Experience based recommendations for improvement of case execution

van Zoest, R.W.

Award date:
2009

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

Experience Based Recommendations for Improvement of Case Execution

By
R.W. van Zoest

Supervisors
Dr. Ir. H. A. Reijers, TU/e
Drs. P. Eertink, Pallas Athena

Dr. Ir. N. Sidorova, TU/e
Ir. M. H. Schonenberg, TU/e

Eindhoven/Apeldoorn, February 2009

Abstract

Experience of finished cases from a process is very valuable. All these finished cases together consist of all best and worst practices. These practices can be found in the case decisions, case results and case identity. Logging this case information into a case base will retain the best and worst practices of the finished cases. This can also be called the “experience” of cases.

The user who handles a case in the process can be supported in his decision-making by the experience of earlier cases. This report is about a decision support service, which is called the recommendation service. In the presented recommendation service the recommendations are based on the case base. These recommendations can be about (1) which resource to allocate to which work item, (2) the sequence and choice of activities in the case path, and (3) decisions within a work item.

It will be shown that this type of decision support helps the user in the decision-making process. Recommendations help in controlling complex cases, while there is flexibility to apply the user’s own opinion. A prototype implementing these recommendations is presented..

Table of Contents

1 Master Graduation Project at Pallas Athena

1.1 Problem area

When executing a process, a defined goal for the case has to be followed. Decision-making during the process enables the achievement of this goal. The problem is that complexity is becoming enormous for business processes, which results in hard decisions. Although users have a limited overview, user intervention is still required, because control by intelligent decision-making systems without user intervention is still a utopia.

Workflow Management Systems (WfMS's) are a good solution to structure and control complex processes. However, WfMS's might provide too much control and reduce the grip of the user at the decision-making. It is therefore important to provide a balance between control and flexibility. Control is required, because certain goals have to be realised. Flexibility is required, because the user of the process must have enough space for her preferred decision.

A recommendation service could be a welcome addition for decision-making process. This service provides support and direction by generating recommendations which can be followed, but the user still can influence decision-making to empower her perception. The idea of the recommendation service is that per case a certain target function is set. The goal can be to optimize some sort of parameter (e.g. cost or time). The service selects similar cases that are executed and logged earlier in the process, and defines a recommendation based on these cases. By using the case base it is possible to utilize the "experience" of similar cases.

1.2 Research question

This master graduation project is about the development and evaluation of a recommendation service. The main research question is:

> *How can the business value of a workflow process be increased, by applying strategies that lead to case-specific recommendations?*

By defining target functions, building a recommendation service and evaluating it, the answer to the research question can be realized. Therefore the topics of the following research goals (RG) have to be a studied and answered:

RG1: *Define target functions.*
RG2: *Describe how to find similar cases in the log file*
RG3: *Describe how to define a recommendation on the basis of the available data.*
RG4: *Design a process that contains a prototype of a recommendation service.*
RG5: *Evaluate the target functions, types of recommendations and the prototype itself.*

1.3 Phases in the project

Although phases will occur in parallel and overlap, the following rough steps can be distinguished.

1. The first step taken is to get familiar with the Protos and FLOWer products of Pallas Athena. This was done via a course and self-study. The recommendation service was built with FLOWer.
2. After getting familiar with FLOWer the target functions were defined. The general hypothesis was that when the process is executed with the recommendations, the process will have better results for the target functions then without them.
3. Research was done about how to mine similar cases from the data log and how to transform this data into a recommendation.
4. The recommendation service was built and tested, this took most of the time.
5. The recommendation service was simulated. After the simulation the target functions and the recommendation service were evaluated.
6. The result of this master graduation project will be a prototype recommendation service and a master thesis where the theory is described and the prototype is evaluated.

1.4 About Pallas Athena

This master graduation project was carried out at Pallas Athena [1] in Apeldoorn. For over 15 years Pallas Athena has been a supplier of BPM software and solutions. Protos and FLOWer[1] are the main products that are developed. Nowadays these are integrated into the BPM|One suite. Pallas Athena provides companies the opportunity to capitalise on the past, optimise the present and anticipate the future.
Pallas Athena has more than 1700 customers and over 1 million users can be found in many different types of companies in all industry sectors. Pallas Athena has about 150 employees. And Pallas Athena's software and solutions are sold in more than 30 countries throughout the world.

1.5 Outline

Chapter two will introduce the topic of the recommendation service with earlier work, a justification and a short description of the different aspects of the recommendation service.

The following three chapters describe in theory the complete recommendation service. Chapter three starts with the design considerations about several topics. Chapter four describes which requirements have to be met. Chapter five describes the design decisions. The structure of these chapters (3-5) are the same. So, the third issue in 3.1 is also the third issue in 4.1 and 5.1.

In chapter six it is described how the prototype of the recommendation service is developed in practice. Chapter seven presents an evaluation of the prototype of the recommendation service.

Conclusions for the recommendation service are made in chapter eight, together with recommendations for future work.

[1] A description of FLOWer is provided in chapter 6.

2 Recommendation Service Introduction

In this chapter the basic idea of the recommendation service is described. In the first paragraph a short introduction is given about the development of the area of Decision Support Systems. In the second paragraph it is discussed why a recommendation service adds value to an organization. In the third paragraph it is explained how the recommendation service works and what types of recommendations are defined. Paragraph four describes how the recommendation service fits in the BPM vision of Pallas Athena.

2.1 Earlier work

Why a recommendation service? Why not develop a fully automated decision making system? The idea of fully automated decision-making systems exists for decades. However fully automated decision making seems to be a utopia. Due to bad design of artificial intelligent applications and no acceptance by the end-users, the trend went from fully automated decision to user decision support from the 1970's until the present [2].

The change to collaboration between the user and the recommendations service, instead of automated decision-making, can also be analysed from the knowledge management view. Three types of knowledge repositories in an organization can be defined [3]: (1) External knowledge, like articles and market research, (2) Structured internal knowledge, like product specifications and production procedures, and (3) Informal internal knowledge, like know-how in the minds of persons. The second and third knowledge repository cannot be managed fully by a person alone, or by a recommendation service alone. Computer support is required to get an overview over the complex and enormous amount of structural internal knowledge. The user has the best perception of the informal internal knowledge. So, the user and the recommendation service together could form a good combination.

In the last forty years many types of Decision Support Systems (DSS) were developed (see also Appendix A). Arnot and Pervan [4] critically analyzed many articles in the DSS area. This research area is large; about 15 percent of the articles in IS journals have the topic of DSS. The conclusion was that DSS research w.r.t. theory is dominated by theory building, but almost not to theory testing, and theory refinement. This is surprising, because the DSS area is evolving for forty years and should have more focus on theory testing and refinement, but it is not. Sceptics would say DSS won't evolve. However, the high focus on theory building can be caused by the fact that DSS is a difficult area, or by ever-developing technical possibilities. New technical possibilities cause development of new DSS movements, like data warehousing and business intelligence, for which theory building is the first step.

There are a lot of examples of recommendation services on the web. Examples are e-commerce sites like Amazon.com where visitors are supported with recommendations like "Customers Who Bought This Item Also Bought" and "Buy This Product and Related Accessories". These

recommendations are derived from other users' preferences, who share the same profile. Recommendation services on the internet are frequently applied for supporting persons with the right product or service.

Optimized case execution is another recommendation services domain. There are a lot of frameworks and applications for decision making. On a high level most of the frameworks and applications contain a stepwise process from Insight to Decision to Action [2]. The output of these frameworks and applications mostly exists of a recommendation list with a preference for the alternatives. This is the support to the user. There is an enormous variety of methods for generating recommendation lists. This is caused by the amount of available techniques and the different environments where the decision-making takes place.
An example of a recommendation list for improved case execution is the product based workflow support described in [5]. Here the next step to be performed is determined by using a strategy in combination with a product data model. The product data model is a description of the structure of the product data elements.
Another example is the recommendation service based on the history of other cases [17]. The optimization can be described by target functions. The recommendations are based on cases that are selected with similarity measures. The last two described services are developed at the Eindhoven University of Technology and implemented in the ProM Framework [6].
The recommendation service in [17] was an inspiration for the prototype described in this report. This earlier recommendation service is also inspired on a technique. This is the case-based reasoning technique (CBR). This technique will be considered in chapter 3 and will lead to the final design in chapter 5.

The goal of this project will be to build an operational recommendation service for a real life business process. By evaluating the service it will be analysed whether implementing a recommendation service into a process will result in improved case execution. The CBR technique and the recommendation service described in [17] will be used as a basis for the prototype.

2.2 Justification

Does the recommendation service add value to a process? The theoretical part of this answer is described here. The recommendation service in practice is evaluated in chapter 7. From a theoretical point of view there are two ways to justify the service.

A recommendation service provides a solution to the trade-off between control and flexibility. WfMS's are a fine solution to structure and control complex processes. However, WfMS's might provide too much control and reduce the influence of the user at the decision-making. It is therefore important to provide a balance between control and flexibility. Control is required, because certain goals have to be realised. Flexibility is required, because the user of the process must have enough room for the final decision-making. A recommendation service is a solution for flexible decision-making while still providing guidance.

A recommendation service is an extension to a business process that provides decision support. Normally, users repetitively take a decision during the process; after each decision a part of the case is executed and finally there will be a case result. This is represented by the dark rectangles in Figure 1. However, for a recommendation service the process is logged and the data is analysed (light rectangles). Current cases can take advantage of the "experiences" of earlier cases. So, the recommendation service is an extension for reusing historically logged data.

The recommendation service is very useful when no well-based decisions can be made. A user should request a recommendation when the following situations occur:

- Context tunnelling [9,10]: The user lacks insight into earlier and yet to be performed work. A user may not be aware of her lack of insight, so automatic recommendations for a case may be required.
- A lack of information: The user has no information available for the decision, for example when an exception occurs.
- A lack of business rules: There are unclear, contradictory, or no business rules for how to perform a certain task, or how to decide on issues for a task.

When these situations occur, the recommendation service will find similar cases and supports the user with a well-based recommendation for decision-making.

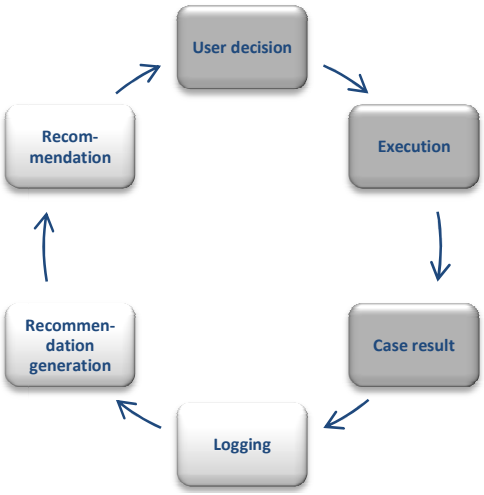

Figure 1: Process (dark) and recommendation service (light)

2.3 Implementation and usage within a business process

There are certain conditions that have to be met before a recommendation service can be used as an operational service. Specialists or managers should decide on several aspects of the recommendation service. Also strategies and alternatives for decisions should be defined clearly.

The recommendation service should only be applied when at least the following requirements are met:

- Management should have a clear understanding of the entire business (e.g. via monitoring). From this understanding a strategy could be defined. This can be translated into an instruction for setting target functions for all cases in a process or individual cases.
- Detailed knowledge about the process has to be available when configuring the recommendations. For example, for attribute-based recommendations the key case properties should be defined.
- Different roles should be distinguished in terms of knowledge, skills, abilities and authorization. These defined roles could be used as different alternatives for the recommendation service.
- The work items should be defined and should have an identity. This makes it possible to distinguish different paths for cases.
- Recommendations require historical logged data. It should be possible to log information about cases in the case base and there should be enough cases, before a reliable recommendation can be generated.

There could be a lot of different situations to implement a recommendation. Recommendation services are no good for the following situations:

- The effort for requesting, generating and handling a recommendation can demand too much of certain resources, like performance of a server, or handling time of the user. An example is the situation where in the past an employee needed 10 seconds to decide, but with a recommendation service he needs two minutes. There has to be a healthy trade-off between the effort made with and without a recommendation service.
- The right answer to the decision is already known, but the information has to be looked up somewhere. A fact is absolutely successful for the problem, where a recommendation can only introduce recommendations with uncertainty.
- The problem is too important and the decision should be made by specialist or managers.
- The impact of having an incorrect recommendation is totally undesirable.

2.4 The recommendation service and the BPM vision of Pallas Athena

The idea of a recommendation service could be implemented within different organizations. Pallas Athena has a certain vision how to improve a process. The recommendation service can be seen as another way to improve a process. This means support within the process execution. As can be seen in the figure below.

Pallas Athena identifies two improvement cycles within the business process: the inner and the outer circle. The emphasis of the inner circle is fully on the operational management of a process. It concentrates on being in control of the daily state of affairs. The outer circle is situated on a more strategic level and here a process' performance is measured and compared to the targets that have been set. Changes in the outer circle are about the structure of the process and have greater risk and impact. Implementation of a modification requires input from higher management and IT. Changes in the inner circle are about process parameters and can be done by operation managers.

The recommendation service is situated in the inner circle and the outer circle. It is situated in the inner circle because it provides real-time decision support during the execution of a case. However, setting the target function is done at the strategic level of the outer circle, where also the process's performance is measured and compared. It is possible to steer the setting of target functions by following the strategy of the outer circle. It is also possible that local managers set the target functions, when they recognize the need for optimization for a certain process parameter.

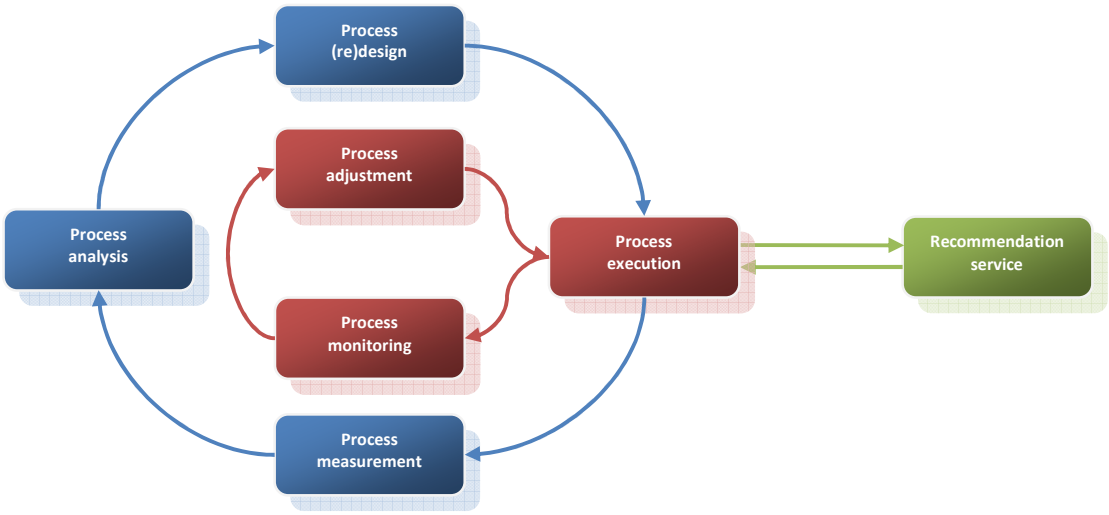

Figure 2: Inner and outer circle in combination with the recommendation service

3 Design Considerations

In this chapter the design considerations about the development of the recommendation service will be explained. Each topic is discussed. Requirements about these issues and remarks will be defined in chapter 4. In chapter 5 it will be discussed which design decisions are made for the recommendation service.

3.1 The CBR technique and the recommendation service

Although the idea is to base the recommendation service on the article of [17], the case-based reasoning technique (CBR) will first be analysed and described.
CBR is a methodology for finding solutions for the current case by analysing already finished, similar cases. All cases which have been executed and finished in the process are stored in a case base, which is a database with cases, their solutions and the results [7]. The case base is used and updated in five steps (Figure 3) [8]. These are (1) Retrieval of cases with similar problems, (2) Reusing the solutions of the retrieved cases, (3) Revising these solutions, (4) Reviewing and evaluating whether the outcome of the formulated solution is acceptable, and (5) Retaining the case in the case base for future analysis.

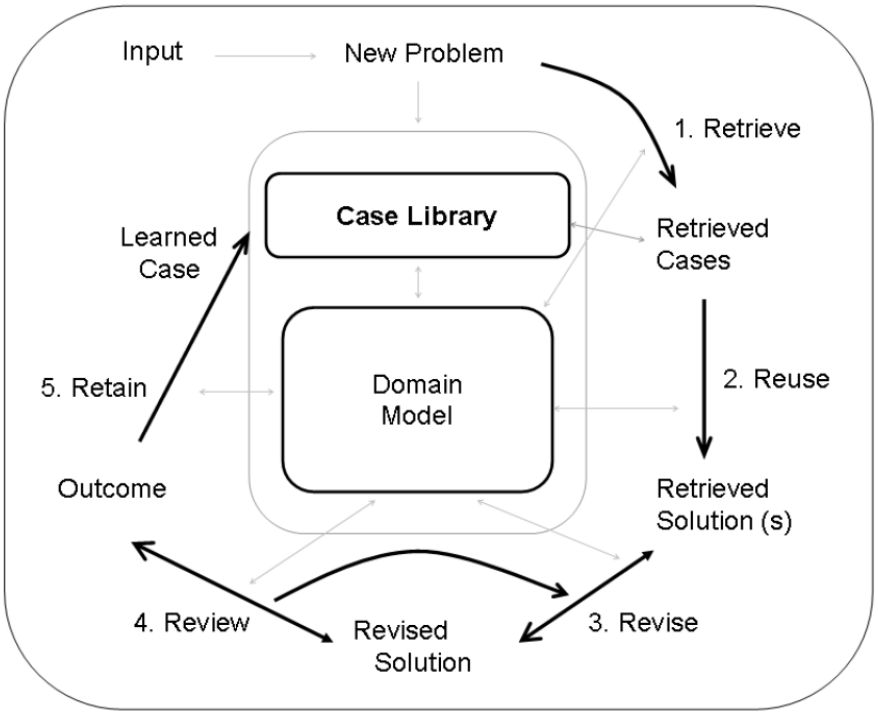

Figure 3: The case based reasoning problem solving cycle [8]

Issues/remarks

1. The CBR technique has from a theoretical point of view the advantage that it reuses cases from the past to learn from behaviour in the past. It is a learning system, so it retains information from past cases to support current cases.
2. The case base (case library) [7,8] can be defined as a database in which the identity, the solutions and the results of cases are logged, see Figure 8. The case base can be seen as a knowledge source (knowledge container) that is used in problem solving [11]. However, rule-based systems have only knowledge sources like facts and rules. For example, the recommendation service in [17] uses the case base, the user's knowledge, the similarity measures and the target function as knowledge sources. When decision support is requested, all these sources can be consulted and used for a solution.
3. From a practical point of view the CBR technique might not be feasible, because of performance issues. This might occur when the case base (case library) contains too many retained cases. Then aggregating information may require too much time and performance of the calculating systems.
4. Another issue is the influence of recently added cases in the case base. When there are extremely many cases in the case base, new trends may not be found, because the much older majority of the cases determine the results.
   Business processes are always influenced by change, e.g. from the environment. This means cases of five years ago might not be that representative any more. Maintenance of the domain knowledge is required. For example, a moving average for the cases could be applied. This means cases will be relevant for a certain period of time. Another solution for this problem could be the idea of qualitative introspective learning. This technique estimates the direction of change and tunes old results for current situations [12].
5. Without cases in the case base the CBR technique cannot be used. So, no feedback without too few cases or an empty case base.
   A cold start is a situation in which the case base has not enough variability. Not all alternatives are applied in the already known finished cases. The danger of a cold start is that recommendations are generated, but the recommendation service cannot choose the best alternative. The alternatives that have not yet occurred will not be recommended and thus not become a part of the case base. However the other alternatives will be recommended, which result in too much emphasis on these alternatives.
   One solution can be that cases in the business process should be processed without decision support from the recommendation service until the moment that the case base contains enough cases.
   Another solution can be to provide variability to the case base by simulating cases. If the amount of possible identities can be calculated, the amount of to-be-simulated cases is known. However, it is required to parameterize the simulation. It might be difficult and complex to predict precisely which actions have which results. More difficult is to set parameters for the simulation for circumstances that are influenced from the outside of the process. Although the parameters might not be totally right, an indication might be enough.
6. Contamination of the case base may occur. Therefore rules are required. It is in the concept of a recommendation service to omit a complex web of rules. However rules will always exist, like ethical and legal rules. Cases with illegal decisions in the case base should be filtered out, because future recommendations can be based on these cases. A solution can be to give the user the possibility at the end of the case to add a comment whether the case represents a well-processed case or not.
   Deleting information from the case base is dangerous, because it might cause a wrong balance. For example, if a case with a very positive result for alternative A is deleted, then automatically alternative B becomes better. Thus, maintenance is important.

3.2 Types of recommendation

1. Recommendation build-up

In this section it is described which types of recommendation can be defined, on what kind of information they could be based and what they recommend. Recommendations can be distinguished by different properties. One property is the input information whereon the recommendation is based. This is also called the case identity, which is the criterion for the similar cases subset. This similar cases subset is the dataset on which the recommendation is based.

1. Recommendation based on the attribute values of a case (attribute-based)
2. Recommendation based on the activity sequence of a case (path-based)
3. A hybrid form of attribute-based and path-based

Another property is for what kind of decision is recommended:

A. Recommendation for the value of a case attribute
B. Recommendation for the next activity in the case path
C. Recommendation for the role that executes a certain activity

In theory all combinations are possible (1A, 1B, 1C, 2A, etc.).

2. Recommendation scope

For the path-based recommendation there is an issue about the scope of the recommendation. Until now the recommendation was assumed to give a suggestion for one decision. For example, it is recommended at status 5 that one should choose option K as the next activity. This is based on the history of other similar cases in the log database. What if the situation occurs that a recommendation service is able to recommend a number of decisions at once (a path recommendation)? This means that the process has such properties that for example the next three steps to be taken depend on the current decision. So, if decision K is taken after status 5, then automatically decision L has to be taken after K and decision M has to be taken after L.

Such recommendations have certain advantages. One advantage is that if the recommendation is similar with the user's opinion and this recommendation is followed, it can be forecasted what capacity is needed in the activities of the recommended path. This is because it is known which current cases follow which paths (because this was determined at status 5). Another advantage might be a lower occupancy rate of the recommendation service, because there are fewer recommendation requests. A disadvantage may be that the requests are tougher to calculate.

3.3 Information structure

1. A possible information structure for the case base is described in [8]. On a high level the cases should be stored as triplets like {problem, solution, outcome} These three parts together form a logical structure for analysing past cases.

3.4 Measuring similarity

Measuring similarity is the technique that will be applied for deriving a subset of similar cases from the case base. In CBR, measuring similarity is represented by the retrieval phase, as can be seen in Figure 3 [8]. Although in CBR cases with similar problems are retrieved, here similar cases are retrieved, assuming that these cases' problems are similar. The requirements to the similarity can be defined by different algorithms. The algorithm will be translated into a logical query that eventually selects the similar cases from the case base.

1. Identity by case attributes

An example of a similarity measuring algorithm is collaborative filtering (CF) and is much applied in news, movie recommendation, or product recommender systems on the internet. CF is applied when items or products cannot be described by explicit properties [13]. CF also assumes that persons with the same user profile agree on a decision in the future [14].
For example, person A and B are both interested in the same movie genre and are therefore assigned to the same type of user profile. The next preference occurs: person A likes movie M. Thus, based on the user profile, it is assumed that person B likes movie M too. In the future, if person B requests a recommendation for a movie, movie M will be recommended, because this is the preference of other persons (i.e. person A) of this profile.
However, applying the CF algorithm for case recommendations is difficult. This is because the person of the example can in the recommendation service be seen as the case in the process, and cases cannot build up history (identity) from earlier trials in the process. It is however possible when the same case flows through different processes.

Content-based filtering (CBF), or attribute-based filtering, derives recommendations based on cases that are equal for some predefined attributes of that case class. The content-based filtering (CBF) algorithm suits better in a case recommendation system than CF does. CBF also performs better than CF, but only if the case contains well-structured content [15].
For example, a recommendation is requested for deciding whether or not to release movie M in the Netherlands, based on the profit of earlier similar cases. Movie M has the attributes genre, actor and released country. For this example, Movie N is the reference set, because only this movie has the same values for the attributes. After analysing the reference set, it was concluded that this movie had a very low profit, thus it is recommended not to be released.

2. Identity by case path

Another measuring algorithm calculates path similarity. It compares the partial case path of the current case with the complete paths of cases from the case base. In [17] several abstractions for defining a case path are provided. These are represented in Figure 4. The prefix is an abstraction that contains the most detailed information about the partial case. It is possible to increase the detailed information of the partial case with all sorts of logistic, financial, etc. properties, like throughput time per activity, etc. However, here only the set of activities are analysed. The lower the abstraction in Figure 4, the less detail is required to become a similar case.
The reason to lower the required level of information that has to be similar depends on the type of activities and the type of process, or the amount of cases in the case base. For example, if similarity only depends on the number of times that a case was denied at a quality check (e.g. activity C in figure 4), then the multi-set abstraction might be more suitable than the prefix abstraction. It is possible to implement the prefix abstraction, because it also contains this information. However, abstractions with more required similarity come at a cost, like more calculation power and calculation time.

Partial case example: A B C D C D C D E , with current state after E		
Abstraction name	**Example set**	**Feasible when it is relevant ...**
Prefix abstraction	<A,B,C,D,C,D,C,D,E>	... that all known details of the partial case are taken into account.
Multi-set abstraction	{A,B,C3,D3,E}	... how often an activity occurred.
Set abstraction	{A,B,C,D,E}	... whether an activity occurred or not.
Partial sequence *(e.g.last four)*	<D,C,D,E>	... what the sequence is in the defined horizon (last X places).
Partial multi-set *(e.g.last four)*	{C,D2,E}	... how often an activity occurred in the defined horizon.
Filtered sequence *(ignore C,D)*	<A,B,E>	... whether an activity occurred in the defined horizon, while only considering certain activities.

Figure 4: Possible path abstractions of cases [17]

In [16] a recommendation service is evaluated by its path-based recommendations. Recommendations based on the Prefix (a.k.a. Sequence), Multi-set and Set abstraction where compared to random recommendations. The Prefix abstraction performed significantly better than the other three alternatives. Therefore the Prefix will be used for the prototype.
The Prefix abstraction allows for a direct comparison between the partial path and the path of a logged case. Thus, the partial path should be 100 percent similar before it is selected in the similarity set. If the Prefix abstraction is selected as similarity measurement, the order and/or the frequency of activities should be relevant.

The advantage of the path-based identity above the attribute-based identity is that path-based similarity measurement is domain independent. For attribute-based identity the attributes have to be selected per process.

3. Quality issue

Monitoring the retrieval of similar cases from the case base is important. In this part of the recommendation service the quality of information is determined. For example, the amount of similar cases that are available for the option list, indicates the quality of the recommendation. Another interesting indicator is required amount of cases in the case base, before recommendation is possible.
If a quality check would be implemented in the recommendation service, the subset with the similar cases is the best point in the recommendation service where the quality can be analysed. If there are not enough similar cases, the option list is based on too few target values (TV's) [2] and the recommendation guarantees no quality. This might cause recommendations which are based on a subset that does not represent the process. The minimum required number of similar cases for a recommendation (minSC) should be larger than the number of alternatives in the recommendation, because it is wishful to base each alternative on minimal one case. If each alternative occurs with the same chance, then each alternative would (on average) have the same number of similar cases.

[2] A TV is the value of a parameter belonging to a specific alternative and a specific TF, as will be described in next chapter. It can be seen as the feedback on a certain alternative for a certain TF.

Then each alternative would be based on the same quantity of TV's. However, the distribution of the number of similar cases to the different alternatives is not known and cannot be assumed to be uniform. So, it is not known that when there are 20 similar cases for 4 alternatives, that there are 5 similar cases for each alternative. For example, if there are 20 similar cases found for the current case, then it is possible for alternative 1 to have 10 similar cases, alternative 2 has 5, alternative 3 has 4 and alternative 4 has only 1 similar case.

4. Calculating target values

Each similar case represents an alternative, with a target value for indicating how good the alternative is. For example, the case 45100382 followed Alternative 2 and had a TV of 86 for TF 3.

The calculation of a TF is based on the total TV for a sequence of activities [16]. This is the sequence of activities of a process from start to end. If a shorter part of the process would be defined for optimization, there will be the risk of local optimization, instead of global optimization (for the whole case). The sequence with the most optimal total target value is the sequence that should be recommended.
It can occur that a local recommendation results in a less optimal total target value than when an alternative is chosen that is not locally optimal, but is globally optimal. For example, two sequences of activities are possible:

Sequence 1:	A(10)	-	B(10)	-	C(10)	-	D(40)	-	E(10)
Sequence 2:	A(10)	-	B(10)	-	C(10)	-	F(10)	-	G(50)

The first sequence has a total target value of 80 and the second has a total value of 90. When recommending based on the target function, sequence 1 is chosen. However, in activity C it is recommended to proceed with activity D, which has a much higher target value than F. The user has to know the whole process well to realise that it is smarter to proceed with D, because of the overall score of the target value.
For these situations where the recommendation is different from the user's opinion it is wise to provide an explanation of the recommendation.
If the total target values of two sequences are approximately equal, then it is wise to analyse an extra target function. In this way it is possible to improve the results for more TF's. Explanation about a multi-TF heuristic will be given section 3.6.

The example of the two sequences can also be translated into an example for attributes, instead of activities. This is because the set of all decisions causes he target value (result) of a case, regardless of the type of decision.

A TV for a case is build up from sub TV's. For example if the TF is minimize costs, the TV will be a calculation of all the registered costs during the processing of the case. The setting of the different costs should be defined and maintained by process managers. The setting of the sub TV's is very important, because all sub TV's together form the result for the TF. The result of a current case will become the feedback over the currently chosen alternatives for future recommendations.
So, based on the properties or identity of a case the costs will be determined. The size of the costs should be predefined by rules, which are defined by process managers. If the time calculation can be determined easily there will be no problem. However, determining the costs for an activity for a case with certain properties might be impossible.

3.5 The target function

The goal of the recommendation service is to support the users of a business process in their decisions [17], but which decision is good and which is wrong? To provide direction to the decision a goal should be aimed for. This goal will be the target function (TF). By selecting the alternative with the most optimal target value (TV) for the selected target function the success of the case will be influenced [3].

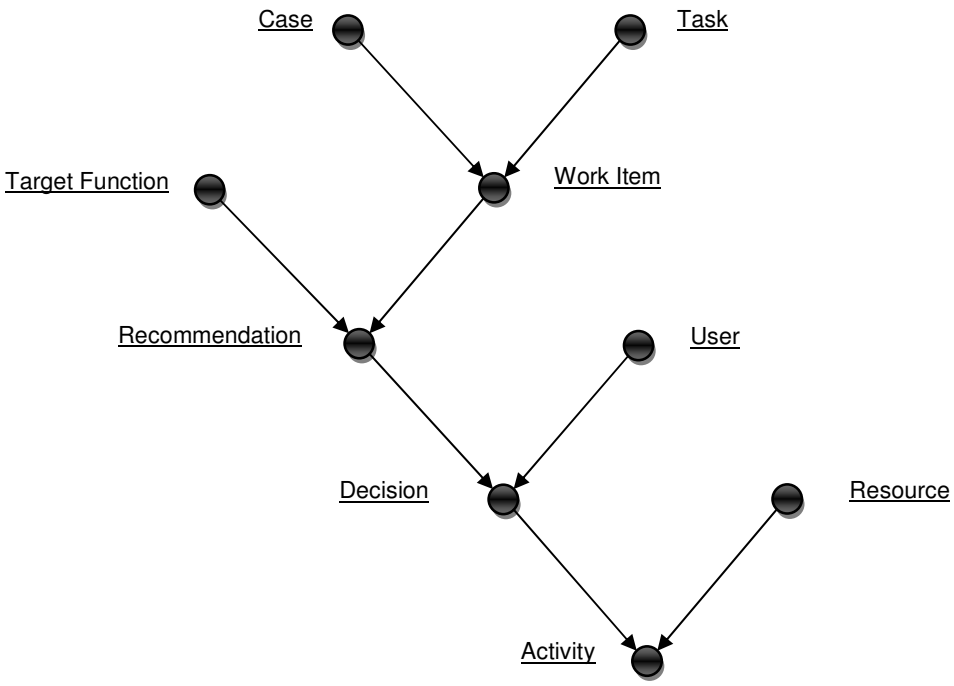

Figure 5: Relationships between concepts

The relationship between a TF and the business process is given Figure 5. The representation is based on [18]. In business processes with or without a recommendation service the Case and the Task together always form a Work Item. A Work Item in combination with a Target Function results in a Recommendation and a not yet enabled Work Item. A Recommendation provided to a User of the recommendation service results in a Decision. The Decision, the Work Item and a Resource are an enabled activity. All Case Activities together cause case results for the Target Functions. In the situation without a recommendation service, the Work Item and the Resource together would be the Activity.

The User is the person who uses the recommendation service and makes the decision. This person could at the same time be the Resource who performs the Activity.
Selecting a TF raises several questions. Who selects the TF? Which TF should be selected? When is the right moment for selecting a TF. How often should a TF be selected: per case, or per process for a certain period?

[3] This conclusion is based on the evaluation in chapter 7.

The preference for a TF for a certain case can be based on the properties of the business process, e.g. the average throughput time of the process is too high, so the throughput time TF will be selected. These properties of the business process could be indicated by key performance indicators (KPI). KPI's are metrics that measure critical success factors of an organization. So, a KPI must be quantifiable and all KPI's together should reflect the organizations goals. Whether the values of the KPI's of a process are good enough should be evaluated against the organization's strategy. This can, for example, be formulated in a balanced score card (BSC). The idea of a BSC is to give managers a quick and complete overview of the business [19]. By translating management information into values for KPI's and providing this information in a dashboard, or management information system (MIS), it is possible to steer an organization.
So, a generic business strategy could influence a decision for a specific case. To enable this, there are three requirements: (1) there should be a business strategy that defines desirable KPI's, (2) KPI's should be monitored, and (3) KPI's should be translated into TF's.

Selecting TF's as an instrument for strategic management is just one way of choosing the TF. By determining the TF's on beforehand, the choice for the TF's is already determined before the start of the case. If the first of the three requirements is not met i.e. there is no strategy, then monitoring KPI's and deciding which TF to select, leads to local optimization of the process (performance management) [20].

Who selects the TF and at what moment, thus depends on whether a strategy is implemented. If there is no strategy, the moment of choosing the target function can be done at several moments during case processing. The later the TF is set, the less influence the TF has on the case result, because fewer decisions will be influenced. However, if no specific strategy is complied to, one should select the TF which returns the greatest benefit. Detecting which TF returns the greatest benefit for the current case, can be done by analysing similar cases. This is the same idea as the selection of the best alternative for a recommendation[4]. It also of influence that more information about the properties of the current case, result in a better similarity subset. So there is a trade-off to be made: postponing the setting of the TF to stimulate better information about the case properties, or selecting the TF as soon as possible, so the TF can influence the case results during more decisions.

3.6 Deriving a recommendation

Deriving a recommendation consists of deriving TV's for the alternatives and choosing the best one. Deriving a recommendation is possible when at the least a TF is selected and there are two or more alternatives. Examples of TF's are: minimize the costs of a case, or minimize the duration of a case. An alternative is a mutual exclusive possibility. For example, alternatives for performance of an activity can be: by a Specialist, by a Manager, by a Consultant or by a Regular Employee. All alternatives ranked by the TF's via TV's form the option list. A schematic representation of the option list can be seen in Figure 6.

[4] Thus, it is possible to develop a recommendation service for suggesting the most profitable TF for a case.

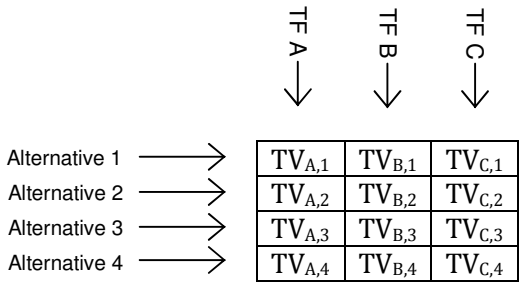

Figure 6: Representation of an option list

A TV is the value of a parameter belonging to a specific alternative and a specific TF. A TV indicates the preference for an alternative, experienced by a similar case. The most optimal TV will become the recommendation [21].
However, the indicator of the TV is not yet defined. When the TF is "minimize the costs of a case", then an example of an indicator can be "the average costs per case". This example uses the same parameter as is mentioned in the TF, namely "costs". Another example of an indicator is "the number of times this alternative was chosen for this TF". In this example for the same TF, another parameter is chosen than "costs", namely "number of occurrences".

There are three dimensions that identify a TV. These are the TF, the Alternative and the Indicator and together they define the recommendation cube. Two of the three dimensions can be set beforehand, because the TF and the Indicator are selected at the start of a case. So, for TF X and Indicator Y there will be a set of TV's as large as the number of alternatives. The alternative that belongs to the most optimal TV from this set will be the recommendation. It is also possible that a recommendation cube is derived for a recommendation where no TF is selected. Then the most optimal value defines which alternative should be recommended, but also which TF should be chosen for this case.

3.7 Opinion versus recommendation

The idea of a recommendation service is to support the user to control complex situations and to provide flexibility for the user to apply her own decision. For the recommendation service a decision is an optimization calculation of a current problem, connected to a target function and a case base filled with solutions from the past.
For the user a decision requires an analysis of the current problem, however the user might consider other factors. The user could be motivated by a reward system, or influenced by context tunnelling, or defines her own norms, or have more trust in certain alternatives than the recommendation service has analysed, or the user (thinks she) is more intelligent.

It is important to have a healthy trade-off between trust and distrust in the service. A user should not fully trust in the recommendation service, because with 100 percent trust there will be no decisions based on the user opinion, which means no new feedback for future decisions. However, there must be some level of trust, because recommendation services have overview in situations which are too complex for the user.
Trust in the recommendation service can be measured and monitored. This can be done by analysing the case base. It could be measured how often the recommendation is the same as the decision and how often not. If all users overrule the suggestions of the recommendation service, there is a low level of trust and action should be taken to gain trust.

More trust can be gained by measuring the results for the cases where recommendations were overruled and where recommendations were followed. Are the results higher for users that followed the recommendations? This information can be used as feedback to the user. If there is too much distrust in the recommendation service, confidence can be gained by providing the user feedback over the earlier made decisions. However, case results are influenced by a lot of different factors, so feedback should be provided with care.

In Figure 8 it can be seen that the user will be influenced by the recommendation. However, the moment of providing a recommendation to the user is important. The recommendation pops-up directly when a decision has to be made, or a recommendation is only showed when it is requested by the user.
Directly providing the recommendation without the request of the user may not be wise. In [22] it is explained that when contradictions in opinions occur, there may be contrary responses, with counter productivity as a result. The user may exaggerate her own opinion and be resistant to the "persuasion" of the recommendation. Also, there will be an increased difficulty on the decision being made, because the user also has to resolve the conflict between the preferences. So, the recommendation should only be showed when it is requested.
As extension to the recommendation the service can provide a logical explanation about the preferred recommendation. In [23] it is said that transparency is an important goal for explanation. When the user interpret the information from the analyses, it will be clear why a certain recommendation is formulated. Explanation can also help by acceptance of justification, relevance, conceptualization and learning.

3.8 The level of information to be displayed

In this section it is described how the recommendations should be formulated and visualised for the users. The previous section was about deriving a preference for the alternatives, with a recommendation containing the most preferred alternative. These preferences have to be translated into a recommendation definition that will be presented to the user. Several different recommendation definitions are presented. Every recommendation has to provide a suggestion about the next decision, although some recommendations may offer more information.

How much information should be displayed to the user? The recommendation service is complete when it reports all available information, but the user should be able to quickly interpret the recommendation. So, there is a trade off in providing information. An overflow of information should be avoided, but not reporting relevant information is undesirable.

Examples of different levels of providing information in a user interface can be seen in Figure 7. Users should experience the different representations, before they choose which formulation is most suitable for them. One way of helping the user to better understand the recommendation message is with adaptive hypertext and hypermedia, like emphasizing, adding or omitting text.

The service suggests to choose alternative *G*.

The service suggests to choose alternative *G*.

A good second option will be alternative *L*.

The service suggests to choose alternative *G*, with a certainty of 90%.

A good second option will be alternative *L*, with a certainty of 89%.

The service suggests to choose alternative *G*.

The service strongly advises not to choose alternative *U*.

The service suggests to choose alternative *G*.

- The history of this case is similar to 93 of the 11423 cases in the case base.
- The top 20% of these cases (with the best results for the target function) were analyzed.
- From these analyzed cases 60 percent choose alternative *G*.

The service suggests to choose alternative *G* for step 6.

If you choose G for step 6, then you should choose option B for step 7
and option R for step 8

Figure 7: Information in the recommendation definition

4 Requirements

The requirement numbers match with the issue numbers in chapter 3 and the design decision numbers in chapter 4.

4.1 About the underlying technique (CBR)

1. The RS should use cases from the past as a resource for feedback for current cases. Retaining cases in the case base should be used as support for decision-generating for future cases.
2. The recommendation service in combination with the CBR technique should not only use knowledge sources like facts and rules like the cases in the case base are. Besides the use of the case base, other knowledge sources should be used like the user's knowledge, the similarity measures and the target function as knowledge sources.
3. The RS should not lack on performance, which can happen when there are too many cases in the case base.
4. The RS should be able to analyse and react to patterns in a short period of time. This means that it should be possible for current cases to influence the recommendations.
5. There should be a warm-up period, before the recommendation service will be run. This means that there should be enough cases in the case base before a recommendation can be generated.
6. Only cases that are executed in line with the vision of the management and permitted by rules should be logged. So, cases with illegal decisions in the case base should be filtered out, because future recommendations can be based on these cases. Only authorized users should be able to filter illegal cases.

4.2 About the types of recommendation

1. The possible types of recommendation are described in section 3.2. Although all types are interesting, for this project there should be three types that can be implemented in the recommendation service.
2. The length of the scope of a path recommendation should be no longer then one. This is because the idea of a recommendation service becomes clear with recommendations for a current decision.

4.3 About the information structure

1. The information structure should describe all information necessary for generating the recommendations.

4.4 About measuring similarity

1. A measurement should be defined that is based on the idea of measuring similarity by the identity of the *case attributes*.
2. A measurement should be defined that is based on the idea of measuring similarity by the identity of the *case path*.
3. It should be possible to influence the quality of the recommendation.
4. It should be avoided that the recommendation service provides a decision which results in a local optimum. So the RS should provide decisions that strive for a global optimum.

4.5 About the target function

A case should have a certain goal (TF) to aim for. There are several requirements for defining a TF:

- If there is a organization strategy, every KPI should correspond to a TF. The KPI's should cover all organizational aspects of an organization to avoid a gap in the measurement set [20]. For example, a BSC is built up of financial, customer, internal business and innovation & learning perspectives. KPI's for the financial and internal business will occur most of the time, but measures about how customers see the organization, and what should be improved (innovation) of a product or service are also important aspects [19].
- The TF (and also the KPI) should be unambiguous. A lot of different persons at different levels in the organisation will work with them.
- The TF should be measurable.
- The data required for the TF should be available in the process and should be logged.

4.6 About deriving a recommendation

An option list should be the framework for the derivation of a recommendation. It should contain information about results per alternative, per target function per indicator.

4.7 About Opinion versus recommendation

There should be a good trade-off in the decision between the user opinion and the derived recommendation. It should be defined how the different phases in the decision making are related to each other.

4.8 About the level of information to be displayed

There should be an interface that provides the information from the recommendation. It should be simple to interpret for the user, but it should also contain the all required information to base a decision on.

5 Design Decisions

This chapter handles the design decisions. Most decisions are made for the practical design. The decisions are implemented. However, some decisions are made for the theoretical design. This means they could and should be applied in a future recommendation service, but are not implemented in the prototype, due to time limits of the project.

5.1 Applying the CBR technique for the recommendation service

1. The RS will use the CBR technique. CBR makes the RS a learning system, as will be shown in the recommendation cycle below.
2. This recommendation service will not only use the case base and the case-based reasoning technique as knowledge sources, but also the user's knowledge, the similarity measures and the target function as knowledge sources. When decision support is requested, all these sources can be consulted for a solution.
3. Extracting and aggregating information from the case base will be done by SQL statements. These statements should be defined shortly, with few calculation steps. If this is not possible less cases will be taken into account for the recommendation generation and pre-calculation will be done . This pre-calculation can be done at night for all cases. Taking less cases into account can be done by only using the last X cases from the case base. These two ideas are not implemented.
4. The RS analyses the case base and via this way it will react to patterns, which means that current cases also influence the recommendations. It will be considered how many cases will be reused as feedback resource. This depends on the number of case on which the recommendation is based on. If a set of cases of the last week provides a better recommendation than a recommendation based on the cases of last month, then it is clear which period should be chosen.
   Reactance to patterns can also be realised by only looking at really similar cases. So, by influencing the similarity of the cases the reactance to patterns can be determined. This comes from the assumption that if a case is 100 percent similar the similar cases subset only has to have one case.
5. There should be a warm-up period, before the recommendation service will be run. So, cases in the business process should be processed without decision support from the recommendation service until the moment that the case base contains enough cases. Enough cases have to be executed and logged to provide a solid base for the generation of recommendations. This will be done by real life execution of the process and by simulating the process.
6. At the end of a decision there is an option to provide information whether the decision should be logged or not. Users should not be able to delete or add information, because they do not have the overview to base this decision on. However, what users might experience as an exception might be a frequently occurring situation or problem in the business process. Case base administrators judge about these cases and make a final verdict whether to log the case or not.

All these design decisions are used to define the process of generating recommendations.

The recommendation cycle

The process of generating recommendations is described by Figure 8. In short this figure will be described. Cases flow through a process. For each case users make decisions for problems. Each completed case has certain results for predefined parameters. All these outcomes per case are logged in a case base (log).
If there are enough cases in the case base, making a request for a recommendation will be enabled. The request can be made by the user or can automatically be triggered by the process. If a request occurs, the recommendation service retrieves the identity of the current case. All available completed cases in the case base are compared to this identity. The cases that are similar are retrieved in a similar-cases subset. This subset of finished cases will be analysed.
Each case contains results for certain parameters. By comparing the parameter results per alternative, it will be determined how good each alternative is. This aggregated information per alternative and per indicator is recorded in an option list (or cube).
Ranking the alternatives in the option list is possible if there is a goal for the case to comply with. This is called the target function. A target function defines which parameter has to be optimal for the current case. The alternatives of the option list will then be ranked. The alternative with the most optimal target function parameter is defined as the recommendation.
The recommendation will be provided to the user. If it is necessary, the aggregated information of the option list will be provided too. Finally, the user decides and a trade-off has to be made between the recommendation and her own opinion.

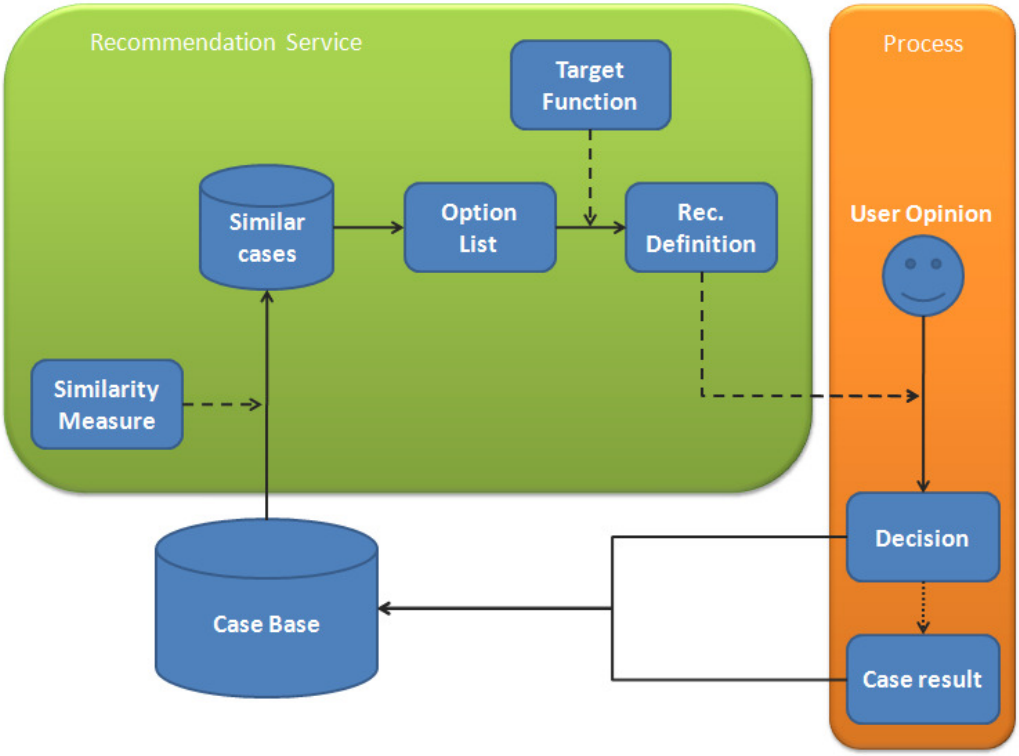

Figure 8: The recommendation cycle

5.2 Types of Recommendation

1. The possible types of recommendation are described in section 3.2. There are three types of recommendations that will be defined and implemented:
   - 1A: the attribute-based recommendation for attribute value decisions
   - 1C: the attribute-based recommendation for decisions about the resource to assign for a specific work item
   - 2B: the path-based recommendation for activity sequence decisions; the path to follow

Attribute based

This type of recommendation is based on the case content of the current case. The case content consists of key data elements (attributes) that together define the identity of the case. For example, a key data element can be the company that sends an invoice, where the company is a value of a data element in the case (invoice). Other examples are the composition of the demand of an order, or the zip code of a delivery, or an indication of familiarity that a customer has with an ordered product. When finished cases are compared to the current case and they share the same values for the key data elements, then they are considered as similar cases.

An example for 1A is that an employee has to choose whether or not to include a covering letter with the product that a customer bought online. The employee has two alternatives: include or not include the letter. The similar subset of cases is based on the key data elements. In this case the key data element can be the familiarity index of the customer. This information was obtained, when the customer bought the product on the web site. A subset with cases with similar familiarity indexes is generated. From this subset the required information about the alternatives can be aggregated. An option list will be generated and a recommendation will be defined and provided. The user will make a decision for one of the alternatives. This decision has an impact on the result parameters of the current case. The decisions will be made, the current case will be finished and the results will be logged in the case base. In this way the decision of the employee will influence future recommendations, irrespective of whether it has a positive or negative result.

An example for 1C is that similar cases will be found, based on the key attributes. However, now a decision has to be made which role(type) will be linked for executing the activity. For example, an invoice has the key-attribute “total amount to be paid”. Cases with the same value for this key-attribute will be selected as similar and the different role(type)s will be the alternatives to be analysed. The role with the best result for a certain TF will be recommended.

Path based

This type of recommendation is based on the past partial path of the current case. The past partial path consists of the sequence of tasks that together define the identity of the case. An important setting is the horizon. The horizon is the length of the path sequence of the current case. The longer the horizon, the greater the similarity should be of cases in the case base.
For example, the current case has a sequence of tasks like: A – B – D – C – E – F and a horizon with a length of 4. Then cases in the case base will be selected when they contain the sub sequence D – C – E – F. For this recommendation type the alternatives are not predefined. All tasks of the process are possible as the next step. The alternatives on the option list now exist of all the tasks that occurred directly after the sub sequence D – C – E – F.

2. In this report the focus will be on a recommendation that contains a solution for only one decision. So, a recommendation with more than one answer will not be handled. Each recommendation recommends only for one decision.

5.3 Defining the information structure

When a recommendation is requested, the recommendation service requests information from the case base and from the process, i.e. from the current case.

The following *case information has to be extracted from the current case and the process*:

1. The type of target function (TF): the case should be optimized for this goal.
2. The type of recommendation that should be generated (path or attribute based).
3. The identity of the case, this can be:
    1. The past partial path of the current case, which contains the sequence of nodes that are within in the range of the horizon. The horizon is drawn backwards from the current node.
    2. The key case attributes that together form the identity. These attributes have to be defined before the start of the process and thus should be the same for all cases in the process. Different data elements can be defined as key data elements which can be used as indicator for similarity with other cases. Examples of attributes that can be defined as the case identity are very domain specific, e.g.:
        - The demand of a customer
        - The location where the applicant of a construction permit lives
        - The covered amount of a life insurance, etc.
4. The location and the type of output of the recommendation:
    1. A path recommendation is requested: the current node should be provided.
    2. An attribute recommendation is requested: the location of that attribute should be provided.
    3. A role recommendation is requested: the next work-item should be provided.
5. The alternatives that have to be analysed by the recommendation service. All alternatives can be taken into account for recommending, but it is also possible to predefine a subset of alternatives. When a subset of alternatives is selected there is more control and less flexibility. The choice is limited, because some alternatives are excluded beforehand. Defining a subset also requires domain specific settings, because process knowledge is required for selecting the alternatives.
   However, it is possible not to make a pre-selection of the allowed alternatives, because the alternatives that were not relevant in the past (or even never occurred in the past), will probably not be preferred by the recommendation service. It is however important to show the user which alternatives are possible, also if they have never occurred before.

Design of the *information structure in the case base*:

An example of a case base is shown in Appendix K. However, the recommendation service in this report focuses on retrieving similar cases, assuming that these cases contain similar problems as the current case has. The cases that are stored have column-properties for

1. The case identity
2. The decisions (choice from alternatives) made for the case and on which TF the decision is based
3. The result for the different TF's.

5.4 Measuring Similarity

1. Attribute-based similarity measurement

The attributes of a case can be defined as the case identity for case comparison. Cases can be distinguished well for this attribute-based similarity measurement, because the values of the attributes can be classified into one of several value ranges. These value ranges can be defined as alternatives for the attribute value.
This results in the next question; how precise do value comparisons have to be? To answer this question the scope of the attribute value has to be defined, and then classified. For example, values for a cost attribute can be classified into classes from 0 to 10.000, from 10.000 to 50.000 and from 50.000 to 100.000 Euros. This means the similarity of the budget attribute can be distinguished into three classes.
This is illustrated in the next example. A finished case is identified by {K=8, L=20, M, Q="good", R="hot", S, T} and the current case is identified by {K=4, L=30, Q="excellent", R="hot", U}. Then the attribute similarity will be 2/4, because the current case has similar attributes for R and L (L classes are defined from 0 to 40 and from 41 to 80, etc). The attributes K=4 and K=8 are not similar (K classes are defined from 0 to 5 and from 6 to 10, etc).

2. Path-based similarity measurement

An important setting for the intensity of similarity is the horizon [16]. The horizon is the sequence of activities, from the current activity to X activities backwards. A finite horizon (recent past) is relevant when only the last X activities have to be taken into account.
For example, if the current case can be identified by {A, B, C, D, C, D, C, E} and the horizon is four, then {C, D, C, E} is considered for comparison.
An infinite horizon is relevant when all activities from the beginning of the case are relevant. An infinite horizon means that the beginning until the current activity of the compared case should be similar.
For example, a finished case is identified by {A, A, C, D, E, A, F} and the current case is identified by {X, X, A, A, C, D, E}, then the finished case is similar.
The longer the horizon, the more similar the selected cases have to be, but also fewer similar cases will be found. It is necessary to find a good ratio between horizon length (intensity) and the number of found similar cases (frequency).

Path activities (but also attributes) that cause no extra distinction in the identity of a case, should be filtered out. Keeping these activities/attributes in the definition of the identity causes unnecessary filtering of cases. Useful filtering may be used to remove certain meaningless activities. For example, if the path recommendation activity is a part of the process model, this activity should be filtered. Otherwise it may occur that the path recommendation activity leads to its own activity.

3. Quality of the recommendation

This problem will not be solved, but a factor (Q) can be defined. The higher the factor, the better the quality. For example, the four alternatives from the example had each on average 5 similar cases. However, it was possible that alternative 4 had only 1 similar case. This was seen by the administrator who configured the recommendation before. The non-uniform distribution of similar cases over the alternatives stays, but the uncertainty that an alternative has too few similar cases can be influenced by demanding a larger similar cases subset.
To realise this, the total subset with the minimum required number of similar cases (defined as minSC) should be at least a factor Q larger than it is now. The minimum required number of similar cases (minSC) can be defined by the number of alternatives for this recommendation (A_rec),

multiplied with a factor (Q) that reassures that the alternatives are on average based on Q values. This can be described as:

$$\min SC_i >= A\_rec_i * Q_i \quad (1)$$

, where $minSC_i$ is the minimum required number of similar cases for a unique recommendation i . A_rec_i is the number of alternatives for recommendation i, and Q_i is the average number of cases per alternative for recommendation i. The higher the Q_i the better the recommendation will be, because the TV of alternative i will be based on more similar cases.

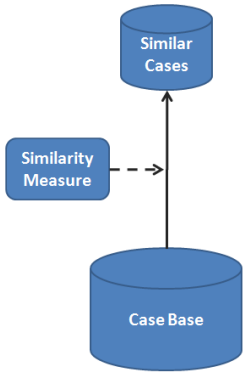

Figure 9: From CB to SC (subfigure of figure 2)

The part of the recommendation service (Figure 9) that retrieves the similar cases (SC_i) from the case base (CB) via a similarity measure (SM_i) can be described as:

$$CB * SM_i = SC_i \quad (2)$$

, where CB is the number of cases in the case base and SM_i is a function that results in a number from 0 to 1 that represents the percentage of the cases in the case base that are similar to the current case. SM_i depends on the type of recommendation i and the identity of the current case. SC_i is number of similar cases for the current case in recommendation i.

If $minSC_i$ is smaller than SC_i, then the quality of the recommendation will be too low. This could be a reason to not provide a recommendation to the user and let the user make his own decision. If SC_i is smaller than $minSC_i$ then it will be allowed to provide a recommendation. Whether the level of quality is defined high enough, can be analysed. This is possible if two sets of cases are compared, where each set had another quality level.

The idea was to link and translate the minimum required set of similar cases into the minimum required set of cases in the case base. However, to do this very detailed information is required. It should be known what the distribution is of the different case identities in the case base. For example, a case with identity X occurs in three percent of the cases in the case base and a case with identity Y occurs in 9 percent of the cases in the case base. Another issue that makes this impossible is that the acyclic behaviour of processes causes an enormous explosion in possible case identities.

However, if it is possible to calculate the minimum required set of cases in the case base, then the required size of the case base is known and together with the case-handling speed the warm-up period can be calculated.

4. Calculating target values

Every logged case gives feedback over the total case and not of a part of it, because this could result in a local optimum. So, by looking at the feedback (in the form of the TV) the whole case is considered and not a part of it.

5.5 Target function definitions

The TF's are categorized into four classes: (1) financial-related TF's, (2) duration-related TF's, (3) user for (role) allocation and guidance of the user, and (4) feedback for customer feedback about the organization and customer feedback about the product or service. When these classes are compared to the BSC perspectives, then (1) is the financial aspect, (2) and (3) are the internal business aspects and (4) is the customer and innovation aspect. One can see that the emphasis is on the financial and internal business aspects, where financial is party also an internal business aspect. One should consider that processes are very specific and there could be a lot of different TF's defined by others. Here, some important TF's are listed.

Financial TF's

1. Business value of a case: when trying to sell insurances for example, the user would like to have recommendations that lead to a sale in the end; hence the TF should express this.
   Cases similar to the identity (past partial path or case attributes) of the current case will be evaluated on the profit they made when they were sent to the Expert and when they were sent to the Regular employee. The current case will be sent to the Expert, if the choice for the Expert resulted in a higher profit in the past, then for the Regular employee.
   Here business value is expressed as profit, but this could also be other parameters. For example, profit as a rate of the turnover.
   In theory it can be assumed that an Expert performs better than a regular Employee. In these situations the work of an Expert results in a higher profit than that of an Employee, despite of the higher salary for an Expert. Thus, a lot of the recommendations recommend to choose for the Expert. It could then be useful to provide some extra constraints to reduce the flow of work to the Expert, like: only cases with a profit that is expected to be higher than 10.000 Euros will have the chance to go to the Expert (see Figure 10).

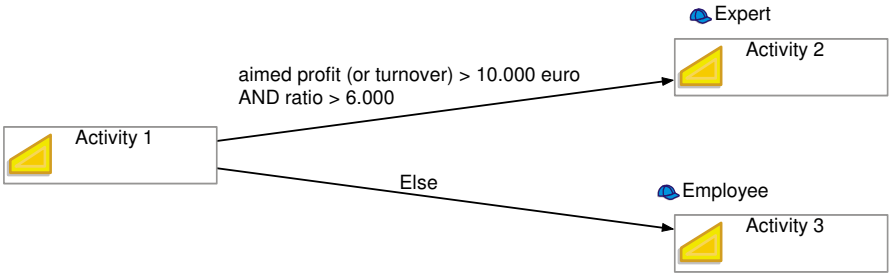

Figure 10: Influencing the business value

Again, usually business rules are applied to control e.g. path decisions. However, the recommendation service will find similar cases for the current case and analyses how frequent the similar cases have been processed by the expert or the employee. It also calculates what the average of all the total case results is. The alternative with the most optimal average is recommended. So there are no preset rules, only an abstraction from the log-file and a consideration about which alternative will be optimal.

2. Business value per time unit [17]: (value for money of a case) if cases that last longer are much more expensive, then it is logical to try to optimize for earning the most money per time unit by ordering the cases in the most optimal sequence. So, this is a recommendation to the user about which case to process first.

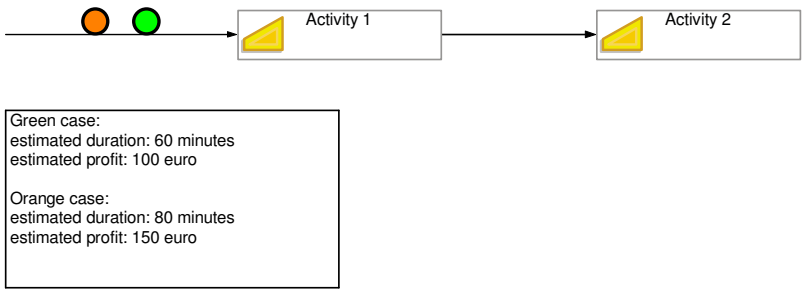

Figure 11: Recommend the order of cases

This TF contains the results of what could be two TF's, namely duration and profit. It transforms these results in the parameter profit per hour. For example, in Figure 11, the green case has 100 Euros profit per hour and the orange case has 112,5 Euros profit per hour. Also for this TF it is possible that cases with a better ratio will be continuously handled first and that the less best cases will be stuck in the process. So there should be rules for handling the less best cases within a certain period or in another process.

3. Costs per case: A very basic and widely understood TF is to recommend to assign a specific role to a case, or assign a specific path or activity to the case path, or to recommend something else, based on the costs.
Why optimize only for "Costs per case" and not also for "Costs per activity"? The answer is that the "Costs per activity" are indirectly taken into account in the cost per case analysis (read also the sequence example in 4.4).

Duration TF's

4. Throughput time: When the goal is to finish cases as soon as possible, the TF for the recommendation service should be to minimize the duration of a case. For example, setting this TF can be based on the information that the process, which is focussed on, is the bottleneck for other processes, or the lead times are longer than required by a service level agreement. When minimizing the duration of the total path the recommendation service needs to know what the throughput, waiting and processing times (TH = WT + PT) are per activity. A requirement for TF's is that the required information is available. In FLOWer only the throughput times are available. So, for the process in FLOWer only the total throughput time of a case is registered.

5. Processing time: Minimising duration when looking at the total processing time per case. This TF may be better than throughput time per case, because processing time is working time without waiting time and can be linear to e.g. wage per hour. Thus, this TF can be also a financial-related TF.

6. Throughput time per activity: is interesting when the resource (person or machine) of the activity is also used by other processes and is a bottleneck.

Minimizing duration can also be done by logistic heuristics for scheduling problems, e.g. First-In-First-Out, Earliest Due Date, Shortest Processing Time and the time-to-readiness of a trigger. Influencing the choice of path, or the sequence of the cases via these rules can sometimes be done in parallel with the recommendation service, but this will not be considered in this project.

User-related TF's

7. Role allocation: Chance of success of a case or even of an activity can be influenced by allocating other roles to that case/activity. For example TF 1 can influence the business value of a case by recommending another role (expert vs. regular employee). Other examples are:
   - Send unique and rare cases to a specific role (e.g. Expert)
   - Send cases which have to be redone for a certain task, because there was an error or a complaint, to a specific role. If it is possible to identify the risk of failing upon a task, then these cases can be sent directly to an expert.

8. User allocation: When it is important to invest in the process' users or control the relationships between users, it is possible to derive recommendations for assigning users to a work item, like:
   - Learning curve of a person [17]: it is possible to determine which users performed which activities. Therefore a manager can decide to influence the recommendation service, such that each user performs each activity with approximately the same frequency, this way the users are constantly learning and developing their skills.
   - Minimize the number of handovers of work in a case. The idea is that when one user performs more tasks this user "knows" the case and thus has less context tunnelling.
   - Centralization[5]: Work can also be allocated by analysing the centralization of a user. For example, when a user is the "crossing point" for a lot of cases, she has a lot of power. This centralization factor can be influenced by reallocating work items.
   - Closeness: Influencing the degree of how near a user is to all other users in a network. This may prohibit that employees become isolated.

9. User support: This TF has the goal to provide optimal guidance to the user. The assumption is that the process is very complex and the user has to make decisions wherefore he has no experience or context. What should the user do? Earlier described TF's could be followed, but they only provide a specific goal. However, it is also possible to do what most users have been doing in that situation.
   This target function calculates what the most applied alternative is for a certain situation. So, this is a recommendation based on frequency of occurrence of a decision. It is about recommending what the average employee would do.

Customer and innovation related TF's

10. Customer satisfaction: As said before, for a balanced strategy, also the customer perspective has to be monitored. If the customers are not satisfied, it should be possible to influence the process

[5] Like centralization and closeness there can be a lot of different indicators for a social network. For more examples of metrics for social network analysis, see: www.insna.org or en.wikipedia.org/wiki/Social_network.

for this aspect. The customer perspective has to do with how the customers see the organization.
Results for monitoring customer satisfaction can be obtained e.g. by providing feedback forms with the delivery of the product or service. Depending on the type of feedback, the type of action can be determined. So, customer satisfaction is a relative term and can only be defined when the type of feedback is known. It could be that customer satisfaction is already defined by other TF's, like the throughput time TF that has to do with service level agreements on lead time.

11. Product satisfaction: This TF has the goal to influence the quality and innovation of the product or service. It is much the same as the previous TF, because this TF should also be monitored via feedback and it is not known in advance what kind of action should be taken.

5.6 Deriving a recommendation

In Figure 12 framework is given of a recommendation cube for the recommendation about which role type has to be chosen, for performing the next work item. If the TF is "maximize profit" and the indicator is "average profit" then the four TV's for the four alternatives will be selected. The alternative connected to the maximum TV will be defined as the recommendation.

Cube (a) contains the TV's for situations where only one TF and one indicator are selected:

$$TV_{i,j,k} \quad (3)$$

,where the index *i* will be the identity of the alternative, index *j* will be the predefined identity of the indicator and index *k* will be the predefined identity of the TF.

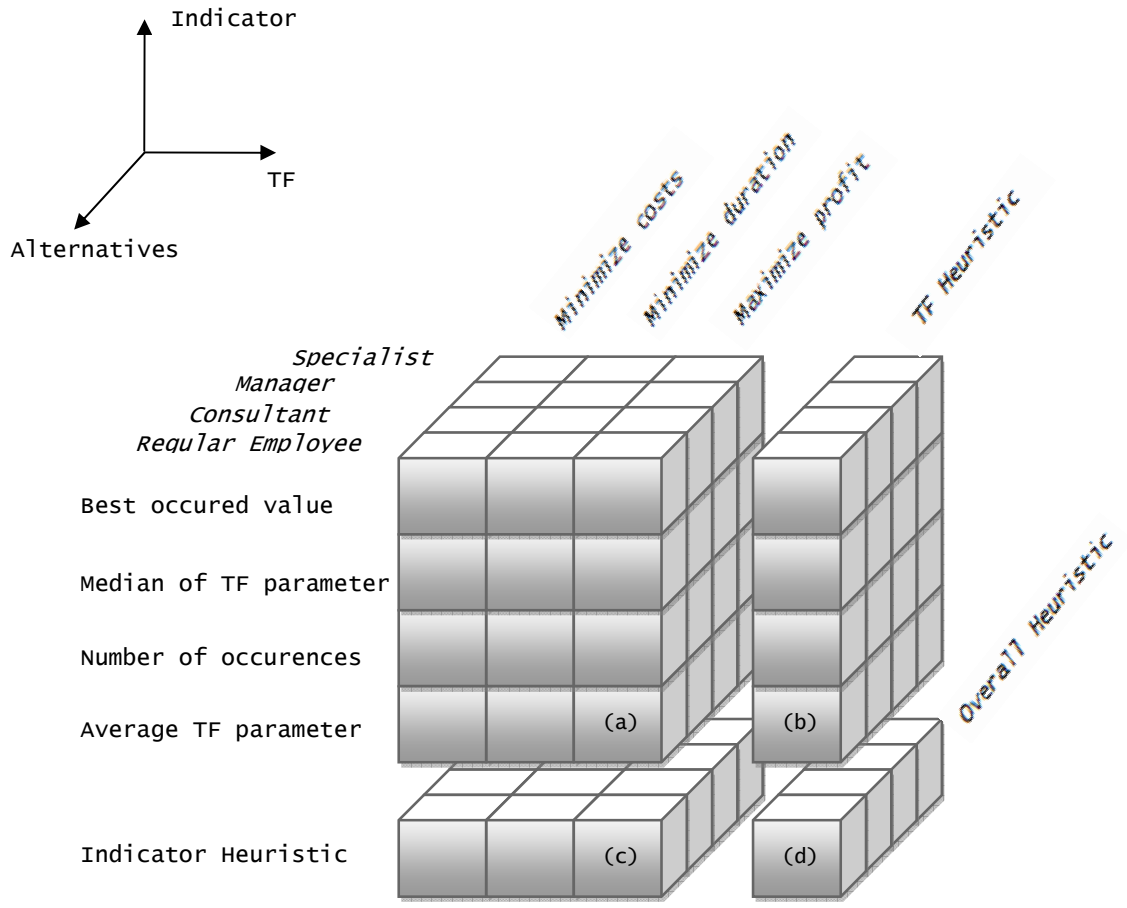

Figure 12: Recommendation cube

Until now only the cube with mark (a) is considered. It is also possible to derive a TV per alternative from all TV's of all TF's for one indicator (cube (b)). This is for example useful when it is interesting to influence the case result for two or more target functions. For example, a heuristic could be that the TV for an alternative is influenced for 70 percent by the first TF "minimize costs" and for 30 percent by the second TF "minimize duration". This TV is mainly determined by the first TF. However, when the TV's for the alternatives of the first TF are almost similar, then the second TF might be the determining factor. An example of comparing and calculating different TV's for different TF's into one TV is shown in Appendix B.
How much influence each TF will have will be defined by ratio's. Setting the ratio's will be driven by trial and error and business process insights. For the TF heuristic only one indicator, but all TF's are selected. A TV for the TF heuristic will look like:

$$TV_{i,j} = \sum_{k=1}^{k=3} r_k * TV_{i,j,k} \qquad (4)$$

,where the index *i* will be the identity of the alternative, index *j* will be the predefined identity of the indicator and index *k* will be the identity of the TF. The *r* will be the ratio for the weight per TF. The sum of the weight ratio's should always be 1.

It is also possible to derive a TV per alternative from all TV's of all indicators and one TF(cube (c)). This is useful because all indicators together form a better TV. Setting the weight ratio's per indicator is a matter of giving higher weight to better indicators. The TF heuristic for deriving a TV per alternative in cube (b) where only one TF, but all indicators are selected, will look like:

$$TV_{i,k} = \sum_{j=1}^{j=4} r_j * TV_{i,j,k} \quad (5)$$

,where the index *i* will be the identity of the alternative, index *j* will be the identity of the indicator and index *k* will be the predefined identity of the TF. The *r* will be the ratio for the weight per indicator. The sum of the weight ratio's should always be 1.
Finally it is also possible to derive a TV per alternative using TV's from all TF's and indicators (cube (d)). The complexity of calculation might become a bottleneck for performance and it should be shown that this heuristic is significant better than the other heuristics. Using all TV's from cube (a) results in the following heuristic:

$$TV_i = \sum_{k=1}^{k=3} r_k * \left(\sum_{j=1}^{j=4} r_j * TV_{i,j,k} \right) \quad (6)$$

,where the index *i* will be the identity of the alternative, index *j* will be the identity of the indicator and index *k* will be the identity of the TF. The *r* will be the weight ratio's. The sum of the weight per type of ratio should always be 1.

5.7 Opinion versus recommendation

In Figure 13 the sequence of steps are defined for decision-making. First the user has to evaluate whether she has a decision or not. If there is consensus, the decision will be taken. Otherwise a recommendation will be requested. If the suggestion of the recommendation is not in line with the user's opinion an explanation could be requested. Finally, if the explanation did not contribute to a consensus about a decision, an expert should be requested to give a second opinion. The degree in which all levels should be used, depend on the importance of the decision.

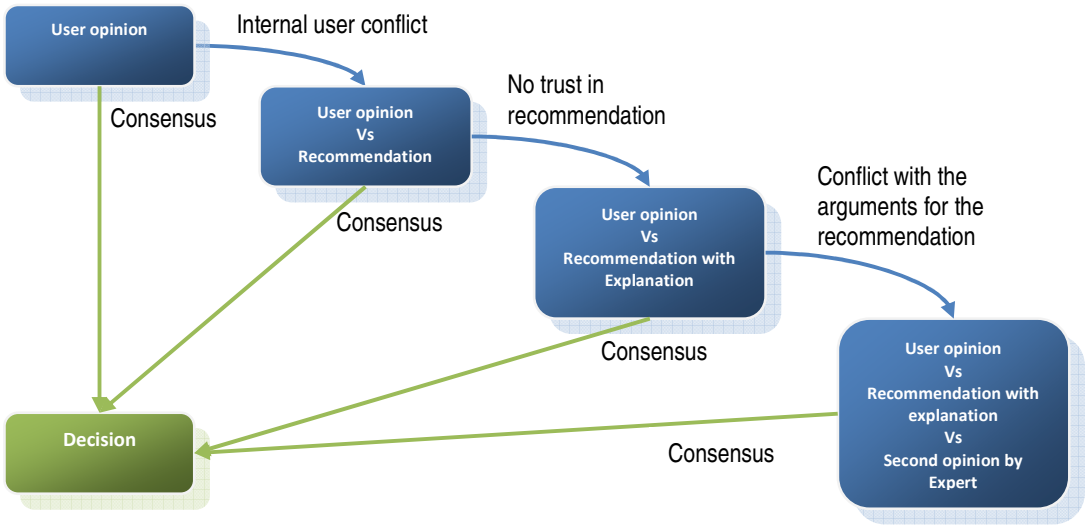

Figure 13: Decision support

5.8 The level of information to be displayed

An example of how information is provided to the user (explanation in the text boxes).

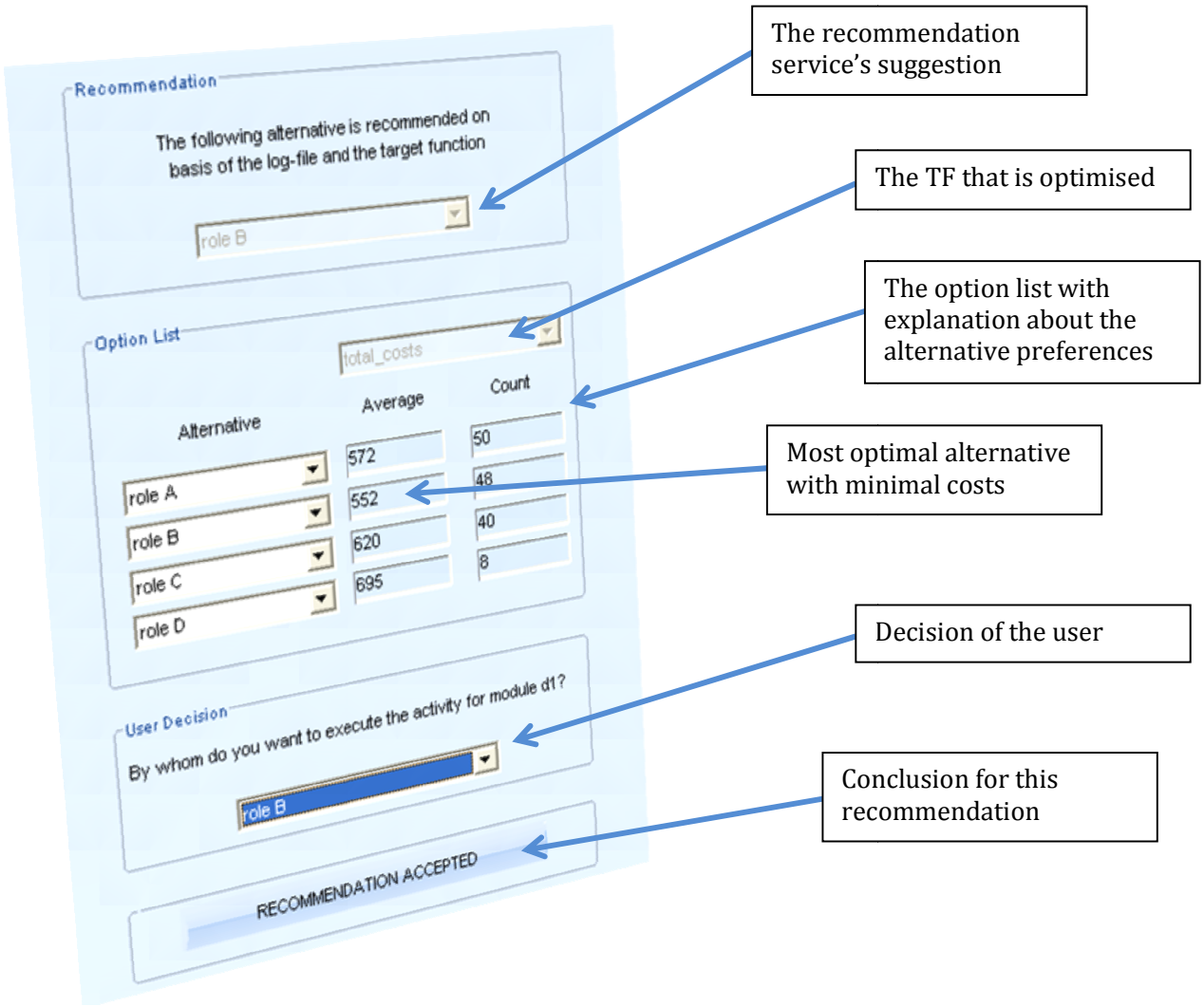

Figure 14: Recommendation interface

6 Developing the Prototype

This chapter describes the development of the prototype of the recommendation service. In Appendix C the work environment is described. In the first paragraph the developed functionality is described. At the end the evaluation and recommendation about a (possible) future implementation of the recommendation service are discussed. The context for developing the prototype, with e.g. information about FLOWer, can be found in Appendix L.

6.1 Development

Before and during the development a number of steps have been defined. These steps contain all required functionality for the developing the prototype and are shortly described in Appendix C.

In FLOWer *the recommendation service is modelled as a process parallel to the business process*. This is very important to realize. So, both processes are built in one FLOWer casetype. This is done because it enhances the easiness of building a prototype. The business process is an example process. It is about the ordering and handling of permits. One case can handle all the required permits (six modules). An activity is mostly represented by setting one data element. The casetype can be divided into four sections. This is also shown in the figure in Appendix E.

A. The configuration phase: this phase is about selecting the right settings from the recommendation service. This part should be managed by the process manager.
B. Start of the business process: the phase where activities, like retrieving customer information and defining the demand, are handled. This part should be performed by the customer.
C. Attribute-based recommendations for helping the user with the processing of the permits (modules). These operational activities should be performed by the employees of the back office of the permits department.
D. Path-based recommendations for the final phase of the business process. These activities should be performed by the employees of the front office of the permits department. They have contact with the customer.

Phase A, C and D will be described from the point of view of the recommendation service. For more context about *phase B* and all other nodes (activities) in these four phases of the casetype, see the description in Appendix G.

Many combinations are possible, only the following will be implemented. In the prototype the path-based recommendations will provide suggestions for paths to follow. The attribute-based recommendations will provide suggestions for assigning roles or persons to work, or how work should be done (e.g. brief versus extensive, or intern versus extern). The last mentioned example can also be seen as a suggestion for a path to follow.

The functionality of the business process and the prototype will be described by four sections, representing the four phases.

A. Configuration phase

This is the part of the pie in Figure 30 that handles the recommendation configuration settings. These settings consist of:

- Selecting the TF. The design decisions about the TF is described in 5.5. The implementation of the target function in the prototype will be described here. The TF should be set at the start of the case. Every recommendation will use this TF. For each case the TV's for each TF will be logged.
- Other settings include: selecting the alternatives, choosing ratios for determining the TV's, quality parameters and the horizon. These settings will be discussed in section C and D.
- All above mentioned settings can be done in the Recommendation Start-up and the Recommendation Setting nodes in the casetype (again see Appendix E).

C. Attribute-based recommendations

These recommendations are based on the similarity of the key attributes. This recommendation is implemented in phase C and applied for modules d1, d2, d3 and for the perform_intern or perform_extern decision. Node specific information can be found in Appendix G. The functionality of the recommendation will be explained below.

In the period previous to the execution of the case, the recommendation will be configured. This will be done in phase A. Per recommendation the location of occurrence should be specified, because this *is a recommendation with a business process-specific configuration setting*. The possible alternatives should be set, the ratios for influencing the TV's per decision should be defined and the key attributes should be selected. *All these settings can be different per recommendation*. The only setting that is common for all recommendations is the level of desired quality. This is embodied by the "minimum number of required cases" setting and the Q factor, as is described in section 5.4.

The following steps will occur real-time when a attribute-based recommendation is requested:
Step 1: Checking the case base. The case base will be checked for the amount of cases. If there are more cases than required the recommendation will be generated, otherwise an empty recommendation will be returned.

Step 2: Retrieving the similar cases. Only cases that have the same key attribute will be selected. These similar cases are the basis for the option list.

Step 3: Calculating the indicators for the alternatives. An indicator is the way in which all TV's will be calculated. In the prototype the average is calculated and the number of times an alternative has been chosen is counted. This is only based on the similar cases.

Step 4: The above calculated TV's are inserted in a temporary table. This table represents the option list for this specific recommendation.

Step 5: Deriving the most optimal alternative from the option list. Determine which alternative should be recommended, e.g. the alternative with the smallest value for the "minimize total costs" TF.

Step 6: Providing the results to the user. The results will be returned and will be shown to the user in the interface. An example of the attribute-based recommendation interface can be seen in Figure 15.

Step 7: Making a choice. The user makes his choice out of the alternatives. The option list can be used as an explanation of the suggestion.

Step 8: Applying the decision. The activity will be executed and the made decisions will be followed.

Step 9: Logging the results. The resulting TV for the activity will be derived. This is done for every TF. If all TV's for the case are available for all TF's, then the results will be logged in the case base. Also the made recommendation, the made decision and the case identity (values for the key attributes) will be logged.

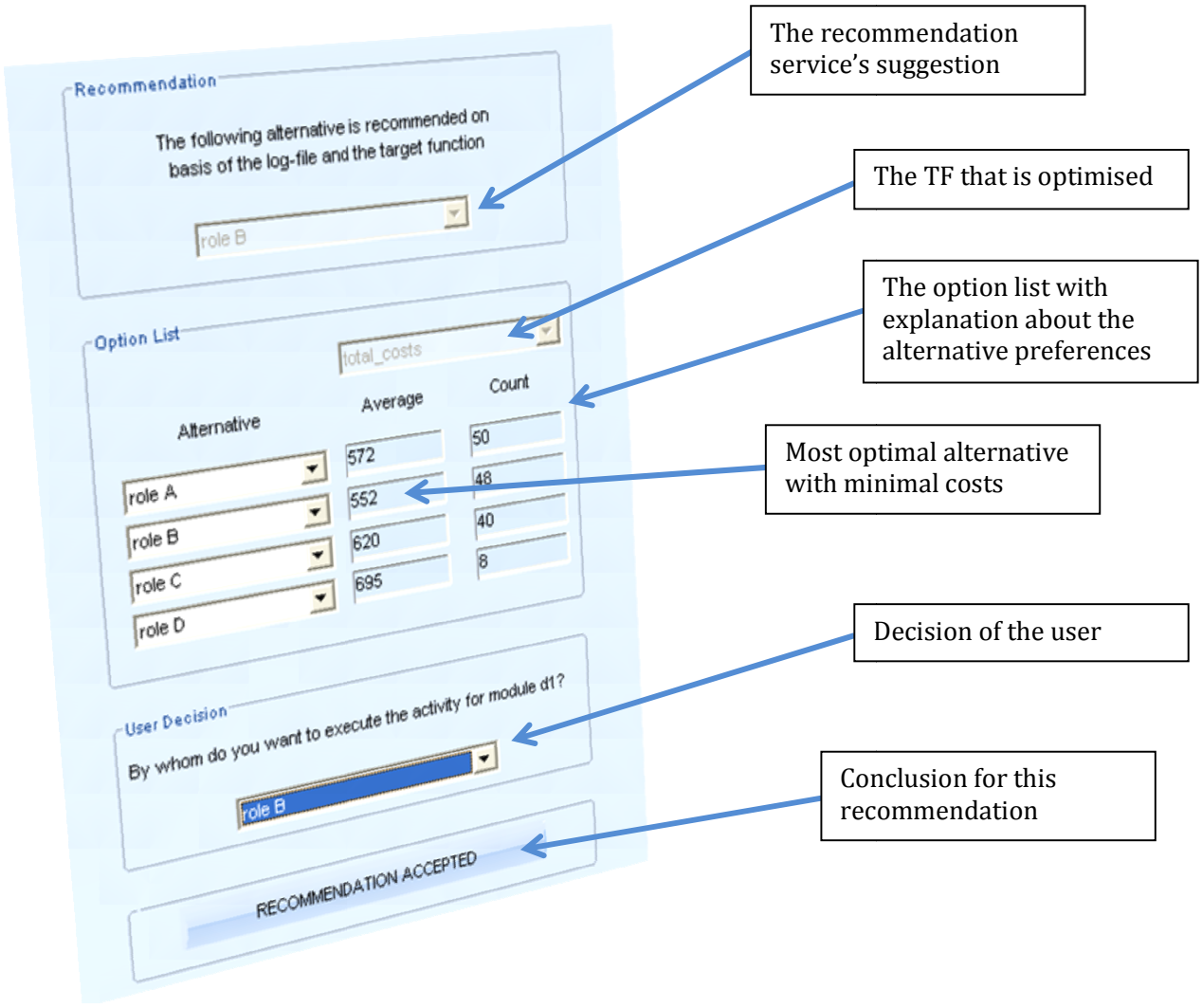

Figure 15: Recommendation interface

D. Path-based recommendations (Final phase)

These recommendations are based on the similarity of the previous partial path of the current case. This recommendation is implemented in phase D and should be applied for all nodes in this phase, except for the last activity in the casetype. Node specific information can be found in Appendix G. The functionality of the recommendation will be explained below.

In the period previous to the execution of the case, the recommendation will be configured. This will be done in phase A. *Path-based recommendations are generic. Thus no domain-specific settings are required.* Still settings have to be selected. These common settings include the level of desired quality. The length of the horizon should also be specified. Again the most important setting is defining the TV derivations. These TV's will become the feedback for future cases and their decisions.
Thus, every path-choice causes its own case specific result. For example, if a case with module 1 is directed to the standard delivery node, it results in a cost of 90 euro. If this case is directed to the quick delivery node it results in a cost of 40 euro. However, if a case without module 1 is directed to a standard delivery, it results in a cost of 20 euro. If this case is directed to quick delivery it results in a cost of 60 euro. Thus each path choice causes a case specific and path specific result.
All possible alternatives should be defined or retrieved. This should also be done in advance of the case execution.

The following steps will occur real-time when a path-based recommendation is requested:
Step 1: Defining the case identity. The current node and the length of the horizon are known. With this information the part partial path can be derived.

Step 2: Checking the case base. The case base will be checked for the amount of cases. If there are more cases than required the recommendation will be generated, otherwise an empty recommendation will be returned.

Step 3: Retrieving the similar cases. Only cases that have the same path identity will be selected. These similar cases are the basis for the option list.

Step 4: Deriving the alternatives. All alternatives that were chosen for this decision at the current node will be selected.

Step 4: Calculating the indicators for the alternatives. An indicator is the way in which all TV's will be calculated. In the prototype the average is calculated and the number of times an alternative has been chosen is counted. This is only based on the similar cases.

Step 5: The above calculated TV's are inserted in a temporary table. This table represents the option list for this specific recommendation.

Step 6: Deriving the most optimal alternative from the option list. Determine which alternative should be recommended, e.g. the alternative with the smallest value for the "minimize total costs" TF.

Step 7: Making a choice. The user makes his choice out of the alternatives. The option list can be used as an explanation of the suggestion.

Step 8: Applying the decision. The case is directed to the right activity in the chosen path.

Step 9: Logging the results. The resulting TV for the activity will be derived. This is done for every TF. If all TV's for the case are available for all TF's, then the results will be logged in the case base. Also the made recommendation, the made decision and the case identity (path) will be logged.

A more technical explanation of the path-based recommendation is provided in Appendix H.

6.2 Improvements for the prototype

Although there is no concrete plan within Pallas Athena for developing a recommendation service, it is possible to implement one. When the recommendation service will be implemented in FLOWer (BPM|One), there will be several issues about adjustments and extensions. Will the recommendation service be an extra functionality in FLOWer or will it be an add-on? How will the data be retrieved? Where will the functionality be coded? How should the recommendation service be modelled in the casetype design? How should it be visualized for the end-user? Where should recommendations be processed? A summary of suggestions follows:

Separate the recommendation service and the business process

The prototype of the recommendation service is build within the same casetype as the business process wherefore it recommends. It will be better to separate the recommendation service from the business process and prevent intertwining as much as possible. If a recommendation service is implemented for a specific business process there will remain enough linkage between the process and the service. This could e.g. be the setting of key attributes for attribute-based recommendations. However, all the functionality like capabilities nodes and interfaces should be on another level than the modelling level of process design. As an example, Figure 17 is provided. Compare the two figures. The top figure is how it is for the prototype and the bottom figure is what a process designer should see. Larger figures are provided in Appendix E and Appendix F.
The differences in the two figures are the components that should be omitted from the process design, this is:

1. The complete content of phase A. This is the process management phase. The setting of the recommendations, the horizon, key attributes and the calculation of the TF parameters (e.g. costs) should be done once for the casetype at the process (re)design. So, not each time a case is created, like in the prototype. This should be implemented in/represented by the "recommendation configuration"-pie part in Figure 30.
   If a strategy is followed the target function is defined at the management phase for all cases. If no strategy is followed, but the most optimal TF is selected, this will be done automatically and not during the process. Only if TF's should be selected manually, there should be a possibility in e.g. the case guide.
2. The recommendation nodes should not be a part of the business process. The path recommendation is a node in the prototype, but it is filtered and not taken into account when path recommendations are generated. It should be possible that recommendations could be requested from the case guide when a user needs support.
3. The recommendation code should also be stored in a separate capability file in FLOWer and not in a capability, which belongs to a casetype of a business process.

Extend the possibility to define the TV calculation

It should be possible to attach more functionality to design objects. For example, it should be possible to insert functions for the calculation of sub TV's per node. This should be implemented like the simulation parameters in PROTOS. However, an overview of all the sub and total TV's at one location improves the overview and easiness of maintenance.

These functions for the calculation of sub TV's per node should also be monitored by a system. The recommendation service retrieves its feedback from these values. So, a monitoring system could help to gain insight into the recommendation service settings and its results.

For example, when TV's are defined in another way, the process might perform differently than before, because its feedback mechanism was redefined. To find out which definition of the TV's has the best result, all definitions and its process results should be logged.

Organize the architecture

The architecture should be organised for better performance, like in Figure 16. The complete recommendation service should be placed on an external server, not on the FLOWer operation server. The case base should have the following categories of case information:

a. Identity of the case: based on the key attributes, or the path sequence
b. Solution: the recommendations made and the decisions taken
c. Outcome: the case results for the different TF's

The case base is only one of the knowledge sources (containers) of the recommendation service. The other sources contain information about:

d. Similarity measures
e. Target Functions
f. Alternatives Selection
g. Quality checks
h. Option list
i. Recommendation definition

The required information from the process model and current case should be extracted from the management information tables, not directly from the casetype. The MI tables are structured and always applicable SQL statements can be defined for these tables, instead of domain specific tables.

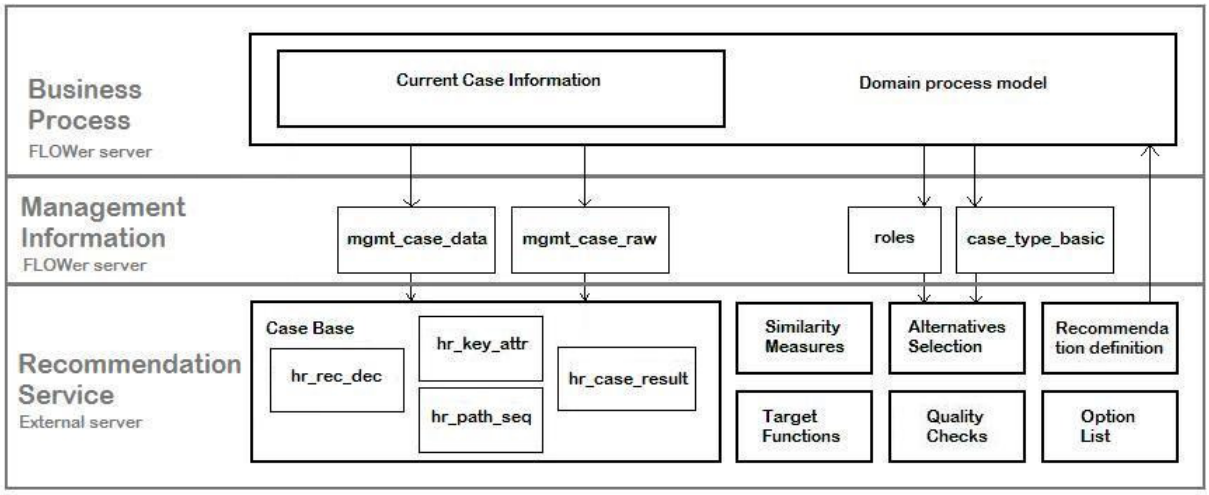

Figure 16: New information architecture

Optimize performance

The prototype uses a MySQL database. This database caused a very bad performance. For example, the table structure should be optimised. In the future it will be better to use e.g. an optimised Oracle database, which has better indexing and keys. Also the capability code should be optimised for a better performance.

The case base is used for retrieving similar cases, or storing new cases. Not every new case should be stored directly in the case base. For example, every night the case base can be maintained and updated.

Dynamic tasks

There is no dynamic task in FLOWer. When recommendations are done about the set of data elements that should be filled, there is no possibility of implementing this in FLOWer. For example, this means that one formfill object shows different data elements in different situations. Another example is that the authorization can be made dynamic, thus for different situations different authorized users.

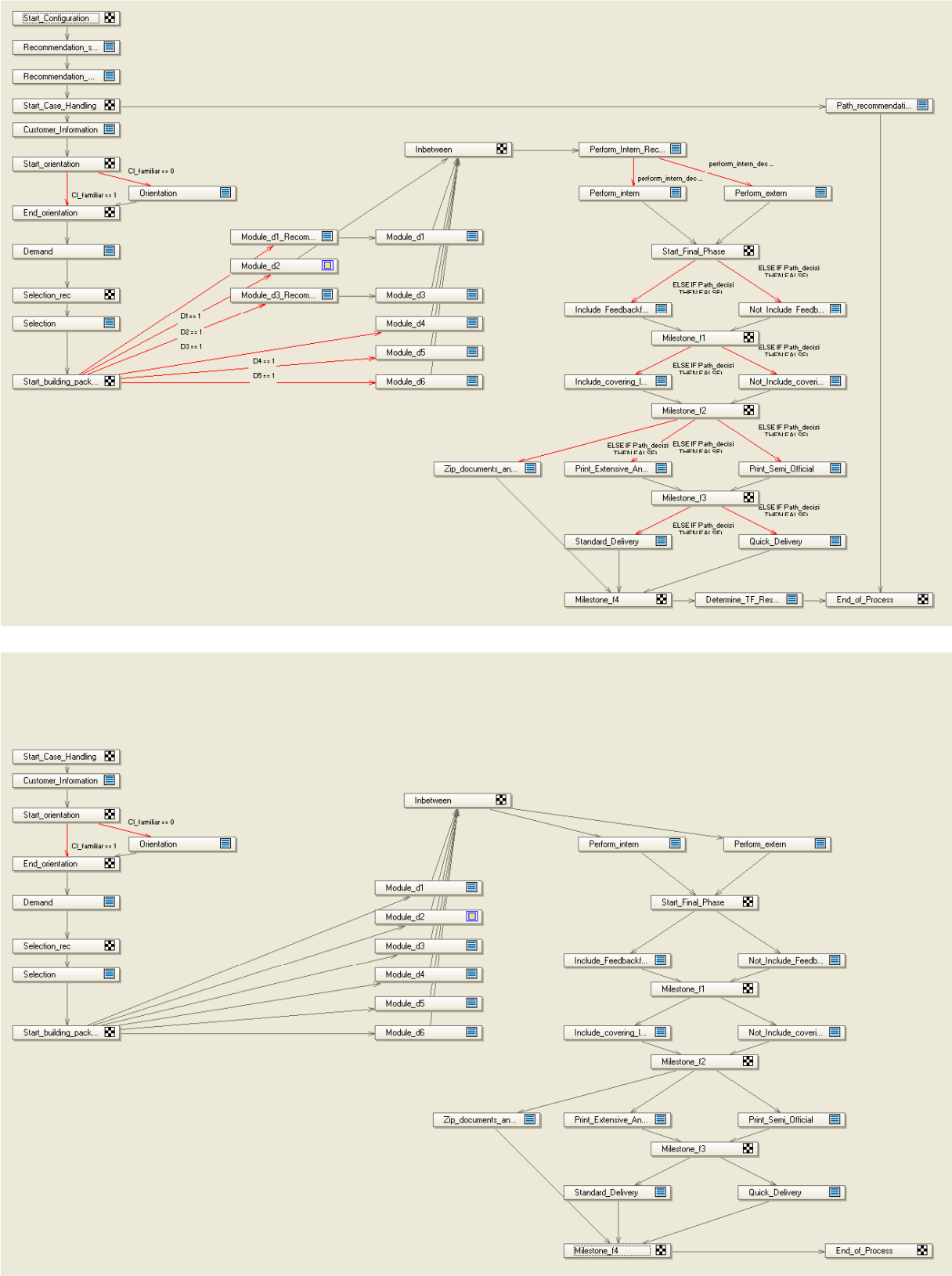

Figure 17: Process model with (top) and without (bottom) the recommendation service

7 Evaluation of the Recommendation Service

The developed prototype is evaluated on simulation output and by real users. The reason for simulation is to provide insight in the recommendation service [26]. The simulation is done by the FLOWer Internal Test Suit (FLITS). The reason for real users is to test what their experience is with the service.

7.1 Set-up and problem definitions

To find whether the recommendation service adds value to a process, two scenarios have to be simulated. The first scenario is random decision-making and not to follow the provided recommendation. The second scenario is decision-making by always following the provided recommendation. The advantage of this plan of simulation is that two extremes will be tested. However, the original idea of the recommendation service is that it provides guidance and suggestions, but the user has the flexibility not to follow this guidance. Always following a recommendation or always following a random decision is not the practice in reality, because a user does not have the chance to overrule faulty recommendations. To simulate the last situation, one has to simulate with real human users, which asks too much effort at this point.
Finding out whether recommendations result in better case results than results from user-decisions do. Random decisions versus recommendations are simulated. A random decision is not a user-decision. User decisions have the advantage of having the potential of being smarter than recommendations. When a user overrules a faulty recommendation, she rectifies the recommendation service and prevents that the recommendation service bases future recommendations on a log with misleading data.
However, simulating random decisions is relatively good, because it provides an average performance which can be compared, and no one can influence the result positively or negatively.

The problem definitions:

1. The main question is: What is the average case result for a target function when always choosing for the recommendation of for the random decision?
   H0: Case result for recommendation > Case result for user's decision
   H1: Case result for recommendation <= Case result for user's decision

2. Are attribute-based recommendations better than random decisions?
   H0a: including the attribute-based identity in measuring similarity provides better case results than without.
   H1a: including the attribute-based identity in measuring similarity provides equal or worse case results than without.

3. Does a larger horizon length provide better recommendations?
   H0: horizon=4 provides better results than horizon=6
   H1: horizon=4 provides equal or worse results than horizon=6

4. Does the recommendation service learn over time?
   H0: A recommendation for the 100th case is worse than for the 1000th case
   H1: A recommendation for the 100th case is equal or better than for the 1000th case

5. Will the recommendation service suffer from a cold start problem?
   H0: The number of different alternatives that are recommended is smaller when recommendations are provided after 50 random cases than after 200 random case
   H1: The number of different alternatives that are recommended is equal or larger when recommendations are provided after 50 random cases than after 200 random case

7.2 Simulation with FLITS

The model
The model will be the developed casetype in FLOWer. The recommendation service and the business process are modelled in one casetype.

Parameterization
The customer information in phase B will be randomly filled from the customer information table in the database.
The target function will randomly be selected by FLITS.
For the costs and profit (and selling price) there is a specific calculation model. This calculation included ratio's for potential alternative in the B and C phase of the casetype. These ratios will influence the costs. So, if alternative B is chosen the costs for task M will be influenced by a specific ratio.
For the nodes influenced by the path recommendation, case specific costs calculation rules are implemented. Every path choice causes its own case specific result. For example, if a case with module 1 is directed to the standard delivery node, it results in a cost of 90 euro. If this case is directed to the quick delivery node it results in a cost of 40 euro. However, if a case without module 1 is directed to a standard delivery, it results in a cost of 20 euro. If this case is directed to quick delivery it results in a cost of 60 euro. Thus each path choice causes a case specific and path specific result.
The parameter time cannot be simulated by FLITS. FLITS can only log the simulation process times instead of process times by real users. So, the minimize time TF cannot be evaluated. However, times could be parameterized like it is done with costs. For example, the parameters that now are defined as costs in Euros could also be defined as time in seconds.

FLITS settings
FLITS is a separate VMware image, but could be approached via a web browser. Instructing FLITS is possible by writing a Perl script. This script contains a procedure about which data elements have to be filled. An example can be seen in Appendix I. The logging of the case results could be shown in a case log by FLITS (example in Appendix J). The case results are also logged by FLOWer.

7.3 Evaluation of the recommendation service

When simulation is finished an analysis can be performed on the output data. This data is retrieved from the database tables.

1. Random versus Recommendation
   The idea is to demonstrate the difference in performance between the situation where decisions are taken random and the situation where decisions are taken by the recommendation service.
   A test is run where first 400 cases with random decisions are executed, continued by 200 cases where recommendations are followed when possible. The result is shown in Figure 18. The figure has costs on the y-axis. On the x-axis are 6 periods. Each period contains 100 case results. The first four periods are the cases with random decisions and periods 5 and 6 are the 200 cases where recommendations were followed when possible.

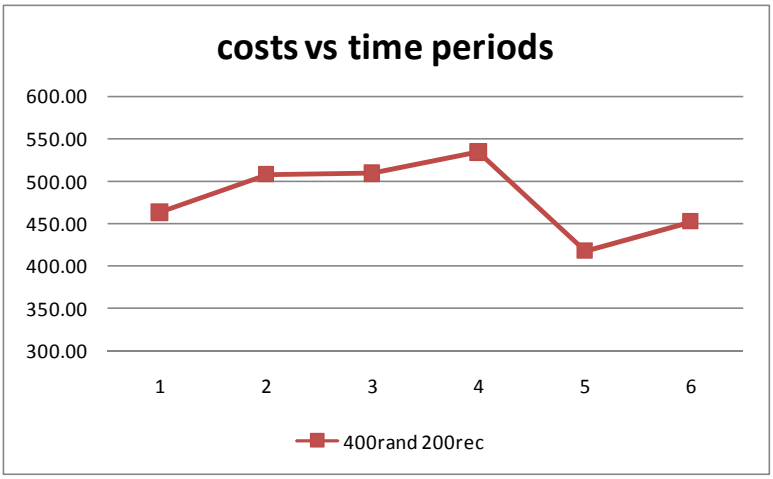

Figure 18: Costs vs time periods

A confidence interval (CI) is calculated for the cases with random decisions. Another CI is calculated for the cases with decisions from recommendations. The results can be seen in Table 1. With a confidence of 95 percent are the random values in the CI interval of (484;522), for recommendation this is (413;455).

Table 1: Confidence Intervals (95%)

Type	Average Costs	St.dev Costs	Confidence	95% CI
Random	503 (over 400 cases)	196	19	(484 ; 522)
Recommendation	434 (over 200 cases)	153	21	(413 ; 455)

Also a heteroscedastic T-test with a two-tailed distribution was used to determine whether the two samples are likely to have come from the same two underlying populations that have the same mean. The result was p=0,00000325, which means the two samples differ significantly. It is clear that the recommendations cause better case results costs than the random decisions do (with a TF to minimize costs).

2. Attribute-based recommendations
   Are attribute-based recommendations better than random decisions? It will be shown that including the attribute-based identity in measuring similarity provides better case results than without.
   In Figure 19 on the y-axis is the number of cases and on the x-axis are two sets. Set 1 contains the first 100 cases with random decisions. Set 2 contains the last 100 cases with recommendations decisions from the sample of 600 cases from the previous section (Random vs Recommendation). So, the first 100 and last 100 (numbers 501-600) of cases are retrieved from the 200 random + 400 recommendation set.
   One specific decision was analysed. This is the role recommendation for module d1. As can be seen in the figure; both sets are about 50 cases large. This is logical, because module d1 was demanded half of the time.
   The results for the first set: 7 times role A, 15 times role B, 14 times role C, 15 times role D. This approaches the idea of random.
   The results for the second set: 50 times role A, 2 times role B. So, for the recommendation cases the choice was in 96 percent of the cases made for role A. This matches with the ratio's of the for the four employee roles. These ratios were set in such a way that Role A was the cheapest role that could be chosen to perform module d1. The result is that attribute-based recommendations are better than random decisions.

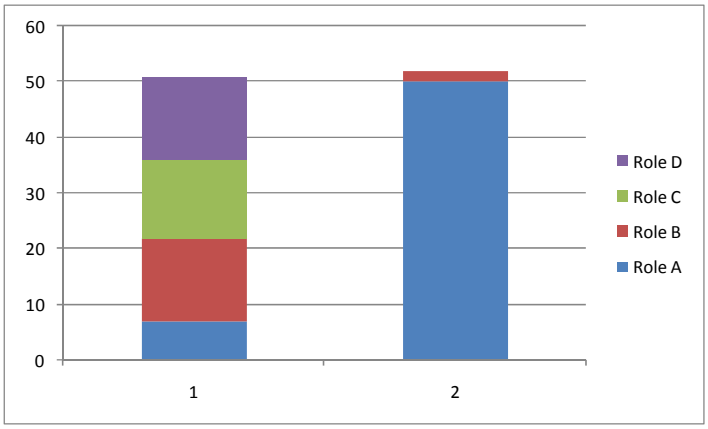

Figure 19: Choice of roles - random vs recommendations

3. Intensity of the similarity measure
   The intensity is for path recommendations the length of the horizon. The issue with the horizon is that everything outside the horizon will not influence the recommendation. However, in the business process for the prototype nodes at the beginning of the process are important for the last nodes in the process. However, to make e.g. the orientation node part of the path sequence for the delivery nodes at the end of the process, the horizon has to have a length of 14 nodes. The variety in the path composition over 14 nodes is too large for a small case base. There will be no similar cases, because the case base should be enormous (50.000+ cases).

4. Learning Recommendation Service
   Detection of a learning service is proven by the example for the first issue of this section (Figure 18). Another example can be seen in Figure 20. It is an example of a learning system, where learning is indicated by the increasing profit. This test was for the first twenty periods random (set 1) and for the last four periods the recommendations were followed (set 2). One period contains the execution of 50 cases. Because there is a lot of variety possible, the averages may differ a little. However, compare the dotted lines that represent the averages of the two sets.
   Again a heteroscedastic T-test with a two-tailed distribution was used to determine whether the two samples are likely to have come from the same two underlying populations that have the same mean. The result was p=0,000474 , which means the two samples differ significantly. It is clear that the recommendations cause better case results for the profit than the random decisions do (with a TF to maximize profit). However, this was only possible with a filled case base (the first 1000 cases). Whether the system keeps on learning over time cannot be concluded.

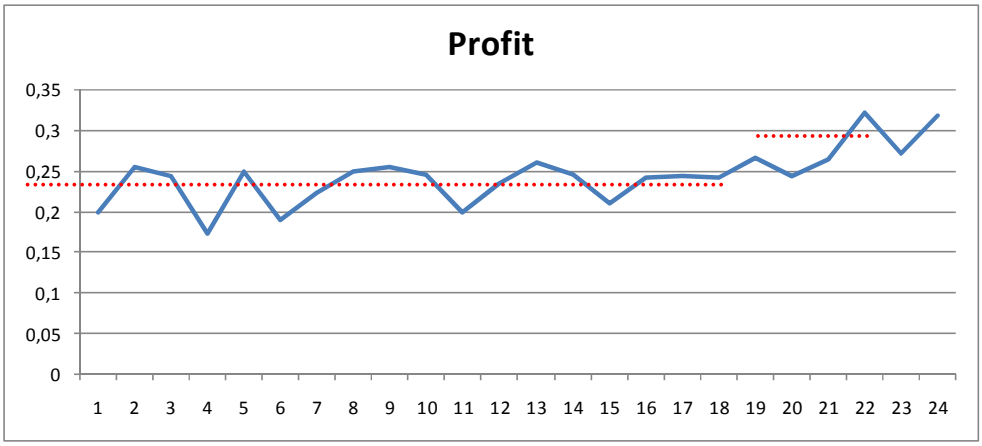

Figure 20: Learning system

5. The Cold start problem
   The cold start problem is described in chapter 3.1. To evaluate whether the recommendation service is influenced by this problem, a test was run with only two random cases in the case base. During the test 397 cases were executed and decisions were taken with recommendations where possible. In Figure 21 the recommended alternatives are shown for the recommendation of module d1. Role A had the most lucrative ratio. However, role A has never been recommended, as is role C. After more thorough analysis it was concluded that role D was always recommended for TF 2 (minimize time) and role B was always recommended for TF 1 (minimize costs). Thus, it is necessary to have enough variability in the case base before recommendations may be generated.

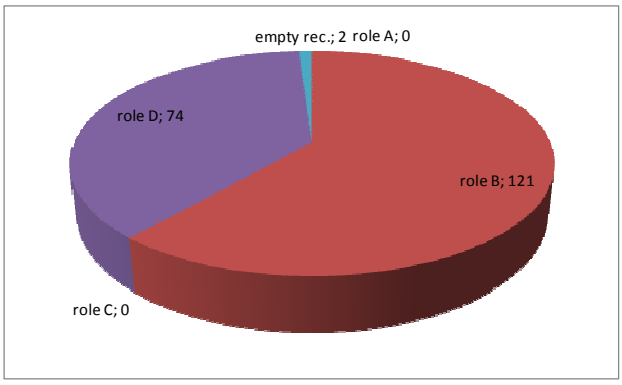

Figure 21: Cold start problem

Evaluation by persons

The evaluation by two real persons was more of a qualitative nature. The users pointed towards the following issues:

The recommendation interface is important for interpreting the concept of recommendations. It is very useful to use for each recommendation the same interface. The prototype uses two different interfaces for the two types of recommendations. This was partly for explaining at the front end of the prototype how the service works.
The path recommendation can in the prototype be requested by redoing the path recommendation node. This was not clear. In a real life service the recommendation should be requested in the case guide.
One of the user mentioned that the option list decreased the level of flexibility. For the attribute-based recommendations the option list is provided to the user as a form of explanation. However, if there are very clear facts in the option list, the user experiences that his opinion is directly weakened and very much influenced by the option list.
In the prototype the user has to enter his path decision into the path recommendation interface. After this, the user will be automatically directed to the right node. This was specially designed for the FLITS simulation, because FLITS needs to know what next data element (nodes0 have to be filled). However, for users it is the easiest if they only have to click once to request a path recommendation. From that event the current node can be calculated, the recommendation can be calculated and provided and the user does not have to insert his decision. This is because the next node that will be opened is his decision.
The other remarks were about interpretation of the business process and questions about the background functionality.

The conclusion by both users was that the thorough analysis of past cases and there results is very welcome information in the decision-making process.

8 Conclusions

Although an evaluation is already provided in 7.3 and improvements for the prototype are provided in 6.2, a short summary of the most important conclusions will be given here.

8.1 About the recommendation service

The concept of recommendations based on the information of already finished cases has proved to add value to a business process. In complex situations the recommendations provide better recommendations than users do. The recommendation service works best when there is a large set of similar cases. This requires a large case base; too few cases in the case base could be a bottleneck. There should be a good ratio between the similarity intensity and the size of the case base. The recommendation service proves to work well in situations where there is yet less variability in the process than the number of cases in the case base can cover.

How cases should be logged in the case base can on a high level be described as: identity – solution – result. The case base is only one knowledge source. The TF definitions, the definition of the similarity measures, the functionality for deriving option lists and choosing the best alternative are all knowledge sources. These are all shown in Figure 16. Only the most intelligent knowledge source misses in this figure, namely the user.

Another defined knowledge source of the recommendation service is the type of recommendation. A recommendation is based on the identity of the case. This identity can be the case path or the values of the key attributes of the casetype. The most important distinction between the two is that the attribute-based recommendation is domain specific and the path-based identity is generic. The output of the recommendation service can be a path recommendation, an attribute value recommendation or a role recommendation for a work item.

A recommendation service cannot aim for a goal without the target function. The setting of the target function can be done by (1) following an organization-wide strategy, (2) following performance optimization goals, or (3) by selecting the TF that adds the most value to the organization (or client).

Determining the values for a current case, e.g. the costs, requires business insight. The decisions will be judged on these values. So, the derivation of the result parameters is very important.

8.2 Future Work

Setting the recommendation service is a difficult job. Therefore one can say the recommendation service should have its own recommendation service. All best and worst practises should be retained in a recommendation service case base. Thus, by monitoring the recommendation service the success of the recommendation system and the level of correctly overruled recommendations can be analysed. The recommendation service can then be criticized on its results. However, this is out of the scope of this project.

A deeper understanding of the types of recommendation could be useful. For example, how can hybrid versions of recommendation types be developed? This means merging the path-based and the attribute-based similarity. This kind of similarity measures are not the only ones. Nearest neighbour, vector space measuring, etc can they also be applied? It is interesting to compare these different techniques.

9 References

1. http://www.pallas-athena.com

2. T. H. Davenport, J. G. Harris. 2005. Automated Decision Making Comes of Age. MIT Sloan Management Review 46, no. 4 (July 1), pp 83.

3. H. C. W. Lau, A. Ning, K. F. Pun, K. S. Chin, W. H. Ip. 2005. A knowledge-based system to support procurement decision. Journal of Knowledge Management 9, no. 1 (January 1), pp 87-100.

4. D. Arnott, G. Pervan. 2005. A critical analysis of decision support systems research. Journal of Information Technology 20, no. 2 (June 1), pp 67-87.

5. I.T.P. Vanderfeesten, H.A. Reijers, and W.M.P. van der Aalst. 2008. Product Based Workflow Support: A Recommendation Service for Dynamic Workflow Execution. In Z. Bellahsene and M. Léonard, editors, Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), volume 5074 of Lecture Notes in Computer Science, pp 571-574. Springer-Verlag, Berlin.

6. http://www.processmining.org

7. S. Chowdhury. 2007. Trends in Database Tools and Technologies. The Business Review, Cambridge. Hollywood:Summer. Vol. 7, Iss. 1, pp 20-25.

8. D.W. Aha, 1998. The Omnipresence of Case-Based Reasoning in Science and Application, Knowledge-Based Systems 11(5-6), pp 261-273, Elsevier.

9. P. Kueng. 2000. The Effects of Workflow Systems on Organizations: A Qualitative Study. In Business Process Management, Lecture Notes in Computer Science nr 1806, pp 301-316, Springer Verlag, Berlin.

10. H. Reijers, J. Rigter, and W.M.P. van der Aalst. 2003. The Case Handling Case. International Journal of Cooperative Information Systems, 12(3), pp 365-391.

11. M. M. Richter. 1995. The Knowledge Contained in Similarity Measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal.

12. Z. Zhang, Q. Yang. 2001. Feature Weight Maintenance in Case Bases Using Introspective Learning. Journal of Intelligent Information Systems. Vol. 16, Iss. 2 (Mar/Apr), p. 95-116, Boston

13. A. Stahl. 2006. Combining Case-Based and Similarity-Based Product Recommendation. Lecture Notes in Computer Science. Advances in Case-Based Reasoning. Springer. Volume 4106. Book chapter. pp 355-369. DOI: 10.1007/11805816

14. T. P. Liang. 2008. Recommendation systems for decision support: An editorial introduction, Decision Support Systems, Volume 45, Issue 3 (June), Special Issue Clusters, pp 385-386, ISSN 0167-9236. DOI: 10.1016/j.dss.2007.05.003.

15. G. Lekakos, G.M. Giaglis. 2007. A hybrid approach for improving predictive accuracy of collaborative filtering algorithms. User Modeling and User - Adapted Interaction. Vol. 17, Iss. 1-2 (Mar), pp 5-40.

16. M.H. Schonenberg, B. Weber, B.F. van Dongen, W.M.P. van der Aalst. 2008. Supporting flexible processes through recommendations based on history. In M. Dumas, M. Reichert, M.C. Shan (Eds.), Business Process Management (6th International Conference, BPM 2008, Milan, Italy, September 1-5, 2008, Proceedings). (Lecture Notes in Computer Science, Vol. 5240, pp. 51-66). Berlin: Springer.

17. B. Weber, B.F. van Dongen, M. Pesic, C.W. Günther, and W.M.P. van der Aalst. 2007. Supporting Flexible Processes Through Recommendations Based on History. BETA Working Paper Series, WP 212, Eindhoven University of Technology, Eindhoven.

18. Workflow Management Coalition: Terminology & Glossary. Document Number WFMC-TC-1011, Document Status—Issue 3.0, February 1999. http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

19. R.S. Kaplan and D. P. Norton. 1992. The Balanced Scorecard – Measures that Drive Performance', Harvard Business Review, Vol. 70, No. 1, pp. 71–79.

20. R.S. Kaplan and D.P. Norton. 2001. Transforming the Balanced Scorecard from Performance Measurement to Strategic Management: Part I, Accounting Horizons, , Vol. 15. No. 1, pp. 87–106.

21. J. Furner. 2002. On recommending. Journal of the American Society for Information Science and Technology. Hoboken:JUl. Vol. 53, Iss. 9, pp 747-763.

22. G.J. Fitzsimons, D.R. Lehmann. 2004. Reactance to Recommendations: When Unsolicited Advice Yields Contrary Responses Marketing Science. Linthicum. Vol. 23, Iss. 1 (Winter), pp 82-94.

23. F. Sørmo, J. Cassens, A. Aamodt. 2005. Explanation in Case-Based Reasoning-Perspectives and Goals. The Artificial Intelligence Review. Dordrecht. Vol. 24, Iss. 2 (Oct), pp 109-143.

24. http://dev.mysql.com/doc/

25. Pallas Athena. 2004. Training Flower Developer Basic. Training Guide. Apeldoorn.

26. W.M.P. van der Aalst and M. Voorhoeve. 2008. Business Process Simulation. Lecture notes 2II75. Eindhoven University of Technology. Unpublished.

Appendix A : Description of the DSS types

Personal DSS (PDSS)
One of the most important DSS in the early days and in the present. It is about the empowerment of the individual and democratization of decision making. PDSS became important, because they were more successful than MIS.

Group Support Systems (GSS)
Decision responsibility is shared by a number of users. GSS is used for decision making, but also for communication and information processing. The group process consists of the group, the task, the group/organizational context, and the system.

Negotiation SS (NSS)
Group-oriented with IT to facilitate negotiations. NSS may be problem oriented or process oriented.

Intelligent DSS (IDSS)
Can be first generation (rule-based expert systems) or second generation (uses neural networks, genetic algorithms and fuzzy logic)

Enterprise Information Systems/ Business Intelligence (EIS/BI)
These are data-oriented DSS, which have a multidimensional view of data (OLAP, data cubes). They deliver enterprise-wide (management) reporting systems. BI is a contemporary term for EIS.

Data Warehouse (DW)
Is a set of databases created to provide information to decision makers. It provides the large scale IT infrastructure for contemporary decision support.

KM-based DSS (KMDSS)
Is used for knowledge storage, retrieval, transfer and application. Managing knowledge is vital for an organizations ability to increase innovation and competitive advantage and support decision-making.

Appendix B : Example of TV calculation

From Table 2 the following can be concluded. If TF 1 was chosen, alternative 2 will be recommended. If TF 2 was chosen, alternative 1 will be recommended.

Table 2: TV's for TF 1 and TF 2

	TF 1: minimize costs (in Euros)	TF 2: minimize duration (in seconds)
Alternative 1	79	**> 60 <**
Alternative 2	**> 77 <**	100
Alternative 3	84	70
Alternative 4	80	90
SUM	320	320

Two options will be explained for TV's derived via the heuristic: 70% * TF1 + 30% * TF2. What will be recommended? The first option uses the rank of the TV's, because the TF's have different parameters (Table 3).

Table 3: TV's for TF-Heuristic, option 1

	TF 1 (**ranked**)	TF 2 (**ranked**)	TF heuristic	New rank
Alternative 1	2	1	0,7*2+0,3*1 = 1,7	1
Alternative 2	1	4	0,7*1+0,3*4 = 1,9	2
Alternative 3	4	2	0,7*4+0,3*2 = 3,4	4
Alternative 4	3	3	0,7*3+0,3*3 = 3,0	3

So, the new rank is the same as TF 1, but place 2 and 1 are interchanged because of the different ranking of TF 2, especially the difference in rank between $TV_{2,1}$ (rank=1) and $TV_{2,2}$ (rank=4).

Another (more complicated and more qualitative) way of calculating the new TV is by normalising the TV and then applying the heuristic (Table 4).

Table 4: TV's for TF-Heuristic, option 2

	TF 1 (**diff.**)	TF 2 (**diff.**)	TF heuristic	New rank
Alternative 1	79/320=0,247	60/320=0,188	0,7*79/320+0,3*60/320 = 0,229	1
Alternative 2	77/320=0,241	100/320=0,313	0,7*77/320+0,3*100/320 = 0,262	4
Alternative 3	84/320=0,263	70/320=0,219	0,7*84/320+0,3*70/320 = 0,249	2
Alternative 4	80/320=0,250	90/320=0,281	0,7*80/320+0,3*90/320 = 0,259	3

So, the new rank is the same as TF 2. The reason that the new ranking sequence is the same as that of TF 2 is because of more deviation at TF 2 than at TF 1. The $TV_{X,1}$ (all values for TF 1) are all very similar. So, when there is not a lot of distinction in TV's for TF 1, the new TV's are determined by TF 2, although TF 2 has only 30 percent of influence in the heuristic.

Appendix C : Setting Up the Work Environment

The work environment exists of a laptop with different software components.

- The hardware exists of a Dell Latitude D610 laptop with an Intel Pentium M processor of 1,86 GHz and 2GB of RAM.
- Microsoft Windows XP SP2.
- Two VMware images. A VMware image is a complete copy of a system, which can be accessed and stored within another system. Image one was the simulation tool FLITS (v2, 2004). VMware image two consists of
    - FLOWer 3.7 SP1. At the time of the project the most stable version. FLOWer manages and supports the execution of activities in the context of a specific case. It is a case management system and has a run-time and a development environment.
    - The Navicat Lite for MySQL database administration tool is used. FLOWer communicates with the database via capabilities including MySQL statements.
- A back-up of the casetype, capabilities and definition of the database is made twice a week. A back-up of the VMware images is made once a month and stored on an external hard disk.

Appendix D : Developing the Prototype

Flower Functionality:

1. Interpret FLOWer's node, data and control objects.
2. Interpret example casetypes
3. Communication with the MySQL database via Mappings
4. Communication with the MySQL database via Capabilities in Mappings
5. Extending capabilities with all possible functionality
6. Implement a Module with simple C code that sets and gets a parameter
7. Extend a Module with Queries in the C code
8. Implement a Publisher that influences the distribution profiles
9. Interpret FLOWer's management information structure (MGMT tables)
10. Select and query MGMT tables
11. Define database tables for the recommendation service
12. Log case results for the recommendation service
13. Calculate similarity via queries, based on the case attributes
14. Implement a Target Function
15. Add a cost factor to the model that could be minimized
16. Derive a attribute-based recommendation option list
17. Define a attribute-based recommendation (with a user interface)
18. Calculate similarity via queries, based on the node sequence of a case
19. Derive a path recommendation option list
20. Define a path recommendation (with a user interface)
21. Implement a quality check on the log size before recommendations will be generated
22. Implement a variable horizon

Implementation:

23. Develop a casetype
24. Implement the recommendation in the casetype
25. Implement ratio's (per alternative) for influencing the TV's

Simulation:

26. Installing and configuring FLITS
27. Write Perl scripts for FLITS to automate simulation
28. Optimize the performance (where possible) of the database and the capability
29. Adapt the casetype to automatically insert the current node in the path recommendation
30. Adapt the casetype to automatically follow the path that is recommended
31. Simulate the casetype for different TF's as well as random as recommendation
32. Analyse the log files of FLITS

Evaluation:

33. Develop an Excel file that analyses the logged results from the recommendation service tables

Appendix E : Business Process With The Prototype - Four Phases

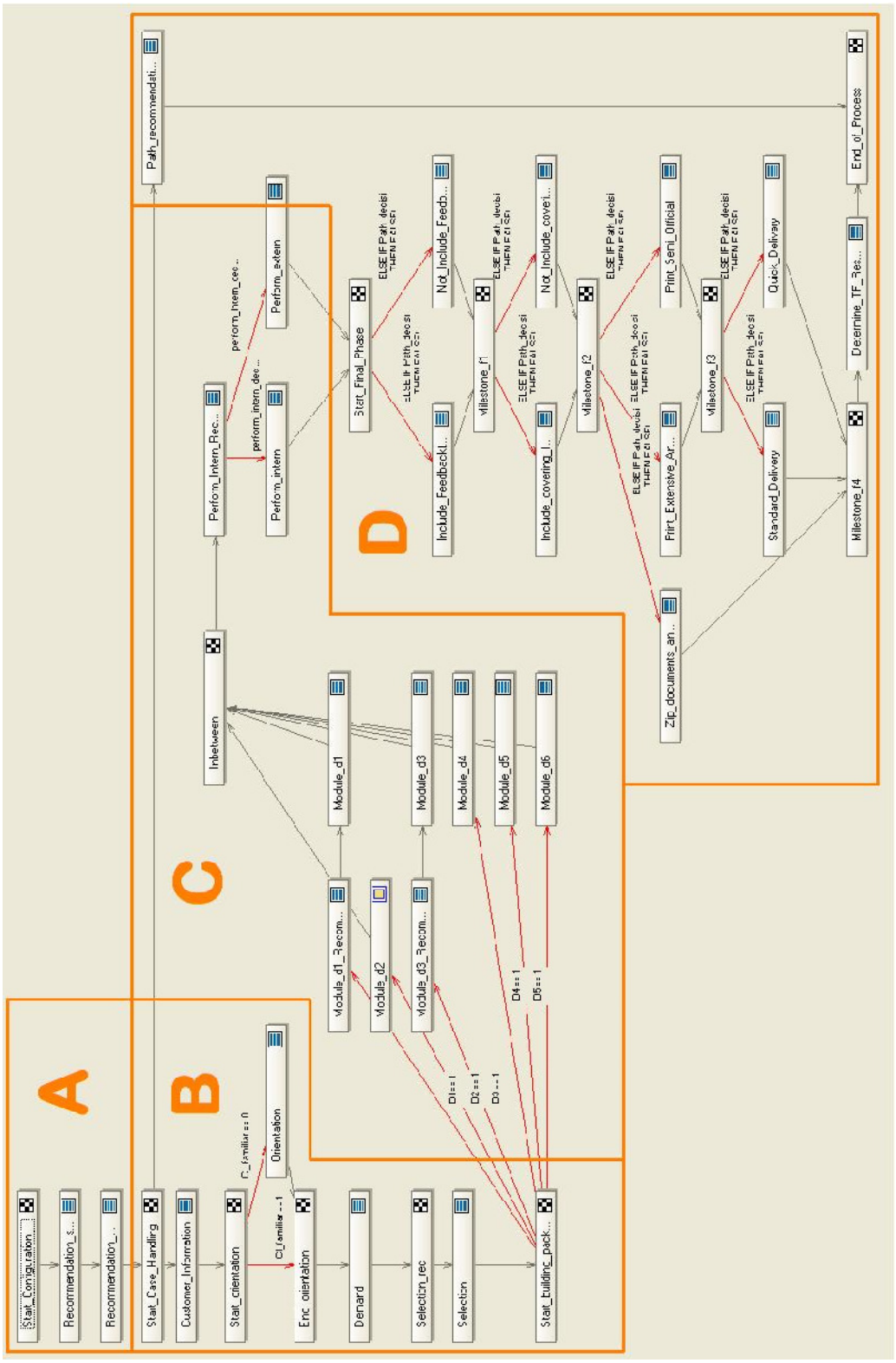

Figure 22: Four phases of the business process and prototype

Appendix F : Business Process Without the Prototype

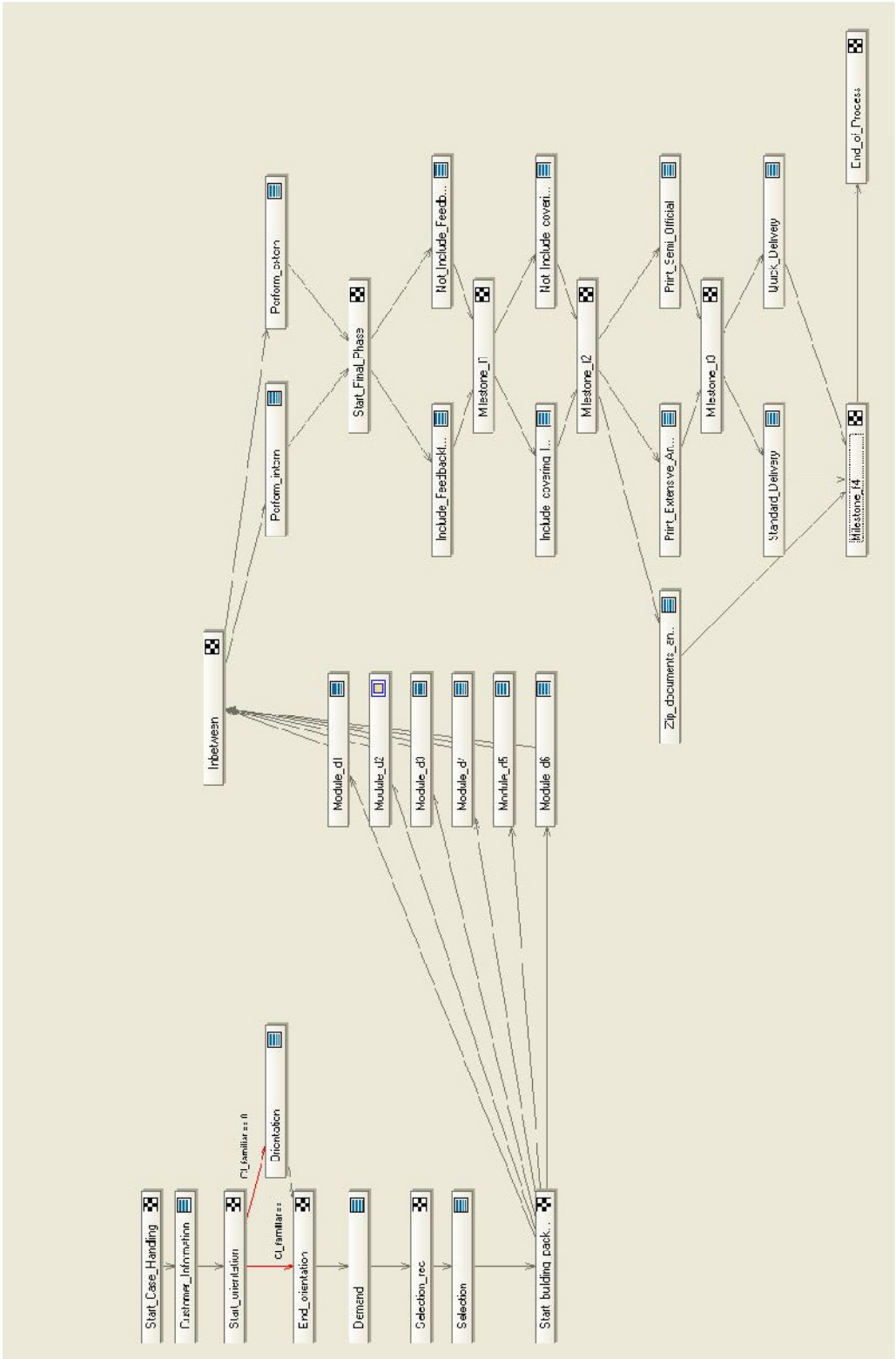

Figure 23: The business process

Appendix G : Four Phases

All nodes (activities) in the casetype will be described shortly and in the same sequence. All these nodes can be found in Appendix E.

A. Configuration phase
   1. Recommendation Start-up:
      - Choose the TF
      - Configure the recommendations: set the recommendation by selecting the alternatives per recommendation and select the case properties.
        The list of potential alternatives is filled with all possible alternatives. For example, one could choose for employee B at the recommendation. This would be a strange choice for the decision about whether to perform task intern or perform the task extern. The alternative "Employee B" makes no sense here, but could be selected. At the analysis for this decision this alternative will be ranked low, because it has never occurred. So, for performance and user overview reasons, only four alternatives can be chosen, but it could be possible to include all alternatives.
        The case attributes are the data elements (case content) that should match during a similarity measurement for a attribute-based recommendation.
   2. Recommendation Setting:
      - By deciding and choosing an alternative, the costs parameter will be influenced. Each alternative has its ratio. This is automatically set. The alternatives with the best ratios should be recommended by the service.
      - The cost parameter is automatically calculated and logged.
      - The minimum size of the case base (number of logged cases) could be defined.
      - The horizon could be defined. In practice the setting of the horizon is implemented directly in the code.

B. Start of the business process
   3. Customer Information:
      - An identity number is attached to the case.
      - Information about the customer is registered, like (1) name, (2) whether the permit request is for personal use or for business use and (3) whether the customer is familiar with the permit process.
   4. Orientation:
      - The decision for addressing this node or not, can be determined by the recommendation service. The customer will always be directed to the orientation node, whether the customer indicated to be familiar with permits or not. During the orientation node the customer will get extra supporting information about the permit order process. At the end of the page the customer has to indicate whether the information was helpful or not.

        This orientation node was build to explain a very simple example of a complete recommendation service for attribute-based (input) recommendations for paths to follow (output):
        - The familiarity attribute is defined as the identity of the case.
        - The indication whether the support was helpful is the result.
        - The target function is to have the highest possible satisfaction rate, by addressing ass few nodes as possible. Thus, if the customer found the

information on average not helpful, this orientation node should not be addressed.
- The case base is the familiarity attribute and the result per case.
- The alternatives are: (1) address the orientation node or (2) not address this node.

For example, the case has a familiarity attribute with the value "familiar". Both alternatives, about addressing this node or not, will be analysed. It is possible that the "address orientation" alternative has the highest satisfaction value. Even if the customer is familiar, this means that the customer indicated that the orientation node provides useful information. (If the orientation node was not addressed, then the question whether extra information would be useful will be raised at the end of the process. So, for both alternative paths there will be a satisfaction result.)
This is a recommendation example. There will be no focus on implementing it in the recommendation service.

5. Demand:
   - The customer selects which permits (modules) he demands.
6. Selection:
   - This is another, but different, example of providing recommendations. The focus of the prototype is on recommendations based on similar cases. The customer's demand is known from the previous node. The following situation occurs: a recommendation should be provided, which indicates whether another selection of the demand is better. For example, if a light and a regular construction permit are demanded, it is advisable to drop one of the two, because they are substitutes of each other.
     Pretend the demand attributes are the case identity. The problem is that the current case demand will be recommended, because similar cases are retrieved based on the demand attributes. So, the key attributes are the same as the attributes about which is recommended.
     In this case a nearest neighbour technique helps, because this technique analyses the attribute space near the current case and seeks for almost similar cases. A simplification of this technique is applied in this node: cases are retrieved that are similar for all demand attributes, except for one.
     There will be no further focus on the nearest neighbour technique. However, this example is a warning for persons that select the attributes for the domain specific attribute-based recommendation.

C. <u>Attribute-based recommendations</u>
This phase applies attribute-based recommendations for modules d1, d2, d3 and for the perform intern or extern decision.

7. Module_d1 Recommendation:
   - If module d1 is demanded, a recommendation node will occur for the role that has to perform the module d1 work item (like the alternatives in Figure 15). The data for this node is calculated in a caption that contains functionality in SQL statements. The functionality of this capability code is explained in chapter 6.1. When a certain role is chosen, the authorization and distribution will be handled automatic via the rules of the organization model and FLOWer's configuration manager.
8. Module_d1:
   - Symbolic work is performed by filling one data element.
9. Module_d2:

- The same type of recommendation as is performed for module d1. It is recommended which type of role has to perform the d2 module; a regular employee or a specialist. D2 automatically follows the recommendation without an intervening user. This is the type of automated decision making that misses the intelligence of the user and the flexibility to head to a certain decision.

10. Module_d3 Recommendation:
    - The same type of recommendation as is performed for module d1. It is recommended how a certain task has to be performed; brief or extended. It is not possible in FLOWer to have dynamic tasks. For example, when it is decided that module d3 should be performed extensive, data elements should be added or deleted from the task.
11. Module_d3:
    - Symbolic work is performed by filling one data element.
12. Module_d4:
    - No recommendation will be derived for this module. Symbolic work is performed by filling one data element.
13. Module_d5:
    - No recommendation will be derived for this module. Symbolic work is performed by filling one data element.
14. Module_d6:
    - No recommendation will be derived for this module. Symbolic work is performed by filling one data element.
15. Perform Intern Recommendation:
    - The same type of recommendation as is performed for module d1. It is recommended whether a certain task has to be performed in the organization or outsourced to another organization. This could e.g. help to control the regulation of workload.
16. Perform Intern:
    - Symbolic work is performed by filling one data element.
17. Perform Extern:
    - Symbolic work is performed by filling one data element.

D. <u>Path-based recommendations (Final phase)</u>

18. Path Recommendation:
    - The functionality of the path recommendations is explained stepwise with figures in Appendix H. For every next activity the path decision has to be made by a user in the path recommendation node. Every path choice causes its own case specific result. For example, if a case with module 1 is directed to the standard delivery node, it results in a cost of 90 euro. If this case is directed to the quick delivery node it results in a cost of 40 euro. However, if a case without module 1 is directed to a standard delivery, it results in a cost of 20 euro. If this case is directed to quick delivery it results in a cost of 60 euro. Thus each path choice causes a case specific and path specific result.
19. Include feedback form or not include feedback form
20. Include covering letter or not include covering letter
21. Zip documents and e-mail or print extensive or print semi official:
22. Standard delivery or Quick delivery
23. Determine TF results:
    - The measured results for the TF's are displayed. The user has also the option to classify the case as (not) representative if this is the matter. However, process managers might decide to exclude the case from the case base, because it heavily biases it. This is also discussed in chapter 3.

Appendix H: Technical Explanation of the Path-based Recommendation

Path-based recommendations
The code can be found in the hr.cap file in the [nodelist_decl] capability.

1. Define a list of nodes that can be used as alternatives for the path-based recommendation. Retrieve these nodes from the management information (MI) table case_type_basic that belong to the current casetype. Do not select data elements, only plan elements (nodes). Of these plan elements only select the node_kinds 2, 3 or 4.

The code can be found in the hr.cap file in the [pr_fill_subset00] capability.

1. The MI table mgmt_case_raw is a table in which all statuses of all nodes are logged. From this table the complete paths of all cases should be retrieved and inserted into the hr_pr_subset00. The nodes have to be finished, may only be of the node_kind 2, 3 or 4. The node_id's belonging to the Path_recommendation node or the Recommendation_System node may not be included in the node-sequence. For performance reasons only the node sequence of the last occurred case should be added, instead of deriving the whole list again.

The code can be found in the hr.cap file in the [pr_path] capability. This capability returns the path-recommendation.

1. The current node is known. From this node the previous node is selected. In Figure 24 an example can be seen of the MySQL statement for selecting which node was the last node before the current node. Because of the structure of the mgmt_case_raw table it is difficult to derive the sequence of nodes. The node before the current node is the node that has the largest update number that is smaller than the update number of the current node. Again, this node may only be a formfill-, user decision-, or system decision node (not e.g. a milestone).

```
#------------------------- 1 ------------------------------
#  find node_id of the node previous to the current node
SELECT              node_id
INTO                node01
FROM                mgmt_case_raw
WHERE               class_id = {class_id}
        AND         object_id = {case_id}
        AND                 node_status = '1'
        AND                 upd_nr      = (SELECT MAX(upd_nr)                          # find lower upd_nr than upd_nr00
                                            FROM      mgmt_case_raw
                                            WHERE     class_id = {class_id}
                                            AND       object_id = {case_id}
                                            AND       node_status = '1'
                                            AND upd_nr < {upd_nr00}
                                            AND       node_id != ALL  (SELECT obj_id          # do not select e.g. milestones
                                                                       FROM      case_type_basic
                                                                       WHERE     obj_kind = ('1' OR '5' OR '6' OR '7' OR '8' OR '9' OR '10' OR '0')
                                                                       AND       class_id = {class_id}
                                                                       )
                                            AND       node_id != (SELECT obj_id          # do not select the PR node itself
                                                                       FROM      case_type_basic
                                                                       WHERE     name = 'Path recommendation'
                                                                       AND       class_id = {class_id}
                                                                       )
                                            )
LIMIT 1;
```

Figure 24: Part of the path recommendation capability code

2. Perform step 1 for as many nodes as the horizon is long.

3. Check whether there are more cases in the case base than required (quality check). If there are more cases than required the recommendation will be generated, otherwise an empty recommendation will be returned.

4. In the next steps the partial path of the current node will be compared to all paths in the hr_pr_subset00 table. First all cases are selected in which the current node exists and insert them into hr_pr_subset01.

5. Select all rows (one node for one case on one row) which contain the case id nodes for all these cases in hr_pr_subset02.

6. Generate per case a string with the sequence of all nodes it followed. Insert these strings into hr_pr_subset03.

7. Define the horizon by concatenating the found sequence of nodes of the current case (step 1 and 2).

8. Compare all strings with the defined current case sequence and write the similar cases into the hr_pr_subset04.

9. Analyse the string and select the node that occurs directly after the current node. For example, in Figure 25the cases 1475 and 1558 are similar for the path-based identity with horizon = 6.

[Table] hr_pr_subset04 @FLOWer_37 (wf)

ile Edit View Window

Memo Hex Image Sort Ascending Sort Descending Remove Sort

object_id	class_id	node_id	next_node	subsequence	node06	node05	node04	node03	node02	node01	current_noc
1475	521	247,248,249,251,252,257,258,260,261,262,264,266,267,273,275,276,279,281	279	262,264,266,267,273,275,276	262	264	266	267	273	275	276
1558	521	247,248,249,251,252,260,261,262,264,266,267,273,275,276,279,281	279	262,264,266,267,273,275,276	262	264	266	267	273	275	276

Figure 25: Similar cases in the hr_pr_subset04

10. Generate an option list table (hr_pr_alternatives) with aggregated information about all alternatives. These alternatives are the found nodes in step 8. The target values of all similar nodes are averaged and the number of occurrences per node is counted. The capability code is shown in Figure 26 and the result in the database in Figure 27.

```
TRUNCATE TABLE hr_pr_alternatives ;

# define all alternatives in the option list
INSERT INTO     hr_pr_alternatives (path_alternative, count, total_costs, total_profit, total_duration)
SELECT          next_node, COUNT(next_node),  (SELECT  ROUND( AVG(total_costs) , 0)
                                               FROM          hr_d_log
                                               WHERE     (object_id, class_id, next_node) IN (  SELECT object_id, class_id, next_node
                                                                                                FROM       hr_pr_subset04 )
                                               ) ,  (SELECT  ROUND( AVG(total_profit) , 0)
                                               FROM          hr_d_log
                                               WHERE     (object_id, class_id, next_node) IN (  SELECT object_id, class_id, next_node
                                                                                                FROM       hr_pr_subset04 )
                                               ) ,  (SELECT  ROUND( AVG(total_duration) , 0)
                                               FROM          hr_d_log
                                               WHERE     (object_id, class_id, next_node) IN (  SELECT object_id, class_id, next_node
                                                                                                FROM       hr_pr_subset04 )
                                                                                                     )
FROM            hr_pr_subset04
WHERE           next_node != '0'
GROUP BY        next_node   ;
```

Figure 26: Generating the option list in the capability code

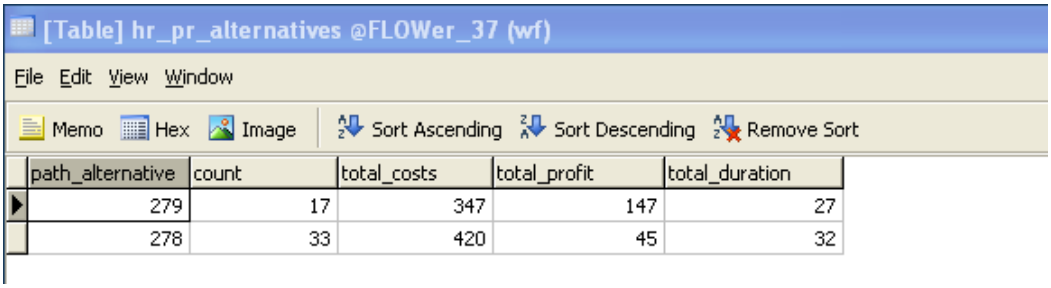

Figure 27: The option list table in the database

11. Select the alternative with the most optimal value for the selected TF. Return this value to the Flower's user interface, which would look like Figure 28.

After requesting a path recommendation all information is automatically calculated and selected. Only the user has to insert his decision. After this decision the user is automatically directed to the selected node.

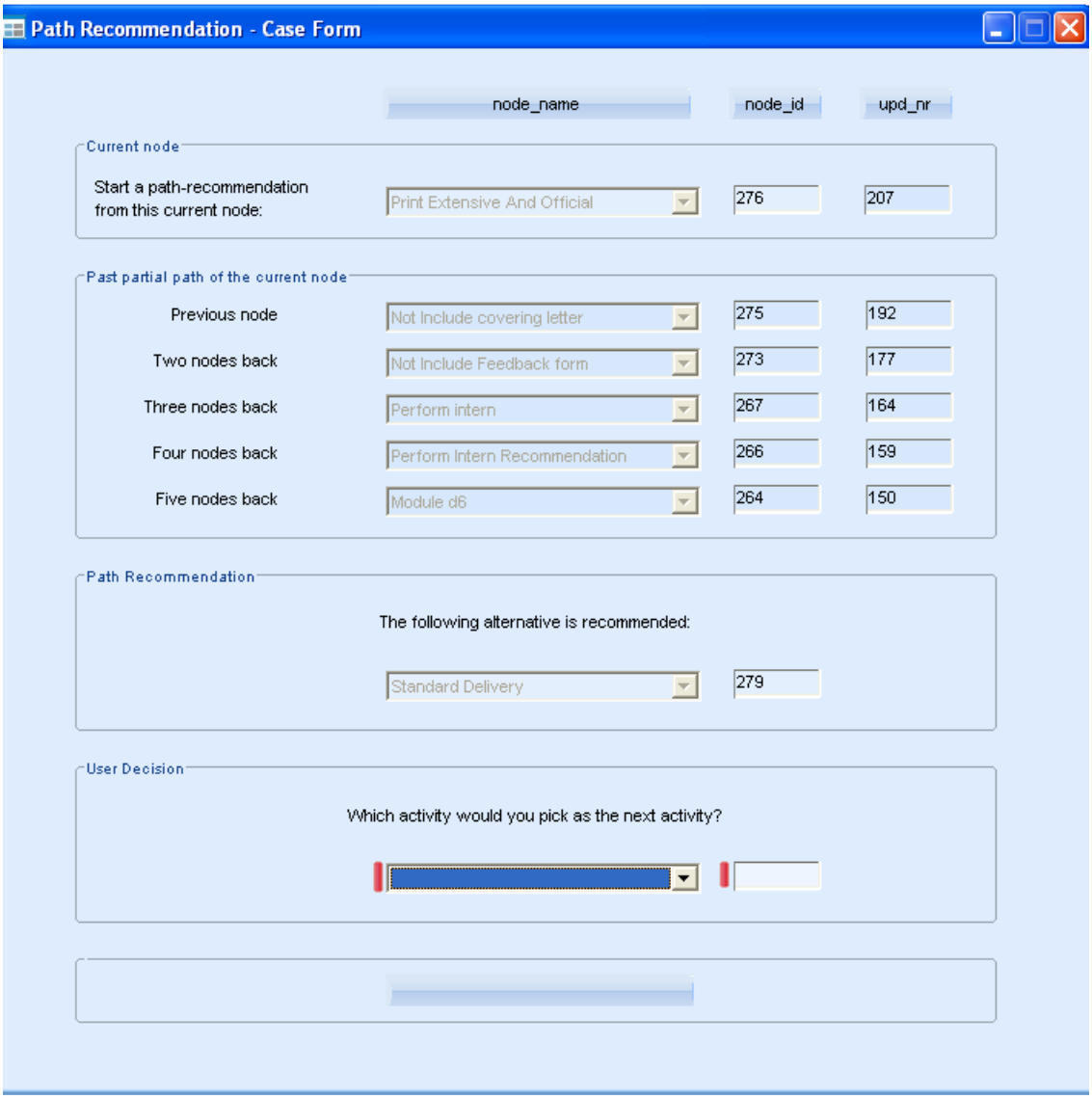

Figure 28: Path recommendation interface

Appendix I : FLITS script

```
204
205 #=====START=========================================================
206 sub create_case{
207
208    $case_hdl = Chif::create_case($con_hdl,$CASE_NAME);
209
210 }
211
212 #=====TARGET FUNCTION and Setting=====================================
213 sub set_target_function{
214
215    my $open_hdl       = Chif::open_case($con_hdl,$case_hdl,'write');
216
217    my @root_node     = Chif::get_wavefront($open_hdl,CHP_ROOT_NODE,WF_NODE,$con_hdl);
218    my $root_node_hdl = $root_node[0]->node;
219
220    Chif::set_data_value($open_hdl,$root_node[0]->node,"Target_function",$tf,$con_hdl);
221    Chif::set_data_value($open_hdl,0,"PR_Horizon",$horizon,$con_hdl);
222    Chif::set_data_value($open_hdl,0,"PR_required_logsize",$required_logsize,$con_hdl);
223
224    Chif::close_case($open_hdl,$con_hdl);
225
226 }
227
228 #=====DEMAND==========================================================
229 sub set_random_demand{
230
231    my $open_hdl    =  Chif::open_case($con_hdl,$case_hdl,'write');
232
233    Chif::set_data_value($open_hdl,0,"CI_satisfaction",$satisfaction,$con_hdl);     # NB bij
elkaar gestopt!
234
235    Chif::set_data_value($open_hdl,0,"D1",$module_1,$con_hdl);
236    Chif::set_data_value($open_hdl,0,"D2",$module_2,$con_hdl);
237    Chif::set_data_value($open_hdl,0,"D3",$module_3,$con_hdl);
238    Chif::set_data_value($open_hdl,0,"D4",$module_4,$con_hdl);
239    Chif::set_data_value($open_hdl,0,"D5",$module_5,$con_hdl);
240    Chif::set_data_value($open_hdl,0,"D6",$module_6,$con_hdl);
241
242    Chif::close_case($open_hdl,$con_hdl);
243
244 }
245
246 #=====REC 01a=========================================================
247 sub follow_recommendation_01{
248
249    my $open_hdl    =  Chif::open_case($con_hdl,$case_hdl,'write');
250    my $rec_01       =  Chif::get_data_value($open_hdl,0,"Mod_d1_role_rec") ;
251
252    Chif::set_data_value($open_hdl,0,"Mod_d1_role_dec",$rec_01,$con_hdl);
253    Chif::set_data_value($open_hdl,0,"mod_d1_dothis",123456,$con_hdl);
254
255    Chif::close_case($open_hdl,$con_hdl);
256
257 }
258
```

Appendix J : FLITS case log file

recommendation_service (real_life_test)						
Nr	**State**	**Type**	**Value in**	**Value out**	**Duur**	**ext. error**
001	✓	Chif::connect_server	Connected to (192.168.229.129) (succeeded as expected)	-	0.013	none
001	✓	Chif::set_locale	Set locale client language to (en)	-	0.000	none
001	✓	Chif::ident_to_server	Identifying as bpm (succeeded as expected)	-	0.000	none
001	✓	Chif::create_case	Created case: HR	2247	1.834	none
001	✓	Run HR	Created a case	-	1.834	none
001	✓	Chif::open_case	Open 521 2247 for write (succeeded)	-	2.466	none
001	✓	Chif::get_wavefront	Wavefront returnd from (0) type (WF_NODE)	-	0.003	none
001	✓	Chif::set_data_value()	Set (Target_function) with chp_dat_setval on data_hdl (2000) with value (1) (succeeded)	-	0.294	none
001	✓	Chif::set_data_value()	Set (PR_Horizon) with chp_dat_setval on data_hdl (2001) with value (4) (succeeded)	-	0.004	none
001	✓	Chif::set_data_value()	Set (PR_required_logsize) with chp_dat_setval on data_hdl (2002) with value (1) (succeeded)	-	0.002	none
001	✓	Chif::close_case	Open case closed (succeeded)	-	0.011	none
001	✓	Run HR	target_function_set	-	2.814	none
001	✓	Chif::open_case	Open 521 2247 for write (succeeded)	-	0.188	none
001	✓	Chif::set_data_value()	Set (CI_satisfaction) with chp_dat_setval on data_hdl (2000) with value (1) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (D1) with chp_dat_setval on data_hdl (2001) with value (1) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (D2) with chp_dat_setval on data_hdl (2002) with value (1) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (D3) with chp_dat_setval on data_hdl (2003) with value (1) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (D4) with chp_dat_setval on data_hdl (2004) with value (1) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (D5) with chp_dat_setval on data_hdl (2005) with value (0) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (D6) with chp_dat_setval on data_hdl (2006) with value (0) (succeeded)	-	0.004	none
001	✓	Chif::close_case	Open case closed (succeeded)	-	0.005	none
001	✓	Run HR	set_random_demand	-	0.227	none
001	✓	Chif::open_case	Open 521 2247 for write (succeeded)	-	0.432	none
001	✓	Chif::get_data_value	Data value for Mod_d1_role_rec	2	0.003	none
001	✓	Chif::set_data_value()	Set (Mod_d1_role_dec) with chp_dat_setval on data_hdl (2001) with value (2) (succeeded)	-	0.002	none
001	✓	Chif::set_data_value()	Set (mod_d1_dothis) with chp_dat_setval on data_hdl (2002) with value (123456) (succeeded)	-	0.003	none
001	✓	Chif::close_case	Open case closed (succeeded)	-	0.004	none
001	✓	Run HR	follow_recommendation_01	-	0.449	none
001	✓	Chif::open_case	Open 521 2247 for write (succeeded)	-	0.163	none
001	✓	Chif::get_data_value	Data value for Mod_d2_roletype_dec	10	0.001	none
001	✓	Chif::set_data_value()	Set (Dothis10) with chp_dat_setval on data_hdl (2001) with value (2222210) (succeeded)	-	0.002	none
001	✓	Chif::close_case	Open case closed (succeeded)	-	0.004	none
001	✓	Run HR	follow_rec_mod_2	-	0.175	none
001	✓	Chif::open_case	Open 521 2247 for write (succeeded)	-	0.455	none
001	✓	Chif::get_data_value	Data value for mod_d3_type_rec	7	0.001	none

Etc.

Appendix K : Case Base

Table 5: The case base : hr_d_log

Column	Description
Casename	Case identification
Key attribute 1 Key attribute 2 Key attribute 3 ...	Key attributes. All elements together define the attribute-based identification.
Target_function	TF identification
Total_costs Total_profit Total_duration ...	Total case results (at least one per target function): indicates the level of success of the case
Recommendation 1 Recommendation 2 Recommendation 3 ...	Logging per recommendation what was recommended
Recommendation 1 decision Recommendation 2 decision Recommendation 3 decision ...	Logging per recommendation what was decided

[Table] hr_d_log @FLOWer_37 (wf)

File Edit View Window

Memo Hex Image Sort Ascending Sort Descending Remove Sort

class_id	object_id	casename	d1	d2	d3	d4	d5	d6	d_count	target_function	total_price	total_costs	total_profit	total_duration	starttime	d1_emp_role	d1_emp_role_dec	d2_roletype	d3_type	d3_type_dec	perform_intern	perform_intern_dec
521	2758	HR-52102758	0	1	1	1	0	1	4	3	640	344	296	19	20090131211810	0	0	10	7	7	6	6
521	2759	HR-52102759	0	1	1	0	1	1	4	3	620	422	198	18	20090131211829	0	0	10	7	7	6	6
521	2761	HR-52102761	0	0	0	0	1	0	1	3	120	160	-40	14	20090131211900	0	0	0	0	0	6	6
521	2762	HR-52102762	0	0	1	0	0	1	2	1	280	208	72	15	20090131211914	0	0	0	7	7	6	6
521	2764	HR-52102764	1	1	0	1	1	0	4	3	780	550	230	19	20090131211947	1	1	10	0	0	6	6
521	2765	HR-52102765	1	0	0	1	1	1	4	1	660	472	188	19	20090131212006	1	1	0	0	0	6	6
521	2766	HR-52102766	0	1	1	0	1	1	4	1	620	422	198	19	20090131212025	0	0	10	7	7	6	6
521	2767	HR-52102767	0	0	1	1	1	1	4	1	540	221	319	19	20090131212044	0	0	0	7	7	6	6
521	2768	HR-52102768	0	1	0	1	1	1	4	1	580	471	109	19	20090131212103	0	0	10	0	0	6	6
521	2769	HR-52102769	1	0	0	1	0	1	3	3	540	362	178	18	20090131212122	1	1	0	0	0	6	6
521	2770	HR-52102770	0	1	0	0	1	1	3	3	440	349	91	18	20090131212140	0	0	10	0	0	6	6
521	2771	HR-52102771	0	0	0	1	0	0	1	3	140	103	37	15	20090131212158	0	0	0	0	0	6	6
521	2772	HR-52102772	0	0	1	0	1	1	3	1	400	318	82	18	20090131212214	0	0	0	7	7	6	6
521	2773	HR-52102773	1	0	1	0	1	1	4	1	700	532	168	20	20090131212232	1	1	0	7	7	6	6
521	2774	HR-52102774	0	0	1	0	1	0	2	3	300	243	57	17	20090131212252	0	0	0	7	7	6	6
521	2775	HR-52102775	0	0	0	0	1	0	1	1	120	200	-80	16	20090131212309	0	0	0	0	0	6	6
521	2776	HR-52102776	1	1	0	0	1	1	4	1	740	533	207	19	20090131212326	1	1	10	0	0	6	6
521	2777	HR-52102777	1	0	0	1	0	1	3	1	540	362	178	18	20090131212346	1	1	0	0	0	6	6
521	2778	HR-52102778	0	0	1	1	1	1	4	3	540	221	319	20	20090131212404	0	0	0	7	7	6	6
521	2779	HR-52102779	1	1	0	0	1	1	4	3	740	553	187	19	20090131212425	1	1	10	0	0	6	6
521	2780	HR-52102780	0	1	1	0	1	1	4	3	620	422	198	20	20090131212445	0	0	10	7	7	6	6
521	2781	HR-52102781	1	1	0	0	1	1	4	3	740	603	137	20	20090131212506	1	1	10	0	0	6	6
521	2782	HR-52102782	0	1	0	1	0	1	3	1	460	421	39	20	20090131212526	0	0	10	0	0	6	6
521	2783	HR-52102783	1	0	0	1	1	1	4	1	660	522	138	21	20090131212546	1	1	0	0	0	6	6
521	2784	HR-52102784	1	0	1	0	1	1	4	3	700	532	168	21	20090131212607	1	1	0	7	7	6	6
521	2785	HR-52102785	1	1	1	0	1	1	5	3	920	546	374	23	20090131212628	1	1	10	7	7	6	6
521	2786	HR-52102786	0	1	1	0	1	0	3	1	520	317	203	19	20090131212652	0	0	10	7	7	6	6
521	2787	HR-52102787	0	1	0	0	1	1	3	3	440	339	101	20	20090131212712	0	0	10	0	0	6	6

Figure 29: The case base of the prototype

Appendix L: Context for developing the prototype

Before developing the prototype different resources were consulted. Knowledge and experience were gathered via a Protos course, lessons in the FLOWer basic developer guide, Pallas Athena's Intranet, the online MySQL developer manual [24], and Pallas Athena's experienced employees.

First, the development context (FLOWer) will be described in more detail. FLOWer is a process management system. The different components can be functionally separated. These components can be seen in Figure 30 [25]:

- Process design: modelling the activities, sub processes, roles, authorization, constraints, sequence of activities, etc.
- Integration with other IT systems: for example with word processor, calculators, or older information systems. For the recommendation service only an external database is integrated.
- Organizational design: setting up the organization for operational activities, like defining the distribution and authorization of work in profiles.
- Distribution of work: employees receive work for which they are authorized and which belongs to their organizational unit. It is possible to sort work for a predefined priority and to apply operational flexibility, like redefining the distribution of work.
- Execution: an employee receives and handles a work item.
- The Recommendation Configuration component is added. The settings for the recommendations should be set here. These could be the type of recommendation, the horizon, the key attributes, the location of the recommendation service, etc.

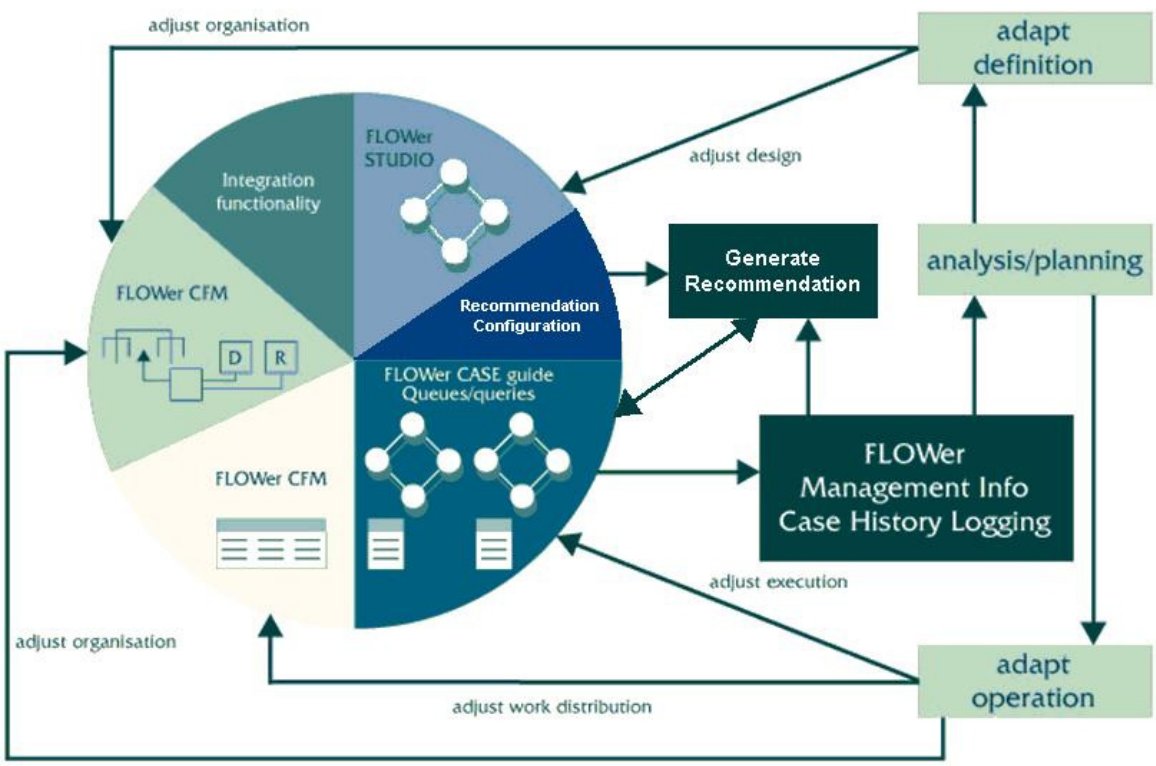

Figure 30: Components of the FLOWer process management system [25]

FLOWer Studio is the process design component. A design consists of about four elements [25]:

- Process flow, for defining the sequence of the process execution.
- Role graph, for defining the hierarchical structure of the different roles.
- Data elements that are required in the process.
- Forms that will be the interface with the user for entering data.

The process in FLOWer is very data-driven. By deriving or filling required data elements, other data elements at other nodes further in the process can become enabled. When all required data elements are filled, a node becomes completed. Different transitions can take place; a node can be in one of these states: executed (completed), skipped, redone (reset), or refused.

The setting of a data element by the user until the storage of that data element, occurs through different layers. The most common and most used method is:

1. The user inserts a value in a control object, like an edit box or a check box.
2. This box is connected to a data element.
3. If all required data elements for a mapping are filled, the values of these data elements will become available in the invoked capability.
4. The functionality in the capability is executed. This functionality could exist of MySQL statements which set or retrieve (aggregated) information in tables.
5. The information will be directed via the same steps backwards to the interface.