviours, high frequent short behaviours need to occur more often, according to the frequency definition that was described above.

The amount of preferred behaviours will probably decrease with increasing lengths of the sequences. On the one hand, longer behaviours do not occur as often as short ones in the creativity map and on the other hand, more possibilities exist for their construction. Therefore, the preferred frequency category presumably contains only a small amount of longer behaviours, although this assumption also depends on the length constraint l_{max}. For example, if this condition is set to 2, long behaviour will not be extracted at all.

6.3.2 Common

The two frequency thresholds f_{min} and f_{max} that were mentioned before are used for the specification of the common frequency category. This includes all behaviours with

a frequency between these two boundaries, which allows to specify the set of common behaviours B_{Co} in the following way.

$$B_{Co} = \{b | f_{min} \leq f(b) < f_{max}\}$$

Common behaviours describe sequences which were performed with an average frequency, where average in this case means a frequency between the minimum and maximum thresholds. An example of a common behaviour might be $\overset{conceptualising}{\longrightarrow}\ \overset{implementing}{\longrightarrow}\ \overset{testing}{\longrightarrow}$, which extends the previously described preferred behaviour with an additional $\overset{conceptualising}{\longrightarrow}$ transition. The software developer commonly needs to conceptualise during the development process, but not as often as implementing or testing. Common behaviours can describe a wide variety of sequences and it is not necessarily the quantitative majority, as a lot of behaviours occur very rarely and are therefore categorised differently. However, they describe common activities of a creator or even a particular subset of creativity maps, such as a whole domain, a project or another constellation.

In contrast to the preferred frequency category, it is rather difficult to further specify the behaviours of this set. Common sequences can be shared in a domain or across creators with similar backgrounds, as they might be used to solve shared problems [10]. The distribution with respect to the lengths of these behaviours is probably similar to the preferred category. This means that the quantity decreases with an increasing sequence length. However, as the frequency threshold is lower than for the preferred set, more long behaviour are able to join this category.

6.3.3 Uncommon

The next category according to the descending order is the uncommon frequency category. The zero and minimum frequency thresholds f_{zero} and f_{min} that were mentioned before are used for its specification. It includes all behaviours with a frequency between f_{zero} and f_{min}, which allows to specify the set of uncommon behaviours B_{Un} in the following way.

$$B_{Un} = \{b | f_{zero} \leq f(b) < f_{min}\}$$

Uncommon behaviours describe sequences that were not performed very frequently by the creator. An example can be the behaviour $\overset{conceptualising}{\longrightarrow}\ \overset{testing}{\longrightarrow}$, which describes a probably uncommon sequential order of actions for a software developer. However, this particular behaviour is still performed in certain situations. These can be rarely occurring moments

or the creator might have performed the actions unconsciously. The uncommon frequency category usually grows larger than the previously specified ones, as a lot of behaviours are performed with a low frequency. Especially long sequences are often uncommon, as the chance of their repetition is very low. However, if the difference between the two frequency thresholds f_{min} and f_{zero} is well defined, it allows to keep this category small, which is beneficial for the extraction of useful information and knowledge.

Further assumptions about the behaviours of this category are difficult, as nearly any sequence of actions can be uncommon for a particular creator. They can describe the unusual treatment of a situation or just generally any behaviour which does not describe a common sequence, as it was mentioned before. The category will probably be larger and also contain more longer behaviours than the two previously defined ones. The amount of sequences also not necessarily decreases with an increasing length. However, this depends on the specification of the two frequency thresholds, which can have fundamental influence on the results.

6.3.4 Zero

Behaviours with a frequency less than f_{zero} or those that do not occur in any of these sets and therefore have a frequency of *0* belong to the zero category B_{Ze}. It is specified in the following way.

$B_{Ze} = \{b|f(b) < f_{zero}\}$, or more detailed

$B_{Ze_E} = \{b|0 < f(b) < f_{zero}\}$: existing set

$B_{Ze_N} = \{b|f(b) = 0\}$: non-existing set.

The zero frequency category splits into two different sets. On the one hand the existing behaviours which are present in the creativity map. They allow to keep the uncommon frequency set small, as behaviours which occur only very rarely can be assigned to the zero category with an appropriate f_{zero} threshold. On the other hand the non-existing behaviours, which describe all activities that were not performed by the creator(s). This set does not specify any particular sequence as such and is empty. Whenever a behaviour occurs in a creativity map which cannot be identified in a another map that is used for a comparison, it belongs to the non-existing part of the zero category. Otherwise all possible sequences that can be constructed from the set of actions would be assigned to this set, of course without the ones that belong to the other frequency categories. It would in turn

become infinite (unless l_min and l_max are specified) and be difficult or even impossible to handle. The non-existing part of the zero category therefore describes a symbolic set and its behaviours can only be determined in combination with the sequences of other frequency categories.

The existing subset of the category probably contains more behaviours than any of the previously defined sets. Especially long sequences that occur only very rarely are assigned to it and can then be treated similarly to the activities that were not performed at all. A main purpose of the zero category, as mentioned above, is to reduce the size of the uncommon frequency category and support the analysis of behaviour dynamics. It can for example be used for the identification of behaviours, which emerged from one creative process to the other. Especially if these behaviours were only performed very rarely, it might be relevant for the analysis. These dynamisms will be explained in the following section, which introduces five frequent behaviour categories based on the four frequency sets that were described in this section.

6.4 Frequent Behaviour Categories

Creativity maps can describe highly dynamic structures. The behaviours that are performed by a creator are able to change their frequency between several maps or even inside different time periods of a single map. Dynamics can be analysed and tracked through the creative process to reveal interesting information about the creator. When did a particular behaviour occur very frequently (preferred), or why did it suddenly change its frequency and occur only infrequently (uncommon) are only two examples of relevant questions that might be asked during the analysis. The four previously defined frequency categories are utilised for the introduction of frequency related behaviour sets. Figure 6.2 depicts five frequent behaviour categories.

The figure shows the different categories of frequent behaviours, in particular *Minimal (f_{max})/Minimal (f_{min})/Maximal Alternating, Emergent* and *Disappearing*. All of them are based on the dynamics in the frequency of a sequence in two or more creative processes or parts of it. A behaviour is then categorised based on an increasing or decreasing number of occurrences. The three *Alternating* categories describe the changes of behaviour frequencies in two directions, the *Emergent* category specifies increasing and the *Disappearing* category decreasing frequencies.

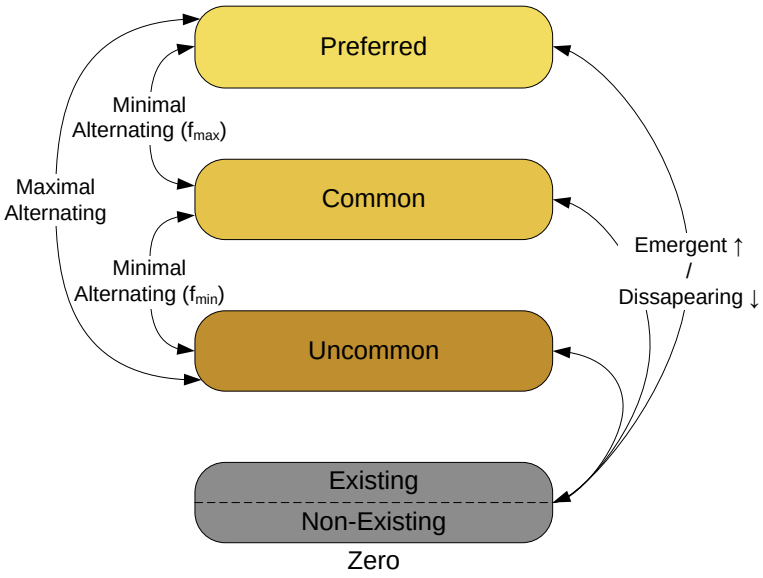


Figure 6.2: Frequent Behaviour Categories

For the specification of the frequent behaviour categories, it is assumed that the frequency categories which were introduced in the previous section were constructed for two creativity maps. In particular, Pr_1, Co_1, Un_1 and Ze_1 for the first map and Pr_2, Co_2, Un_2 and Ze_2 for the second one. Each frequent behaviour category is identified as $F_{C_1-C_2}$, where C_1 describes the frequency category of the first map and C_2 the one of the second map.

6.4.1 Minimal/Maximal Alternating

Minimal Alternating (f_{max})

The minimal alternating (f_{max}) category describes behaviours that change their frequency and alter between the frequency categories preferred and common. Particularly two different cases exist for this behaviour type. Firstly a previously common sequence that increases its frequency so that is becomes preferred and secondly the same scenario in the opposite direction. Minimal alternating (f_{max}) behaviours, as illustrated in Figure 6.2, are specified in the following way.

$F_{Co-Pr} = \{b|b \in (B_{Co_1} \cap B_{Pr_2})\}$: Minimal Alternating (f_{max}) Increasing

$F_{Pr-Co} = \{b|b \in (B_{Pr_1} \cap B_{Co_2})\}$: Minimal Alternating (f_{max}) Decreasing

This category represents very fine grained dynamics in the behaviour development, which means that the actual difference in the frequency is probably not very high. The set F_{Co-Pr} represents behaviours with a higher preference than before. Reasons for this, such as environmental situation or problems during the creative process may occur more often and lead to a repetition of these activities. In contrast, the set F_{Pr-Co} describes behaviours that are performed less often.

Minimal Alternating (f_{min})

The minimal alternating (f_{min}) category describes behaviours that change their frequency and alter between the frequency categories common and uncommon. Similar to the minimal alternating (f_{max}) set, two different cases exist. Firstly a previously common sequence that decreases it frequency and becomes uncommon and secondly the same scenario vice versa. The minimal alternating (f_{min}) behaviours, as illustrated in Figure 6.2, are specified in the following way.

$F_{Un-Co} = \{b|b \in (B_{Un_1} \cap B_{Co_2})\}$: Minimal Alternating (f_{min}) Increasing

$F_{Co-Un} = \{b|b \in (B_{Co_1} \cap B_{Un_2})\}$: Minimal Alternating (f_{min}) Decreasing

Minimal alternating (f_{min}) behaviours are very similar to the minimal alternating (f_{max}) ones and can be studied for the same fine grained changes in the behaviour development, which allows to determine low dynamics of activities. The set F_{Un-Co} contains behaviours that were possibly used in some rare situations and developed to become more frequent. In contrast, the F_{Co-Un} set represents sequences of actions, which are not performed regularly any more. Alternative ones may have been identified or previous problems and situations in the creative process are no longer treated in the common way. The creator might have also dismissed some of the common behaviours.

Maximal Alternating

The maximal alternating category describes behaviours that change their frequency and alter between the frequency categories uncommon and preferred. Differences in the frequencies of these behaviours need to be higher than for the minimal alternating sequences to allow them to skip over the common category. The specification is similar to the two previously defined sets and a behaviour b is categorised as maximal alternating if any of

the following two conditions is satisfied.

$F_{Un-Pr} = \{b|b \in (B_{Un_1} \cap B_{Pr_2})\}$: Maximal Alternating Increasing

$F_{Pr-Un} = \{b|b \in (B_{Pr_1} \cap B_{Un_2})\}$: Maximal Alternating Decreasing

Maximal alternating behaviours describe particularly high dynamics in the behaviour development. For example, if a creator performed an action very rarely (uncommon) and changes it to become very frequent (preferred) towards the end of the project, possible effects might be revealed. If a linkage between the changes and an enhancement in the efficiency [13] is possible, these patterns become useful for the creativity support. Abrupt changes of an activity might additionally be the effect of creativity assistance. For example, if support tools illustrate possible flaws in the creative process, sudden frequency increase as well as decrease of some behaviours can be the result.

6.4.2 Emergent

The emergent category contains behaviours that previously belonged to the zero category and are now part of one of the three categories uncommon, common or preferred. They were only rarely or not at all used in the previous creativity map and can now be identified due to an increasing frequency. The zero set distinguishes between existing and non-existing behaviours, which is considered in the emergent category as well. The letter "X" is used as a place holder and represents one of the three sets uncommon, common or preferred. A behaviour is emergent, if any of the following conditions is satisfied.

$F_{Ze-X} = \{b|b \in (B_{Ze_1} \cap (B_{Un_2} \cup B_{Co_2} \cup B_{Pr_2}))\}$, or more detailed

$F_{Ze_E-X} = \{b|b \in (B_{Ze_{1E}} \cap (B_{Un_2} \cup B_{Co_2} \cup B_{Pr_2}))\}$: existing set

$F_{Ze_N-X} = \{b|b \in (B_{Ze_{1N}} \cap (B_{Un_2} \cup B_{Co_2} \cup B_{Pr_2}))\}$: non-existing set.

The second and third line specify more detailed versions of the emergent behaviours, which distinguish between the existing and non-existing set in the zero category. Sets marked with "E", like $B_{Ze_{1E}}$ specify existing behaviours and sets marked with "N", like $B_{Ze_{1N}}$ specify non-existing ones. Figure 6.2 illustrates three possibilities for emergent behaviours with respect to their destination set. It allows for a further refinement of this category into *Uncommon Emergent, Common Emergent* and *Preferred Emergent*. This distinction supports a more detailed analysis but might increase its complexity at the same time. In

particular, the following three cases exist for a more detailed determination of emergent behaviours.

$$F_{Ze-Pr} = \{b|b \in (B_{Ze_1} \cap B_{Pr_2})\}$$

$$F_{Ze-Co} = \{b|b \in (B_{Ze_1} \cap B_{Co_2})\}$$

$$F_{Ze-Un} = \{b|b \in (B_{Ze_1} \cap B_{Un_2})\}.$$

Similar to the previous distinction between existing and non-existing behaviours in the zero category, it is also possible to further distinguish each of the detailed emergent behaviours that are shown above. One example is the *Preferred Emergent* category, which is additionally related to existing behaviours. It is specified in the following way.

$$F_{Ze_E-Pr} = \{b|b \in (B_{Ze_{1E}} \cap B_{Pr_2})\}$$

The remaining behaviours can be constructed analogously. An emergent behaviour might be studied for the identification of environmental or other external events that influenced the creative process. One example is collaboration [99], which possibly leads to unexpected activities and behaviours that were not present whilst the creator was working in isolation. Creativity, as it was mentioned in the third axiom of creativity (Section 3.2) is an emergent phenomenon. Exactly this property can be analysed and evaluated with the definition of the emergent behaviour category. Solutions to previously non-existing problems might be revealed, leading to an extraction of new information about the creative process. Another reason for emergent sequences of actions can be the hopping phenomenon, which was described in Section 3.3.

6.4.3 Disappearing

The disappearing category describes behaviours that change their frequency and alter between any of the three frequency categories preferred, common or uncommon and the zero set. A sequence of actions disappears, if its frequency decreases to less than f_{zero}. If two creativity maps are compared, then all behaviours which cannot be identified in the second map but in the first one automatically belong to this category as well. A behaviour is therefore disappearing, if any of the following conditions is satisfied.

$F_{X-Ze} = \{b|b \in (B_{Un_1} \cup B_{Co_1} \cup B_{Pr_1}) \cap B_{Ze_2})\}$, or more detailed

$F_{X-Ze_X} = \{b|b \in (B_{Un_1} \cup B_{Co_1} \cup B_{Pr_1}) \cap B_{Ze_{2E}})\}$: existing set

$F_{X-Ze_N} = \{b|b \in (B_{Un_1} \cup B_{Co_1} \cup B_{Pr_1}) \cap B_{Ze_{2N}})\}$: non-existing set.

Similar to the emergent behaviour category, the letter "X" is used as a place holder and represents any of the three sets uncommon, common or preferred, "E" stands for the existing and "N" the non-existing subset of the zero category. A further distinction between *Preferred Disappearing, Common Disappearing* and *Uncommon Disappearing* is possible, which might be helpful for a fine grained and detailed analysis. These disappearing behaviours are defined in the following way.

$F_{Pr-Ze} = \{b|b \in (B_{Pr_1} \cap B_{Ze_2})\}$

$F_{Co-Ze} = \{b|b \in (B_{Co_1} \cap B_{Ze_2})\}$

$F_{Un-Ze} = \{b|b \in (B_{Un_1} \cap B_{Ze_2})\}$

Similar to the distinction between between existing and non-existing behaviours in the zero category, it is possible to further distinguish each of the detailed disappearing behaviours that are shown above. One example is the *Preferred Disappearing* category, which is additionally related to existing behaviours. It is specified in the following way.

$F_{Pr-Ze_E} = \{b|b \in (B_{Pr_1} \cap B_{Ze_{2E}})\}$

The remaining behaviours can be constructed analogously. Disappearing behaviours can be studied for the extraction of information about the creator and actions that were neglected. The dismissal of these activities was maybe advantageous, but it is also possible that behavioural patterns changed and efficient sequences of actions with a major impact on the creative process became irrelevant. Creativity support tools can remind the creator about them to allow for a review of specific creation periods. Disappearing and emergent behaviours may also be closely linked, as new or more efficient behaviours possibly replace old ones. This relationship might as well be identified in the previously defined alternating behaviour categories.

6.5 Extraction of Frequent Behaviours

It was mentioned at the beginning of this chapter that frequent information represents the essential behaviours of creativity maps. Frequency and frequent behaviour categories

are utilised for the specification of a creator's behaviour development. For example, it is possible to construct several PCMs from a single creativity map, each representing a different creation stage. Comparing these maps based on frequent behaviours emphasises differences and similarities, which enable to reason about the dynamics of a creative process. It is of course necessary to construct the frequency categories first, before the frequent behaviour categories can be studied. Figure 6.3 illustrates the process of frequent information extraction.

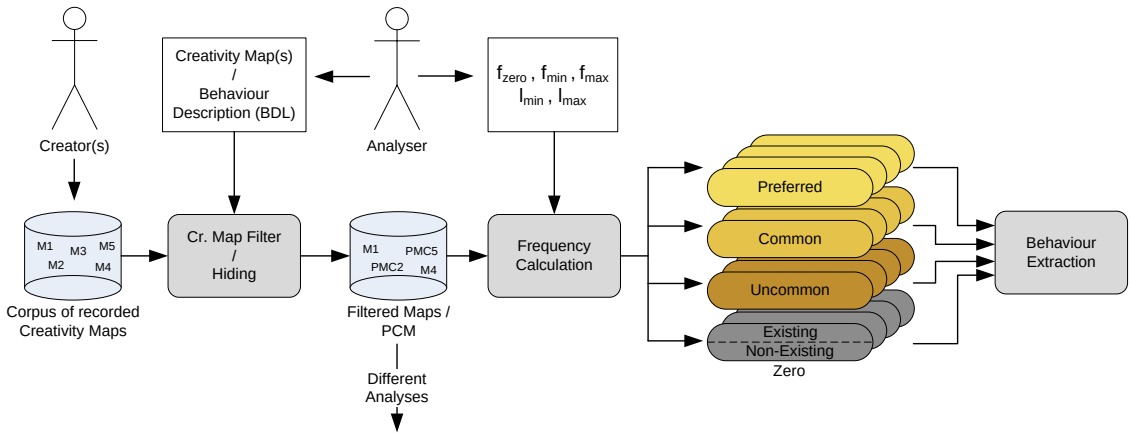


Figure 6.3: Frequent Behaviour Extraction Process

The figure shows that the corpus of creativity maps is constructed by one or more creator(s). Relevant creativity maps can be filtered beforehand, based on the purpose of the analysis and an additional corpus reduction is possible through information hiding, which was explained in detail in the previous chapter. Partial Creativity Maps (PCMs) can be constructed for an aim oriented information extraction approach with the help of behaviour description. Once the filtered and reduced corpus is created, the creativity maps can be further analysed. Other possible analyses, for instance visual exploration [27] are illustrated by the arrow that points to the bottom of the figure. This is one reason for a separate description of the hiding process, as it not necessarily interacts with the information extraction approach. The frequency calculation component uses the reduced corpus for the extraction of behaviours and their storage into the frequency categories preferred, common, uncommon and zero. These sets are constructed for each single creativity map or group of maps.

After the frequency calculation and the categorisation into the different frequency sets, the extraction of the behaviour dynamics follows. This step compares a number of categories for the determination of behaviour developments, which are represented by

any of the previously mentioned frequent behaviour categories *Minimal (f_{max})/Minimal (f_{max})/Maximal Alternating, Emergent* or *Disappearing*. Creativity support tools can utilise this information, as the examples that were described in the previous sections illustrated.

6.5.1 Frequency Calculation

The frequency calculation is responsible for the construction of different frequency categories and the extraction and classification of the corresponding behaviours from the reduced corpus of creativity maps.

Behaviour Identification

Each map is traversed with the depth-first search [91] and any identified behaviour is inserted into a frequency category. The length constraints l_{min} and l_{max} additionally reduce the number of behaviours in the search space and also affect the depth-first traversal. Instead of visiting only single transitions, a window of length l_{max} is sliding through the map and determining all behaviours with a length between l_{min} and l_{max}. They are identified by creating all suffix-behaviours which are present in the current window. Let suffix(b,n) describe the suffix of b, starting from position n, including the action label at n. For example, for the behaviour $b = \xrightarrow{edit}\xrightarrow{read}\xrightarrow{dis}\xrightarrow{edit}\xrightarrow{dis}$, the operation produces suffix(b, 3) $= \xrightarrow{dis}\xrightarrow{edit}\xrightarrow{dis}$ or suffix(b,5) $= \xrightarrow{dis}$. The window that slides through the creativity map usually contains a behaviour b of length l_{max}. Whenever this sequence changes, the sub-behaviours suffix(b,1), suffix(b,2), ..., suffix(b,$|b| - l_{min} + 1$) are inserted into the frequency categories. This is only one technique for the behaviour extraction and it might also be possible to use the prefix instead. However, if the suffix is used, the sliding window never needs to leave the creativity map and it is only necessary to fill it successively one time at the very beginning. If the prefix is used, the window always needs to slide until the last transition of every branch occurs at its beginning.

The creativity map usually contains several branches that need to be traversed. Two or more of them can share a common prefix, which needs to be considered only once to avoid counting certain behaviours again. It was mentioned before that the depth-first traversal returns to a state, whenever it is the source of an unvisited branch. The first transition of this branch adds up to an unvisited behaviour with the previous $l_{max} - 1$ transitions, going backwards from the branching state. Instead of revisiting these parts, the frequency

calculation process will store the last $l_{max} - 1$ transitions of the sliding window into a hash map, whenever a state with multiple outgoing transitions is reached. The key is the state label and the value specifies the window content. Every time a new branch is visited, the transitions are requested from the hash map and recopied into the window.

Construction of the Frequent Behaviour Categories

As the traversal continues, the behaviours are inserted into the frequency categories according to the previously mentioned suffix calculation technique. Every sequence is initially stored in the zero category and will then successively raise up into the other sets, as its map count increases. Figure 6.4 depicts this process.

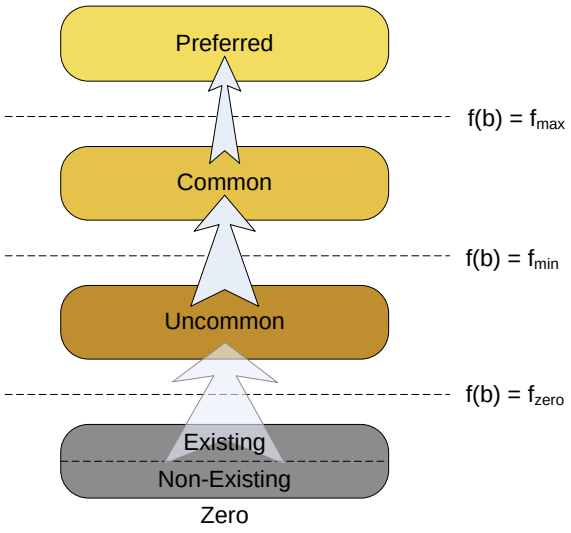


Figure 6.4: Frequency Calculation

The figure illustrates how the different frequency categories are filled successively as the behaviour count increases, whenever a sequence is identified. If any of the thresholds f_{zero}, f_{min} or f_{max} is reached, a behaviour moves up one level into the next set. Sequences will successively ascend from the bottom, which means that the *Uncommon, Common and Preferred* categories are empty at the very beginning. The size of the arrows represents the number of behaviours that change between the levels. It was mentioned before that the zero category is partly symbolic so that the transition to the uncommon set is depicted as a transparent arrow. Obviously, a large number of sequences that can be identified in a creativity map moves from the zero to the uncommon class, which is indicated by the largest arrow. A moderate amount of the behaviours in the uncommon category moves upwards one level. Some of the common sequences might be performed more frequently

and therefore become a member of the preferred category. This arrow is illustrated very small, as probably only few activities are preferred.

The arrows do not necessarily illustrate the correct amount of behaviours for each category, as this depends on the configuration of the three frequency thresholds. It can be assumed that the zero set and particularly the existing subset contain the most items, as especially long sequences of actions have a very low chance of occurring. In contrast, the preferred category possibly contains the fewest behaviours. The specification of the frequency thresholds might require additional knowledge and experience, as the third case study in Chapter 8 illustrates. It emphasises that these boundaries need to be configured rather small in order to gain useful results.

6.5.2 Behaviour Extraction

The next step in the information extraction process is the determination of the frequent behaviour categories. It was mentioned before that they describe the dynamics of the creative process for a creator or a specific combination of creativity maps. The extraction of frequent behaviour categories can only be performed if two maps are compared. The previously described step finished with the construction of the frequency categories for each creativity map. These are now combined and checked for overlapping sequences. The particular set that is being extracted depends on the purpose of the analysis. If for example the impact of collaboration [47] with respect to new behaviours should be studied, it is possible to determine all behaviours that emerged from one creativity map to the other. As mentioned before, it is also possible to specify emergent or disappearing behaviours in greater detail. Figure 6.5 depicts two examples for the behaviour extraction.

The figure shows the four frequency categories that were constructed for each creativity map. It highlights minimal alternating (f_{max}) behaviours that are part of the preferred set for the first map and become common in the second map as a result of a frequency reduction. The second example shows emergent behaviours, which were not present in the first map but belong to the uncommon frequent behaviour set of the second map. This particular type of behaviours can be specified more detailed as F_{Ze_N-Un}.

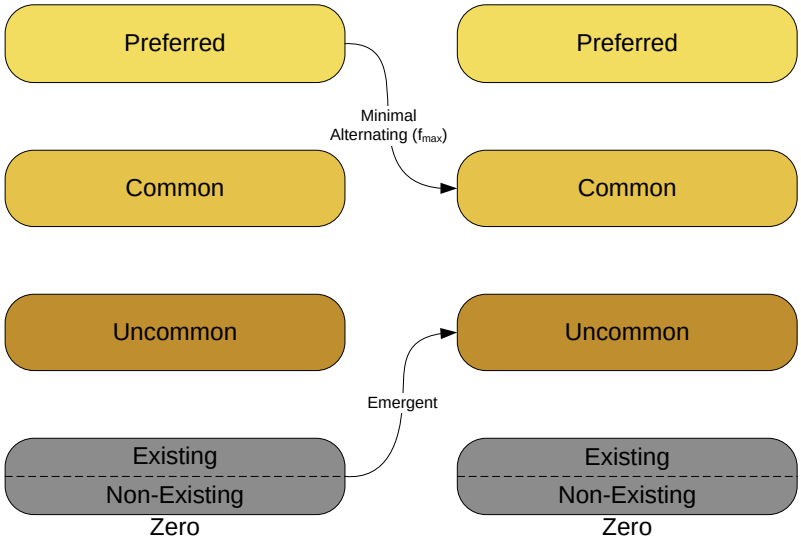


Figure 6.5: Behaviour Extraction

Behaviour Tracking

The previous example illustrated the dynamics of two behaviours for two distinct creativity maps. It is also possible to track one or more sequence(s) through a corpus of maps, which enables an exact determination of a behaviour development. The analyser might for instance be alerted, if an "uncommon" change is discovered to allow for the extraction of additional information like environmental influences. It can be utilised for the creativity support, when similar situations occur or related changes in the frequency of other behaviours are discovered. This might also be relevant for teams of creative individuals [9]. Figure 6.6 depicts an example of five creativity maps together with the illustration of one behaviour development. It is marked as a read line and alternates between the frequency categories as it is performed more or less frequent in the respective map.

The presented behaviour is uncommon in the first two maps and becomes maximal alternating afterwards, in particular F_{Un-Pr}, as its frequency increases. A decreasing frequency between M_3 and M_4 lets this sequence become minimal alternating (f_{max}). No change of the frequency category can be determined for the last two creativity maps. Especially the abrupt change between M_2 and M_3 might become important for the analysis in order to identify its reasons. However, the particular information that will be extracted and compared depends on the requirements and aims of the analysis, which is not part of this approach. The specified framework and processes allow for the extraction of any previously defined frequency and frequent behaviour category.

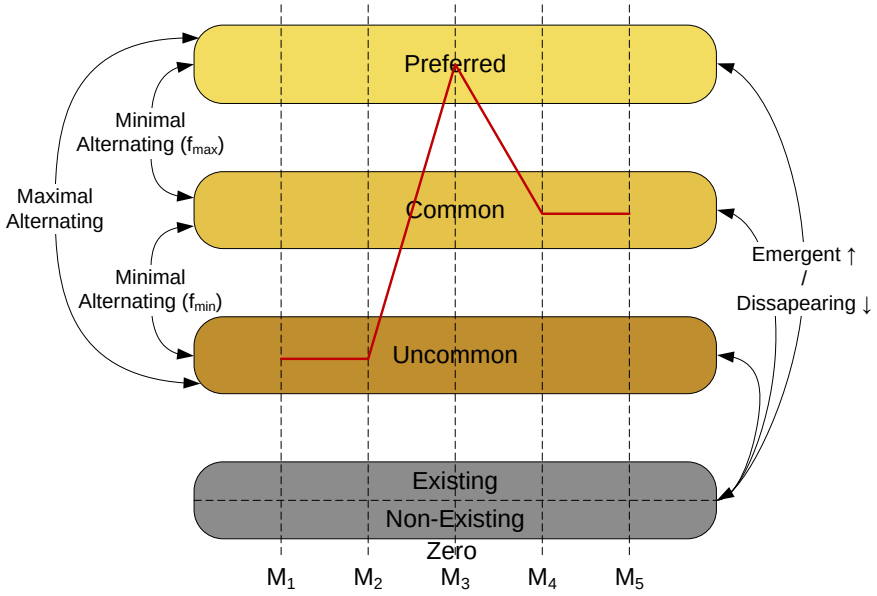


Figure 6.6: Development of a Behaviour

6.6 Summary

This chapter explained the frequent information extraction approach. A metric for frequency was introduced and the search space for frequent behaviours was specified. It was explained how the two length thresholds l_{min} and l_{max} are used for a further limitation of relevant sequences. The four distinct frequency categories *Zero, Uncommon, Common* and *Preferred* were defined with the aid of the three thresholds $f_{zero}, f_{min}, f_{max}$ and their purpose for the analysis was discussed. It was shown how a slightly modified depth-first traversal that utilises a sliding window can be used for the frequency calculation. Furthermore, the five frequent behaviour categories *Minimal* (f_{max})*/Minimal* (f_{min})*/Maximal Alternating, Emergent* and *Disappearing* were introduced and is was explained how they are used to determine the dynamism of a creative process. The frequent behaviour extraction process was discussed and its combination with the previously introduced information hiding approach was illustrated. An example of a behaviour development was illustrated in the last part of this chapter.

Chapter 7

Tool Support

Objectives

- Introduce the De Montfort Creativity Assistant (DMCA) and its layer based architecture.
- Describe the Collaborative Editor and its design.
- Introduce the De Montfort Creativity Mapper (DMCM) and explain its architecture and implementation.
- Present the Creativity Map Construction Engine (CMCE) together with its implementation and design.
- Describe the Information Mining Engine (IME)
- Present the Knowledge Repository and its architecture.

7.1 Introduction

This chapter explains the prototype tool support for the presented research. It discusses the implementation and design of the De Montfort Creativity Assistant (DMCA), which represents an extendible framework that has been developed for the support of creativity and creative processes. Its layer based architecture is explained and each of the layers

is presented in detail. The DMCA offers a pluggable design, which allows for the integration of tools from other domains. Architectures and implementations of the default components are explained and it is illustrated how they become part of the overall design. The facilities for the construction of creativity maps and information mining are presented. They illustrate the implementation of the proposed approaches. A mapping facility, particularly the De Montfort Creativity Mapper (DMCM), which is responsible for the observation of creative processes is introduced and its design and implementation are discussed.

7.2 De Montfort Creativity Assistant (DMCA)

The DMCA constitutes the overall tool support and implements the techniques and processes that were introduced in this thesis. It is designed to computationally support creativity and the creative process. The idea is to create a facility that is not restricted to a single domain, but allows to develop creative ideas in a multidisciplinary context. As the creator moves through the different stages of the creative process, the tool helps to analyse the gathered data and support the creation. The creative process and the artefact that is being created will be captured. Especially the performed actions must be present at any time to enable a creator the returning to previous stages.

7.2.1 Requirements

Ben Shneiderman introduced a number of design principles for creativity support tools, which were discussed in Section 2.5. The design of the DMCA follows similar principles and satisfies the following requirements.

Web-Enabled The DMCA should support a time and place independent access to allow for convenient work conditions. It should not be bound to any platform or workplace to gather a wide variety of creators. To fulfil these requirements, the DMCA has been designed as a web-enabled tool. The server side is responsible for the data storage and handling, the client side allows users to create and retrieve artefacts. A web-enabled tool has is advantageous, as clients are always using the latest version. No updates need to be distributed, they are instead integrated into the tool once and then automatically downloaded by each client. In contrast, a standalone application

would need to apply patches and it is never guaranteed that all users are installing them.

Data-Centric The previous chapter illustrated that data is the essential part in creativity support. It needs to be treated as a first-class citizen, which means that its creation, sharing and modification should be realised as conveniently as possible. A centralised data storage ensures that client information is saved in one place and allows external tools to access it for analysis purposes. It was mentioned before that the artefact and the creative process should be visible to the entire system.

Collaboration One design principle for creativity support tools, which is also mentioned by Shneiderman is collaboration. Tools should provide possibilities to collaboratively create artefacts and share them with others. This needs to be possible for a variety of domains, or even across them. It is therefore necessary to integrate facilities for different types of creators, like writers, artists or musicians. These tools need to be seamlessly integrated into the overall design and support team work.

7.2.2 Layer Based Design

The DMCA is designed as a modular, layer-based system, which provides the needed flexibility for a convenient handling and an easy adaptation to different domains. Each of these layers is responsible for specific tasks and is explained in the following. Figure 7.1 depicts the DMCA design, which splits into the components *De Montfort Creative Environment (DMCE), De Montfort Creativity Mapper (DMCM), Data Presentation, Creativity Mining Engine* and *Knowledge Repository.*

De Montfort Creative Environment (DMCE) The DMCE represents the tools that are mainly used by the creators. It contains facilities which allow for collaboration and data creation. These components are able to link several collaborators and enable them to work together time and place independent. It is an essential part of the system that collects and stores data in a centralised repository for further analysis.

De Montfort Creativity Mapper (DMCM) The DMCM is a separate component of the DMCA, which is used for the recording and visualisation of creative processes. It stores the constructed creativity maps in the Creativity Map Repository (CMR) of the Knowledge Repository. The DMCM can be invoked as an integrated facility inside the Collaborative Editor or as a standalone application.

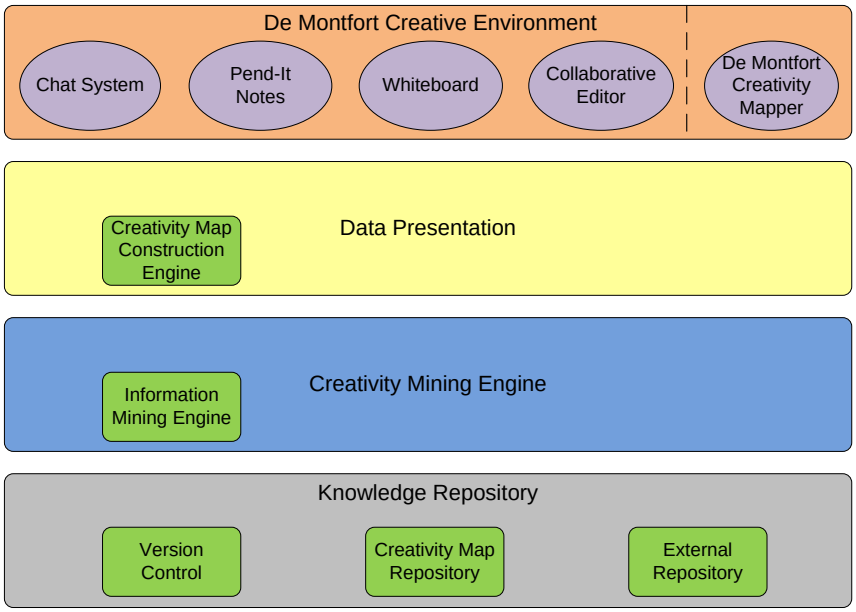


Figure 7.1: Architecture of the De Montfort Creativity Assistant (DMCA)

Data Presentation The Data Presentation layer transforms the user input into a computationally processable structure. It works bidirectionally and structures the data from the Knowledge Repository so that it can be displayed and vice versa. This is essential for the storage and retrieval of creativity maps. One part of the Data Presentation layer is the CMCE, which is responsible for the construction of creativity maps from the recorded creative processes.

Creativity Mining Engine The Creativity Mining Engine analyses the generated user data and creates new information and knowledge for the creativity support. One part of the Creativity Mining Engine is the Information Mining Engine (IME), which realises the hiding and extraction techniques that were described in this thesis.

Knowledge Repository The Knowledge Repository saves the data that is handled by the DMCA. It is responsible for the storage of any generated content, especially the creativity maps and artefacts. The Knowledge Repository splits into three subcomponents. Firstly the version control that stores the different versions of the artefact, secondly the CMR which contains all constructed creativity maps and thirdly the external repository that is linked to external resources like the internet or libraries in order to extend the internal knowledge and enable a broader analysis.

Each layer of the DMCA is flexible and can be extended with new components, for instance new links to external knowledge for the Knowledge Repository. The key concept of the

tool is to allow creators of any domain to capture their creative processes and generated artefacts in all stages of the creation processes. The collected data is then analysed to create valuable information and knowledge for the user feedback. It allows to assist the creative process of a creator and possibly improves and accelerates the process of artefact creation.

The development of the DMCA and its tools, as depicted in Figure 7.1 were developed by myself and Keno Buss, who is another Ph.D. student at the STRL. The components that I developed are in particular:

- Collaborative Editor
- De Montfort Creativity Mapper (DMCM)
- Creativity Map Construction Engine (CMCE)
- Information Mining Engine (IME)
- Version Control
- Creativity Map Repository (CMR)

It needs to be mentioned that the initial idea of designing the DMCM as a toolbar with several buttons was given by my supervisor Prof. Hussein Zedan. Issues regarding the components mentioned above were also sometimes discussed during meetings with Keno Buss and Prof. Hussein Zedan.

7.3 De Montfort Creative Environment (DMCE)

The DMCE is the top most layer of the DMCA. It integrates several collaboration components, which create a convenient environment for the creation of artefacts. One requirement at an early stage of the design was the integration of a collaborative editing facility, which allows to create, share and modify documents. Together with this, several other facilities were developed for the DMCE, namely a *Chat System, Whiteboard* and *Pend-It Notes*. This section presents an overview of the current DMCE components and explains their client/server architecture. The Collaborative Editor will be described in greater detail in the following section and is therefore not further mentioned here.

7.3.1 Collaboration Components

Chat System A Chat System enables the communication between creators to discuss modifications of the artefact (i.e. document) or give feedback. It distinguishes between several channels. On the one hand a global one, where all creators are able to communicate with each other and on the other hand special project channels. They allow for conversations about particular projects without interrupting or disturbing other collaborators.

Whiteboard The Whiteboard is a collaborative drawing space that enables the creation and sharing of sketches. It is more convenient in some situations to illustrate an idea or problem with a quick sketch. The Whiteboard can for instance be used as a storyboard where scenes and characters are designed. Collaborators have the choice between several shapes like rectangle, square or circle and can also draw freely with the pen tool.

Pend-It Notes Pend-It Notes enable collaborators to hold the results of the day, write down initial ideas or store reminders for important information. They represent a personal and unshared repository.

Additionally to the previously described collaboration components, a Project Management Facility that is responsible for the creation and management of projects was developed. It represents the main user interface element, which summarises and invokes the collaboration components and provides convenient access to them. As the initial version of the DMCA was developed mainly for the writing domain, projects are realised as hierarchies of documents. This allows to create and structure them in a flexible and customised way. Figure 7.2 depicts a screenshot of this facility.

The file operations implement functionalities for the creation and deletion of projects, files and folders. The important and export of files is currently not integrated, but might be realised in future developments. The task of the access control management is to provide mechanisms for file sharing with other collaborators and to manage their access during collaboration. The next two buttons start the Chat System and Whiteboard and the right-most button invokes the Pend-It Notes component. A double-click on a document, for instance *introduction* or *chapter1*, opens it in the Collaborative Editor. All previously described collaborative components are invoked from the Project Management Facility. If a new one is added, a new button should be integrated into this tool.

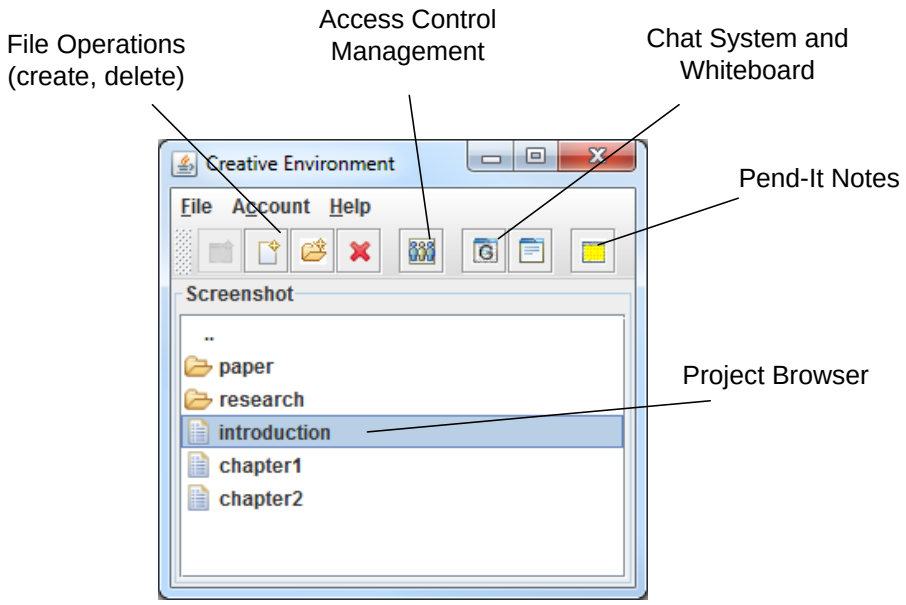


Figure 7.2: GUI of the Project Management Facility

7.3.2 Client/Server Architecture

Each of the collaboration facilities splits into a client and a server component. This satisfies the required web-enabled design, which was mentioned before. The client component allows for (collaborative) modification of an artefact and possibly links several collaborators. It enables the observation and capturing of the creative process and represents the interface between the user and the whole system. User input and feedback for creativity support are given through this component. Some parts of the client application depend on the domain, which it is used for so that its appearance can change. A writer might need different tools than a musician or a painter. This is realised via customisation of the DMCE layers.

The server component enables collaboration and is responsible for the handling of client requests and the storage and processing of user generated data. Any created content is transported from the client to the server and will be stored in the Knowledge Repository. It is furthermore processed by the Creativity Mining Engine for the creation of user feedback and support if necessary. To allow for collaboration, the server manages and exchanges all changes to an artefact. This information needs to be processed and distributed in the right order to keep the creators synchronized. This essential feature is realised via a centralised server that receives and processes any form of communication. To allow for a standardised information exchange between collaborators, any kind of data is first wrapped into events

and then transported. This encapsulates the data and supports a clear and convenient interface. It also allows for an easy extension, as new components can simply define new events.

7.4 Collaborative Editor

The Collaborative Editor is a required tool that needed to be integrated into the DMCE. It implements an editing facility, which enables writers to create, share and collaboratively edit of documents. Especially the synchronisation of a document that was opened by several users is an essential task, which can easily lead to different versions amongst the collaborators if not being performed correctly. This section presents the GUI of the Collaborative Editor and explains its design.

7.4.1 Graphical User Interface (GUI)

The Graphical User Interface (GUI) of the Collaborative Editor is kept clear and only the main editing features are implemented. This allows writers to focus on the document, instead of spending time for familiarising with the software. As a document can be shared for collaboration, additional features for a convenient handling of multiple clients were implemented. For instance, a collaborator list displays all creators with their current status (online or offline) and text passages of particular authors can additionally be highlighted with distinct colours. Figure 7.3 depicts a screenshot of the GUI. The DMCM is explained later in Section 7.5.

Editing Modes

One aim of the DMCA is to support the creative process of a creator. The Collaborative Editor implements this requirement in the form of multiple document modes. They model different periods that a writer joins during the creation of a document. Robert Dilts studied the creative process of Walt Disney [29] and developed the idea of these three modes. He realised that Walt Disney had three distinct phases in his creative process, the Dreamer, Realist and Critic phases. The model from Dilts was adapted for the Collaborative Editor and is called the Dreamer, Maker Critic model.

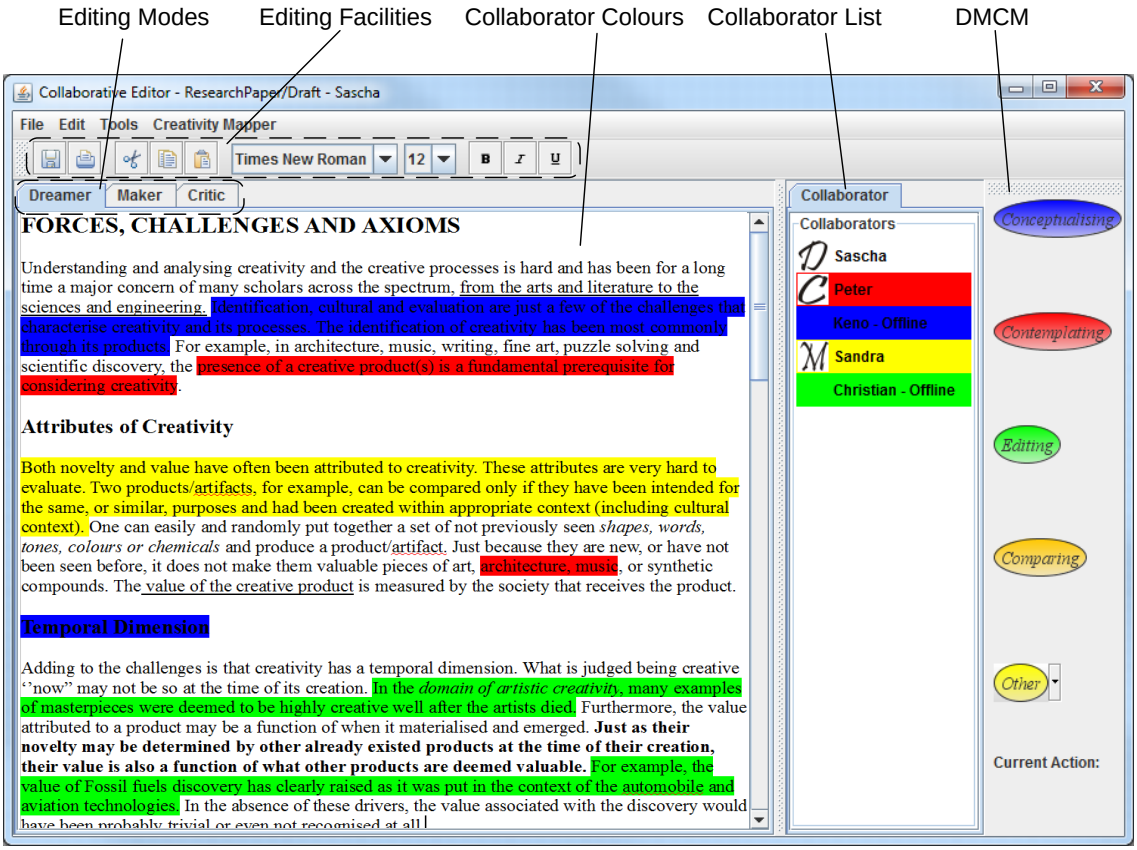


Figure 7.3: GUI of the Collaborative Editor

Dreamer Mode The Dreamer mode involves remembering, imagining, fantasising and inventing. One possible imagination is a stroll on the beach, enjoying the views and imagining distant countries across the ocean. The dreamer mode itself does not create the document, it collects material that is used in the construction process.

Maker Mode The Maker mode includes using this material and crafting, building, synthesising, interpreting, constructing and organising a document from it. This is similar to returning home from the beach, recalling the walk and using it as the basis of a story, poem or other work.

Critic Mode The Critic Mode validates the created work, forms the document and removes failures. For instance, the spelling is checked or paragraphs are rewritten.

The presented phases usually correspond to specific situations. For example, the Dreamer mode may occur in those circumstances that can usually be described as inspirational or meditative. This might be the case, when the writer is alone or in particular locations.

The Maker mode is often associated with the use of patterns, such as certain kinds of notebooks, pens, computers, or other tools. The Critic mode is the most distant phase, which is entered when the document or parts of it are finished. It describes situations, where the text is edited and rechecked to create the final result. Writers can cycle between the described modes several times, repeating this process in different permutations.

The Collaborative Editor supports these modes by splitting a document into three independent texts. To be able to distinguish between them, each mode is implemented as a separate pane. The resulting three panes are added to one tabbed pane for a convenient switching, as depicted in Figure 7.3. Text which is written in one mode cannot be overwritten by another one. Each of the modes is stored as a separate document in the version control system, which is described in Section 7.8.1. The Collaborative Editor implements a copy and paste functionality that allows to combine the modes or parts of them. The Collaborator List additionally displays the current mode of a creator with one of the letters (D)reamer, (M)aker or (C)ritic in front of the name. This enables users to identify which of the three documents each of the other creators is currently editing.

Editing Facilities

The editing facilities that are used to modify the appearance of the text are displayed in a toolbar at the top. From left to right, the buttons implement the following functionalities: save the document, print the document, cut, copy, paste text, change the font of the selected text, change the size of the selected text, format the selected text bold, italic and underline. They enable quick access to the most important editing features.

Collaborator Colours

Each collaborator is assigned a unique and randomly chosen colour when joining an editing session. It is a system-wide colour, which can also be used in other components if necessary. Double-clicking on a collaborator name in the Collaborator List highlights the text that was written by this creator in the particular colour. It enables other collaborators to get an overview of the document's composition. Once the colour for a collaborator has been switched on, newly inserted text will be marked in real time to enable a tracking of the editing activities.

The Collaborator List

The Collaborative Editor includes a Collaborator List that contains all creators who have ever edited the document. An additional status for each user (online or offline) allows to identify the current collaborators. Offline clients are not working on the document at the moment, but have previously edited it. They have the term " - Offline" appended to the names; otherwise only the name is displayed. New collaborators are added to the list, when they join a shared document the first time. As mentioned in Section 7.3 and particularly illustrated in Figure 7.2, collaborators can be added to a file with the Access Control Management.

7.4.2 Revision History

It was mentioned before that the created documents are stored in the version control system of the Knowledge Repository. Every time the document is saved by a user, a new version is created. To review this information, the Collaborative Editor includes a facility that allows to retrieve data from the version control. This enables to review different revisions and compare them in order to get an overview about their similarities and differences. Figure 7.4 depicts a screenshot of the previously described component.

The figure illustrates that the revision history facility is seamlessly integrated into the Collaborative Editor. The list on the right-hand side, next to the DMCM shows all revisions that were saved by the user or other participating collaborators. Each of them is displayed with a timestamp, allowing for the tracking of the document creation.

The revisions can be selected and displayed in the editor, with the possibility to revert to previous document versions. This is a flexible way to return to preceding stages of an artefact, as it was for example mentioned in the construction process of creativity maps. The facility provides the necessary information that are required in addition to the automatic comparison of states. Whenever a creator uses it, the creative process automatically returns to the corresponding state and continues from there.

It is furthermore possible to compare two document revisions in order to reveal similarities and differences. This scenario is illustrated in the screenshot. The two texts are shown beside each other at the top, labelled with the according revision numbers. A comparison of the documents based on characters, words and paragraphs is shown underneath. Each of the bar diagrams presents one of these properties. The pie chart illustrates the relation

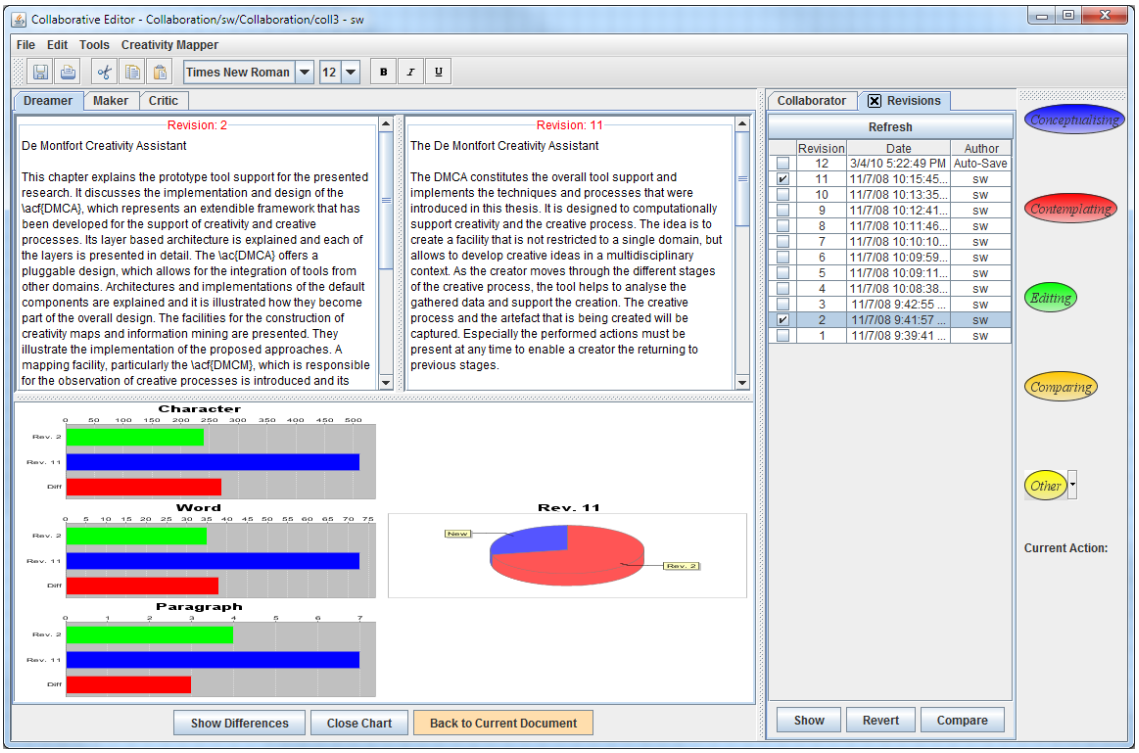


Figure 7.4: GUI of the Revision History Facility

between old and new content that can be identified in the newer revision of the comparison.

7.4.3 Design

It was mentioned before that each component of the DMCE splits into a client and server element. The communication between both components is realised via standardised events, which encapsulate the data and ensure a convenient communication. The UML class diagrams that are presented in this section illustrate only the most important classes and methods; parameters are additionally omitted to keep them clear.

Collaborative Editor Client

The client of the Collaborative Editor is represented by the GUI of the word processing tool that was described previously. Its editing operations are sent to the editor server, which is responsible for their correct distribution to the other users in order to ensure synchronised documents at all sides. Whenever events are received from the editor server,

the client needs to process them and integrate the results into the document in order to keep it synchronised. Figure 7.5 depicts the UML class diagram of the Collaborative Editor Client.

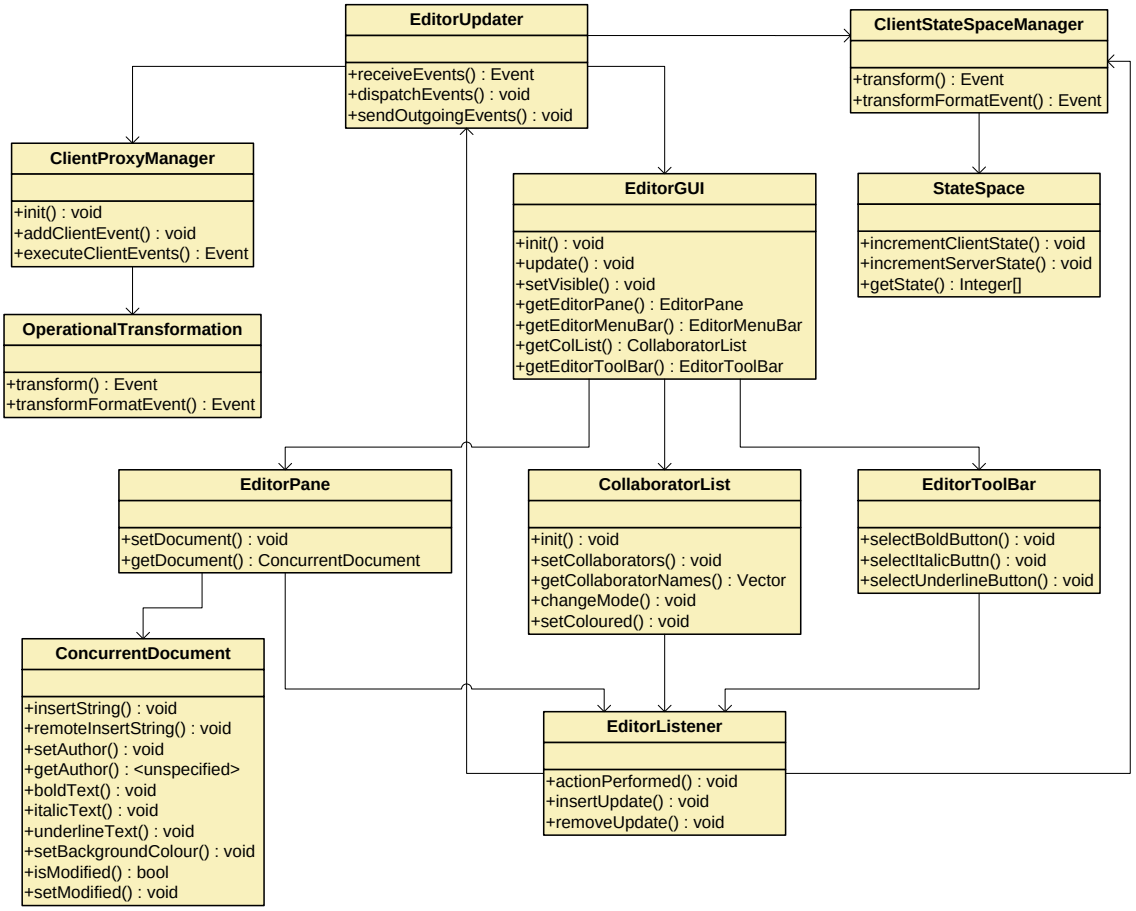


Figure 7.5: UML Class Diagram of the Collaborative Editor Client

The diagram in the figure illustrates how the Collaborative Editor Client splits into a number of elements. These are only the main classes which are necessary to describe its main functionalities. There exist additional classes for the GUI that are not described in further detail, as they are not essential for the understanding of the editor. The depicted components realise the following tasks.

EditorUpdater The EditorUpdater component of the editor client handles incoming events from the server. These events usually represent the editing actions, like insert or delete, which were performed by the other collaborators. The EditorUpdater integrates them into the document.

ClientProxyManager The ClientProxyManager is used by the EditorUpdater for the document synchronisation. It keeps track of the local and remote modifications to ensure updated documents on both sides. The component is also responsible for the construction and sending of events, which need to be broadcasted to the connected clients.

OperationalTransformation The OperationalTransformation synchronises the incoming editing events and adjusts them for a correct integration into the document. For instance, insert and delete operations need to be preprocessed as previously performed events are able to interfere. This mainly includes recalculating their positions.

EditorGUI The EditorGUI represents the wrapper component of the Collaborative Editor, which keeps all elements at a centralised place. This design protects components from unauthorised access and supports a clear and simple structure.

EditorPane The EditorPane contains the document and implements all editing and handling functionalities. For instance, it implements possibilities to change the text format or export the document to the local hard disc.

ConcurrentDocument The ConcurrentDocument represents the written text. It includes functionalities to insert text remotely or locally, which is important for the integration of modifications into the document. This component also allows to change the text appearance, like bold, italic or underline.

CollaboratorList The CollaboratorList is the list of all collaborators who have previously worked on the document or are currently editing it. It allows creators to identify collaborators and all users that have edited the document so far. The collaborator list explicitly displays the current online status of each creator.

EditorToolBar The EditorToolBar represents the tool bar that contains selected functionalities of the menu to enable a more convenient and probably faster access. It includes different buttons for format modifications like bold, italic and underline. These features are well known from other editors. As mentioned before, the Collaborative Editor implements the main functionalities of a word processor and the EditorToolBar contains only the essential editing facilities.

EditorListener The EditorListener implements the listener for all actions that are performed by the EditorGUI components. It is responsible for the transformation of these editing actions into events, which are then sent to the editor server.

StateSpace The StateSpace keeps track of local and remote document modifications.

ClientStateSpaceManager The ClientStateSpaceManager manages the state space and tracks the number of performed client and server operations.

Collaborative Editor Server

The Collaborative Editor Server represents the counterpart of the Collaborative Editor Client. It handles the communication between clients and is responsible for the document processing. This includes for instance loading or saving and the modification and distribution of client events. Every loaded document creates an editing session at the server side. Collaborators who decide to work on the document are able to join it. A session is responsible for the communication between the server and its clients and encapsulates the information from unauthorised access.

The Collaborative Editor Server keeps track of the document modifications and stores them in a version control system, which enables clients to recall any particular version if needed. Figure 7.6 depicts the UML class diagram of the Collaborative Editor Server.

The diagram in the figure shows the different components of the Collaborative Editor Server. Some of the elements, like the *OperationalTransformation* or *StateSpace* are shared between the client and server. The components that are depicted above implement the following tasks.

EditorServer The EditorServer is the central component that receives all editing events from the Collaborative Editor clients. It analysis each received event and invokes the SessionManager for the further processing.

SessionManager The SessionManager handles the sessions that are constructed for each loaded document on the server side. It is responsible for the construction and removal of all editing sessions as well as their user management, in particular adding and removing clients.

Session The Session encapsulates the (collaborative) editing activity into a component that manages one document and its collaborators. It processes the editing events of the clients, synchronises them with the document on the server side and distributes the resulting events to the collaborators. Each session is identified by a unique ID.

ServerDocument The ServerDocument implements the shared document of an editing session on the server side. It is split into three separate documents, each representing a different mode as it was explained in Section 7.4.1.

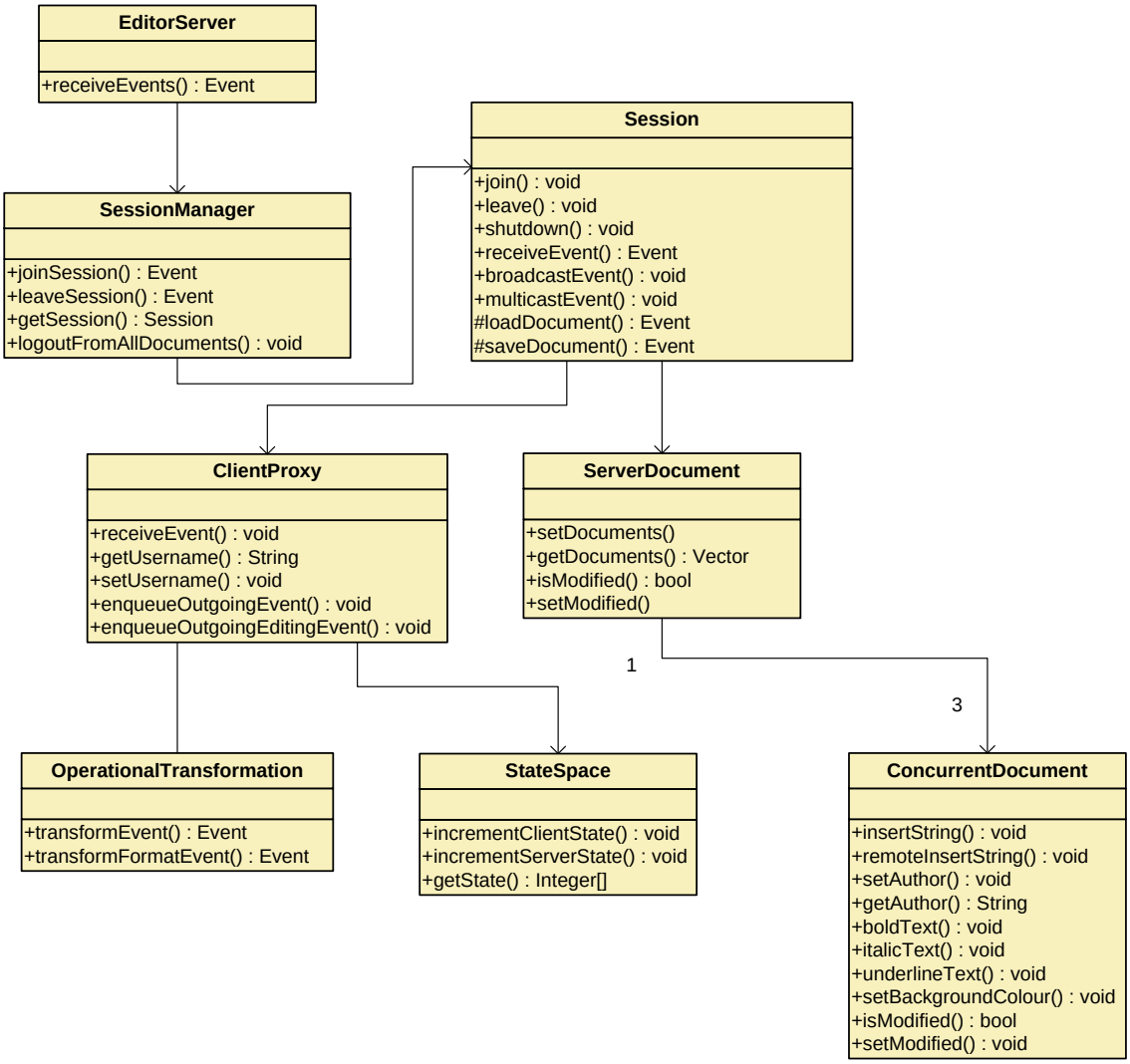


Figure 7.6: UML Class Diagram of the Collaborative Editor Server

ConcurrentDocument The ConcurrentDocument represents the written text. It includes functionalities to insert text remotely or locally, which is important for the integration of modifications into the document. This component also allows to change the text appearance, for instance bold, italic or underline.

ClientProxy The ClientProxy is an implementation of a single client on the server side. It monitors the modifications and keeps both the client and server document synchronised to ensure a failure free collaboration.

OperationalTransformation The OperationalTransformation synchronises the incoming editing events and adjusts them for a correct integration into the document. For instance, insert and delete operations need to be preprocessed as previously

performed events are able to interfere. This mainly includes recalculating their positions.

StateSpace The StateSpace keeps track of local and remote document modifications.

7.5 De Montfort Creativity Mapper (DMCM)

The De Montfort Creativity Mapper (DMCM) is the component of the DMCA which observes and visualises the creative process of a creator. It captures the actions that were performed during the creation of an artefact and stores them in the Knowledge Repository. As mentioned in Section 4.2.1, the approach for the creative process observation is a user interactive capturing system. A creator needs to interact with this tool and keep it updated about the currently performed action. The DMCM is then able to calculate and incrementally generate the creativity map for the momentary creation. This section presents the GUI of this tool and discusses its design afterwards.

7.5.1 Graphical User Interface (GUI)

The facility is designed as a toolbar that provides various buttons, each representing a single action (e.g. contemplating, reading, discussing). A user is required to press the button that most accurately describes the current mood. Actions are customisable and can be modified with a configuration tool. The DMCM can be used as an integrated facility within the Collaborative Editor, like shown on the right-hand side in Figure 7.3 or as a stand-alone application along with any other software or working environment. This section focusses on the latter use case.

De Montfort Creativity Mapper (DMCM)

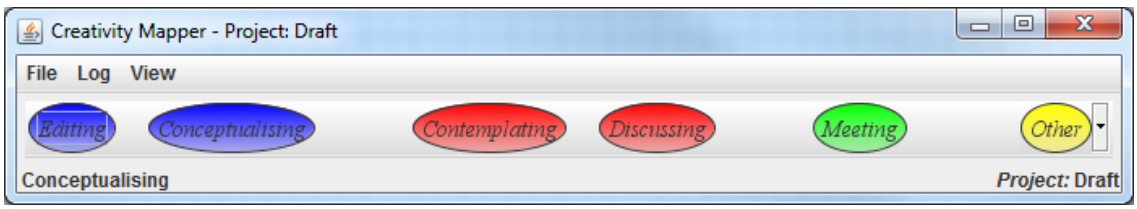


Figure 7.7: Screenshot of the DMCM

Figure 7.7 depicts a screenshot of the main DMCM facility. The buttons of the DMCM represent the personalised actions of a creator. A number of default buttons, which were defined according to the purposes of the writing domain are provided by the facility. They specify the actions *Conceptualising, Contemplating, Editing, Comparing* and *Other*. The Other button allows to quickly add new actions to the DMCM. Any currently performed activity is displayed at the right-hand side. The colour of a button represents a viewpoint, which means that the DMCM in the figure contains three distinct ones. A customisation of the DMCM is possible with the creativity mapper configuration dialogue.

Creativity Mapper Configuration

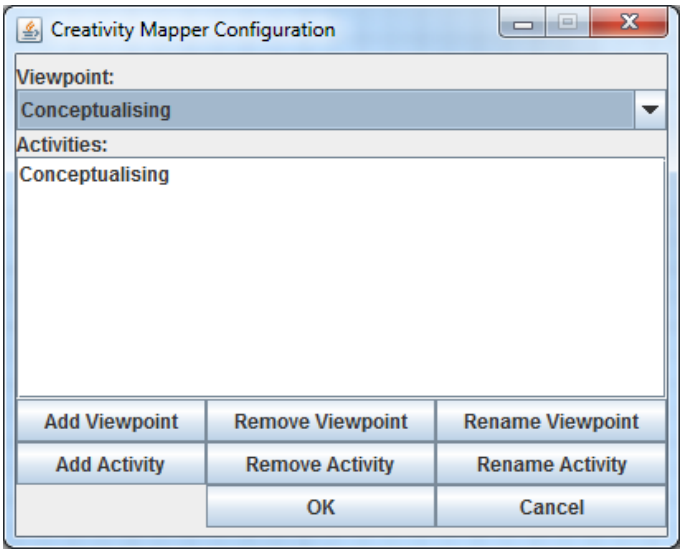


Figure 7.8: Screenshot of the DMCM Configuration Dialogue

Figure 7.8 illustrates a screenshot of the DMCM configuration dialogue. The creativity mapper configuration dialogue allows to modify the buttons of the DMCM. As it is shown in the figure, the dialogue contains a drop-down menu at the top that includes all viewpoints. Once a viewpoint was selected, the corresponding set of actions is displayed in the list field underneath. The creator can then use the buttons at the bottom of the dialogue to modify these elements. As illustrated, new viewpoints and actions can be added and existing ones can be removed or renamed. Each action creates a coloured button in the DMCM.

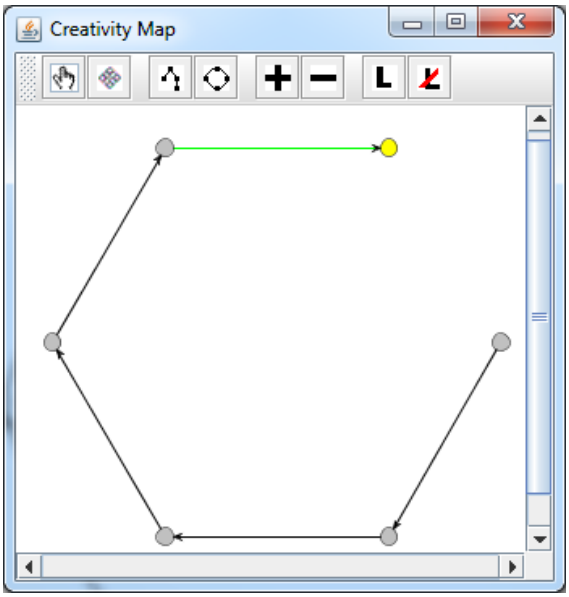


Figure 7.9: Screenshot of the DMCM Visualiser

Creativity Map Visualiser

The creativity map visualiser, as depicted in Figure 7.9, displays the creativity map for the current creation. It shows the transitions and states and allows clients to visually analyse a map. The majority of the window is used for the illustration in the white area. Buttons in the toolbar at the top implement browsing facilities. Starting from the left, the first button switches to picking mode, which enables the selection of one or more nodes and allows to drag them around. The second button switches to transformation mode in order to move the whole map in any direction and navigate to a particular position. The third button arranges the creativity map in a tree layout, which enables the visual identification of hierarchical structures. The fourth button arranges the nodes in a circle layout. The next two buttons implement zooming operations and the last two switch the transition labels on and off.

7.5.2 Design

The DMCM is designed as a client/server system similar to the collaboration components of the DMCE. It uses the existing structures for the communication between client and server, in particular the events. An action listener on the client side observes any action that is performed by the creator. Each action is wrapped into an event and send to

the DMCM server. This component construct several mapping sessions, similar to the editing session of the Collaborative Editor. The events are forwarded to these sessions and the corresponding creativity maps are constructed. The DMCM-Server is furthermore linked to the Creativity Map Repository (CMR) to enable a permanent storage of the collected data. This section illustrates the design of both the client and server component. The presented UML class diagrams show only the most important classes and methods; parameters are additionally omitted to keep them clear.

DMCM Client

The DMCM client includes the three components that were discussed previously. The UML class diagram of this component is depicted in Figure 7.10.

CreativityMapperGUI The CreativityMapperGUI represents the wrapper component of the DMCM, which keeps all GUI elements at a centralised place. This design protects components from unauthorised access and keeps the structure clear and simple.

CreativityMapperMenuBar The CreativityMapperMenuBar represents the menu of the DMCM.

CreativityMapperToolBar The CreativityMapperToolBar is the main component that is responsible for the recording of the creative process. A creator is required to press one of its buttons for every performed action in order to guarantee a correct observation.

CreativityMapperListener The CreativityMapperListener is the listener for the performed actions. This component transforms the observed activities into events and forwards them to the EventHandler.

EventHandler The EventHandler sends the observed events to the server component of the DMCM, where the creativity map is constructed and stored in the CMR.

CMConfigGUI The CMConfigGUI is a GUI for the customisation of the DMCM. It allows to add, remove or rename viewpoints and actions, which in turn modifies the appearance of the mapper.

CMConfigManager The CMConfigManager manages the action and viewpoint configurations of the DMCM. They are stored on the local hard disc to allow for a

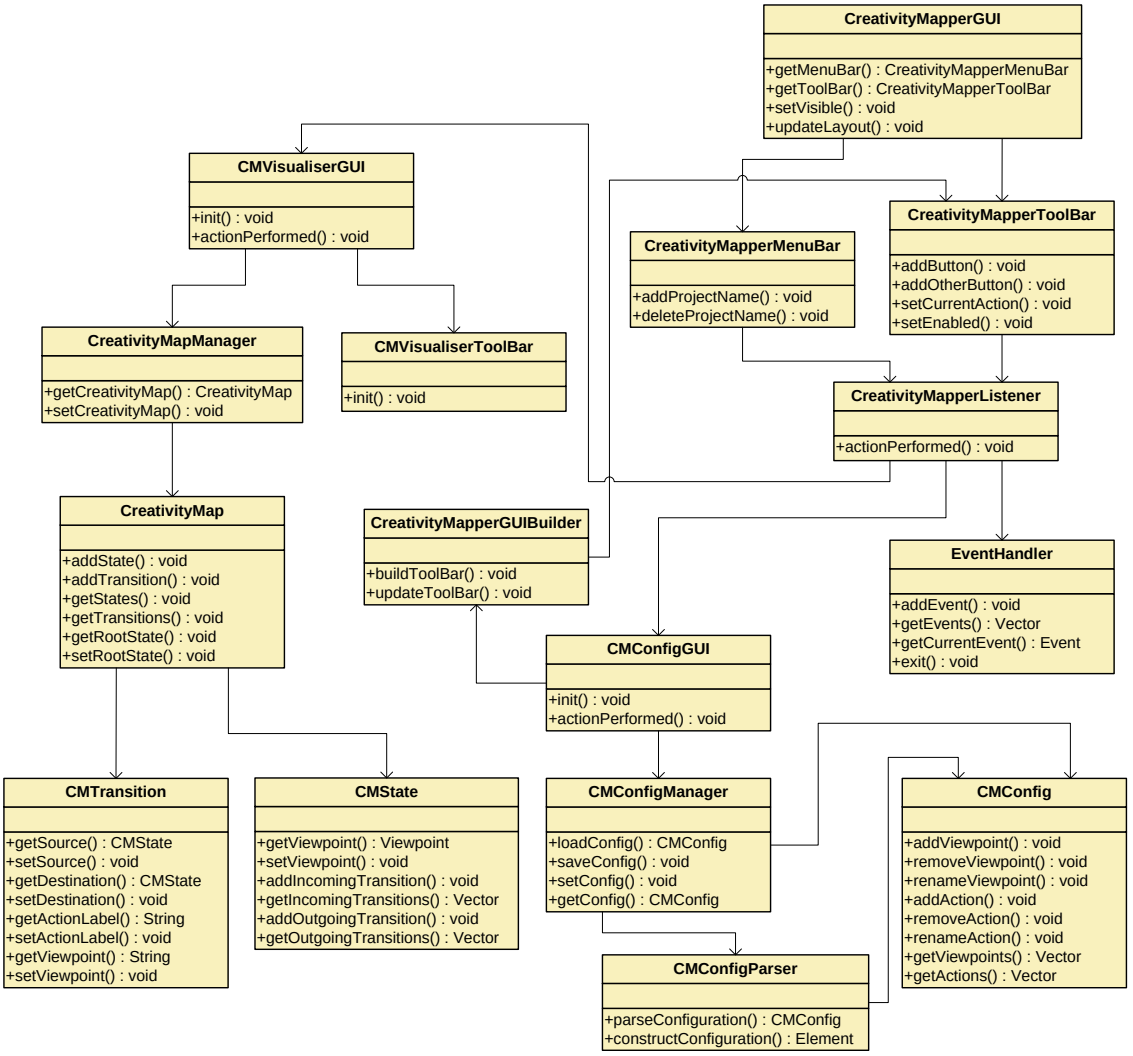


Figure 7.10: UML Class Diagram of the DMCM Client

continuous work with a customised DMCM after it has been configured once. The CMConfigManager is the only component that is able to access a configuration.

CMConfigParser The CMConfigParser parses a configuration and constructs a CMConfig object as representation. It is also used vice versa for the creation of a storable configuration from a CMConfig object.

CMConfig The CMConfig represents the configuration of viewpoints and actions for the DMCM. It stores the name of each viewpoint together with its set of actions.

CreativityMapperGUIBuilder The CreativityMapperGUIBuilder creates the GUI of the DMCM. It is responsible for its initial construction and modification, whenever

the configuration was modified. It guarantees a dynamic and interactive adaptation of the user interface.

CMVisualiserGUI The CMVisualiserGUI implements the main window that is used to visualise the creativity map on the client side. It allows users to zoom in and out of a map, display and hide transition labels and arrange it according to different graph layouts [52] [79]. Actions that are performed by the creator are added and visualised in real time.

CMVisualiserToolBar The CMVisualiserToolBar is the tool bar that implements the previously mentioned functionalities.

CreativityMapManager The CreativityMapManager is a major component that contains and manages all creativity maps during runtime. Access to the maps is only granted by this component.

CreativityMap The CreativityMap represents the corresponding data structure of a creativity map. It contains CreativityMapStates and CreativityMapTransitions.

CreativityMapState The CreativityMapState represents a state of the creativity map. It contains the viewpoints of the creation, for instance artefact or time. As mentioned in Section 3.3, time is realised as a timestamp, representing the moment when the activity of the state's outgoing transition begins.

CreativityMapTransition The CreativityMapTransition represents a transition of the creativity map. It contains a start and end state, an action label and the name of its viewpoint. An additional variable that stores the original action label during a hiding process is included as well.

DMCM Server

The DMCM server stores and manages the creativity maps that were constructed for the users of the DMCM client. Figure 7.11 depicts the UML class diagram of the DMCM server.

CreativityMapperServer The CreativitMapperServer receives the events that are sent from a DMCM client. Each received event is analysed and processed by the CMSessionManager.

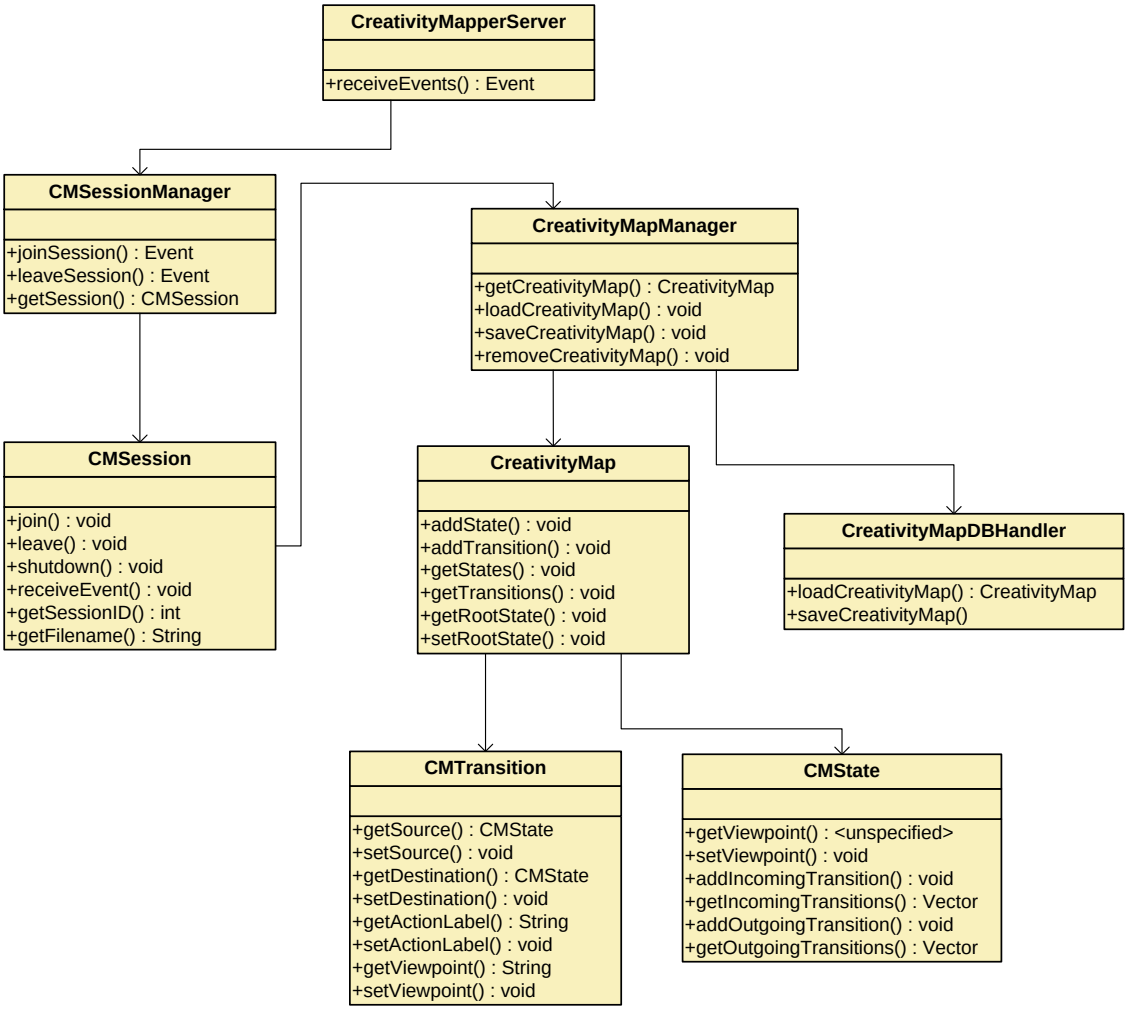


Figure 7.11: UML Class Diagram of the DMCM Server

CMSessionManager The CMSessionManager handles the sessions that are created for each creativity map construction. If the DMCM is executed as an integrated facility within the Collaborative Editor, a creativity mapping session is started together with every editing session. It automatically contains all collaborators. If the DMCM is invoked as a standalone application, a unique session is constructed for each user. The CMSessionManager is responsible for the creation and removal of sessions as well as their user management, in particular adding and removing clients.

CMSession The CMSession encapsulates all necessary information of the creativity mapping process. It can be joined by several clients and receives actions from the session manager. These events are processed by the CreativityMapManager.

CreativityMapManager The CreativityMapManager incrementally creates and manages the creativity maps for every client of a mapping session with the help of the CMCE. It invokes the CreativityMapDBHandler whenever a map should be stored in or loaded from the CMR.

CreativityMapDBHandler The CreativityMapDBHandler represents the link to the CMR. It is responsible for the transformation of a CreativityMap object into a storable format and vice versa.

CreativityMap The CreativityMap represents the corresponding data structure of a creativity map. It contains CreativityMapStates and CreativityMapTransitions.

CreativityMapState The CreativityMapState represents a state of the creativity map. It contains the viewpoints of the creation, for instance artefact or time. As mentioned in Section 3.3, time is realised as a timestamp, representing the moment when the activity of the state's outgoing transition begins.

CreativityMapTransition The CreativityMapTransition represents a transition of the creativity map. It contains a start and end state, an action label and the name of its viewpoint. An additional variable that stores the original action label during a hiding process is included as well.

7.6 Creativity Map Construction Engine (CMCE)

The CMCE is a component of the Data Presentation Layer. It is responsible for the construction of creativity maps from the initial sequential creative process, as explained in Section 4.2. It is necessary for this process to identify similar states and redirect their outgoing transitions to create a map structure. A main requirement of the CMCE is flexibility in order to enable the comparison of several viewpoints and the use of different metrics. Figure 7.12 depicts the UML class diagram of the CMCE. It represents only the most important classes and methods; parameters are additionally omitted to keep the diagram clear.

StateComparatorInterface The StateComparatorInterface defines an interface that needs to be implemented by any concrete state comparator. It contains one *compareStates* method, which is responsible for the comparison of two states. This design allows for a flexible exchange of comparators and their metrics.

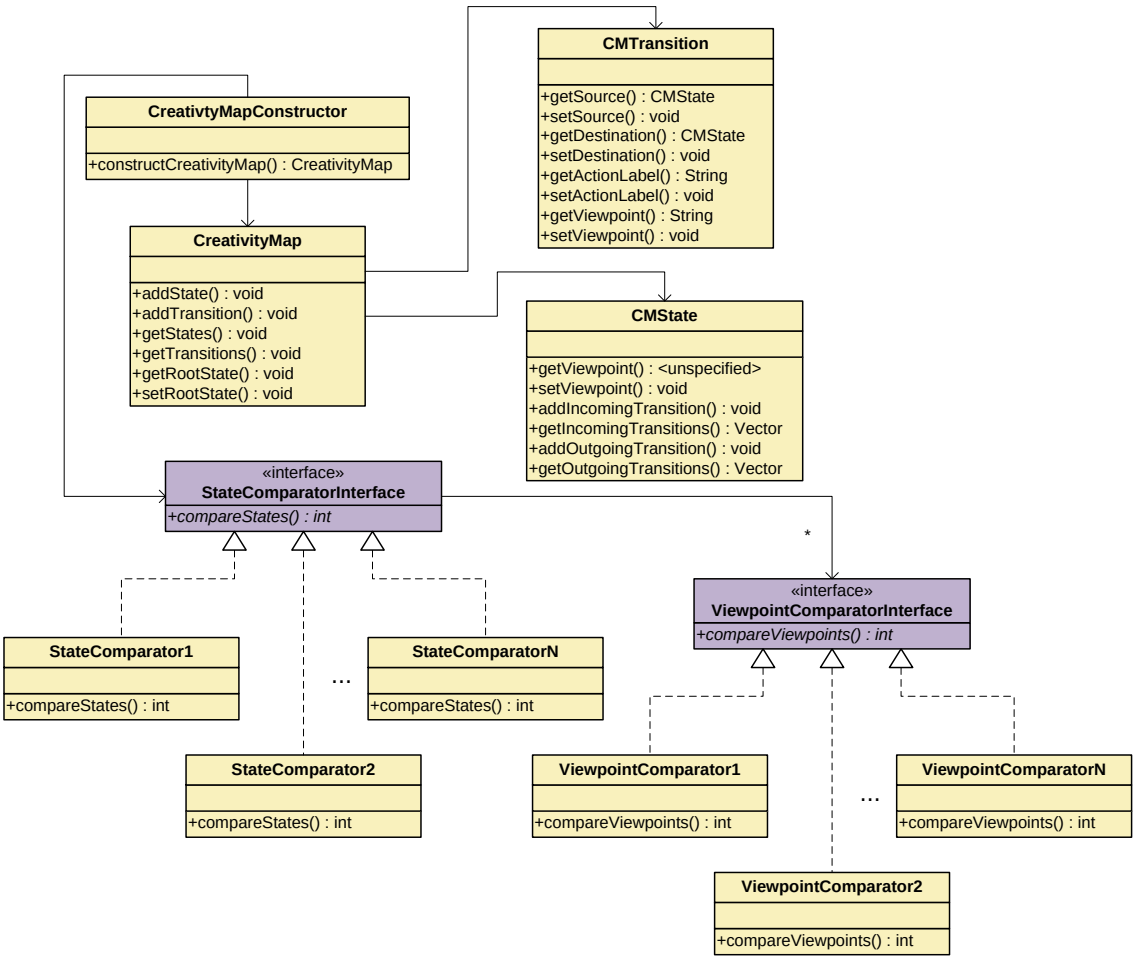


Figure 7.12: UML Class Diagram of the CMCE

StateComparator The StateComparator is a concrete implementation of the *StateComparatorInterface* that needs to implement the *compareStates* method. It utilises ViewpointComparator objects for the concrete comparison of viewpoints.

ViewpointComparatorInterface The ViewpointComparatorInterface defines an interface that needs to be implemented by any concrete viewpoint comparator. It contains one *compareViewpoints* method, which is responsible for the comparison of two values from the same viewpoint. This design enables a flexible exchange and reusability of concrete ViewpointComparators.

ViewpointComparator The ViewpointComparator is a concrete implementation of the *ViewpointComparatorInterface* that needs to implement the *compareViewpoints* method.

CreativityMapConstructor The CreativityMapConstructor creates the creativity map from the initial sequential creative process. It utilises *StateComparators* objects for the comparison of states. The transitions of similar states are repositioned as described in Section 4.2.3.

CreativityMap The CreativityMap represents the corresponding data structure of a creativity map. It contains CreativityMapStates and CreativityMapTransitions.

CreativityMapState The CreativityMapState represents a state of the creativity map. It contains the viewpoints of the creation, for instance artefact or time. As mentioned in Section 3.3, time is realised as a timestamp, representing the moment when the activity of the state's outgoing transition begins.

CreativityMapTransition The CreativityMapTransition represents a transition of the creativity map. It contains a start and end state, an action label and the name of its viewpoint. An additional variable that stores the original action label during a hiding process is included as well.

The presented design of the CMCE provides a high degree of flexibility. Viewpoint comparators can be exchanged between multiple state comparators. These can themselves be reused for different creativity map constructions.

7.7 Information Mining Engine (IME)

The Information Mining Engine (IME) is a component of the Creativity Mining Engine. The whole layer is responsible for the analysis of creative processes as well as creativity maps and realises the steps that were described in the knowledge creation process. It splits into several elements that are all designed to perform specific actions. One of them is the IME. It implements the information extraction and hiding approaches that were discussed in this thesis. The UML class diagrams that are presented for both implementations show only the most important classes and methods; parameters are additionally omitted to keep them clear.

7.7.1 Information Hiding

The information hiding process was presented in Chapter 5. It conceals irrelevant behaviours of a creativity map and possibly reduces its size, which is for instance important

for a compact visualisation. The resulting PCM enables aim oriented analyses as only relevant states and transitions need to be processed. It was mentioned before that the information hiding process converts a behaviour description into a BDA, which is then used to identify behaviours in a creativity map. This process is realised with the Thompson and powerset constructions. Figure 7.13 depicts the UML class diagram of the information hiding component.

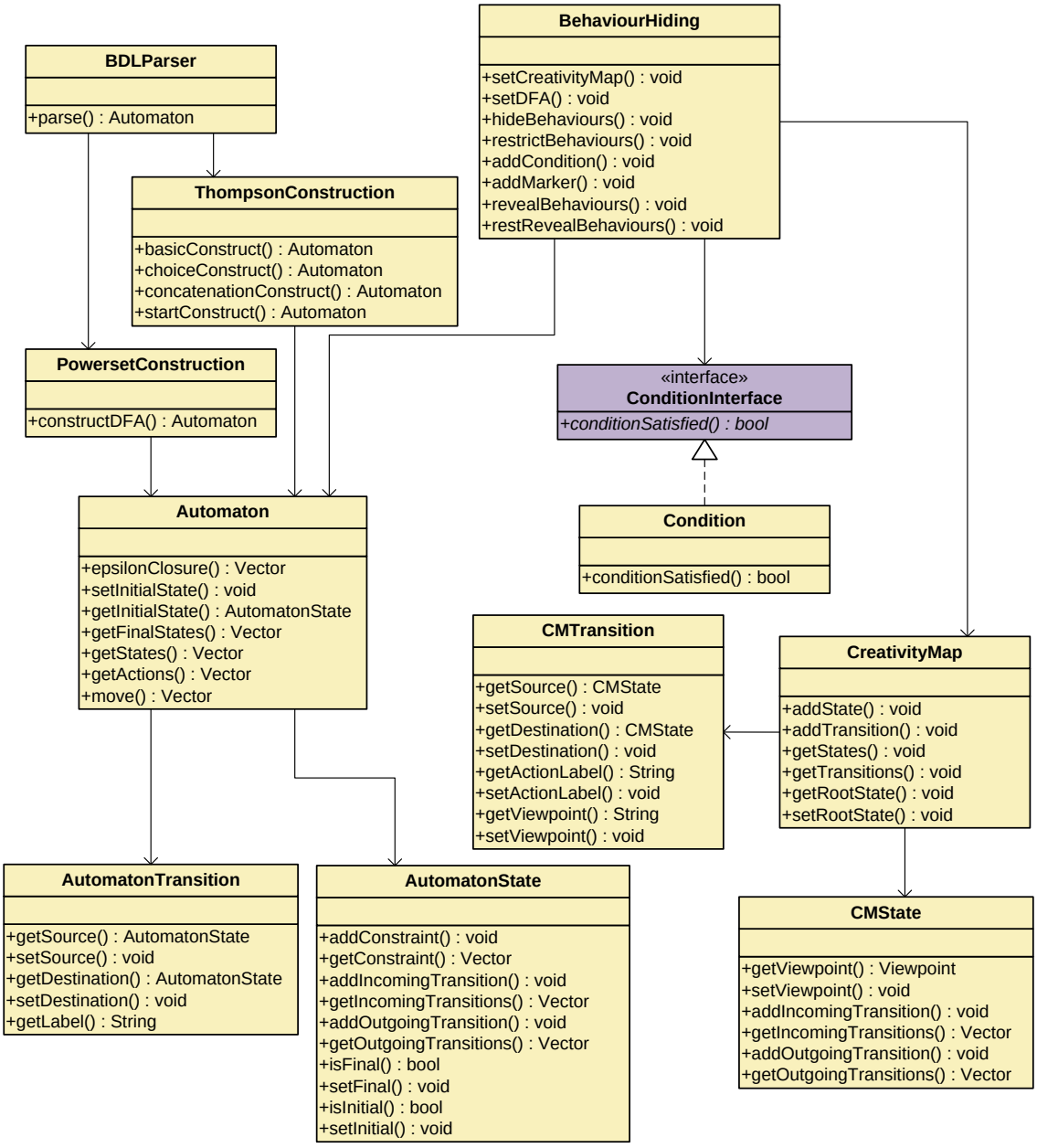


Figure 7.13: UML Class Diagram of the Behaviour Hiding Component

BehaviourHiding The BehaviourHiding represents the behaviour hiding process. It realises the four operations hiding, restriction, revealing and restrictive revealing that were introduced in this thesis. This component uses a DFA as a representation of the BDA and traverses the creativity map to identify and mark matching behaviours, which are then processed based on the used operation.

ConditionInterface The ConditionInterface defines the interface for the state conditions. It contains the single method *conditionSatisfied*, which returns if a viewpoint of the state satisfied this condition. This interface needs to be implemented by any concrete state condition.

Condition The Condition is a concrete implementation of the *ConditionInterface*. An examples for a condition that was mentioned in this thesis was a word count restriction for the artefact viewpoint. The component needs to implement the *conditionsSatisfied* method.

BDLParser The BDLParser parses a behaviour description that is expressed in the Behaviour Description Language (BDL) and constructs an ϵ-NFA based on the elements that were described in Section 5.5.2. The powerset construction is used afterwards to convert the ϵ-NFA into a DFA.

ThompsonConstruction The ThompsonConstruction creates an ϵ-NFA from a behaviour description. The automaton is created from a set of constructs that are combined to build the ϵ-NFA for the whole behaviour description.

PowersetConstruction The PowersetConstruction converts the ϵ-NFA that was created with the Thompson construction into a Deterministic Finite Automaton (DFA). As mentioned before, DFAs represent more convenient tools for the behaviour hiding process.

Automaton The Automaton represents any finite automaton that is needed in the hiding approach. This includes the ϵ-NFA after the modified Thompson construction as well as the DFA after the powerset construction. The Automaton contains a number of AutomatonStates and AutomatonTransitions.

AutomatonState The AutomatonState is a state of the automaton, which contains an action label.

AutomatonTransition The AutomatonTransition represents a transition of the automaton. It contains a label that is used for the behaviour matching. Special δ transitions include additional state conditions and ϵ transitions do not require an input symbol.

CreativityMap The CreativityMap represents the corresponding data structure of a creativity map. It contains CreativityMapStates and CreativityMapTransitions.

CreativityMapState The CreativityMapState represents a state of the creativity map. It contains the viewpoints of the creation, for instance artefact or time. As mentioned in Section 3.3, time is realised as a timestamp, representing the moment when the activity of the state's outgoing transition begins.

CreativityMapTransition The CreativityMapTransition represents a transition of the creativity map. It contains a start and end state, an action label and the name of its viewpoint. An additional variable that stores the original action label during a hiding process is included as well.

7.7.2 Frequent Information Extraction

The frequent information extraction process was explained in Chapter 6. Frequent behaviours contain essential information about the creator and are important for the further study. The input of the extraction process is a corpus of creativity maps, which is used for the construction of the frequency categories *Preferred, Common, Uncommon* and *Zero*. These categories are then utilised during the behaviour extraction to build the frequent behaviour categories *Minimal (f_{max})/Minimal (f_{min})/Maximal Alternating, Emergent* and *Disappearing*. Figure 7.14 depicts the UML class diagram of the information extraction component. It illustrates the integration of both the frequency categories and frequent behaviour categories into the design.

BehaviourExtraction The BehaviourExtraction implements the techniques for the frequent behaviour extraction. It creates both the frequency categories based on the thresholds f_{zero}, f_{min} and f_{max} and the frequent behaviour categories. This component also allows to track behaviours, as described in Section 6.5.2.

FrequencyCagtegory The FrequencyCagtegory represents one of the four frequency categories *Preferred, Common, Uncommon* or *Zero*. One object is constructed for each of these sets. It stores all behaviours that belong to this category and allows to remove, modify and retrieve them. The categories are constructed for each creativity maps or PCMs that should be analysed.

FrequencyCategorySet The FrequencyCategorySet summarises the four frequency categories for a creativity map into one component. Any of these categories can be

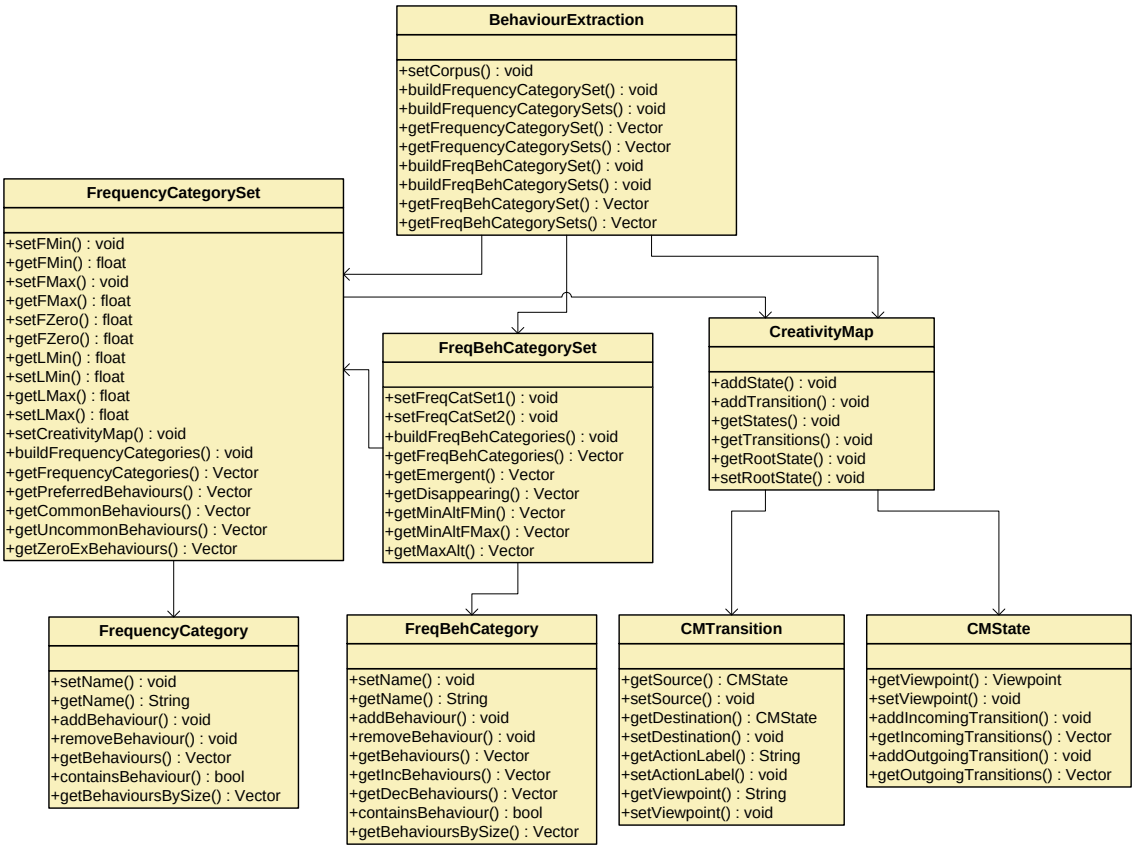


Figure 7.14: UML Class Diagram of the Behaviour Extraction Component

retrieved separately or all together as a set. The component allows to configure the length and frequency thresholds according to the personal needs.

FreqBehCategory The FreqBehCategory represents one of the frequent behaviour categories *Minimal (f_{max})/Minimal (f_{min})/Maximal Alternating, Emergent* or *Disappearing*. Similar to the FrequencyCategory component, one object is constructed for each of these categories. It allows for the distinction between incrementing and decrementing behaviours, which are present in all categories except for *Emergent* and *Disappearing*.

FreqBehCategorySet The FreqBehCategorySet summarises the five frequent behaviour categories that were created for two creativity maps into one component. Any of these categories can be retrieved separately or all together as a set.

CreativityMap The CreativityMap represents the corresponding data structure of a creativity map. It contains CreativityMapStates and CreativityMapTransitions.

CreativityMapState The CreativityMapState represents a state of the creativity map. It contains the viewpoints of the creation, for instance artefact or time. As mentioned in Section 3.3, time is realised as a timestamp, representing the moment when the activity of the state's outgoing transition begins.

CreativityMapTransition The CreativityMapTransition represents a transition of the creativity map. It contains a start and end state, an action label and the name of its viewpoint. An additional variable that stores the original action label during a hiding process is included as well.

7.8 Knowledge Repository

The Knowledge Repository represents the storage space for all data that is handled by the DMCA. It is divided into the three distinct components: version control system, Creativity Map Repository (CMR) and external repository. The version control system stores the different revisions of the documents that are created with the Collaborative Editor and manages their access. The CMR stores the creativity maps that are generated with the DMCM. Every map is saved in this database, which enables a convenient and centralised access for the analysis. The external repository is a dynamic connection to knowledge that exists outside of the DMCA. This includes for instance external libraries or documents that can be queried for artefacts which are not present in the current creation zone.

7.8.1 Version Control

The documents that are created with the Collaborative Editor are stored in a subversion [16] version control system. It is able to save different revisions of a file, which in this case allows to track the creation process of documents. Versions can easily be retrieved and compared for the emphasis of modifications. The version control system of the Knowledge Repository is the only part of the DMCE that has access to the subversion system, which prevents it from unwanted harm. Access to the version control system within the tool is realised with the SVNKit library. It is a free client library for subversion that is purely written in Java [41]. Figure 7.15 depicts the UML class diagram of the version control component. It represents only the most important classes and methods. Parameters are additionally omitted to keep it clear.

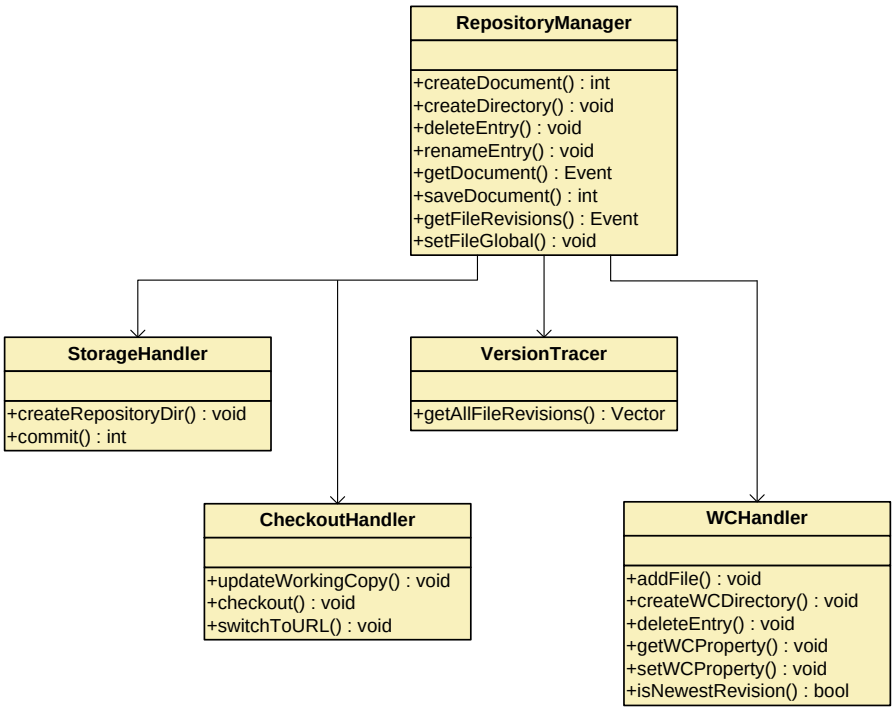


Figure 7.15: UML Class Diagram of the Version Control

RepositoryManager The RepositoryManager executes all operations that need to access the version control system. Its functionalities are encapsulated into several components. Each of them implements a set of functionalities from the subversion system. This design creates one central component that delegates access to the version control and prevent the subversion system from unwanted access.

StorageHandler The StorageHandler is responsible for committing changes to the version control system. When a document is modified with the Collaborative Editor and saved afterwards, the SvnServer component eventually commits these changes, which then automatically creates a new version.

CheckoutHandler The CheckoutHandler loads a document from the repository to the local working copy of the server, which is defined as a *checkout* operation. If it is necessary to revert to an old revision of the document, the CheckoutHandler needs to checkout this version from the subversion system. It can then be communicated to the client and used afterwards.

VersionTracer The VersionTracer allows for the collection of version numbers and information of a document that can then be presented in the Collaborative Editor. The user can review these information before reverting to a previous document revision.

WCHandler The WCHandler manages the working copy of the version control system. A working copy represents the locally stored files that were checked out from the subversion system to the hard disc of the DMCA server. The WCHandler is responsible for the creation or deleting of new files and folders inside the working copy and enables the attachment of additional information, so called properties to files. A property that every document possesses for its identification is the "author". The files inside the working copy are committed to the version control system by the StorageHandler.

7.8.2 Creativity Map Repository (CMR)

The Creativity Map Repository (CMR) represents a storage space that saves all creativity maps which were captured with the DMCM. It keeps the maps at a central place in the system to guarantee a convenient access for other components. All necessary information about the creative process of a person, domain or other group of users can be retrieved from this repository. It can then be used for the creativity support and behaviour mining. The repository can possibly be opened for external resources, which enables additional processing by experts from several domains. A relational database [20][19], in particular the open source database management system PostgreSQL [42] is used to implement the CMR. Figure 7.16 depicts the Entity-Relationship Model (ERM).

user The user entity (primary key = userID) represents any user who works on a project and records the creative process with the DMCM. It consists of a user name and a password.

project The project entity (primary key = projectID) represents a project that a user is working on. It stores a userID as foreign key to refer to its creator together with a projectName. Various projects can be stored for each user.

creativityMap The creativityMap entity (primary key = cmID) describes a creativity map. This entity contains the current state currState, which specifies the state where the creative process stopped when the map was saved the last time. The creative process continues from this state after the map was loaded. The creativityMap entity additionally stores an initial state, which is the root state of the creativity map and the projectID of the project it belongs to.

transition The transition entity (primary key = transitionID) represents a creativity map transition. It stores the id of the start state (start1ID), end state (state2ID)

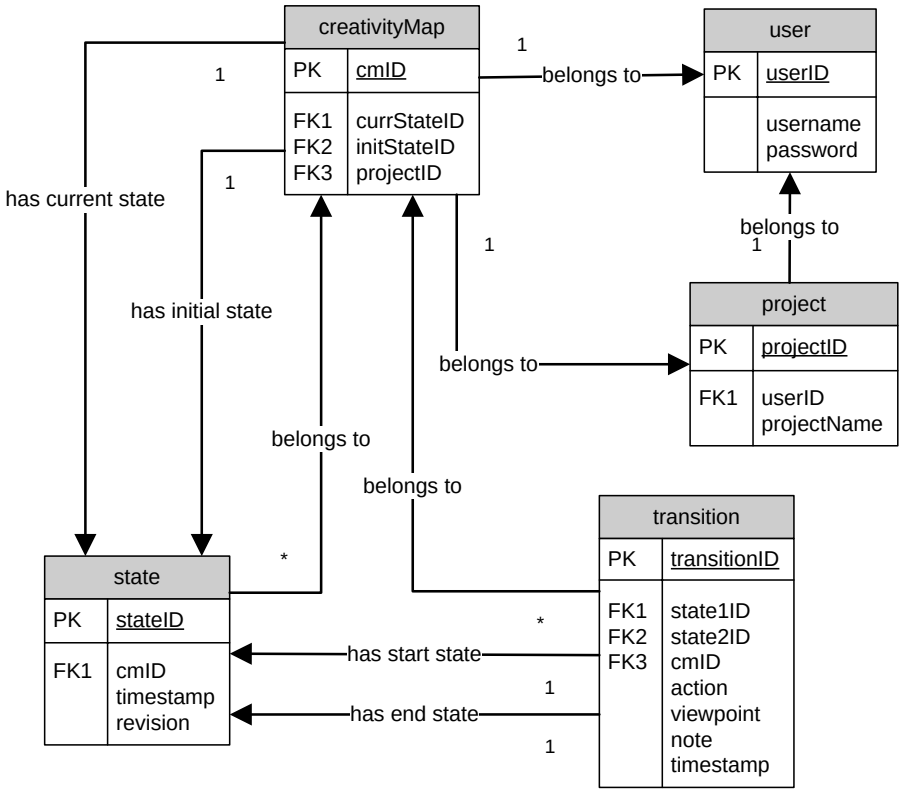


Figure 7.16: ERM of the CMR

and creativity map it belongs to (cmID), together with an action, a viewpoint, a timestamp and a note for additional information.

state The state entity (primary key = stateID) represents a state of a creativity map. It contains the id of the creativity map it belongs to (cmID), a timestamp and the revision number of one particular document that is stored in the version control system of the Knowledge Repository. The last component is a representation of the artefact viewpoint.

The state entity of this initial ERM contains the two viewpoints *time* and *artefact*. It needs to be further adapted, if additional viewpoints should be observed during the creative process. They can for instance be stored directly in the database, like the timestamp, or as a reference to another system, like the revision number of the version control system.

7.8.3 External Repository

The External Repository extends the knowledge that is stored inside the DMCA via connections to external facilities. One possible resource is the World Wide Web (WWW), which contains large amounts of information from different domains. It has grown to a platform of (collaboratively) generated data that is stored centralised for an enhanced accessibility. Its representations are diverse and include for instance documents in online libraries or journals or simple web-pages that are build with the help of user content. The External Repository integrates tools that query data from these resources and present it in a processable format. This assists the creative process during situations where a creator is "stuck" and needs support. If needed, users can also actively utilise the external repository to search for artefacts or other information.

External resources are outside of the user's creation zone. To integrate them into the creative process, they need to be stored in any of the viewpoints. One example is a list of literature that was read from an external library during the creative process. It can be stored in a literature viewpoint, if the creator specified it accordingly. Another example is a paragraph that was cited in the document, or other parts of an external text which was used. The latter resources are already present in the artefact viewpoint and integrate into the existing structures. However, whenever this kind of data is used in the own creative process, it might be necessary to cite the original appearance. This helps other creators to identify original sources and avoids plagiarism. The system can assist this process by reminding the user when content is copied from an external resource. Another issue is the quality of external resources. Only the developer is able to maintain a high standard by checking these objects prior to an integration into the DMCA.

The use of external data requires some kind of interface that allows for a standardised access. For instance, search engines and other data repositories sometimes offer web service that can be queried remotely to access these resources. Developers can then build tools and integrate them into the Knowledge Repository. The main advantage of the described approach is the use of external elements within the DMCA, without the necessity to switch between tools and possibly deflect the user's attention from the creative process. This is one main requirement of the DMCA and the reason for its pluggable design. However, if no such interfaces or services are provided, the creator is required to switch to external tools like a web browser to access these resources.

7.9 Summary

This chapter explained the prototype tool support for the proposed research. It was shown how the motivation and aim for a creativity support tool are implemented in the initial version of the DMCA. Each of its layers was discussed in detail. It was illustrated that the DMCA and especially the DMCE offer a flexible and extendible environment, which enables collaboration without being confined to a single domain. The design and GUI of the Collaborative Editor illustrated the simplicity and adaptation to the needs of a collaborative word processor. The De Montfort Creativity Mapper (DMCM) was introduced as a powerful tool for the observation and mapping of the creative process. Its flexible design enables a convenient adaptation to the custom requirements of its users. Architecture and implementation of the Creativity Map Construction Engine (CMCE), which manages the construction of creativity maps were introduced. The Information Mining Engine (IME) that is part of the Creativity Mining Engine was explained and the realisation of the information mining approaches that were discussed in this thesis was presented. It was explained how the components of the behaviour hiding and extraction processes were transformed into corresponding software components. The Knowledge Repository was described at the end of this chapter.

Chapter 8

Case Studies

Objectives

- Present a case study of a creativity workshop, where the creative processes of all participants were recorded. Problems with the developed prototype tools are also identified.
- Present a case study with long recording time of the creative process and a large creativity map to demonstrate the information hiding approach.
- Present a case study that uses a corpus of creativity maps to demonstrate the information extraction approach.
- Demonstrate and evaluate the need and practical applicability of the presented research.

8.1 Introduction

This chapter demonstrates and evaluates the applicability of the presented research with the help of three case studies. The first one illustrates the use of the DMCA by a group of people during a workshop that was held at the Software Technology Research Laboratory (STRL) in cooperation with the Institute of Creative Technologies (IOCT). It demonstrates the construction process of creativity maps and the flaws during the observation of the creative process. The second case study presents a long recording time of the

creative process. It shows how a creativity map grows large and demonstrates the information hiding approach to conceal irrelevant behaviours. The third case study presents the extraction of frequency related information from a corpus of 10 creativity maps. It explains behaviour tracking and demonstrates its use for the analysis and assistance of the creative process.

8.2 Creative Writing Workshop

The first case study presents a workshop that was held at the Software Technology Research Laboratory (STRL) at De Montfort University (DMU) and demonstrates the creativity map construction process. It shows how the DMCA is used simultaneously by several users to capture their creative processes with the DMCM. Individual sets of actions were used for the recording during the workshop and the DMCM was customised by each user to satisfy the personal requirements.

A group of 8-12 people with different backgrounds and a common interest in creative writing should be found for this workshop. Invitations were sent beforehand via email to specific persons and email-lists in general. The workshop was held on the 30th July 2009 at the STRL with 9 people, representing a mix between students, university staff and people that were employed outside the university. It was organised as a 3 hour session in the morning from 9.30 a.m. - 12.30 p.m., where the group used the DMCA and especially the Collaborative Editor to create and edit documents. At the same time, each creative process was recorded with the DMCM. The agenda of the workshop was divided into the following three parts.

1. Starting the DMCA and writing a short text about oneself to get used to the software, especially the Collaborative Editor (10 - 15 minutes).
2. Writing a short text about a stimulus item (e.g. watch, necklace, sunglasses, etc.), which was provided by Prof. Sue Thomas (ca. 25 minutes) followed by a break (20 minutes).
3. Editing the written text to reduce the number of words by half (ca. 25 minutes).

Every participant worked on a separate PC and started the DMCA directly from the webpage (http://dmu-ca.ioct.dmu.ac.uk/) in a single user mode, without any form of collaboration. All necessary requirements like the installation of the latest Java Runtime

Environment (JRE) or Java Development Kit (JDK) [40] were finished and tested in advance. Each stage of the workshop was followed by a short break, where the participants had the possibility to talk to each other and discuss things in general. After the actual exercises were finished, a group discussion about the session and the DMCA took place. Valuable feedback on the tool and the whole workshop was given, which is also presented in this case study.

8.2.1 Creativity Map Construction

Recording of the Creative Process

A presentation was held at the beginning of the workshop to introduce the DMCA and the DMCM. The functionalities of these tools were presented and a short presentation demonstrated the use of the software. During the first stage of the workshop, each participant was able to familiarise with the DMCA for 10-15 minutes and no recording of the creative process was performed. The documents which were generated with the Collaborative Editor were not used for the analysis or construction of the creativity map. This first stage was mainly for the collection of background information about the participants and their interest in creative writing. As mentioned before, the task was to write a text about oneself.

The DMCM was used during the second and third stage of the workshop to observe and capture the creative processes of the participants. Each generated creativity map was stored in the CMR of the Knowledge Repository and the documents that were created with the Collaborative Editor were saved in the version control system. The participants had to adapt the DMCM to the personal needs, before the second part started. In order to help them, a predefined set of actions and viewpoints with respect to the creative writing domain was used as the default configuration. These were in particular:

Editing The Editing action is performed when the user is editing the document. This can be for instance writing new text, deleting existing text passages or copying and pasting.

Conceptualising The Conceptualising action describes an activity where the user is thinking about the concept of the document. Usually a first structure is created in mind before the writing process starts.

Contemplating The Contemplating action is performed, when the user thinks about the current topic. This includes the actual document as well as other related thoughts. Contemplating is a domain independent action and can probably be identified in a variety of creativity maps.

Comparing The Comparing action describes the situation where a user compares different version of the created document. The Collaborative Editor integrates a comparison functionality, which allows to display any two versions side by side with highlighted differences. Whenever this facility is used, it is assumed that a comparing action is performed.

A viewpoint is created for each of the previously described actions. This means that the *editing* viewpoint includes the *editing* action, the *conceptualising* viewpoint the *conceptualising* action and so on. As the Collaborative Editor was used during the workshop, the DMCM was invoked as an integrated facility. Figure 8.1 depicts the GUI of both components with the default configuration of the DMCM, representing each viewpoint in a different colour.

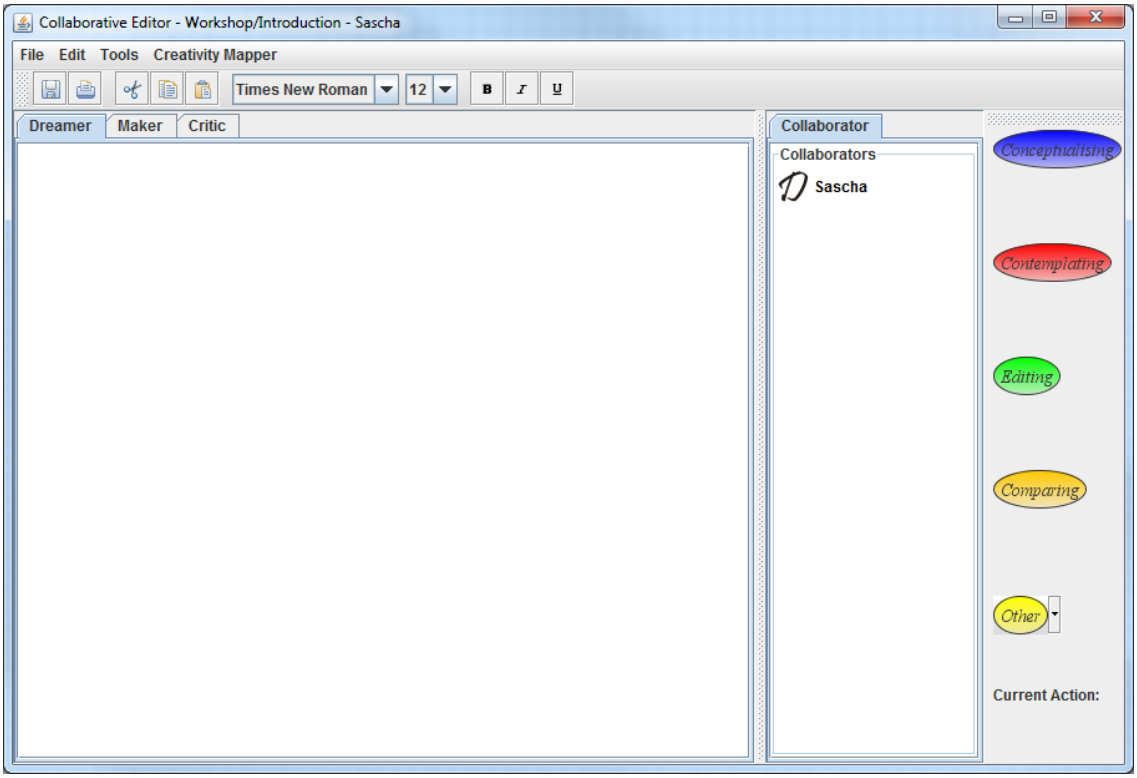


Figure 8.1: GUI of the Collaborative Editor with integrated DMCM

After the first stage of the workshop was finished, each participant had to choose one of the stimulus items that were provided by Prof. Sue Thomas. This included for instance sunglasses, a watch or a necklace. The Collaborative Editor was invoked as shown in Figure 8.1 with a plain text document afterwards and the second phase of the workshop started, where 25 minutes were spent to write about the stimulus item, followed by a 20 minutes coffee break for possible conversations. Each participant was instructed to record the creative process during this period. In the 25 minutes of the third phase, the participants were asked to reduce the previously created document by half and continue with the recording of the creative process. After this stage ended, the recording and writing was finished, the DMCA was closed and the captured creative process of the two 25 minute sessions was stored in the Knowledge Repository. All document versions that were created with the Collaborative Editor were additionally saved in the version control system. A fragment of one observed creative process of a participant is depicted in Figure 8.2.

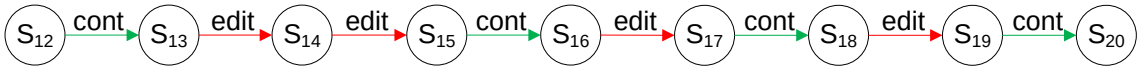


Figure 8.2: Fragment of a Captured Creative Process

State Comparison

As mentioned before, the initial creative process describes a chronologically ordered sequence of actions. However, this does not necessarily describe the correct structure, as the creator possibly moved back to a previous stage during the creation, resulting in several branches and a creativity map structure. For the construction of this map, similar states need to be identified in the first step. This process is based on the similarity measurement of states that was described in Section 4.2.2 and additional information which is supported by the creator. It was mentioned before that some situations where the creator returned to a previous idea are difficult to record and require additional information. The described comparison process is therefore only one part of the identification of similar states.

The metric that was chosen for this case study utilises the artefact viewpoint, which stores the document about the stimulus item that was created during the second and third phase of the workshop. This viewpoint was picked, because it is definitely present in all captured creative processes and furthermore represents the essential information for the observation of creativity, like mentioned in the first axiom of creativity A0 in Section

3.2. The comparison process of two artefact viewpoints defines the included documents as similar if they are identical, implying that they refer to the same version in the version control system.

The Collaborative Editor integrates a facility which enables the user to review every revision of the currently loaded document, as described in Section 7.4.2. It furthermore allows for a reversion to any of these versions, which then loads the corresponding document into the editor. Whenever this happens, the creator simultaneously returns to this stage in the creative process. This additional information that is provided by the participant is integrated into the construction process, which is responsible for the creation of a map structure. Figure 8.3 depicts a small sequence of a creative process with additional transitions to emphasise the returning phases. These dotted arrows are only a visual aid and not part of the actual creative process that is stored in the CMR. Each of the states contains the two viewpoints *artefact* and *time*. The artefact viewpoint stores the revision number of the version control system and the time viewpoint saves the timestamp that is not further discussed in this example.

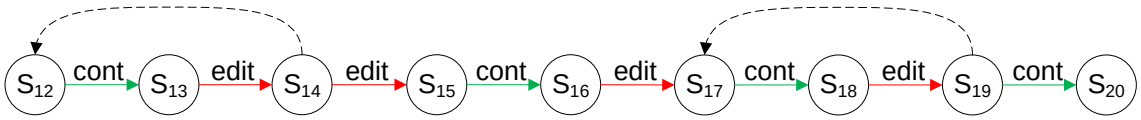


Figure 8.3: Fraction of a Captured Creative Process with Revert Transitions

All states of the creative process need to be checked for identical document versions. It is necessary to integrate the provided information from the creator into this process. Table 8.1 illustrates the result of the state comparison phase for one participant. It lists all version numbers of the document that are at least shared by two states.

Document Version	States
2	S_1, S_2
6	S_6, S_8
9	S_{10}, S_{11}
10	S_{12}, S_{14}
14	S_{17}, S_{19}
17	S_{21}, S_{25}
23	S_{28}, S_{32}

Table 8.1: Similar States with Respect to Document Versions

The states that are not shown in Table 8.1 failed in the comparison process. This implies that the participant of the workshop did not return to any of them in the creative process.

Transition Repositioning

The transition repositioning process is responsible for the construction of the map structure from the initially sequential creative process. Outgoing transitions of similar states are repositioned based on the results of the state comparison step. As described in Section 4.2.3, the state with the lowest timestamp becomes the source for these transitions. Table 8.2 illustrates these states coloured in red.

Document Version	States
2	S_1, S_2
6	S_6, S_8
9	S_{10}, S_{11}
10	S_{12}, S_{14}
14	S_{17}, S_{19}
17	S_{21}, S_{25}
23	S_{28}, S_{32}

Table 8.2: Highlighted Lowest Timestamp

Any outgoing transition of a state that is not marked needs to be repositioned. For example, the transitions of state S_2 need to change its source to S_1, the ones of S_8 to S_6 and so on. Figure 8.4 depicts a small fraction of the creative process from one of the workshop participants after the described step is finished. This is also the final phase that ends the creativity map construction process.

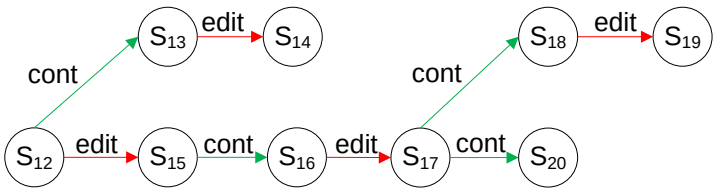


Figure 8.4: Fraction of a Constructed Creativity Map

It is shown in Figure 8.4 that all states of the original sequential process are kept in the resulting creativity map. This is necessary to retain all information, which can become important for the analysis. The whole map that includes the presented fraction is depicted in Figure 8.5. It is relatively small, due to the short recording time of 50 minutes. The branches of the map are not very long, which allows to assume that the participant realised very quickly, if the current behaviour leads to the desired result. Whenever this was not the case, he returned to a previous stage.

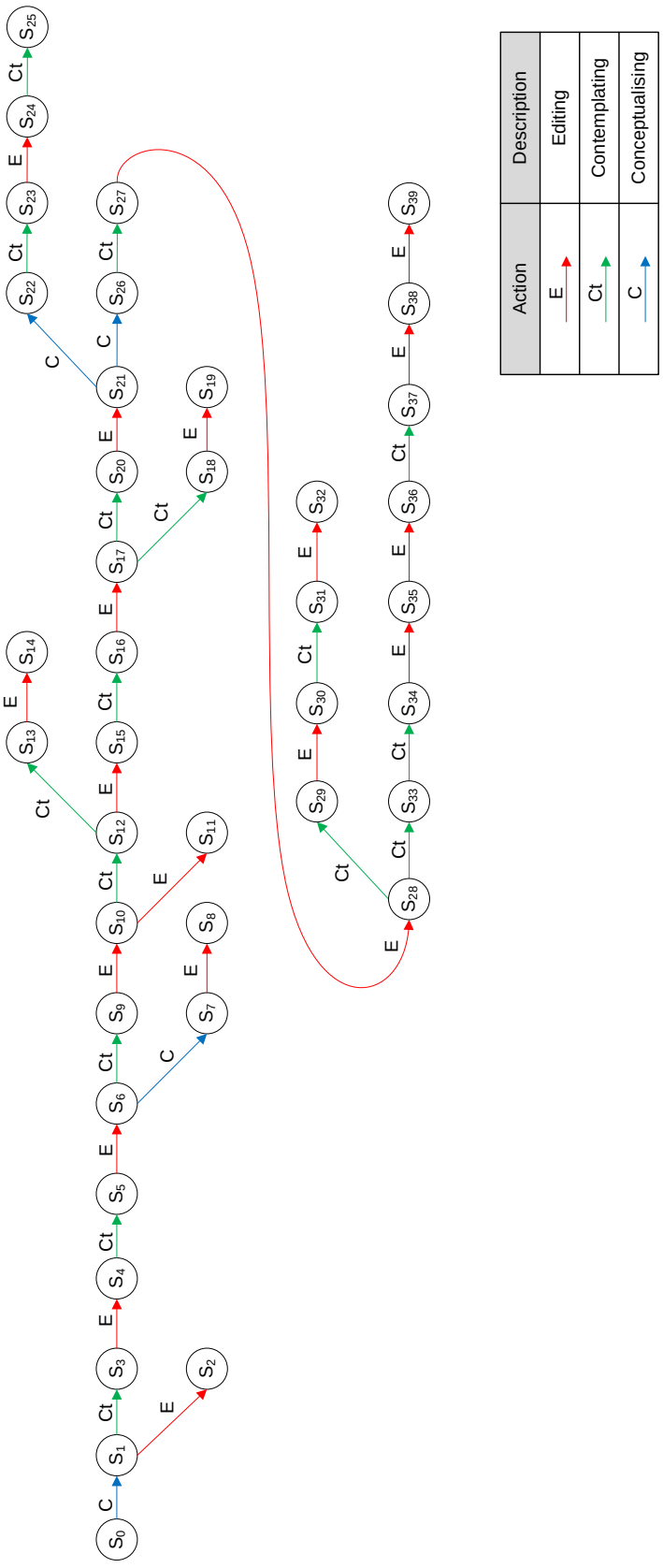


Figure 8.5: Creativity Map of a Workshop Participant

The creativity maps of all workshop participants are located in Appendix A. Some of the attendants added new actions to the DMCM before or during the two recording sessions. A number of maps is very small and one even remains a simple sequence, which describes the situation where the participant did not return to a previous stage of the creative process at any time.

8.2.2 Feedback from the Workshop Participants

After the recording and editing phases of the creativity workshop were finished, all participants met for a discussion session of about 1 hour, where they had the possibility to give feedback on the De Montfort Creativity Assistant (DMCA) and especially the De Montfort Creativity Mapper (DMCM). Although the time spent with the tools was rather short, valuable information could be collected, which allowed for interesting insights and inspired new ideas for future improvements.

The design and functionality of the Collaborative Editor were perceived positively by most of the participants. As mentioned before, only the essential editing features are implemented to keep the interface clear. All users liked the idea and added that most editing facilities contain too many possibilities that are disturbing on the one hand and complicate the location of certain configurations on the other hand. They also welcomed the fact of having an integrated facility which allows them to keep track of the document and review previous versions. The comparison feature was especially helpful for the identification of modifications and the ability to revert to previous document versions.

Differentiated feedback was given on the DMCM. The idea of observing the creative process of a person in order to build creativity maps was perceived positively by the participants. However, different opinions existed for the interface and the recording technique. As described in Section 4.2.1, the DMCM is realised as a user interactive tool, which allows for a simple and flexible implementation without additional post-processing to determine the performed actions. Some of the participant were satisfied with this approach, others perceived the DMCM as being disturbing in a number of situations during the workshop. It was especially mentioned that the interaction with the DMCM was sometimes distracting.

Summarising, it can be pointed out that the concept of creativity maps and the DMCA as initial tool support were judged favourably. However, the different opinions about the DMCM and especially the mentioned distraction from the creative process illustrated that

improvements are required. Possible directions for future developments are described in Chapter 9, particularly in Section 9.6.

8.3 Conference Paper

The second case study demonstrates the information hiding process. It illustrates how a longer recording period leads to a large creativity map which also contains irrelevant information. This leads to the construction of a PCM as the condensed representation of relevant viewpoints and actions. This time, the DMCM was used to record the creative process during the creation of a research paper.

8.3.1 Creativity Map Construction

Only the main parts of the three construction steps are explained, as the previous case study already illustrated this process in detail. The resulting creativity map is presented at the end of this section.

Recording of the Creative Process

An external word processing tool was used during this case study for the creation of the research paper, because it was written with the LaTeX type setting system, which is not supported by the Collaborative Editor at the moment. Each version of the document was stored in an external version control system, which was accessible to allow for the construction of the creativity map. The following actions were used by the person to configure the DMCM.

Editing The Editing action is performed when the user is editing the document. This can be for instance writing new text, deleting existing text passages or copy and paste.

Conceptualising The Conceptualising action describes an activity where the user is thinking about the concept of the document that is going to be created. Usually a first structure is created in the mind before the writing process starts.

Comparing The Comparing action describes the situation where a user compares different version of the created document.

Discussing The Discussing action represents conversations with the supervisor, other authors or colleagues to agree on the current outcomes and talk about the further development of the research paper.

Contemplating The Contemplating action is performed, when the user thinks about the current topic. This includes the actual document as well as other related thoughts. Contemplating is a domain independent action and can probably be identified in a variety of creativity maps.

Reading The Reading action is performed when literature about the topic is read or reviewed. For instance, the "Literature Review" section of a research paper describes one stage where the reading action is probably performed. However, it is also possible that is does not occur at all.

The first three actions *editing, conceptualising* and *comparing* are grouped into the *artefact* viewpoint. The next two actions *discussing* and *contemplating* build the *review* viewpoint and the last action *reading* is stored in the *reading* viewpoint. Figure 8.6 depicts the DMCM, which has been configured with the previously described actions. It is invoked as a standalone facility, as the Collaborative Editor is not used in this case study.

Figure 8.6: DMCM Configuration for the Research Paper

State Comparison

It was mentioned before that an external version control system was used during the creation of the research paper. It is not part of the DMCA but was accessible afterwards. The artefact viewpoint, which stores the document was therefore used to determine similar states in the creative process. Two artefacts are compared based on the same measurement, which was used in the first case study. This means that they are similar, if each character matches so that the documents are identical. Similar to the previous case study, additional information about the situations when the creator returned to a previous state

was provided. It was integrated into the overall construction process and combined with the previously described identity measurement for artefacts.

Transition Repositioning

The transition repositioning step is similar in all creativity map constructions. As mentioned before, similar states are grouped into a set and the state with the lowest timestamp becomes the source for all outgoing transitions. The particular map that was recorded during the creation of the research paper contains 211 transitions and can barely be displayed on a single page. Especially the positioning of transitions and nodes to a more structured form would require additional space. Figure 8.7 depicts this creativity map; an illustration of the collapsed form will be presented later.

8.3.2 Information Hiding

One aim of information hiding is to reduce the size of a creativity map and display only relevant transitions and states. The distinction between relevant and irrelevant is made in advance and can always be adjusted for the particular purposes. For this case study, the analyser is interested in three different creation phases of the conference paper. Firstly from the beginning of the recording to the end of the first week, secondly from the beginning of the second week to the end of the third week and thirdly from the beginning of the fourth week to the end. A number of actions and viewpoints, which are irrelevant for the further analysis are additionally hidden for each of these creation phases.

For the first stage from the beginning to the end of the first week, the analyser is especially interested in the concept and the first creation steps of the research paper. In particular, the actions *discussing*, *conceptualising* and *reading* should remain in the creativity map for this period. These activities and their relations contain information about the early creation process, which usually includes preparation in the form of discussions, concepts and literature review.

For the second stage of the creation phase from the beginning of the second week to the end of the third week, the analyser is interested in the main editing actions that are contained in the *artefact* viewpoint and their relation to the *discussing* action. This phase probably describes the main two weeks of the research paper editing and the period where most of the content is created.

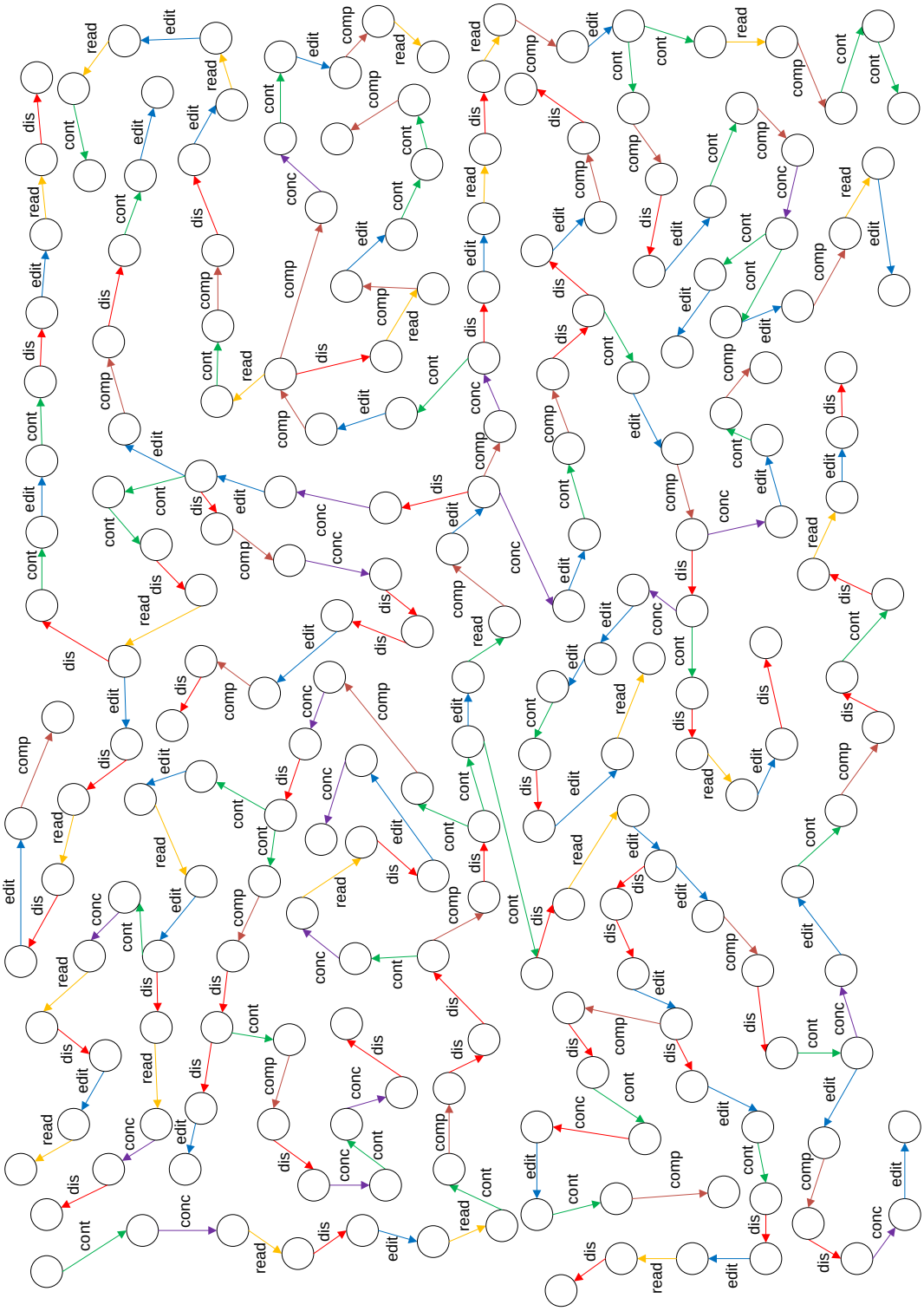


Figure 8.7: Creativity Map of Research Paper

For the third stage from the beginning of the fourth week to the end of the recording, the analyser is interested in the review process of the research paper. The relevant actions for this stage are *comparing* and any activity of the *review* viewpoint. This last stage of the creation process probably includes a review of the generated content and the comparison with previous versions.

Behaviour Description

The hiding process constructs a PCMs that reduces the original creativity map to the previously mentioned actions. To enable the distinction between the three different creation periods, it is necessary to introduce a set of conditions for the time viewpoint. It was mentioned before that a timestamp for each performed action is stored inside the states of a creativity map, which allows to define the following four conditions.

1. Condition C_1 : The timestamp needs to be less than or equal to the end of the first week.
2. Condition C_2 : The timestamp needs to be higher than the end of the first week.
3. Condition C_3 : The timestamp needs to the less than or equal to the end of the third week.
4. Condition C_4 : The timestamp needs to be higher than the end of the third week.

The first condition C_1 is used to restrict the behaviours to only the first week, which was defined as the first important stage for the analysis. C_2 and C_3 specify the second phase of the research paper creation, from the beginning of week 2 to the end of week 3. Condition C_4 is finally used for the specification of the fourth phase starting from week 4 to the end of the recording. These four conditions are utilised for the specification of the following behaviour description B_1 in the BDL, which also specifies the previously mentioned relevant actions for each of the time periods.

```
B1 = (<S-A=Discussing-C1>|<S-A=Conceptualising-C1>|<S-A=Reading-C1>)|
     (<C2-V=Artefact-C3>|<C2-A=Discussing-C3>)|
     (<C4-V=Review-S>|<C4-A=Comparing-S>)
```

This behaviour description defines the behaviours that should remain in the creativity map and is therefore used as a parameter of the restriction operation (/). Let M be the

recorded creativity map that is depicted in Figure 8.7. The restriction operation that creates the desired PCM M' is expressed in the following way.

```
M' = M / {  B1  }
```

As mentioned before, it is in many cases more convenient to use this operation for the hiding process. However, it is also possible to modify the behaviour description B_1, so that it can be used as a parameter of the hiding operation ($\backslash$). The previously defined conditions C_1 to C_4 can stay the same, but it is necessary to define the transitions that are complementary to the ones described in B_1. B_2 represents the according behaviour description.

```
B2 = (<S-A=Contemplating-C1>|<S-A=Editing-C1>|<S-A=Comparing-C1>)|
     (<C2-A=Contemplating-C3>|<C2-A=Reading-C3>)|
     (<C4-A=Conceptualising-S>|<C4-A=Editing-S>|<C4-A=Reading-S>)
```

The hiding operation with B_2 as parameter leads now to the equivalent PCM that is constructed with the restriction operation presented above. This can be expressed in the following way.

```
M' = M / {  B1  } = M \ {  B2  }
```

The example should only illustrate, how the restriction operation can be converted into a hiding operation by an adaptation of the behaviour description. However, B_1 is used as a parameter of the restriction operation in the following.

Behaviour Description Automaton (BDA)

After the behaviour description B_1 has been defined, it is converted into a BDA that is then used during the hiding process in order to match the described behaviours in the creativity map. An ϵ-NFA representation of the BDA is created first with an adapted version of the Thompson construction and the components that were described in Section 5.5.2. The initial step of this process substitutes all viewpoints and AnyEdges with the corresponding actions. This leads to the following behaviour description B_1'

```
B1' = (<S-A=Discussing-C1>|<S-A=Conceptualising-C1>|<S-A=Reading-C1>)|
```

```
(<C2-A=Conceptualising-C3>|<C2-A=Editing-C3>|
<C2-A=Comparing-C3>|<C2-A=Discussing-C3>)|
(<C4-A=Discussing-S>|<C4-A=Contemplating-S>|<C4-A=Comparing-S)
```

The ϵ-NFA representation of the BDA is not further explained, as it only represents an intermediate step. It was mentioned before that this type of automaton is more difficult to handle, due to a possibly non-deterministic choice of transitions. The powerset construction is therefore used to convert the ϵ-NFA into a DFA. The DFA that has been created for the behaviour description B_1' is depicted in Figure 8.8.

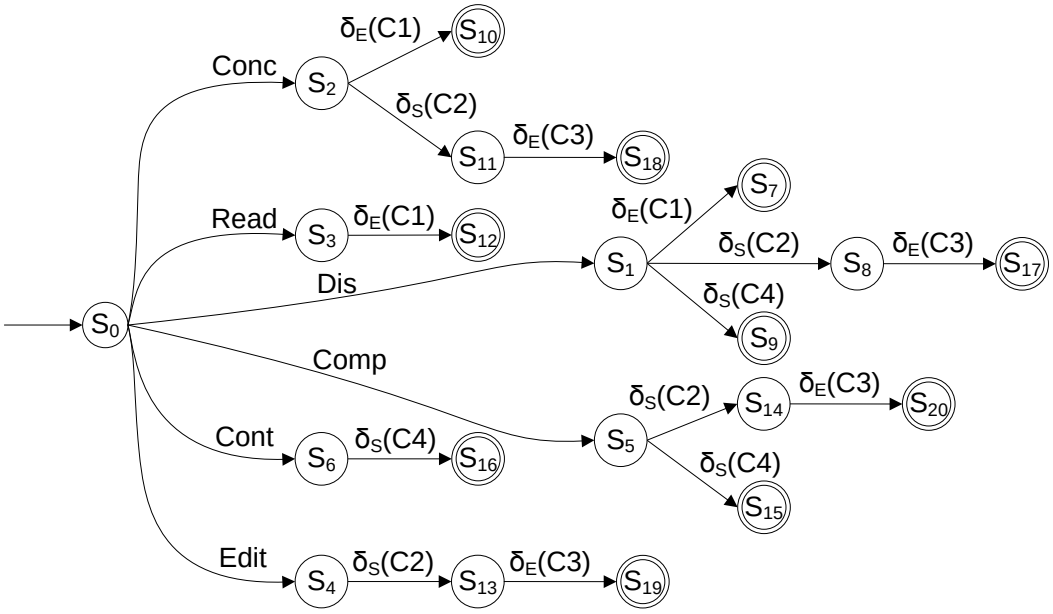


Figure 8.8: BDA in DFA Representation

The BDA in the figure illustrates a choice between the six transitions *Conceptualising, Reading, Discussing, Comparing, Contemplating* and *Editing* immediately after the initial state S_0. Depending on the chosen transition, the conditions $C1, C2, C3$ or $C4$ or a combination of them needs to be satisfied in order to reach a final state.

Behaviour Matching

The previously described BDA is used to search for the specified behaviours in the creativity map M. Any encountered sequence is marked and remains in the resulting PCM,

as the restriction operation is used. All unmarked behaviours are relabelled with α. As these α transitions are not relevant for this case study, they are going to be collapsed in a succeeding step. Figure 8.9 depicts the corresponding PCM. The states are displayed at mostly identical positions compared to the original map in Figure 8.7 in order to emphasise the size reduction as a result of the information hiding process.

Collapsed states are displayed as squares, similar to their specification in Section 5.6. The PCM contains many of them, which implies that the hiding process was able to conceal a lot of previously visible states and transitions. It was mentioned before that these elements can also be pruned, if they are no longer needed in order to reduce the physical size of the PCM.

Another possibility, which considers the amount of time that an activity consumed is presented in Figure 8.10. It illustrates the same creativity map as shown in Figure 8.9. However, the size of a node is proportional to the amount of time that the activity of an incoming transition took. This additional information allows for instance to distinguish between a short and long discussion that is followed by an editing activity. The information can be integrated into further analysis and ultimately used to support the creative process.

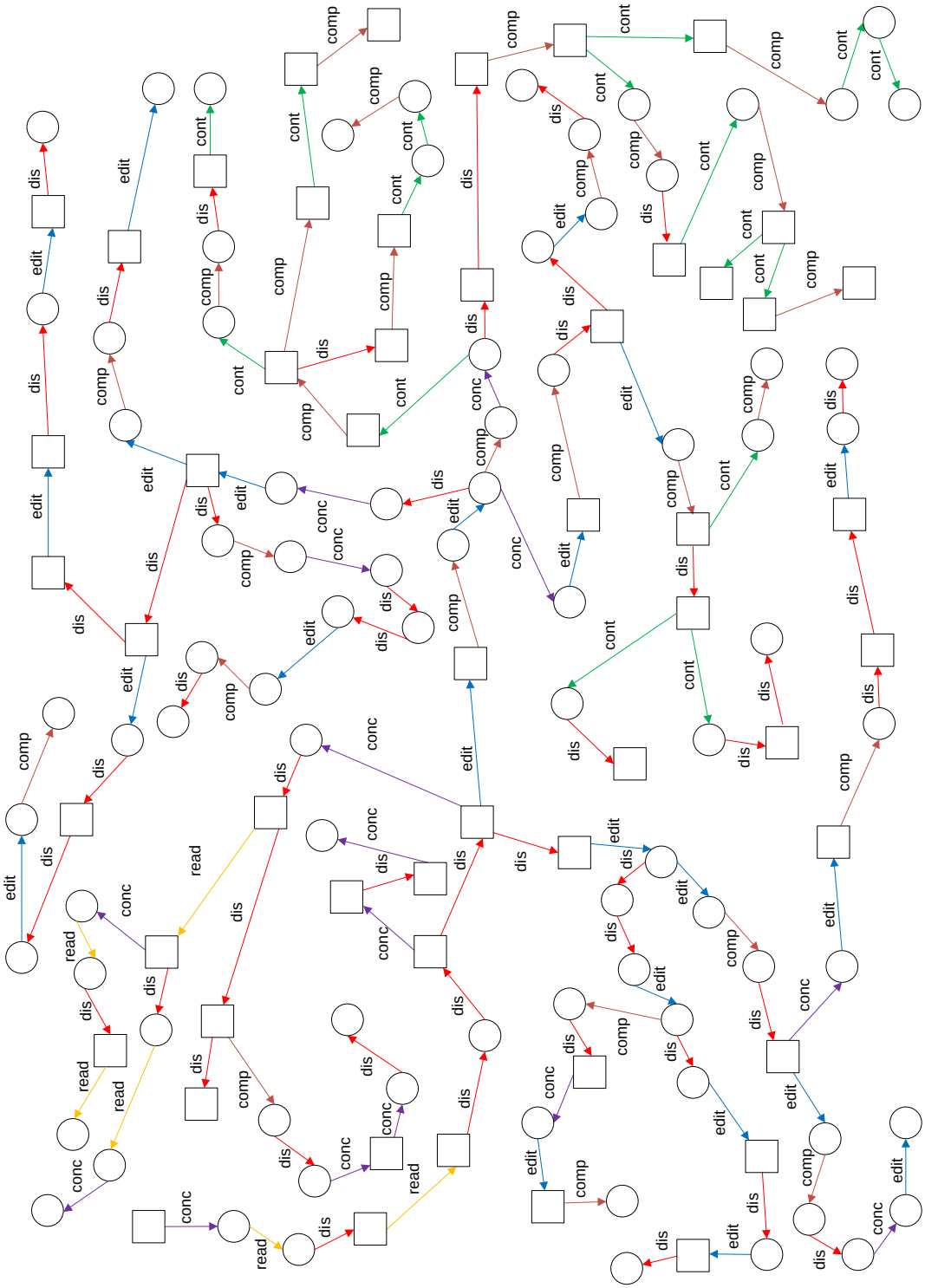


Figure 8.9: Collapsed Creativity Map of Research Paper

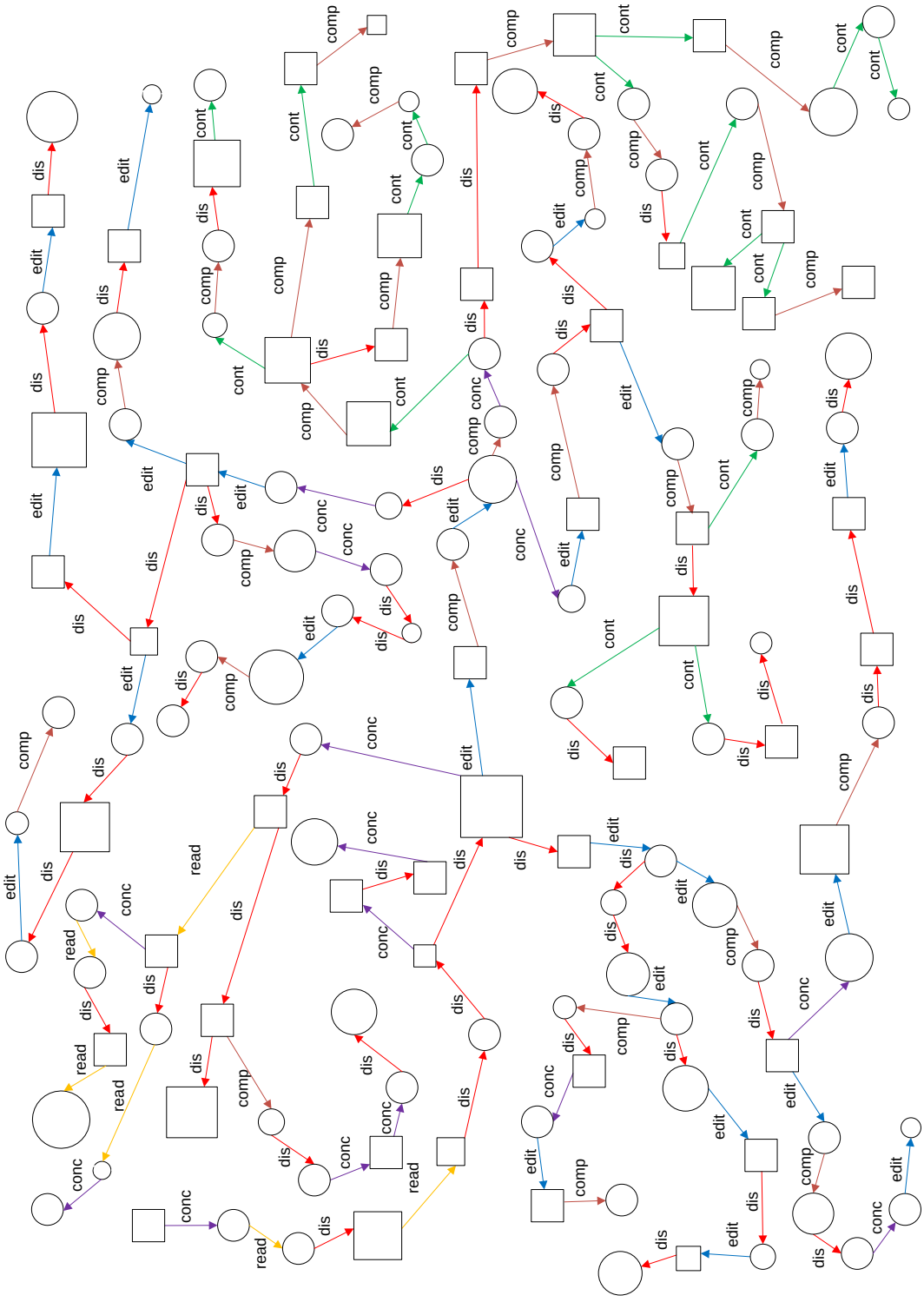


Figure 8.10: Collapsed Creativity Map with Time Information

8.4 Software Development

The third case study demonstrates the frequent information extraction process. The creative processes that were involved in a number of software projects were recorded with the standalone version of the DMCM for a time period of 6 month. Especially the amounts of behaviours for several frequency thresholds are emphasised and it is explained how this information is used for the configuration of $f_{zero}, f_{min}, f_{max}$. Frequency and frequent behaviour categories are extracted and examples of behaviour tracking and its purposes for the analysis are presented.

8.4.1 Creativity Map Construction

Recording of the Creative Process

The software development process that is described in this case study was performed in an external environment, which is not part of the DMCA. An external version control system was used for the storage of the different source code versions, similar to the previous case study. The following actions were used by the software developer to configure the DMCM.

Implementing The Implementing action represents the creation of source code by the software developer. This action is for instance similar to the editing action of a writer, which was mentioned in the previous case study.

Comparing The Comparing action is performed, when the developer compares different version of the created source code.

Testing The Testing action is performed when the software or parts of it are tested and evaluated. This is a common activity in the software development process, which can happen at any stage.

Designing The Designing action describes an activity that involves defining the software design and architecture. This might be UML class diagrams or sketches. The design of a software component should be defined before the implementation begins.

Conceptualising The Conceptualising action describes an activity where the requirements of the software and concepts are specified. They are usually determined before the actual design or implementation starts. However, it might be necessary to adjust the concept at some point during the development process.

Meeting The Meeting action is performed, when the software developer meets with team members who are also working on the project.

Discussing The Discussing action represents informal discussions about the project.

Contemplating The Contemplating action is performed, when the software developer thinks about the current topic. This includes the actual project as well as other related thoughts. Contemplating is a domain independent action and can probably be identified in a variety of creativity maps.

The first three actions *implementing*, *comparing* and *testing* are grouped into the *artefact* viewpoint. The following two actions *designing* and *conceptualising* build the *design* viewpoint and the last three actions *meeting*, *discussing* and *contemplating* are summarised into the *review* viewpoint. Figure 8.11 depicts the DMCM that has been configured with the previously described actions.

Figure 8.11: DMCM Configuration for the Software Projects

State Comparison

It was mentioned before that the version control system was not accessible during this case study. In contrast to the previous two case studies, it was impossible to use the automatic identification of similar states. Additional information about the situations when the creator returned to previous stages was of essential importance. It was provided by the software engineer, who was also instructed to collect this information. This situation illustrates the difficulty of an automated creativity map construction process.

Transition Repositioning

The transition repositioning is executed identically to the process described in the last case study. Outgoing transitions of similar states are repositioned to the state with the

lowest timestamp. The result of this construction process is a corpus of 10 creativity maps, each containing between 800 and 2000 transitions. This set is used for the information extraction process that is described in this case study.

8.4.2 Frequent Information Extraction

Information extraction creates valuable information about the frequent performance of behaviours and the relationship between actions, which can be used in the analysis. It was mentioned above that the search space for frequent information can become quite large. Additional conditions were introduced for its reduction to only necessary behaviours in order to support an aim oriented study. In this case study, the minimum behaviour length was adjusted to $l_{min} = 2$ and the maximum behaviour length to $l_{max} = 5$. Short behaviours of length 1 are usually not relevant, as they are probably preferred and long behaviours with more than 5 transitions can become difficult to handle. The relationship between transitions from the beginning and end might not be strong enough to extract useful properties. Additional information hiding and filtering steps as depicted in Figure 6.3 were not performed, as all maps are relevant for the analysis.

Table 8.3 shows quantitative details of the creativity maps corpus. It illustrates the number of distinct behaviours between lengths 2 and 5 for each of the maps M_1 to M_{10}. Furthermore, the amount of occurrences for these different sets is described. It can be observed that the number of behaviours increases with the length, while the quantity of occurrences decreases at the same time. However, the amount of occurrences does not change very much due to the large size of the creativity maps, which allows even long behaviours of length 5 to occur in every branch. The size of each map that is shown in the second column is similar to the number of occurrences of length 1 behaviours. As mentioned before, these sequences are not further considered in this case study.

The table illustrates that the smallest creativity map M_5 contains 827 transitions and the largest one M_{10} inludes 2119 transitions. It can be observed that the amount of different behaviours of length 2 alternates between 46 and 54. This is only a slight variation, as the number of possible behaviours of this size is $8^2 = 64$ and therefore also relatively small. It increases for longer behaviours, because a larger variety of them can be performed by the software developer. The second part of the table illustrates the number of occurrences for the different behaviour lengths, which is of course higher than the previously mentioned counts, as every performance of a behaviour is considered. There is only a small difference between the number of occurrences and the size of each creativity map.

Map	Transitions	Behaviours					Occurrences $\|B_{\|n\|}\|$			
		2	3	4	5	$\sum$	2	3	4	5
M_1	979	46	109	168	223	546	977	975	973	971
M_2	2066	54	152	230	315	751	2065	2064	2063	2062
M_3	1536	50	115	178	237	580	1535	1534	1532	1530
M_4	1262	47	106	170	240	563	1261	1260	1259	1257
M_5	827	48	114	178	249	589	826	825	823	821
M_6	1714	53	135	215	283	686	1713	1712	1711	1710
M_7	1654	51	145	228	330	754	1653	1651	1649	1647
M_8	1426	53	136	218	296	703	1425	1424	1422	1420
M_9	2119	54	146	235	326	761	2118	2117	2116	2115
M_{10}	1305	47	122	188	263	620	1304	1303	1302	1301

Table 8.3: Details of the Creativity Map Corpus of the Software Development

Frequency Categories

The first step of the information extraction process is the construction of frequency categories. It was mentioned before that the three thresholds f_{zero}, f_{min} and f_{max} need to be defined in advance by the analyser. They can be adapted for every creator or project; it is however advantageous to keep the thresholds identical for the same creator to produce useful and reusable results for the analysis. It is not always easy to determine them and further adjustments might be necessary, considering circumstances like the corpus size.

The corpus which is used in this case study contains creativity maps that are relatively large. Each of the maps consists of 827 to 2119 transitions. They obviously cannot be displayed, which was already problematic for the creativity map of the previous case study. The software developer chose eight distinct actions, which results in a variety of possible behaviours especially for longer sequences. Long behaviours are therefore probably less frequent than short ones, which needs to be considered during the configuration of the frequency thresholds.

The previously mentioned facts illustrate, at least to a certain degree, how background knowledge can support the information extraction process. If the frequency thresholds are configured too high, only very few or no behaviours at all are able to become frequent. In contrast, if they are adjusted very small, a large number and variety of behaviours easily becomes common or even preferred. To illustrate this scenario, several frequency thresholds were chosen and the quantity of behaviours with a frequency higher than this boundary was calculated. Figure 8.12 depicts the according behaviour percentages. Each bar represents one creativity map, for example "1" stands for M_1.

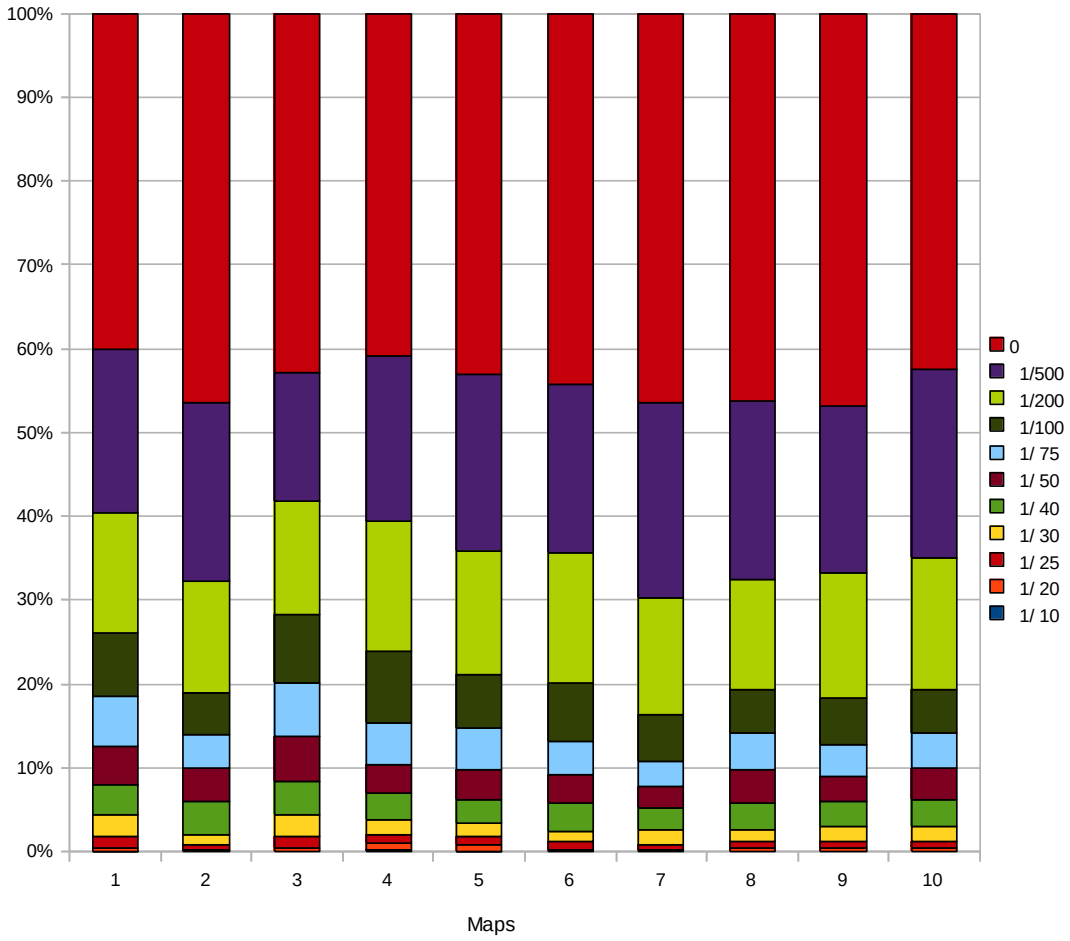


Figure 8.12: Behaviour Distribution on a Percentage Base for Several Frequency Thresholds

The figure illustrates that the parameters need to be adjusted rather small to receive useful results. For example, for the frequency thresholds 1/10 or 1/20, the percentage of behaviours with a frequency higher than this boundary is only very low. About 20% of the behaviours in creativity map 2 have a frequency higher than or equal to 1/50. Especially longer behaviours occur rarely, if the threshold is chosen too high. For this case study, the three parameters f_{zero}, f_{min} and f_{max} are specified in the following way.

$f_{zero} = \frac{1}{200}$

$f_{min} = \frac{1}{100}$

$f_{max} = \frac{1}{25}$

All three thresholds are chosen rather small, so that at least some behaviours are able to reach the common and preferred categories. For example, if the parameter f_{max} would have been configured to 1/10, only 3 behaviours of the whole corpus are able to join the preferred frequency set. As mentioned before, background knowledge and some attempts are helpful for the adjustment. Table 8.4 illustrates the amounts of behaviours according to previously described configuration.

Map	Preferred	Common	Uncommon	Zero (Existing)
M_1	19	44	131	352
M_2	9	55	150	537
M_3	18	55	109	398
M_4	15	31	166	351
M_5	13	35	154	387
M_6	13	41	186	446
M_7	11	31	184	528
M_8	13	46	140	504
M_9	12	36	194	519
M_{10}	11	44	176	389

Table 8.4: Sizes of the Frequency Categories

Although the frequency thresholds were chosen very small, the quantity of behaviours for the different categories is still not high. Only between 9 and 19 behaviours are classified as preferred and between 31 and 55 as common. The results illustrate that the particular configuration of these parameters is able to reduce or increase the size of the frequency categories. It furthermore allows to adjust the number of behaviours that are classified as zero, which can be used for frequencies that are irrelevant for the study.

Frequent Behaviour Categories

Which maps or frequency categories are compared in particular depends on the analysis. The comparison of succeeding creativity maps allows for a fine grained study that probably emphasises short term changes in behaviours. This can for example be used for the monitoring of activities and the assistance of the user if specific variations are revealed. Long-term changes in behaviours can be analysed, if succeeding creativity maps are grouped, so that they represent a particular time period. It is then possible to compare these sets and analyse the behaviour developments. The analyser can in both cases be alerted if uncommon changes were identified, such as a sequence of actions that was minimal alternating or even stayed in the same frequency category for a certain time period and number of creativity maps and then became maximal alternating or disappearing. This

case study emphasises three behaviours of the software developer. It demonstrates the classification, monitoring and extraction of information to support the analysis of creative processes.

The different frequency categories for each of the 10 creativity maps were already constructed in the previous step. They are ordered chronologically, so that the first map represents the earliest creative process, followed by the second map and so on. This allows to generate the according frequent behaviour categories and track the development of activites. The monitoring process that is presented in this case study starts at the first creativity map and ends at the last one. It focusses on the short term dynamics of behaviours and presents a number of different cases, which illustrate numerous interpretations of the results. Figure 8.13 depicts three distinct behaviours and their developments from the first creativity map M_1 to the last one M_{10}.

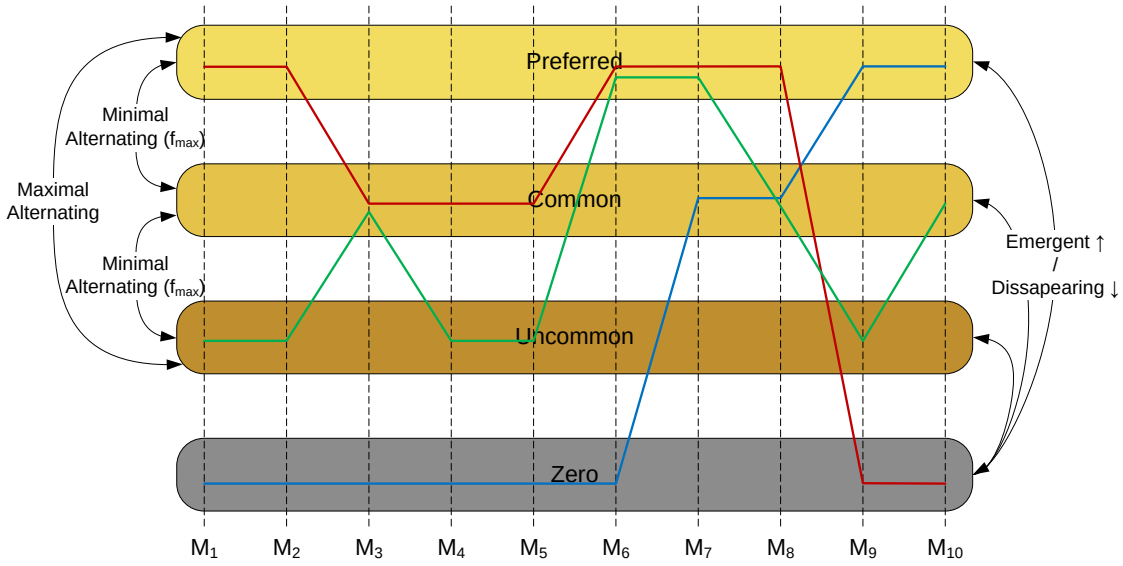


Figure 8.13: Development of Three Behaviours

Each of the three behaviours is represented as a coloured line in the figure. The illustrated developments allow for a visual analysis of the activities and a possible observation of "turning points" and uncommon changes. Reasons for this, such as environmental conditions or other issues can then be revealed and utilised to assist the creative process.

The red behaviour $\overset{Discussing}{\longrightarrow}\ \overset{Conceptualising}{\longrightarrow}$ is preferred at the beginning. It is classified as minimal alternating (f_{max}) between the maps M_2 and M_3 and also between M_5 and M_6. The behaviour was used with a high frequency in the majority of the projects and disappears after the map M_8. A closer look at it reveals that the order of the actions

describes a very common sequence, which can probably be identified in many creative processes of software developers. The sequence furthermore seems to be an important part and its disappearing illustrates that no discussion or conceptualising steps in this particular order were performed in any of the last two creativity maps.

Especially this radical change needs to be considered for the analysis and the creativity support. It was pointed out that this behaviour was an essential part of the creative process in the majority of creativity maps. If it is possible to determine its importance, maybe in terms of efficiency or the quality of the artefact, it might be necessary to attract the attention of the software developer to those behaviours that disappeared in order to reveal problems or even positive developments. Whenever an decreasing frequency of a behaviour between two creativity maps or PCMs can be identified, it is possibly be useful to interact with the creator in order to influence the creative process. This can also be realised in "real-time", for example when the creative process of map M_9 is recorded.

One possible way to interact with the software developer can be a pop-up window showing a message that illustrates the disappearing of a long time preferred and possibly essential behaviour. The user can then inform the system about project related changes, such as different team members or modifications of the environment, which lead to these dynamics. Due to the importance of the particular sequence, this might have helped the software developer in many situations. A reminder to have a discussion with colleagues would be a first and possibly necessary step for the support of the creative process.

The green line in Figure 8.13 represents the behaviour $\overset{Contemplating}{\longrightarrow}\ \overset{Discussing}{\longrightarrow}\ \overset{Implementing}{\longrightarrow}$. It is classified as uncommon at the beginning, which means that its frequency is between f_{zero} and f_{min}. The behaviour belongs to the minimal alternating (f_{min}) category between the creativity maps M_2, M_3 and M_4. It is then maximal alternating from map M_5 to map M_6, as it changes from the uncommon to the preferred set. The behaviour finally changes back to become uncommon and is classified as common in the last creativity map M_{10}. It is present in all maps and any of the frequent behaviour categories, except for the zero set. A behaviour development of this kind can probably be identified in several creative processes.

The interesting fact about this behaviour is that it leads to an implementation action, which is part of the artefact viewpoint. It is therefore at least a part of an artefact behaviour, which was presented above in Section 5.3.1. As mentioned before, the sequence abruptly became preferred from map M_5 to M_6. This means that the two actions *contemplating* and *discussing* are essential at least in this period in order to reach an *implementing*

action and modify the artefact. The behaviour is useful for the creativity support, as a reminder about the preceding actions can be displayed, whenever the software developer is "stuck" and needs assistance. A stimulus, maybe in the form a message that reminds the person to have a discussion with team members in order to get new ideas or evaluate the current ones can be presented.

The blue line represents the behaviour $\overset{Conceptualising}{\longrightarrow}\ \overset{Discussing}{\longrightarrow}\ \overset{Implementing}{\longrightarrow}\ \overset{Comparing}{\longrightarrow}$. It belongs to the zero category until creativity map M_6 and occurs the first time in map M_7, where it emerges from M_6 and is classified as common. The behaviour furthermore belongs to the minimal alternating (f_{max}) category, as it changes its frequency from common to preferred between the maps M_8 and M_9. A closer look reveals that this particular order of actions describes a probably uncommon activity of a software developer. However, as it is performed in the last four creativity maps, some circumstances might have changed.

The blue behaviour contains, similar to the green one, at least parts of an artefact behaviour, in particular $\overset{Conceptualising}{\longrightarrow}\ \overset{Discussing}{\longrightarrow}\ \overset{Implementing}{\longrightarrow}$. Its frequency increases steadily after the creativity map M_6. Compared with the green behaviour, it can be derived that a discussing action before the implementation is important in many cases. Together with a succeeding *comparing* action, it furthermore develops to a preferred activity of the creative process. Especially the situation where the behaviour emerges can become relevant for the analysis. It might be used to extract possible environmental factors for the creativity support, like collaboration, which leads to new, previously not determined behaviours. The ideas and considerations of collaborators can stimulate the software developer and influence the creative process. If the described behaviour leads to better results or is assigned a particular importance, the person needs to be informed about it. However, this kind of study always has to consider the created artefact, in particular the software or source code.

The three previously described examples illustrate possible behaviour developments. It was mentioned that the creator needs to be an integrated part of the creativity support and interact with the system in order to gather useful information. However, the particular analysis of the behaviour developments is not part of this research. It should be demonstrated how both the frequency and frequent behaviour categories are used for the illustration and revelation of creative process dynamics.

8.5 Summary

This chapter illustrated the applicability of the proposed approaches. Three different case studies were presented, each focussing on a particular part of the research. The first one explained the use of the DMCA during a workshop with a group of 9 people and emphasised the construction of creativity maps. The second case study demonstrated the information hiding process, which lead to the construction of a PCM. A behaviour description that included a mixture of state conditions and transition labels was used to introduce three distinct time periods with respect to the creative process. A different set of transitions was hidden for each of them, based on the specified purposes of the study. The last case study illustrated the information extraction process for a software developer and a corpus of 10 creativity maps. It was explained how the frequency and frequent behaviour categories are used for the description of behaviours and their dynamics. Three example behaviours were particularly emphasised and possible analyses of uncommon changes in their developments were discussed. Reminders and notifications were briefly explained as first ideas for the creativity support. The DMCM was used in all three case studies to record the presented creative processes.

Chapter 9

Conclusion and Future Work

Objectives

- Evaluate and summarise of the presented work.
- Describe the limitations of the approach.
- Propose future work.

9.1 Summary of the Thesis

This thesis discussed the mapping of the human creative process onto creativity maps. It was explained how irrelevant behaviours can be hidden on the one hand and frequency related information can be extracted on the other hand. The presented approach explained the construction process of creativity maps from the initial sequential creative process. It was shown how an individual similarity measurement based on viewpoints is used for its creation. A hierarchical classification of creativity maps into domain, creator and project categories was explained and it was shown how a customised set can be used for the retrieval of any personalised combination of maps. The role of data, information and knowledge was identified and the knowledge creation process was presented. It was especially emphasised in this context that information, with respect to creativity maps, is represented as sequences of transition. These behaviours were defined and their granularity and unification was discussed.

The information hiding approach specifically illustrated how irrelevant behaviours of large creativity maps can be concealed in order to reduce their size. Differences between observable and non-observable behaviours were explained and Partial Creativity Maps (PCMs), which contain both hidden and visible transitions were introduced. Furthermore, a formal language, namely Behaviour Description Language (BDL) for the convenient definition of behaviour descriptions was introduced. Different information hiding, restriction, revealing and restrictive revealing operations, which all seamlessly integrate with the BDL were specified. It was explained how a behaviour description is converted into a Behaviour Description Automaton (BDA) and used for the behaviour matching. Additional minimisation techniques for PCMs, in particular collapsing and pruning were introduced to further reduce the visual and physical map size.

The frequent information extraction approach defined a frequency metric for behaviours of creativity maps. It was illustrated how these sequences of actions are classified into the four distinct categories *Preferred, Common, Uncommon* and *Zero*. They are constructed with the help of three frequency thresholds (f_{max}, f_{min} and f_{zero}). The zero category was further divided into an existing and non-existing set. All frequency categories were then utilised for the extraction of behaviour dynamics, which specify changes that can be identified between two or more creativity maps. This allowed for the introduction of the five distinct frequent behaviour categories *Minimal (f_{max})/Minimal (f_{min})/Maximal Alternating, Emergent* and *Disappearing*. It was explained how the information extraction and hiding approach can be combined to build one overall process.

The prototype tool support of this research was presented in the form of the De Montfort Creativity Assistant (DMCA). This initial tool illustrates the implementation of a framework for creativity support; it allows collaborators to create and share artefacts and record creative processes. A modular design realised as a pluggable system enables the integration of tools from any domain. Especially the Collaborative Editor was presented, as the initial version of the DMCA was mainly constructed for the creative writing domain. The De Montfort Creativity Mapper (DMCM), which represents a domain independent facility for the recording of creative processes was introduced. It was furthermore illustrated how the constructed creativity maps are stored in the Creativity Map Repository (CMR). The design of the Information Mining Engine (IME), which implements the presented information hiding and extraction approaches and its integration into the whole tool were explained.

The practical value of the presented approach was proven with the aid of three case studies. The first one described a creative writing workshop with 9 participants that was

held at the STRL. It focussed on the construction of creativity maps. The second case study explained the construction of a behaviour description that allows for the distinction of three time periods in a creativity map. It was shown how the information hiding process is used to conceal irrelevant information in each of the phases. The frequent information extraction process for a corpus of 10 creativity maps was demonstrated in the third case study. It explained the dynamism of three distinct behaviours and the tracking of unexpected changes in their developments. Interpretations for a possible creativity support were presented.

9.2 Evaluation

Five research questions were specified in Section 1.3. They are evaluated based on the presented thesis.

Is it possible to record the creative process computationally?

The recording of creative processes was discussed in Chapter 4. It was explained that a user interactive recording process is preferred, because no additional knowledge for the identification of the performed actions is necessary. The implementation of this system, in particular the DMCM was discussed in Chapter 7. It was illustrated how its customisable design allows creators to adapt the tool to the personal needs. The initially recorded creative process describes a chronologically ordered sequence of actions. Section 4.2.2 introduced the strategy that is used for the identification of similar states, which are then utilised for the construction of the creativity map structure. This comparison is kept flexible and can be defined for any viewpoint of the creative process. However, it was also emphasised that additional knowledge about situations when the creator returned to previous stages is usually required to assist the identification of similar states. The resulting creativity map represents the recorded creative process.

How can the creative process be used to mine for behavioural information?

The three entities data, information and knowledge were specified in Chapter 4. It was explained that data describes the transitions of a creativity map, information the connected sequences and knowledge the behavioural patterns. The relationship between these three elements was illustrated as the knowledge creation process, which describes the transformation from data to information and information to knowledge. Especially the specification of information as behaviours in the form of consecutive sequences of actions was emphasised. The information mining process is therefore interested in their extraction and

processing. One possibility that was discussed in Chapter 6 is the extraction of frequent information. Different representations of behaviours were explained and their unification and granularity were discussed.

Can (temporarily) irrelevant information of the creative process be hidden?

Chapter 5 introduced a formal language for the specification of behaviour descriptions, namely BDL. It allows to define every behaviour of a creativity map in the form of a behaviour description, which is then used for the information hiding process. The language additionally contains possibilities for the specification of state conditions, which enables the integration of state values into the information hiding process. A behaviour description integrates seamlessly with the presented hiding, restriction, revealing and restrictive revealing operations, which describe flexible possibilities for the information hiding and uncovering. It is converted into a BDA, which is used during the depth-first traversal of a creativity map in order to hide irrelevant information and construct a PCM. Collapsing and pruning techniques, which allow to further reduce the visual or physical map size were also discussed in the last part of Chapter 5.

What kind of frequency related information can be extracted from the creative process and how can it be used to describe the dynamism of behaviours?

The frequent information extraction process was discussed in Chapter 6. A frequency metric that considers the length of a behaviour was introduced and it was explained how it is utilised for the classification of behaviours into the frequency categories *Preferred, Common, Uncommon* and *Zero*. These classes are distinguished with the help of the three freely adjustable frequency thresholds f_{max}, f_{min} and f_{zero}. The development of a behaviour and the dynamics of creativity maps can be tracked with the aid of the five frequent behaviour categories *Minimal (f_{max})/Minimal (f_{min})/Maximal Alternating, Emergent* and *Disappearing*). They allow to analyse creative processes and recognise unexpected changes. An example of this behaviour tracking was explained in the third case study of Chapter 8.

How can the proposed research be used to implement initial tools for computational creativity support?

The DMCA and DMCM as the prototype tool support for this research were introduced in Chapter 7. The former provides facilities for the collaborative creation, sharing and modification of artefacts. Its different layers illustrated an initial and extendible design of a creativity support tool. The DMCM creates possibilities for the recording, construction, storage and analysis of creative processes. It includes facilities for the visualisation of creativity maps and integrates the approaches that were presented in this thesis. A

number of ideas for possible creativity support were described in the third case study. It was explained that assistance can be integrated in the form of messages or feedback as a first step.

9.3 Advantages of the Proposed Approaches

It was mentioned in Chapter 2 and particularly in Section 2.2 that the creativity map as a representation of the creative process has several advantages. The main problem in existing models of creativity [97][11][21] is a non-tangible representation of the creative process. It is usually described as an unconscious process, mainly tacking place inside the head of a creator. Analysis processes that refer to this data, such as data or information mining, become more difficult if not impossible. In contrast to this, a creativity map represents the process in a format that can be stored and analysed. By recording the activities of a creator, the whole process becomes tangible and allows for the creation of reproducible results. The Creativity Map Repository (CMR), which was described in Section 7.8.2 illustrates a reusable data repository. It can be utilised for several analysis tasks, also addressing access outside the De Montfort Creativity Assistant (DMCA).

The described information hiding approach in Chapter 5 enables a convenient expression of creative behaviours. It allows for the construction of behaviour descriptions, which represent sets of behaviours, also containing place holders like the ANY transition or state conditions and markers. Any behaviour of a creativity map can be addressed with this formal language, which in turn enables a more aim oriented and focussed analysis. It is of course developed based on the creativity mapping model and related to its components, in particular states and transitions. The information extraction approach, which is described in Chapter 6 introduces four frequency categories and is able to represent the dynamism of creative processes. In difference to the existing data and sequence mining approaches that were reviewed in Section 2.6.2, the approach focusses on the whole range of frequent information. This includes rarely occurring behaviours as well as preferred ones, as not only the most frequent ones are relevant for the support of the creative process.

A major advantage of creativity maps is the possibility to utilise them for computational analysis and support. Especially as creativity is of high interest in academia and businesses [23][98], these tools can for instance support the efficient production of (desired) outcomes. The specification of data, information and knowledge in Section 4.4 helps in understanding this process and can be integrated into analysis processes when required. Chapter 7 presented an initial tool support for the described approaches. The De Montfort Creativity

Assistant (DMCA) is designed as an extendible software component with the aim to support the creative process of a creator. Its different structures that were described, emphasize the convenient integration and usability of the creativity mapping model. The ability to share and collaboratively modify artefacts allows teams of creative individuals to connect and work together [47].

Summarising, it can be pointed out that the creativity mapping model and its applications are advantageous over the models that were reviewed in this thesis. A tangible representation, which also allows for the reproduction of outcomes, builds the foundation for the information hiding and extraction approaches. Amongst other things, the initial tool support illustrates the implementation of the proposed models and their advantages.

9.4 Validation

The developed models and algorithms were validated with the help of case studies and experiments. The three case studies that were presented in this thesis are one major part of the validation. They show creativity maps from creative writers as well as software engineers. Two other experiments, which are not particularly mentioned here but need to be considered are discussed in [107]. One of them illustrates the creativity map of a musician and the other presents the map of a software engineer. The five experiments therefore cover the distinct domains (creative) writing, music and software engineering. They emphasize the adaptability to various circumstances and the possibility of integrating the dynamics of creative processes. This application of creativity maps in different domains demonstrate and validate the generality of the approach and its domain independence. The particular scenarios, which are described in Chapter 8 illustrate a detailed evaluation of the presented approaches.

It is important to note that the whole project is still in its initial stage and the presented research is the first outcome. This means that further experiments and case studies need to be conducted for the collection of additional data, information and knowledge. In turn, the validation process becomes incremental, starting with the initial results that are presented in this research. As new creative process are recorded and therefore creativity maps are constructed, the Creativity Map Repository (CMR) grows and validation processes are able to resort to larger data sets. New information and knowledge can be generated with respect to the knowledge creation process that was described in Section 4.4 and stored in the system. This might then help to further refine or extend the described approaches, possibly leading to a more detailed validation. However, it is important to note that

the existing data definitely allows for a validation of the proposed approaches, as it was demonstrated in this thesis. The three case studies validate the model of the creative process also elaborating the proposed approaches. A sufficient validation has therefore been successfully conducted in the proposed research.

It was mentioned before that a creativity map records the activities during the creation of an artefact. This rather personal data complicates the determination of correctness of the presented approaches. For example, some creators prefer to work by themselves, whereas others require collaboration for suitable results [47][99]. Approaches which are supportive for the former might not necessarily lead to the same results for the latter. Other circumstances such as a submission deadline or environmental restrictions can become relevant as well. Especially the individual reactions might become essential and possibly need integration into the existing structures and processes. It is then necessary to adapt the described techniques for the most suitable support of the creative process. The mentioned situations illustrate the difficulties of a general validation as a result of individual and maybe even unpredictable circumstances.

To summarise the arguments mentioned before, the presented case studies, examples and experiments have been used for a successful validation. It was illustrated that they build a sufficient foundation for the described processes. In order to incrementally refine and improve the validation, it is recommended to conduct further studies.

9.5 Limitations

Recording of the Creative Process

The approach for the recording of creative processes is designed as a user interactive system. The reasons for this choice were discussed in detail in this thesis. However, it was mentioned in the user feedback of the first case study that the system can lead to a distraction of the creator. This means that some people will possibly not use the DMCM at all. Alternative user interface designs or forms of user interaction might be necessary. They can still integrate with the existing GUI of the DMCM.

State Conditions

The state conditions, which were defined in the BDL are used during the information hiding process. They need to be specified by the analyser, as illustrated in the second case study. Sufficient knowledge about the creative processes and the domains that are going

to be studied is mandatory, as it is additionally necessary to implement the corresponding metric for the comparison of viewpoints. The values that are stored in the states of a creativity map must therefore be known.

Specification of the frequency thresholds f_{zero}, f_{min} and f_{max}

The three frequency thresholds f_{zero}, f_{min} and f_{max}, which are used for the distinction of the four frequency categories *Preferred, Common, Uncommon* and *Zero* need to be specified in order to enable the frequent information extraction. It was illustrated in the third case study that these parameters must be chosen rather small. The thresholds probably change for different creators, projects or domains, which requires sufficient knowledge about creative processes in order to guarantee meaningful results.

Tool Support

The implementation of this approach is still in the prototype stage and some of the described components need further development and improvement. The components that are integrated in the DMCE primarily focus on the Collaborative Editor and the writing domain.

9.6 Future Directions

Improvements in the Recording Process

The recording of creative processes and its implementation in the form of the prototype tool DMCM were presented. However, some of the participants perceived this particular approach as disturbing, like the feedback of the creative writing workshop in the first case study illustrated. It is therefore necessary to discuss about improvements in the recording process or other maybe automated techniques that can be integrated into the DMCA.

Integration of Collaboration

The role of collaboration in creative processes and especially its importance for creativity support were not addressed in this thesis. However, it was illustrated that creativity maps and the underpinning mathematical model are capable of handling collaboration. Research about collaborative creativity, for instance software development teams, is a future direction that would benefit the DMCA development. The tool support already offers interfaces for collaboration, which allows for a convenient integration of any results into an existing structure.

Unification of Behaviours

Creativity maps are used as a domain independent representation of creative processes. To enable a comparison of these maps or alternatively frequent behaviours from different domains, it is necessary to define a mapping between similar terms. As it was mentioned in this thesis, domains probably use various words to express a similar action. Techniques like NLP or even more advanced comparison approaches need to be integrated to enable a unification. This would be beneficial for the comparison of creativity maps.

Integration of Tools from Different Domains

The prototype of the DMCA includes the Collaborative Editor, which was mainly developed for the writing domain. Additional tools that enable the collaboration between creators from other domains would help to gather a larger number of users and in turn more creativity maps. The analysis and extraction process would then be able to construct knowledge about a variety of creative processes, which can be utilised for the creativity support.

Bibliography

[1] R. L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.

[2] Charu C. Aggarwal and Philip S. Yu. Data mining techniques for association, clustering and classification. In *Methodologies for Knowledge Discovery and Data Mining*, volume 1574, pages 13–23, 1999.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. pages 207–216, 1993.

[4] Rakesh Agrawal, Manish Mehta, John C. Shafer, Ramakrishnan Srikant, Andreas Arning, and Toni Bollinger. The quest data mining system. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 244–249. AAAI Press, 2–4 1996.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. pages 3–14, 1995.

[7] Silvano Arieti. *Creativity: The Magic Synthesis.* Basic Books, 1976.

[8] Franz Baader. Logic-based knowledge representation. *KI*, 3:8–16, 1996.

[9] Min Basadur. Leading others to think innovatively together: Creative leadership. *The Leadership Quarterly*, 15(1):103 – 121, 2004.

[10] Min Basadur, Pam Pringle, Gwen Speranzini, and Marie Bacot. Collaborative problem solving through creativity in problem definition: Expanding the pie. *Creativity and Innovation Management*, 9:54–76(23), March 2000.

[11] Margaret Boden. *Dimensions of Creativity.* MIT Press Cambridge, Massachusetts, 1994.

[12] Margaret Boden. *The Creative Mind: Myth and Mechanisms.* Routledge, London, 2004.

[13] James T Brady. A theory of productivity in the creative process. *IEEE Comput. Graph. Appl.*, 6:25–34, May 1986.

[14] Keno Buss. *Behavioural Patterns.* PhD thesis, De Montfort University, 2010.

[15] Keno Buss, Sascha Westendorf, and Hussein Zedan. Mining for behavioural knowledge and information in the creative processes. In *Second International Conference of Creativity and Innovation in Software Engineering. Ravda (Nessebar), Bulgaria.*, 2009.

[16] Brian W. Fitzpatrick C. Michael Pilato, Ben Collins-Sussman. *Version Control with Subversion.* O'Reilly Media, Inc., 2008.

[17] Ming-Syan Chen. Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10:209–221, 1998.

[18] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *Ieee Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.

[19] E. F. Codd. Derivability, redundancy and consistency of relations stored in large data banks. *IBM Research Report, San Jose, California*, RJ599, 1969.

[20] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[21] M. Csikszentmihalyi. Implications of a systems perspective for the study of creativity. In R. Sternberg, editor, *Handbook of creativity*, 1999.

[22] Mihaly Csikzentmihalyi. *Creativity: Flow and the Psychology of Discovery and Invention.* HarperCollins, 1996.

[23] Anne Cummings and Greg R. Oldham. Enhancing creativity: Managing work contexts for the high potential employee. *California Management Review*, 40(1):22–38, 1997.

[24] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[25] Edward de Bono. *Six Thinking Hats*. MICA Management Resources, 1st edition edition, 1985.

[26] Rocco De Nicola and Frits Vaandrager. Action versus state based logics for transition systems. In Irène Guessarian, editor, *Semantics of Systems of Concurrent Processes*, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer Berlin / Heidelberg, 1990.

[27] Maria Cristina Ferreira de Oliveira and Ham Levkowitz. From vidual data exploration to visual data mining: A survey. In *IEEE Transactions on Visualization and Computer Graphics*, volume 9, pages 378–394, 2003.

[28] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer-Verlag, Heidelberg, 4th edition, October 2010.

[29] Robert Dilts. *Strategies of Genius*. Meta Publications, 1995.

[30] Maged El-Sayed, Carolina Ruiz, and Elke A. Rundensteiner. Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 128–135, New York, NY, USA, 2004. ACM.

[31] Ben Shneiderman et al. Creativity support tools: Report from a u.s. national science foundation sponsored workshop. *International Journal of Human-Computer Interaction*, 20(2):61–77, 2006.

[32] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Knowledge Discovery and Data Mining*, pages 82–88, 1996.

[33] Colin Fidge. A comparative introduction to csp, ccs and lotos. Technical report, 1994.

[34] Gerhard Fischer. Social creativity: Turning barriers into opportunities for collaborative design. 2004.

[35] Isaksen G. and Treffinger D. J. Creative learning and problem solving. In A. L. Costa, editor, *Developing minds: Programs for teaching thinking*, volume 2, pages 89–93, 1991.

[36] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73:82–98, 1999.

[37] John S. Gero and Ricardo Sosa. Creative design situations - artificial creativity in communities of design agents.

[38] Arthur Gill. *Introduction to the Theory of Finite-State Machines.* McGraw Hill, 1962.

[39] Paul Giudici. *Applied Data Mining: Statistical Methods for Business and Industry.* John Wiley & Sons Ltd., 2003.

[40] James Gosling. The feel of java. *Computer*, 30:53–57, 1997.

[41] James Gosling and Henry McGilton. *The Java Language Environment. A White Paper.* sun, 1995.

[42] The PostgreSQL Global Development Group. *PostgreSQL 8.4.4 Documentation.* The PostgreSQL Global Development Group, 2009.

[43] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[44] U. Hahn, N. Chater, and L.B.C. Richardson. Similarity as transformation. *Cognition*, 87:1–32, 2003.

[45] J. Han, Y. Fu, K. Koperski, G. Melli, W. Wang, and O. Zaane. Knowledge mining in databases: An integration of machine learning methodologies with database technologies, 1995.

[46] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2000. ACM.

[47] Andrew B. Hargadon and Beth A. Bechky. When collections of creatives become creative collectives: A field study of problem solving at work. In *Organization Science*, volume 17, pages 484–500, 2006.

[48] Evan Heit. Features of similarity and category-based induction. In *Proceedings of the Interdisciplinary Workshop on Categorization and Similarity*, pages 115–121, 1997.

[49] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[50] Wang Hua, Ma Cuiqin, and Zhou Lijuan. A brief review of machine learning and its application. pages 1 –4, dec. 2009.

[51] International Organization for Standardization. *ISO/IEC 14977:1996: Information technology — Syntactic metalanguage — Extended BNF*. International Organization for Standardization, pub-ISO:adr, 1996.

[52] Peter Eades Roberto Tamassia Ioannis G. Tollis, Giuseppe Di Battista. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.

[53] D. J. Isaksen, S. G. Treffinger. Celebrating 50 years of reflective practice: Versions of creative problem solving. *Journal of Creative Behavior*, 38:75–101, 2004.

[54] Scott G. Isaksen and Donald J. Treffinger. *Creative Problem Solving: The Basic Course*. Bearly Limited, 1985.

[55] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms, 1997.

[56] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[57] Mortazavi-Asl B. Jianyong Jian Pei, Jiawei Han. Mining sequential patterns by pattern-growth: the prefixspan approach. In *IEEE Transactions on Knowledge and Data Engineering*, volume 16, pages 1424–1440, 2004.

[58] Micheline Kamber Jiawei Han. *Data Mining. Concepts and Techniques.: Concepts and Techniques*. Morgan Kaufmann, 2000.

[59] Cem Kaner and Walter P. Bond. Software engineering metrics: What do they measure and how do we know? In *10th International Software Metrics Symposium*, 2004.

[60] Brian W. Kernighan and Rob Pike. Regular expressions: Languages, algorithms and software. Technical report, Bell Laboratories, 1999.

[61] Doug Kimelman, Bruce Leban, Tova Roth, and Dror Zernik. Reduction of visual complexity in dynamic graphs. In *GD '94: Proceedings of the DIMACS International Workshop on Graph Drawing*, pages 218–225, London, UK, 1995. Springer-Verlag.

[62] Michael D. Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 1254–1259, Mahwah, NJ, 2005. Erlbaum.

[63] Douglas B. Lenat and John Seely Brown. Why am and eurisko appear to work. *Artificial Intelligence*, 23(3):269 – 294, 1984.

[64] David D. Lewis and Karen Spärck Jones. Natural language processing for information retrieval. *Commun. ACM*, 39(1):92–101, 1996.

[65] Yu-Tung Liu. Creativity or novelty?: Cognitive-computational versus social-cultural. *Design Studies*, 21(3):261 – 276, 2000.

[66] Heikki Mannila. Methods and problems in data mining. In *ICDT*, pages 41–55, 1997.

[67] Pamela McCorduck. *Aaron's Code: Meta-Art, Artificial Intelligence and the Work of Harold Cohen*. W H Freeman & Co, 1990.

[68] Antonio S. Micilotta, Eng Jon, and Ong Richard Bowden. Detection and tracking of humans by probabilistic body part assembly. In *Proceedings of British Machine Vision Conference*, volume 1, pages 429–438, September 2005.

[69] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[70] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81:231–268, 2001.

[71] S. Moran and K. Nakata. Ubiquitous monitoring and human behaviour in intelligent pervasive spaces. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 4, pages 1082 –1087, aug. 2009.

[72] R. S. Nickerson. Enhancing creativity. In Robert J. Sternberg, editor, *Handbook of Creativity*, pages 392–430, 1999.

[73] A. Nurnberger, R. Seising, and C. Wenzel. On the fuzzy interrelationships of data, information, knowledge and wisdom. In *Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American*, pages 1 –6, june 2009.

[74] Alex F. Osborn. *Applied Imagination: Principles and Procedures of Creative Problem-Solving*. New York: Scribner's, 1953.

[75] Alex F. Osborn. *Applied Imagination: Principles and Procedures of Creative Problem-Solving: Third Revised Edition*. New York: Scribner's, 1963.

[76] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, pages 251–258, New York, NY, USA, 1999. ACM.

[77] Jonathan A. Plucker and Joseph S. Renzulli. Psychometric approaches to the study of human creativity. In Robert J. Sternberg, editor, *Handbook of Creativity*, pages 35–61, 1999.

[78] A.J. Pretorius and J.J. Van Wijkk. Visual analysis of multivariate state transition graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):685 –692, sept.-oct. 2006.

[79] Ioannis G. Tollis Roberto Tamassia, editor. *Graph Drawing: DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 10 - 12, 1994*. Springer, 1st edition, 1995.

[80] Jennifer Rowley. The wisdom hierarchy: representations of the dikw hierarchy. *J. Inf. Sci.*, 33(2):163–180, 2007.

[81] Mark A. Runco and Shawn Okuda Sakamoto. Experimental studies of creativity. In Robert J. Sternberg, editor, *Handbook of Creativity*, pages 62–92, 1999.

[82] R. Saunders and J. S. Gero. Artificial creativity: A synthetic approach to the study of creative behaviour. *Computational and Cognitive Models of Creative Design*, V:113–139, 2001.

[83] Rob Saunders. *Curious Design Agents and Artificial Creativity*. PhD thesis, University of Sydney, 2002.

[84] James G. Shanahan. *Soft Computing for Knowledge Discovery: Introducing Cartesian Granule Features*. Kluwer Academic Publisher, 2000.

[85] R.N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function: Part i. *Psychometrika*, 27:125–140, 1962.

[86] R.N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function: Part ii. *Psychometrika*, 27:125–140, 1962.

[87] Ben Shneiderman. Creating creativity for everyone: User interfaces for supporting innovation. Technical Report CS-TR-3988, 1999.

[88] Ben Shneiderman. Supporting creativity with advanced information-abundant user interfaces. Technical Report CS-TR-4042, 1999.

[89] Usama Fayyad Gregory Piatesky-Shapiro Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, pages 37–54, 1996.

[90] Rudi Studer Stefan Decker Dieter Fensel Steffen Staab. Situation and perspective of knowledge engineering. 2000.

[91] Mark Stefik. *Introduction to Knowledge Systems*. Morgan Kaufmann, 1st edition, June 1995.

[92] Robert J. Sternberg, editor. *Handbook of Creativity*. Cambridge University Press, 1999.

[93] Ken Thompson. Programming techniques: Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, 1968.

[94] A. Tversky. Features of similarity. *Psychological Review*, 2:327–352, 1977.

[95] Gertjan van Noord. The treatment of epsilon moves in subset construction. In *Finite-Sate Methods in Natural Language Processing, Ankara. CMP-LG/9804003*, pages 61–76, 1998.

[96] P.E. Vernon, editor. *Creativity*. Penguin Books, 1970.

[97] Graham Wallas. *The art of Thought*. Jonathan Cape, 1926.

[98] Ching-Wen Wang and Ruey-Yun Horng. The effects of creative problem solving training on creativity, cognitive type and r&d performance. *R&D Management*, 32(1):35–45, 2002.

[99] Andy Warr and Eamonn O'Neill. Understanding design as a social creative process. In *Proceedings of the 5th conference on Creativity & cognition*, C&C '05, pages 118–127, New York, NY, USA, 2005. ACM.

[100] Robert W. Weisberg. *Creativity: Beyond the Myth of Genius*. W.H. Freeman & Company, 2nd edition, 1993.

[101] G. A. Wiggins. Towards a more precise characterisation of creativity in AI. In R. Weber and C. G. von Wangenheim, editors, *Case-Based Reasoning: Papers from the Workshop Programme at ICCBR'01*, pages 113–120. Washington, DC: Naval Research Laboratory, Navy Centre for Applied Research in Artificial Intelligence, 2001.

[102] Geraint Wiggins. Categorising creative systems. In Bento, Cardoso, and Gero, editors, *Proceedings of the IJCAI'03 Workshop on Creative Systems*, 2003.

[103] Wendy M. Williams and Lana T. Yang. Organizational creativity. In Robert J. Sternberg, editor, *Handbook of Creativity*, pages 373–391, 1999.

[104] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[105] Lawrence F. Young. Knowledge-based systems for idea processing. *SIGMIS Database*, 22:46–52, February 1991.

[106] Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. In *Machine Learning*, volume 42, pages 31–60, 2001.

[107] H. Zedan, A. Cau, K. Buss, S. Westendorf, S. Thomas, and A. Hugill. Mapping human creativity. In *Proceedings of the 12th Serbian Mathematical Congress, Novi Sad*, 2008.

[108] Hussein Zedan, Sascha Westendorf, and Keno Buss. The effect of collaboration and co-creation on the creative processes. In *Second International Conference of Creativity and Innovation in Software Engineering. Ravda (Nessebar), Bulgaria.*, 2009.

[109] Du Zhang and J.J.P. Tsai. Machine learning and software engineering. pages 22 – 29, 2002.

[110] Minghua Zhang, Ben Kao, and Chi-Lap Yip. A comparison study on algorithms for incremental update of frequent sequences. *Data Mining, IEEE International Conference on*, 0:554, 2002.

[111] Bin Zhou, Daxin Jiang, Jian Pei, and Hang Li. Olap on search logs: An infrastructure supporting data-driven applications in search engines. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 1395–1404, New York, NY, USA, 2009. ACM Press.

[112] Chaim Zins. Conceptual approaches for defining data, information, and knowledge: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 58(4):479–493, 2007.

Appendix A

Creativity Maps of the Creativity Workshop

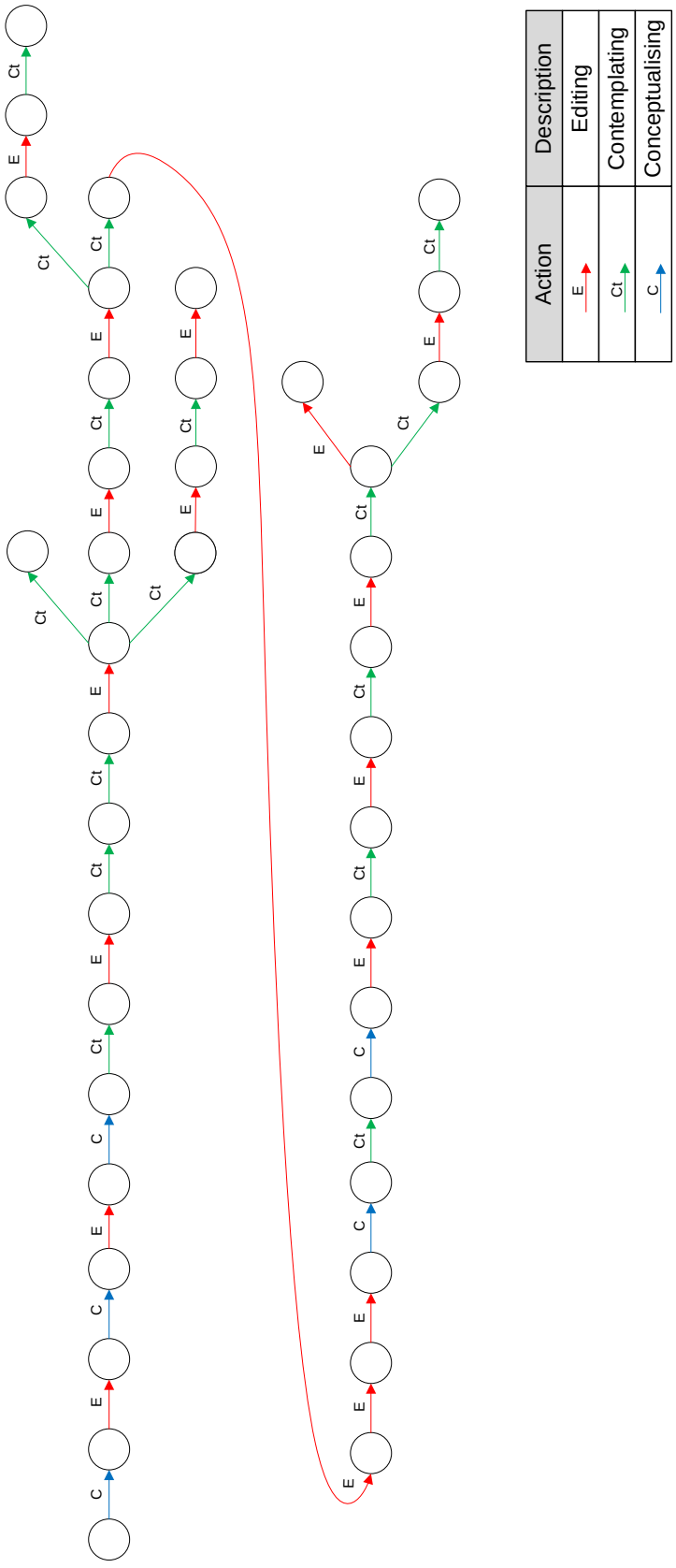


Figure A.1: Creativity Map - Participant 1

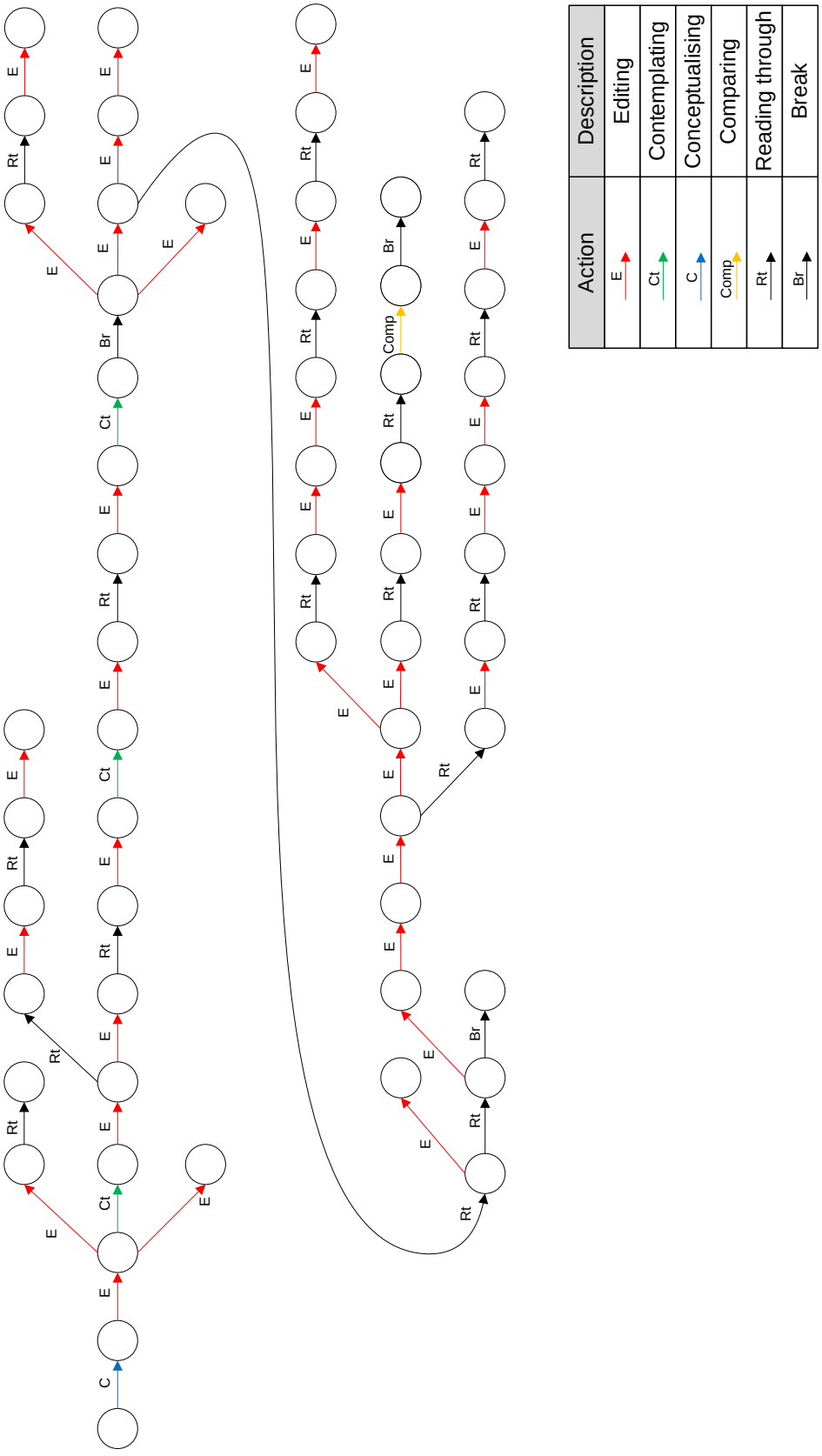


Figure A.2: Creativity Map - Participant 2

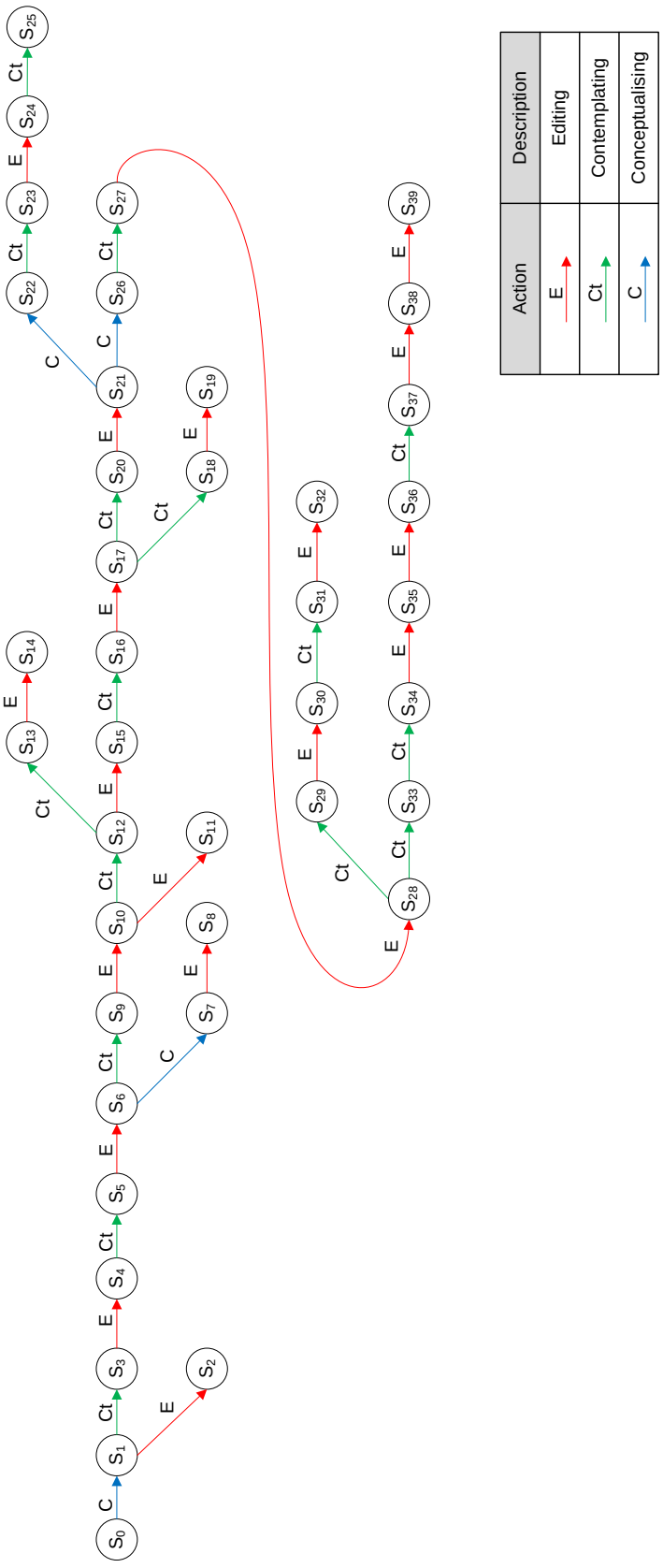


Figure A.3: Creativity Map - Participant 3

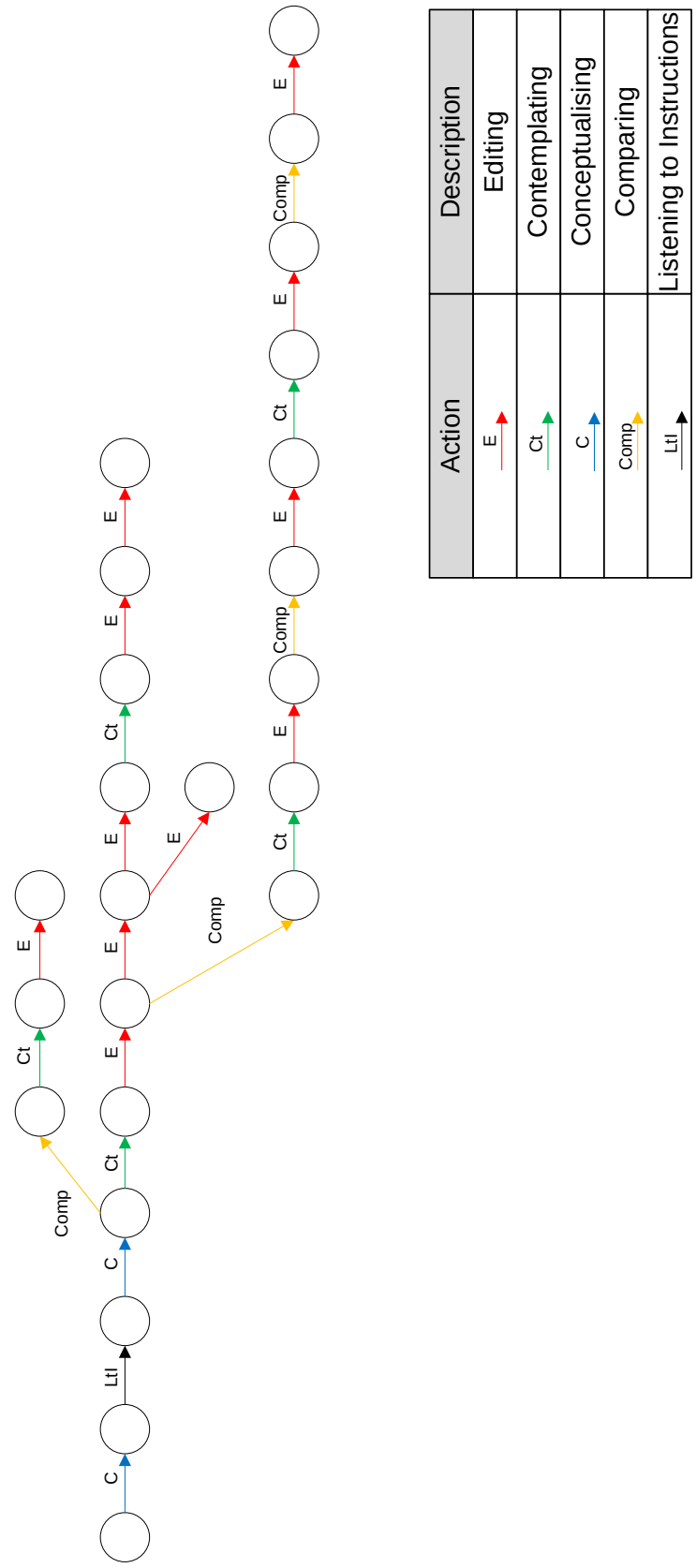


Figure A.4: Creativity Map - Participant 4

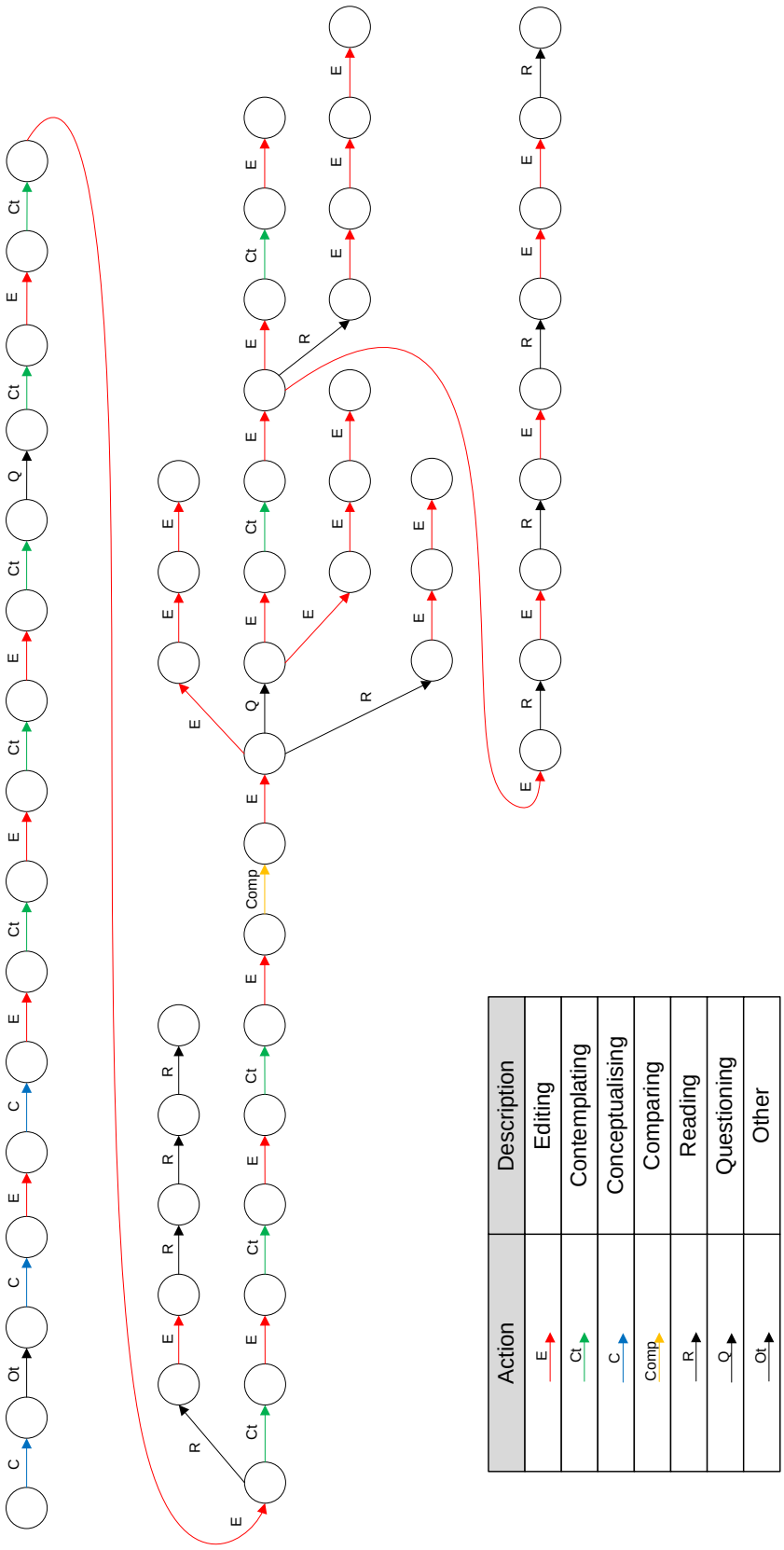


Figure A.5: Creativity Map - Participant 5

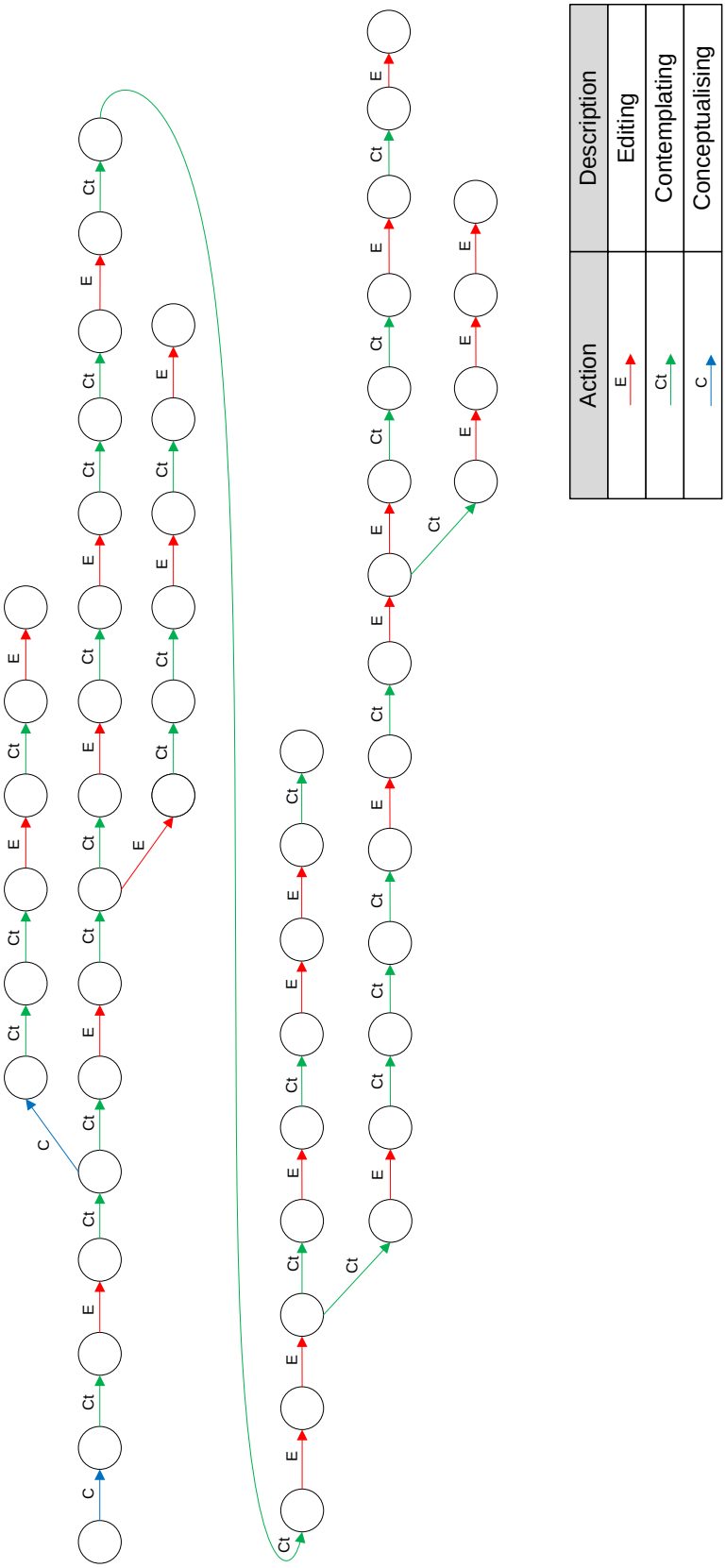


Figure A.6: Creativity Map - Participant 6

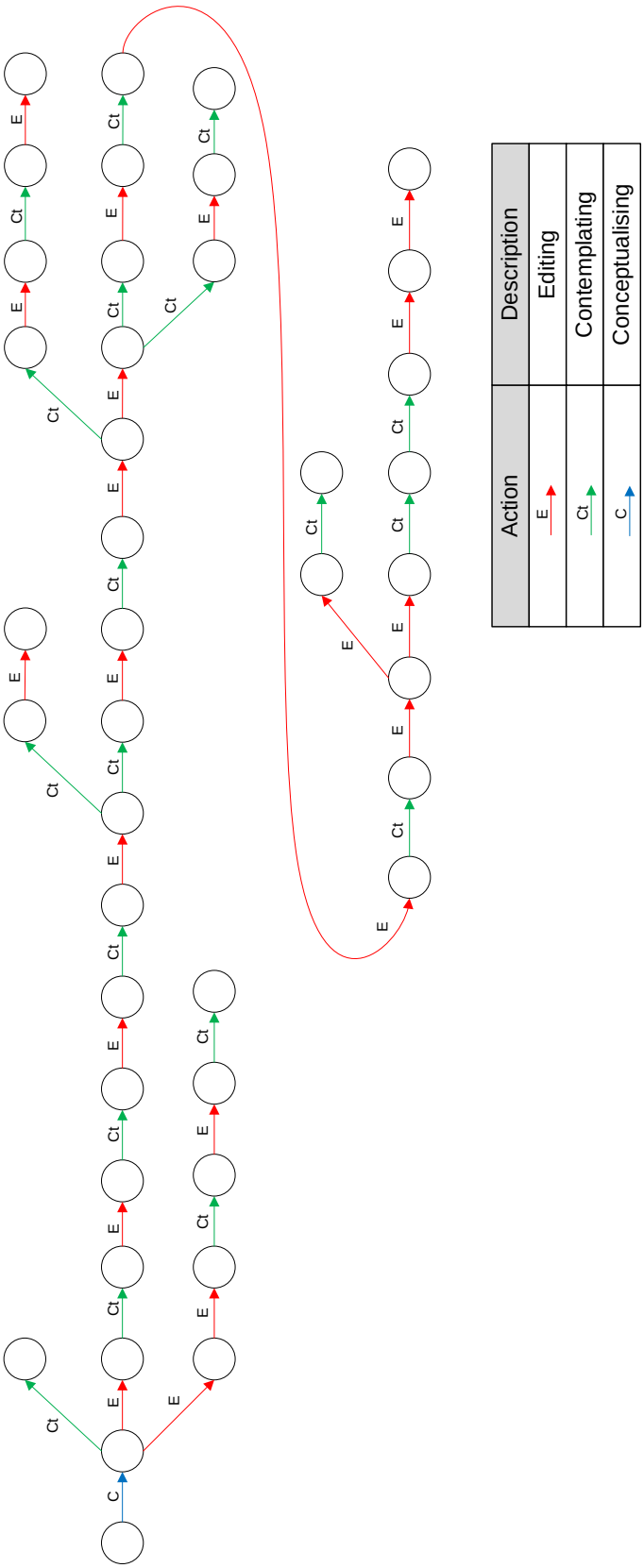


Figure A.7: Creativity Map - Participant 7

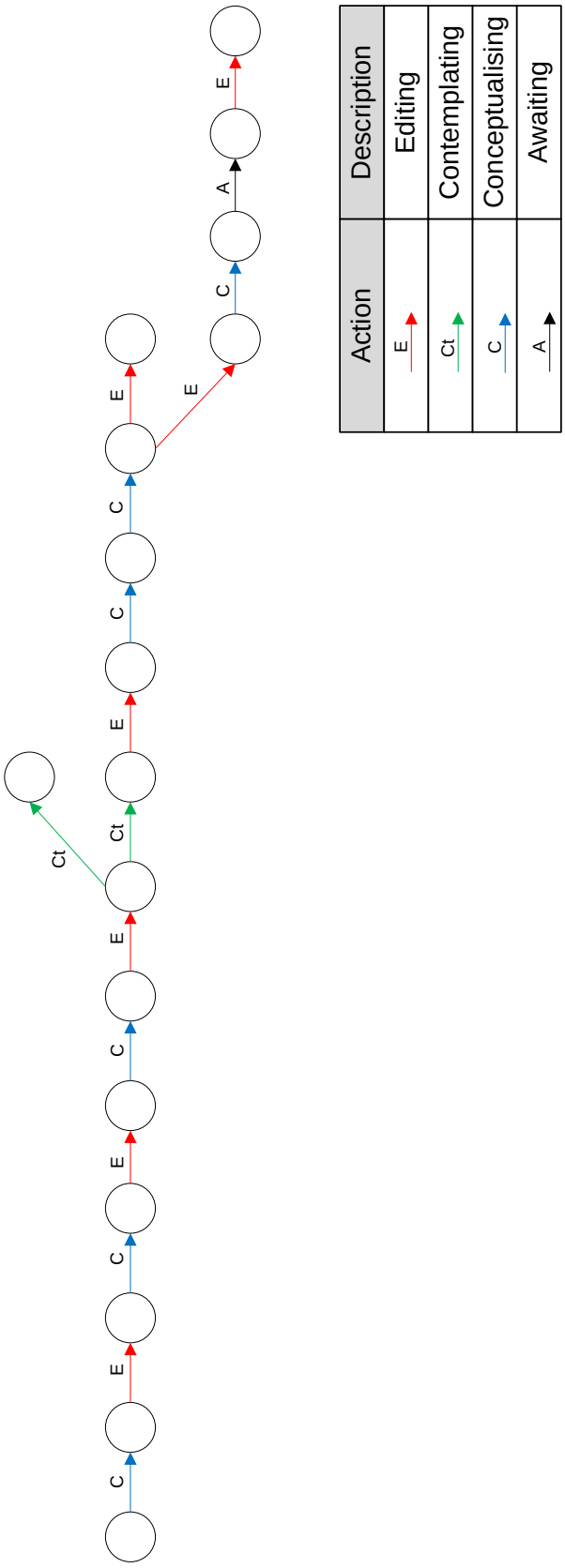


Figure A.8: Creativity Map - Participant 8

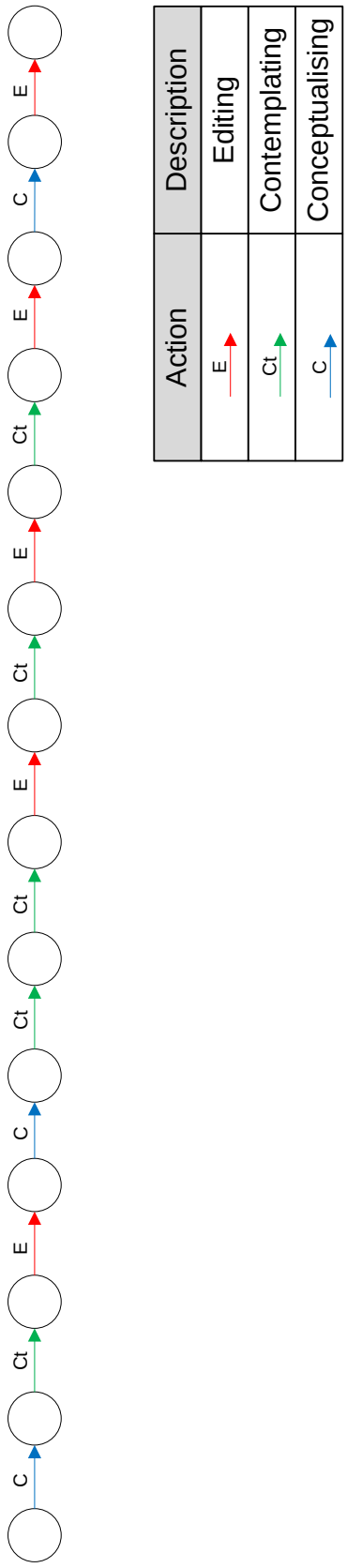


Figure A.9: Creativity Map - Participant 9