

Towards a Requirements Engineering Process Model

Armin Eberlein
Department of Electrical & Computer Engg.
University of Calgary, Canada
e-mail: eberlein@enel.ucalgary.ca

Li Jiang
Department of Computer Science
Dandong Normal College, China
Email: jiangl@enel.ucalgary.ca

Abstract

The creation of a development process is a challenging task. The application, customization and refinement of generic process models into fine-grained process steps suitable for a specific problem domain requires major work. This paper first reviews a generic framework for requirements engineering as well as a domain-specific framework. It then outlines some basic principles for a methodology that helps the process engineer develop a process model that considers domain knowledge, as well as constraints and priorities for a particular project. The methodology is called REPM (Requirements Engineering Process Model) and uses mathematical definitions to generate an initial process model to be refined in an iterative manner using the feedback of the process designer.

1 Introduction

Software development has been a challenging task for several decades. Although many good ideas have been suggested, the increasing complexity of software systems seems to make it difficult to achieve a drastic break-through in software quality. There appears to be a consensus that a well-defined process is an absolute necessity for any good software development. But putting the standard models, such as the waterfall model, the spiral model, etc. [1] into practice is a non-trivial task. These models are too coarse-grained to be used directly. They require a lot of additional work to include domain and project-specific constraints and to get the level of granularity that is needed to guide the software development. Experienced people are needed that can fill in the gaps. But even then, there are still major hurdles to overcome. This paper gives some initial ideas on a framework that can be used to assist in the definition of a domain specific, fine-grained process model.

Section 2 outlines Pohl's 3-dimensional framework for requirements engineering. Section 3 shows how this framework has been customized to suit the telecommunication domain. Section 4 describes a new methodology called REPM (Requirements Engineering Process Model) that was inspired by previous work and that can be used to develop process models for various domains.

2 Pohl's 3-Dimensions of Requirements Engineering

Pohl worked on a process-centred framework for requirements engineering (RE) [2] that has been widely published and cited. According to Pohl, the essence of the RE process is the transformation of requirements that are vague, expressed in natural language and incomplete into a complete and formal specification that is agreed by all stakeholders. Based on this, Pohl proposed three dimensions of requirements

engineering (see Figure 1) that could be applied to various problem domains. Pohl assumes that the whole RE process can be split into small sub-processes along the three dimensions which are supposed to be independent of each other. The three dimensions are defined as specification, representation and agreement [3]. The specification dimension deals with the degree of requirements understanding at a given time. At the beginning of the RE process the specification of the system and its environment is more or less opaque. Focusing on this dimension, the aim of RE is to transform the operational need into a complete system specification through an iterative process of definition and validation. The representation dimension copes with the different representations used for expressing knowledge about the system. Within RE there are three categories of representations. The first category includes all informal representations, such as arbitrary graphics, natural language, etc. The second category subsumes the semi-formal languages such as ERD and SADT. The third category covers formal languages such as specification languages (e.g., VDM, Z) or knowledge representation languages (e.g. ERAE, Telos). The agreement dimension deals with the degree of agreement reached on a specification. At the beginning of the RE process each person involved has his/her own personal view of the system. Negotiation is required until a so-called common system specification has evolved, i.e., a specification on which the entire RE team has agreed. Getting from the initial input to the desired output is an iterative process consisting of different actions. An activity that is aimed at progress in one dimension may also affect other dimensions. For instance, improving one dimension may advance or cause a setback in another dimension.

Pohl's RE process framework is a first step towards a generic framework that can be used in various domains. However, it also has several limitations: Pohl's framework assumes the usage of formal specifications. Some companies might not be interested in a formal definition of the requirements but prefer semi-formal notations, such as DFDs, UML. Additionally, the agreement dimension implies that agreement increases continuously as the specification progresses. Reality shows that many projects have initially complete agreement of all stakeholders, but as the specification progresses, conflicts between stakeholders' interests increase. Only later in the final stages of the specification, the agreement will (hopefully) increase again. These are some of the reasons why we think that Pohl's original framework

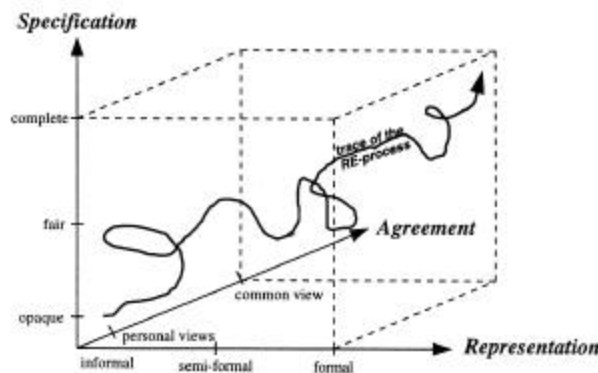


Figure 1: Pohl's 3-dim framework for requirements engineering

needs to be customized for different domains. An example of how this framework can be customized for the telecommunications domain (especially for telecommunication services development) is given in Section 3.

3 The Requirements Assistant for Telecommunication Services (RATS)

In our previous work we developed a 3-dimensional framework for the requirements engineering process of telecommunication service development [4]. This work was inspired by Pohl's generic framework, however, it addresses some of the weaknesses of Pohl's approach and customizes the requirements engineering process to telecommunication service development. The framework defines three dimensions (completeness, refinement and formality) that are of major concern during telecommunication service development and that can be seen as orthogonal to each other. This orthogonality ensures that the dimensions are fairly independent of each other. The dimensions have states which allow the definition of responsibilities, milestones and metrics for the service development process. The states were defined based on the constraints given by a collaborating telecommunication service provider who wanted to use the Intelligent Network (IN) architecture [5] as well as a formal specification language called SDL [6]. Based on these parameters, the framework was customized to the format given in Figure 2.

The RATS methodology has been developed based on the assumption that the start of service development and the final service specification are on opposite corners of the cube defined by the three dimensions (see Figure 2). This assumption was also made in Pohl's work. However, while Pohl stated that the trace of the RE process is an arbitrary curve within the cube spanned by the three dimensions [7], we assume that in most cases a smooth progression of the development in all three dimensions is required. Working only in the direction of one single dimension and neglecting the other two dimensions reflects generally not a good development process. The "ideal" progress in the development is therefore assumed to be along the diagonal. However, real progress will much more look like a wavy curve oscillating around the "ideal" diagonal. This assumption has not been proven, however, while deriving the RATS methodology guidelines from the RATS 3-dimensional framework this assumption was found to be useful [4].

The RATS methodology consists of a series of actions that need to be performed in

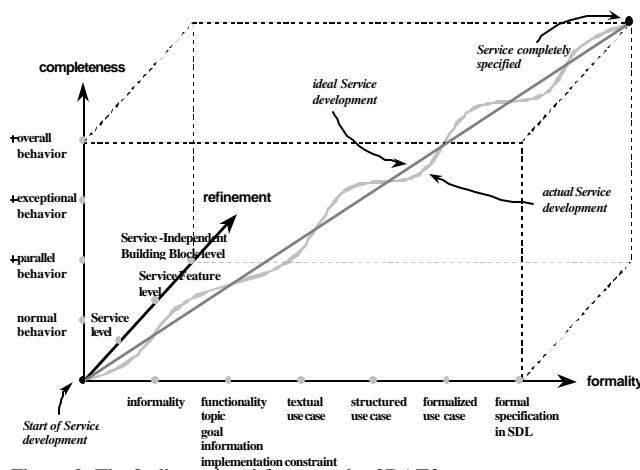


Figure 2: The 3-dimensional framework of RATS

order to get from the starting point to the end point in the development. These actions are coordinated in such a way, that the diversion from the diagonal in Figure 2 remains at a minimum. Carrying out this exercise and using domain knowledge, the RATS methodology guidelines have been developed. The

detailed derivation of the guidelines from the 3dimensional framework can be found in [4]. The two guiding principles were the proximity of the development to the “ideal” diagonal, and the domain knowledge that provides constraints.

A great advantage of the RATS guidelines is that they have a relatively fine granularity and are very domain-specific. This allows the implementation of the guidelines into an expert system that can provide answers to the developers as to what should be done next. A prototype of this expert system has been developed [4]. However, the development of the methodology guidelines had to be done “by hand” in a very time-consuming manner as it required the handling of complex constraints. Certain activities need to be done in a specific order, even though this order might lead further away from the “ideal” diagonal. Not being able to define any constraints very much limits the usability of the RATS methodology. We, therefore, felt that defining a more generic framework, that allows the definition of n dimensions with an arbitrary number of states on each dimension, together with the ability to define constraints as well as relationships between the various states, would greatly enhance the usefulness of a process framework. This is described in the next section.

4 Formal Description of REPM

4.1 Introduction

This section now describes ideas on how the previously described work can be expanded into a generic requirements engineering process called REPM that can be used to develop a fine-grained, domain-specific process. REPM uses Pohl’s idea of dimensions to establish a foundation for a requirements engineering process. It also takes the idea of domain-specific definitions of the dimensions from RATS, together with the assumption that the diagonal of the cube can in a first instance be assumed to represent the “ideal” development. RATS was seen as the validation of this assumption. REPM now is an attempt to formalize the procedure that was used to derive the methodology guidelines of RATS from the 3-dimensional framework shown in Figure 2. It goes beyond the capabilities of the previous work by allowing the process engineer to define domain-specific constraints between process activities. This assumes that it is possible to pre-define the development path that can be taken through the activities of a process model. This has been seen as restrictive by some researchers [8][9]. However, we find that our framework represents a well-balanced trade-off between the genericity of the framework (i.e., it can be customized to various domains) and the ability to provide guidance during the development (i.e., it contains a list of successive activities), two aspects that are generally mutually exclusive.

4.2 Abstraction and definitions

We abstract the 3-dimensional requirements engineering frameworks described in Sections 2 and 3 into an n -dimensional vector space with arbitrary numbers of generic states in each dimension. This means that the current status of the development can be defined by a unique set of coordinates within the n -dimensional space. This abstraction allows us to use sound mathematical definitions for the framework and to automate the calculations. The 3-dimensional frameworks described in the previous sections can be seen as instantiations of this more generic model.

Before we describe the mathematical calculations, we want to give an overview of some of the definitions used:

n : denotes the number of dimension, $n \in \mathbb{N}$

D_i : represents dimension i ($i=1,2,\dots,n$)

M_i : corresponds to the number of states in dimension i

$m_{i,j}$: represents the distance of the j^{th} state in dimension i . To simplify the subsequent calculations, we give each state a unique number. For instance, $m_{1,3}$ means the 3rd state in dimension 1 (D_1), so $m_{1,3} = 3$.

We assume that the states in each dimension are equally separated, for example in dimension j the distances between $m_{i,j}$, $m_{i+1,j}$ and $m_{i-1,j}$, $m_{i,j}$ are equal.

dh : represents the distance of a given point in the n -dim space to the diagonal

\vec{V}_i : symbolizes the vector for dimension i ($i=1,2,\dots,n$)

r_i : represents the unit vector for dimension i , i.e. $|\vec{r}_i|=1$, we abbreviated it as $r_i = 1$, for all i , ($i=1,2,\dots,n$),

\vec{V}_{diag} : corresponds to the diagonal vector in the n -dimensional space of the

problem domain, i.e. the “ideal” development.

In order to reduce the complexity of this discussion, we assume the whole distance

in each dimension to be 1, i.e., $|\vec{V}_i|=1$. This means that the distances of m_j to

$m_{i,j+1}$ and $m_{i,j}$ to $m_{i,j+1}$ (i?’) are generally not the same.

4.3 Mathematical basis for a generic framework

The RATS 3-dimensional framework points out that the “ideal” route is the diagonal that goes from the initial starting point to the final endpoint. This assumption is reasonable since it allows smooth progress in all three dimensions. However, both of the above mentioned frameworks also mention that real development is more realistically represented by a random curve along the diagonal of the 3-dimensional space (see Fig. 3). Expressed in mathematical terms, the best development process has the minimum distance from the diagonal. This reduces the problem of searching for the best development process to a mathematical minimization problem for which we can use vector theory.

The development can then be seen as a sequence of vectors that leads from the starting point O in one corner of the space to the destination D in the opposite corner (see Figure 3 for a 3dimensional example). If all distances are normalized, we can define for the n -dimensional space:

$$\vec{V}_{diag} = r_1 + r_2 + \dots + r_N = \sum r_i$$

$$r_i = 1 \quad i \in \{1,2, \dots, n\} \quad (1)$$

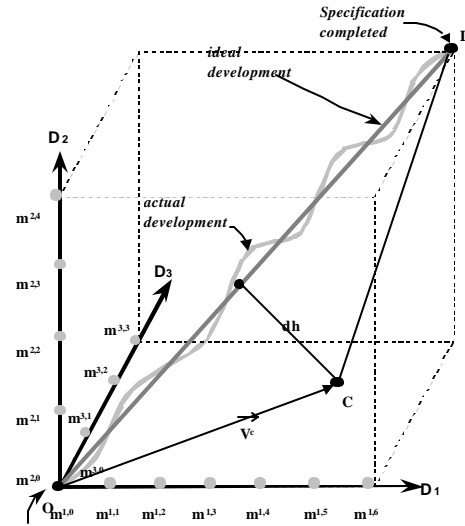


Figure 3: Example of REPM with 3 dimensions

From this we can conclude the following:

$$\left| \vec{V}_{diag} \right| = \sqrt{1 + 1 + \dots + 1} = \sqrt{N} \quad r_i = 1 \quad (2)$$

In the example shown in Figure 3 with n=3, we get the following: $\left| \vec{V}_{diag} \right| = \sqrt{3}$

Suppose that the current position is at point C (see Figure 3), the current vector is \vec{OC} , which we call vector \vec{V}_c . We can then write vector \vec{V}_c as:

$$\vec{V}_c = \frac{m_{1,C_1}}{M_1} r_1 + \frac{m_{2,C_2}}{M_2} r_2 + \dots + \frac{m_{N,C_N}}{M_N} r_N \quad (3)$$

$$\text{So } \left| \vec{V}_c \right| = \sqrt{\left(\frac{m_{1,C_1}}{M_1} \right)^2 + \left(\frac{m_{2,C_2}}{M_2} \right)^2 + \dots + \left(\frac{m_{N,C_N}}{M_N} \right)^2} \quad r_i = 1 \quad (4)$$

where $m_{1,C_1}, m_{2,C_2}, \dots, m_{N,C_N}$ are the projected distances on each dimension.

$$\text{Let } r_i = \frac{m_{i,C_i}}{M_i} \quad i \in \{1, 2, \dots, n\} \quad (5)$$

r_i represent the ration of the projected distances, then equation (4) can be rewritten as:

$$\left| \vec{V}_c \right| = \sqrt{r_1^2 + r_2^2 + \dots + r_N^2} = \sqrt{\sum r_i^2} \quad i \in \{1, 2, \dots, n\} \quad (6)$$

Let's study \vec{V}_{diag} and \vec{V}_c first. In order to reduce the complexity of the problem, we redraw the triangle ΔOCD in Figure 3 as follows:

Using vector algebra theory, we can derive the following conclusion from equations (1), (3), (5):

$$\vec{V}_{CD} = \vec{V}_{diag} - \vec{V}_c = (1 - r_1)r_1 + (1 - r_2)r_2 + (1 - r_3)r_3 + \dots + (1 - r_N)r_N \quad (7)$$

In an arbitrary triangle ΔABC with three sides a, b, c and the angle θ between a and b, we derive the following equations using the cosine-sine theorems:

$$c^2 = a^2 + b^2 - 2ab \cos \theta \quad (8) \quad \sin q = \sqrt{1 - \cos^2 q} \quad (9)$$

For the triangle ΔOCF in Figure 4 we can state:

$$dh = \left| \vec{V}_c \right| \sin \theta \quad (10)$$

$$\text{Assume } a = \left| \vec{V}_{diag} \right|, \quad b = \left| \vec{V}_c \right| \quad \text{and} \quad c = \left| \vec{V}_{CD} \right|.$$

Replacing the terms in equation (10) with (8), (9) and further with (2), (6), we get:

$$dh = \sqrt{\left(r_1^2 + r_2^2 + \dots + r_N^2 \right) - \frac{\left(r_1 + r_2 + \dots + r_N \right)^2}{N}} \quad (11)$$

Furthermore, we can infer and get the following definition for the distance from the diagonal:

$$dh = \frac{\sqrt{\left(r_1 - r_2 \right)^2 + \left(r_1 - r_3 \right)^2 + \dots + \left(r_1 - r_N \right)^2 + \left(r_2 - r_3 \right)^2 + \left(r_{N-1} - r_N \right)^2}}{\sqrt{N}} \quad (12)$$

$$\text{with } r_i = \frac{m_{i,C_i}}{M_i} \quad i \in \{1, 2, \dots, n\}$$

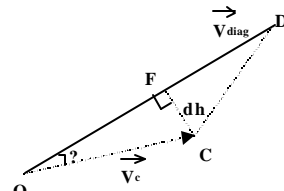


Figure 4: Vector calculations

4.4 Considering Constraints

We now have to derive a trace calculation algorithm based on the previous calculations. However, domain and project-specific development constraints also have to be considered. Some development activities have to be done in a certain order to make sense. These constraints will cause further derivation from the “ideal” diagonal. The definition of these constraints and their addition to the algorithm leads to the following challenges:

- 1) During the calculation of $dh[i]$, there may be situations in which the calculated results are equal, i.e. $dh[i]$ and $dh[j]$ may be the same.
- 2) Some states may be requested to happen before other states that, according to formula (12), should be done first. Let’s assume again a 3-dimensional example: Formula (12) resulted in the following process sequence: $m_{1,1}$, $m_{2,1}$, $m_{3,1}$, $m_{1,2}$, $m_{3,2}$, $m_{2,2}$, $m_{1,3}$, $m_{3,3}$. How can we consider a constraint that, for instance, requires that $m_{2,2}$ has to be done before $m_{1,2}$?
- 3) Some states may have to be done in parallel with each other. Using again the 3-dimensional example from above: How can we express the condition that e.g., $m_{2,2}$ must be performed in parallel with $m_{1,2}$?

In order to solve these problems, we introduce the following concepts:

Weight: This is a constant that is given to each dimension according to the importance of this dimension. For example, for the 3-dimensional framework discussed above, we can give a weight of 3 to the dimension with the most states, a weight of 1 to the dimension with the least number of states and a weight of 2 to the remaining dimension.

Current State: This is the state that the model is supposed to select according to the mathematical calculation based on equations (11) or (12). But whether this state will be selected depends on the constraints defined. We denote the current state as $CS_{i,j}$, i.e., the j^{th} state of dimension i .

Priority State: A state that must be done before the Current State. This is expressed by a constraint on the Current State. Each state is likely to have several priority states. Priority States are denoted as $PC_{k,l}$.

Parallel State: A state that must be done in paralleled with the Current State. This is also a kind of constraint on the Current State. Most states are likely to have several other states in parallel. We denote Parallel States as $PA_{m,n}$.

Priority Constraints: A priority constraint is a constraint with the following relationship:

$PC_{i,j}$ must be done before $CS_{k,l}$ which can be denoted as $PC_{k,l} \mapsto CS_{i,j}$.

The priority constraints of a system can be seen as the set of all Priority States with respect to the current state. It can be written as:

Priority Constraints = $\{ PC_{k,l} \mapsto CS_{i,j} \}$ ($PC_{k,l}$ is a Priority State; $CS_{i,j}$ is the Current State, $k,l,i,j \in \mathbb{N}$)

Priority Constraints have the following properties:

- (1) They are exclusive. If $PC_{k,l} \mapsto CS_{i,j}$ then $CS_{i,j} \not\mapsto PC_{k,l}$
- (2) They must not contain any logical errors. E.g., $PC_{1,3} \mapsto CS_{1,2}$ is not a valid constraint definition.
- (3) They can be linked in order of priority, such as $PC_{3,2} \mapsto PC_{2,3} \mapsto CS_{1,4}$

There are certain constraints that are implicit from the way the states are ordered on one dimension. We call them implicit constraints:

Implicit Constraints = $\{ m_{i,k} \mapsto m_{i,k+1}, i \in \{1,2,\dots,n\}, k \in \{1,2,\dots,M_i-1\} \}$.

If we put the states in the order of $m_{1,1}, m_{1,2}, \dots, m_{1,6}$ along Dimension 1, we imply $m_{1,1} \mapsto m_{1,2}, m_{1,2} \mapsto m_{1,3}$, etc.

Parallel Constraints: This is a constraint with the following relationship:

$PA_{k,l}$ must be done in parallel with $CS_{i,j}$ which can be denoted as $PA_{k,l} \parallel CS_{i,j}$

The parallel constraints of a system can be seen as the set of all Parallel States with respect to their current state. It can be written as:

Parallel Constraints = $\{ PA_{k,l} \parallel CS_{i,j}, i,j,k,l \in \mathbb{N} \}$

Parallel Constraints have the following properties:

- (1) They are permutable, i.e. if $PC_{k,l} \parallel CS_{i,j}$ is a parallel constraint, then $CS_{i,j} \parallel PC_{k,l}$ must also be a parallel constraint.
- (2) They must be without logical errors. For instance, $PA_{2,2} \parallel CS_{1,2} \parallel PA_{2,4}$ is not a valid parallel constraint.
- (3) They can export (or derive) some priority constraints. For instance, if $PA_{3,5} \parallel CS_{2,4}$ is a parallel constraint, then $PC_{3,4} \mapsto CS_{2,4}$ must be a priority constraint.

Generally speaking, all priority constraints have the following relationship:

$\{ PC_{k,l} \mapsto CS_{i,j} \}$

Here are some examples: ($n,m > 1$)

1 to 1 $PC_{1,3} \mapsto CS_{2,2}, PC_{3,3} \mapsto CS_{4,2}$

1 to n $PC_{1,3} \mapsto CS_{2,2}, PC_{1,3} \mapsto CS_{4,2}$

n to 1 $PC_{1,3} \mapsto CS_{2,2}, PC_{3,3} \mapsto CS_{2,2}$

n to n $PC_{1,3} \mapsto CS_{2,2}, PC_{3,3} \mapsto CS_{2,2}, PC_{3,3} \mapsto CS_{2,4}$

With these definitions, we can now address the problems that we mentioned at the beginning of this section:

Problem 1) has been solved by giving each dimension a weight according to its importance and the requirements of the system domain. This helps decide in which direction to go in situations where a step in one direction results in the same distance from the diagonal as a step in another direction. For example, if we give dimension 1 a weight of 3, dimension 3 a weight of 2, and after calculating $dh[i]$ we get $dh[1]=dh[3]$, then we should choose dimension 1 due to the higher weight of this dimension.

Problem 2) has been solved by checking the priority constraints after calculating $dh[i]$. In order to determine the direction of the next step, we not only have to calculate the minimum $dh[i]$ but we also have to consider any possible constraints that may be defined. If there are no constraints for the current state, then we can choose direction i according to the minimum $dh[i]$.

Problem 3) can be solved by considering the parallel constraints after having checked the priority constraints.

Having provided solutions to these three problems, we can now define the algorithm:

1. If $n=1$:

The process follows a single line.

2. If $n \geq 2$ do the following:

(1) Define all n dimensions together with their weights and their states.

(2) Do the following steps 1) to 3) until all states in all dimensions have been traversed:

1) For $i=1$ to n do

try to go one step in the direction of each dimension and calculate dh using formula (12) at each step (we use $dh[i]$ to represent the calculated result of dh when a step is attempted in the direction of dimension i) and store all the values $dh[i]$ as well as the reference information for each dimension i .

2) **If** the minimum value of $dh[i]$ is unique

record the information of the route with the minimum dh (i.e., the dimension number i), then go to 3)

Else

among those routes that have the same minimum dh , select the one with the highest weight and record the information (i.e., the dimension number i), then go to 3)

3) **If** there are priority constraints

take the priority state of the current state as selected state. Go one step in the direction of the dimension that was selected and keep the information of the dimension and other relevant information in an array.

Else

go one step in the direction that we selected in 2) and keep the information of the dimension and other relevant information in an array.

4) **If** there are parallel constraints

deal with the parallel constraints (store the corresponding information which will be required for the display)

3. Display the result.

4.5 Implementation and Simulation

We are working on a prototype that allows the automatic generation of an initial process model after the user defined the dimensions of the problem domain, the states on each dimension and the constraints between these states (see Figure 5). The process engineer can then look at the outcome of the process calculation and further refine in an iterative manner the process model using the tool. The current prototype already provides some of the anticipated functions, such as process model definition and modification, process definition and modification, process calculation and display, etc. The prototype as well as the underlying framework are still under construction and require further modifications and optimization.

5 Conclusions and Future Work

This paper presented a new process-engineering framework. The novelty of this framework is its genericity and its ability to aid the process developer in defining a fine-grained, domain-specific process model. The kind and number of dimensions as well as the constraints are specific to the domain and the project. The current

prototype implementation of the framework suggests an initial process model derived from the dimensions, their states and the constraints, which have to be provided by the tool user.

The next step of this work requires a refinement of the prototype. This work will most-likely cause some optimization of the framework. Once the prototype is completed, we will work on a medium-scale example. Although the validity of the approach has already been shown by the RATS framework, we want to investigate the feasibility of the generic framework for different domains. Once a set of process models exist, we want to look at developing a repository of best-practice process models. This process model database will allow us to store process models together with a description of the domain for which they are suitable. These models are likely to be medium-grained since they require further adaptation to the company-specific procedures.

References

- [1] Pressman R.S. Software Engineering – A Practitioner’s Approach, McGraw Hill, 2001
- [2] Pohl K. Process-Centered Requirements Engineering, John Wiley & Sons, 1996
- [3] Pohl K. The Three Dimensions of Requirements Engineering: A Framework and its Application, International Journal of Information Systems, 19, 3, pp. 243-258, 1994
- [4] Eberlein A. Requirements Acquisition and Specification for Telecommunication Services, PhD Thesis, University of Wales, Swansea, UK, 1997
- [5] Magedanz T. and Popescu-Zeletin R. Intelligent Networks, Thomson Computer Press, 1996
- [6] ITU-T Z.100 Recommendation Specification and Description Language (SDL), ITU, 1993
- [7] Pohl K. The Three Dimensions of Requirements Engineering, Proceedings of the 5th International Conference on Advanced Information Systems Engineering, 1993
- [8] Wynekoop, J.D. and N.L. Russo. System Development Methodologies: Unanswered Questions and the Research-Practice Gap, Proceedings of the 14th International Conference on Information Systems, 1993
- [9] Rolland C. and N. Prakash. From Conceptual Modelling to Requirements Engineering, Annals of Software Engineering, 10, pp.151-176 ,2000

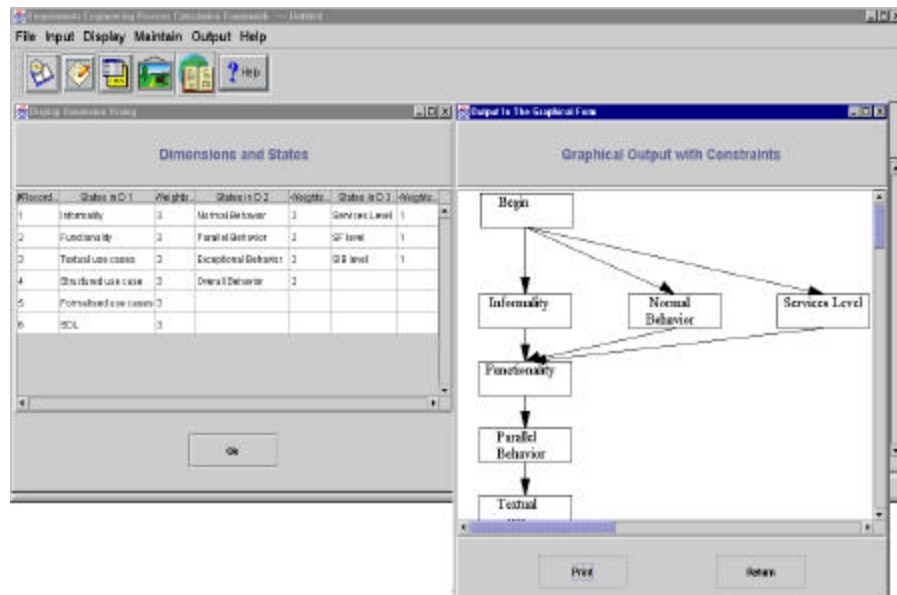


Figure 5: Screen shot of the REFM prototype