Technical Report

Nr.: FIN-008-2008

The Causal-Based Software Process Modelling

Karsten Richter, Reiner R. Dumke

Arbeitsgruppe Softwaretechnik

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Nr.: FIN-008-2008

The Causal-Based Software Process Modelling

Karsten Richter, Reiner R. Dumke

Arbeitsgruppe Softwaretechnik

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 10 MDStV):

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Reiner Dumke
Postfach 4120
39016 Magdeburg
E-Mail: dumke@ovgu.de

http://www.cs.uni-magdeburg.de/Preprints.html

Auflage: 81

Redaktionsschluss: 20.10.2008

Herstellung: Dezernat Allgemeine Angelegenheiten,
Sachgebiet Reproduktion

Bezug: Universitätsbibliothek/Hochschulschriften- und
Tauschstelle

The Causal-Based Software Process Modelling

Karsten Richter, Reiner R. Dumke

University of Magdeburg, Dept. of Computer Science, Postfach 4120,
D-39016 Magdeburg, Germany
{richter,dumke}@ivs.cs.uni-magdeburg.de,
http://ivs.cs.uni-magdeburg.de/sw-eng/us/

Contents

Abstract

This Technical Report includes a causal-based modelling of software measurement processes in order to clarify the real situations in the software metrics application field. A first overview about existing semantic network approaches shows the problems and possible benefits using these formal techniques in the software engineering area. The definition and extension of the causal modelling using causal networks helps to understand the relationships between the different measurement artefacts and their causalities. The description of first applications of our approach demonstrates the empirical reasoning of the software improvement processes in a holistic manner.

1 Semantic Networks in Software Engineering

1.1 Semantic Networks

In a first approximation, we consider the semantic connections between the software metrics based on the empirical characteristics in a lose manner. The method for visualisation should be semantic networks. *Semantic networks* are symbolic-based methods of reasoning [Pearl 2000]. They are a kind of knowledge representation schemes involving nodes and links between nodes. The nodes represent objects or concepts and the links represent relations between nodes. A simple example of a semantic network considering the software development involvements are given in the following figure based on [Laird 2006].

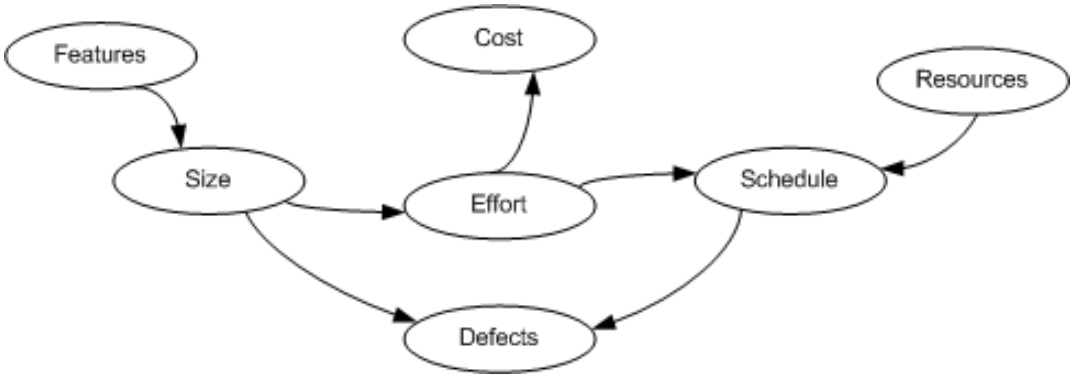


Figure 1: Semantic network of software development dependencies

1.2 Examples in Software Product and Process Descriptions

Typical kinds of links are ‘is-a’ or ‘has’. Another simple semantic network is given in Figure 2 and shows the relationships of a software project.

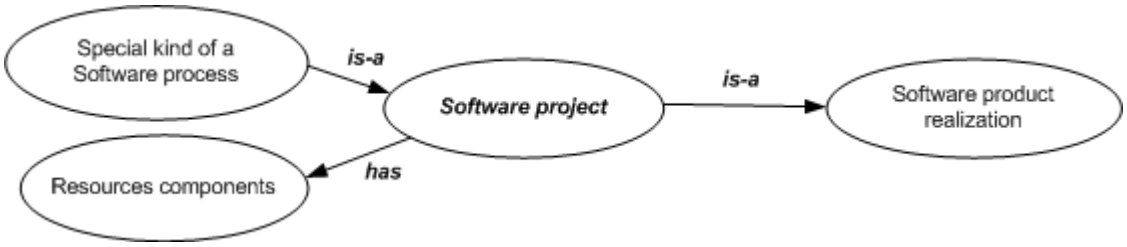


Figure 2: A simplified semantic network of a software project

A well-known example of a semantic network is given by [Furgeson 1998] without any comments in the Figure 3 which includes the software process evaluation standards and methods.

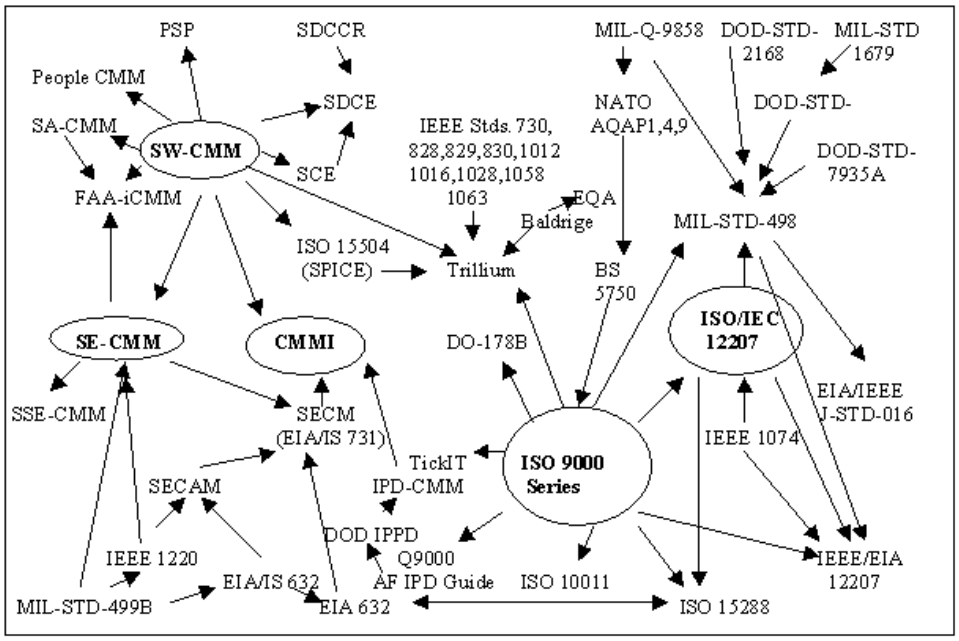


Figure 3: Dependencies of software evaluation methods and standards

This network allows us to consider the different parts and sources of the well-known software process evaluation standards and techniques. In order to use the semantic networks for the representation of the *empirical knowledge* in software engineering, we will consider an empirical interpretation of the network in Figure 2, described in Figure 4.

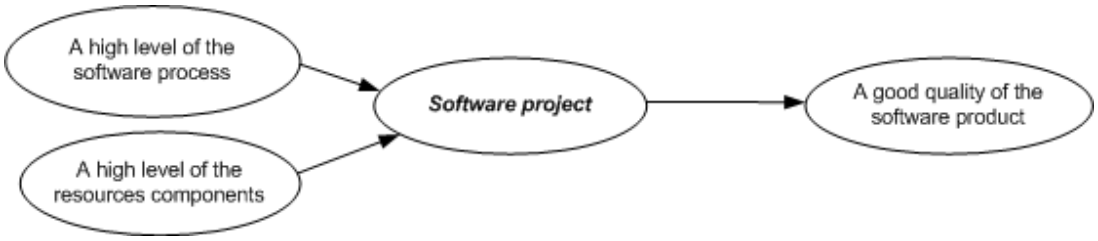


Figure 4: Example of a semantic network including empirical aspects

Addressing the basics of the project management, CMMI considers the following components for the management of the IT processes [SEI 2002] (where QPM stands for Quantitative Project Management, IPM for Integrated Project Management, IPPD for Integrated Product and Process Development, RSKM for Risk Management, and ISM for Integrated Supplier Management).

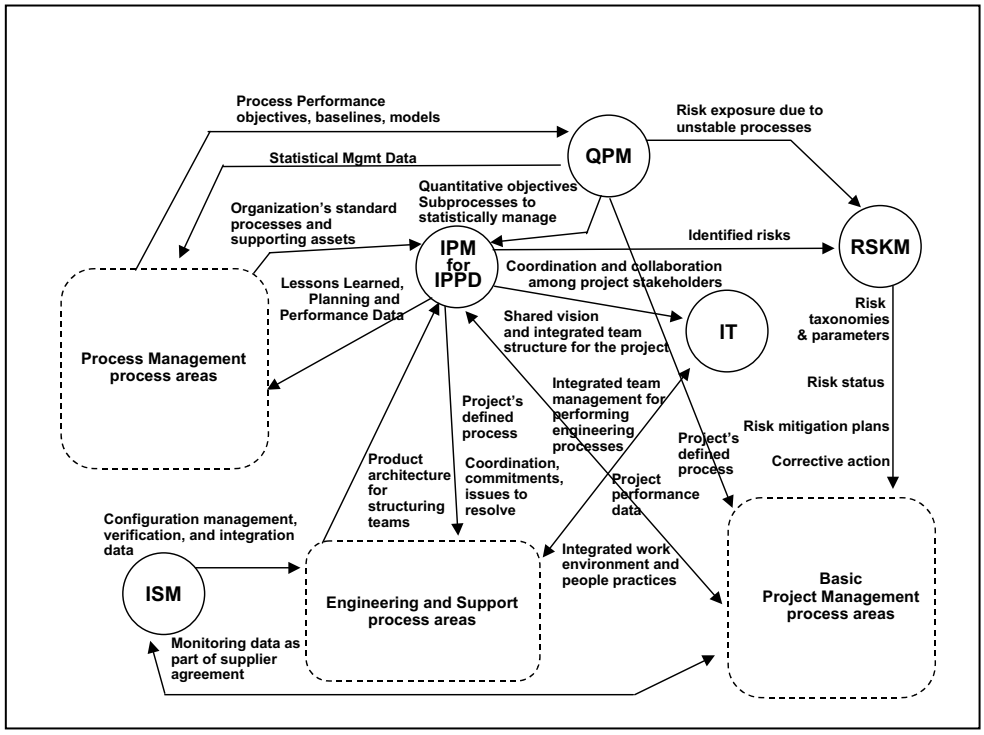


Figure 5: The CMMI project management process areas

Of course, this *semantic network* is helpful in addressing the management aspects which are necessary for quality processes. But this description is mostly rough and implicit.

In [Dumke 2006c] we have shown a level-based approach in order to use software process-related semantic networks for process description and explanation. We have been started with a definition of software process modelling using a description from Wang and King [Wang 2000] as following: *"Software process model (SPM) is a process of a model system that describes process organisation, categorization, hierarchy, interrelationships, and tailorability."* A complete presentation of these aspects which we must consider in the management of software processes is shown in following simplified chart.

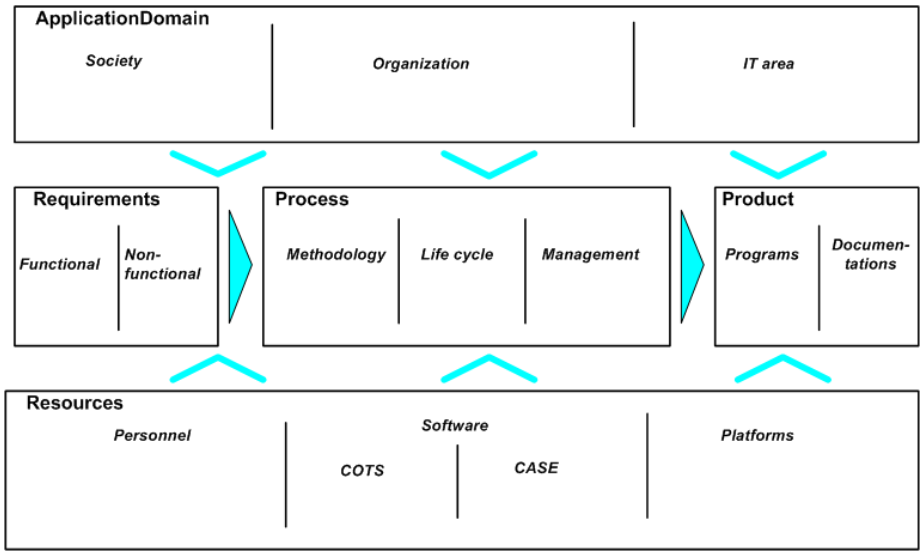


Figure 6: A holistic presentation of the software process involvements

Furthermore, the following definition by Wang and King [Wang 2000] is helpful as a kind of improvement of our SPM. *"The software process establishment (SPE) is a systematic procedure to select and implement a process system by model tailoring, extension, and/or adaptation techniques."* The general software process model (SPM) is qualifying to a software process establishment (SPE) as a concrete IT adaptation. We use this consideration in order to improve our SPM with more details. The next figure includes a refinement of the general structure described in figure 6 and the derived process indicators and criteria [Dumke 2006c].

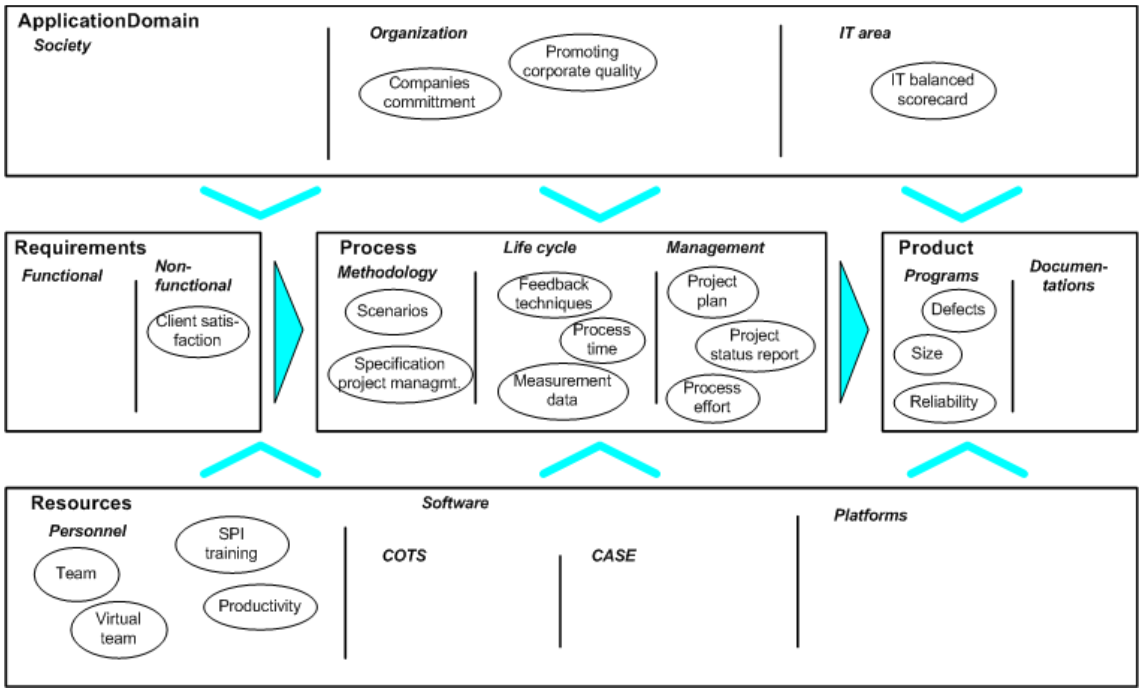


Figure 7: The SPE based process network

Note that our SPE refinement of the SPM based on the scientific investigation leads us to the nodes of our semantic network approach only. But we could establish the following situation:

- *The aspects of process evaluation do not involve the area of software (as COTS or CASE), platforms and society aspects like cultural background and marketplace.*

- *Simple process management criteria are not addressed to the product documentations as is the typical situation in the agile development approaches.*

Considering the improvement context of software processes, the following definition by Wang and King [Wang 2000] is helpful again: *"A process improvement model (PIM) is an operational model that provides guidance for improving a process system's capability by changing, updating, or enhancing existing processes based on the findings provided in a process assessment."* We will choose some laws and process principles and rules [Dumke 2006c] and derive the following simple kind of a process related semantic network.

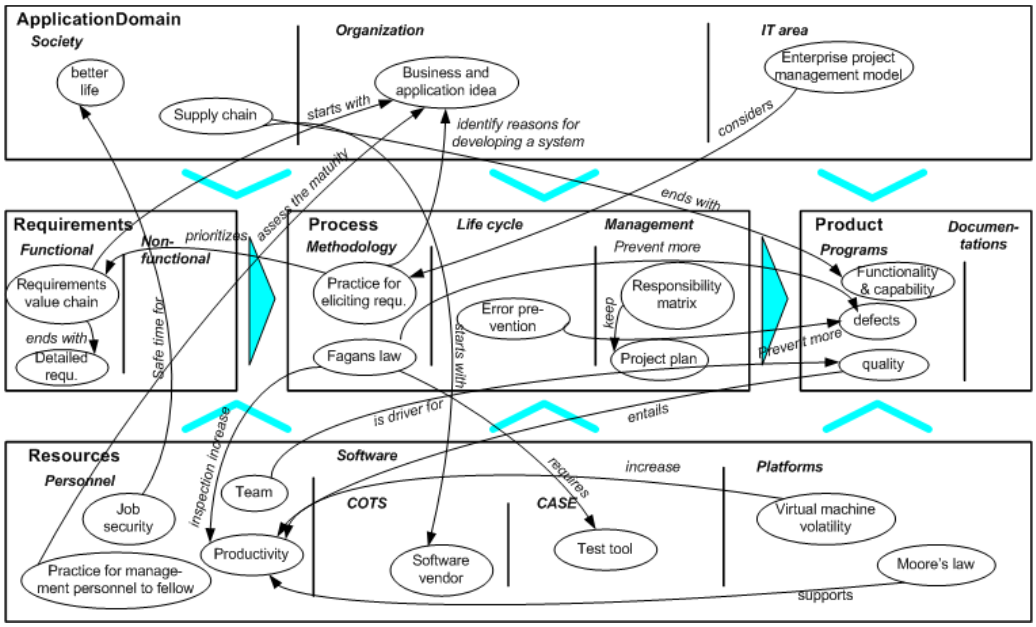


Figure 8: The PIM based process related semantic network

Analyzing this simple kind of process related semantic network, we can establish the following:

- *Based on the considered approaches in the literature, we can see a better coverage of the involved software process related areas*

- *The derived semantic network is helpful for identifying quality improvements principles and laws for process controlling*

- *Considering the roots of this semantic network gives a good orientation for the essential process aspects that would lead to best process improvement and management.*

In order to qualify our process semantic network by quantitative experience, we consider an empirical (based) process model. Therefore, the following definition by Wang and King [Wang 2000] is helpful again: *"An empirical process model (EPM) is a model that defines an organized and benchmarked software process system and best practices captured and elicited from the software industry."* As rules of thumb, experiments and case studies will chose some typical kinds which include a lot of such experience [Dumke 2006c]. The following figure 9 shows the involvement of such empirical studies in our kind of semantic network.

Of course, our network presentation is very simple but suggests some of the following short characteristics:

- *The relationships between the network components are quantified and can support the process management in this manner*

- *Using the kinds of semantic network applied before, we can identify some tasks for improving the process relationships for higher software process control.*

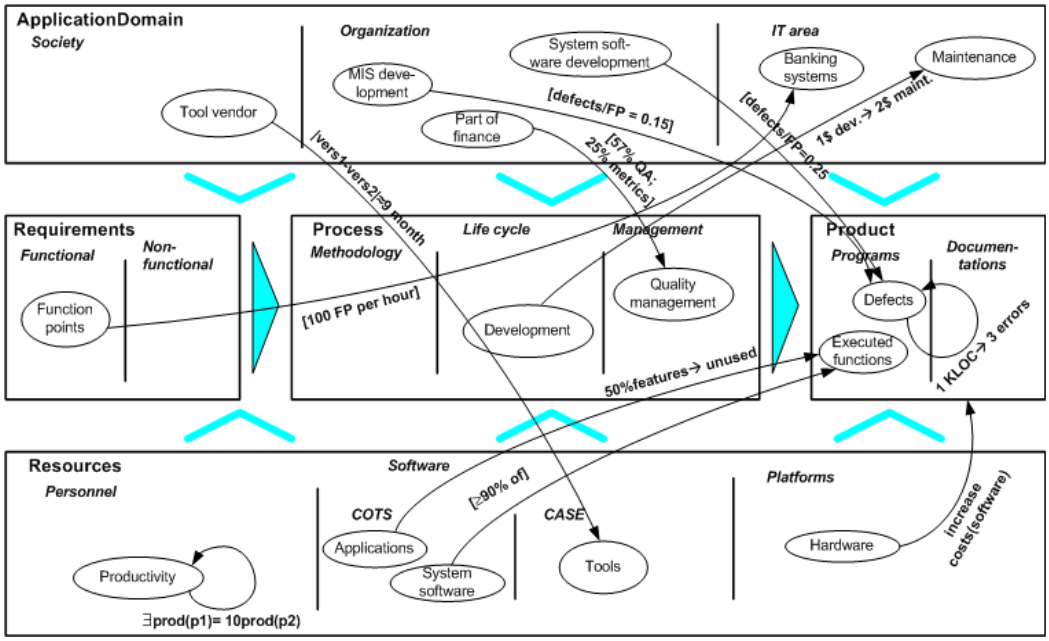


Figure 9: An EMP based process related semantic network

Using metrics leads us to the highest level of process related semantic network because of quantified relationships between different process involvements. We will start with a simple definition again addressing to the process measurement as following: *"Software process measurement model (SPM) is a model that defines an organized and benchmarked software process system and applies process metrics and measures with quantitative measurement characteristics."* In order to derive a semantic network based on these characteristics we will consider the experiences [Dumke 2006c]. We will choose only some of these process metrics in the figure 10 in order to better see the principal structuring of process related involvements.

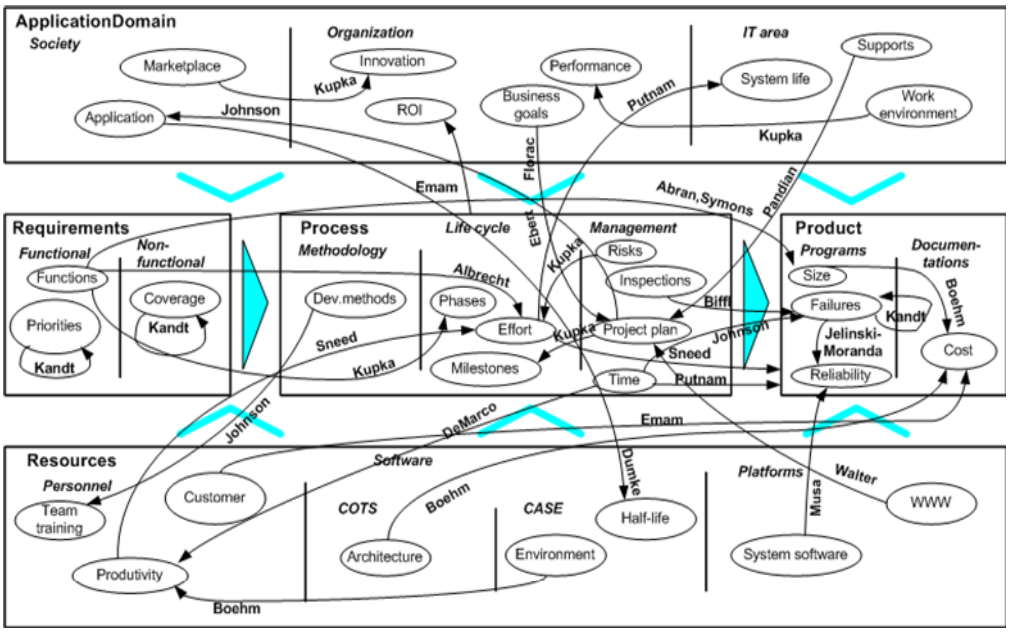


Figure 10: A SMP based process related semantic network

This kind of semantic process related network helps us in the following manner:

- *The semantic network shows the process components which are involved in the quantitative process management.*
- *It is possible to find out the components which are necessary to measure in order to apply the chosen process metrics*
- *Finally it is possible to estimate the level of quantitative management as a fundamental basis of process management level.*

In this section we considered the software process involvements in an explicit (shown) manner by using different detailed semantic networks which should guarantee the following:

- *In every level of measurement-based process management we can see the missing parts or the parts of weak kinds of evaluation.*
- *The semantic relationships between the process related aspects should be explained definitely and could be analyzed through their appropriateness in a changing IT area.*

1.3 The Approach of Fenton et al.

Fenton at al. creates an approach by using probabilistic networks in order to better explain the software defect prediction [Fenton 2001]. The basic semantic network in this approach is the following.

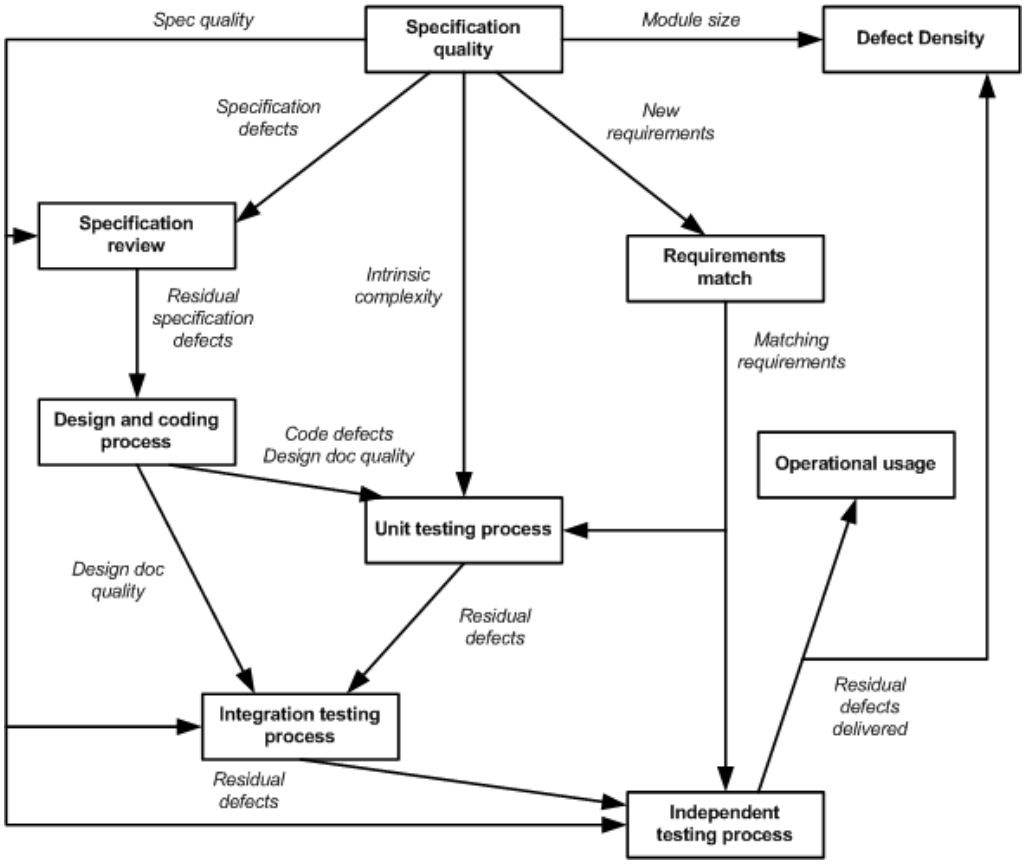


Figure 11: A defect relationship semantic network

This resulting network takes account of a range of product and process factors from throughout the lifecycle of a software module. A derived semantic network considering the specification quality mainly was build in the following manner.

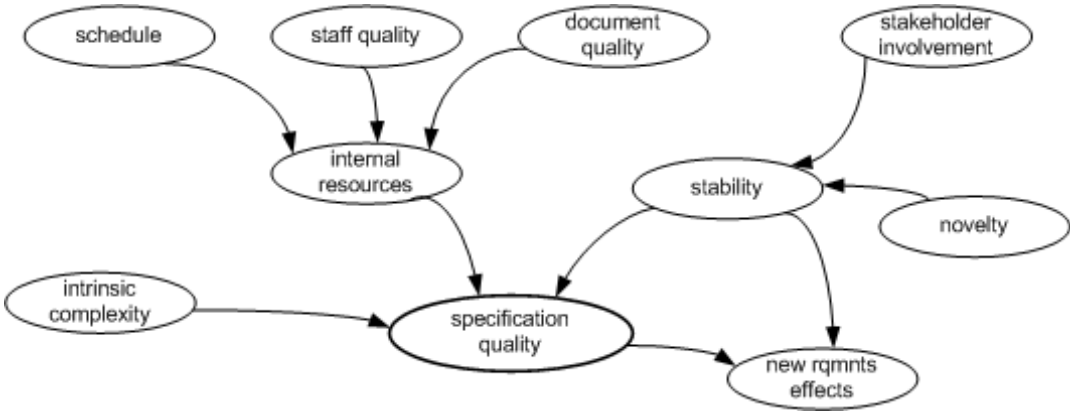


Figure 12: A quality relationship semantic network

Using special ordinal attributes for the different kind of relationships leads to a better characterization and explanation of the specification qualities.

1.4 Semantic Hierarchies and Levels

Considering the detailed descriptions of the different levels of semantic network based process description can be ad to a situation in order to build structures about these levels. Such structures could be built as

- ***Network hierarchy*** as define of refinements and subparts of the different nodes or edges of the created semantic network
- ***Network levels*** as summarize the different relationships constructed in the different levels of process involvements based on improvements, rules of thumb and metrics.

Therefore, our derived figures of semantic process networks show different kinds of management levels shown in the following figure [Dumke 2006c].

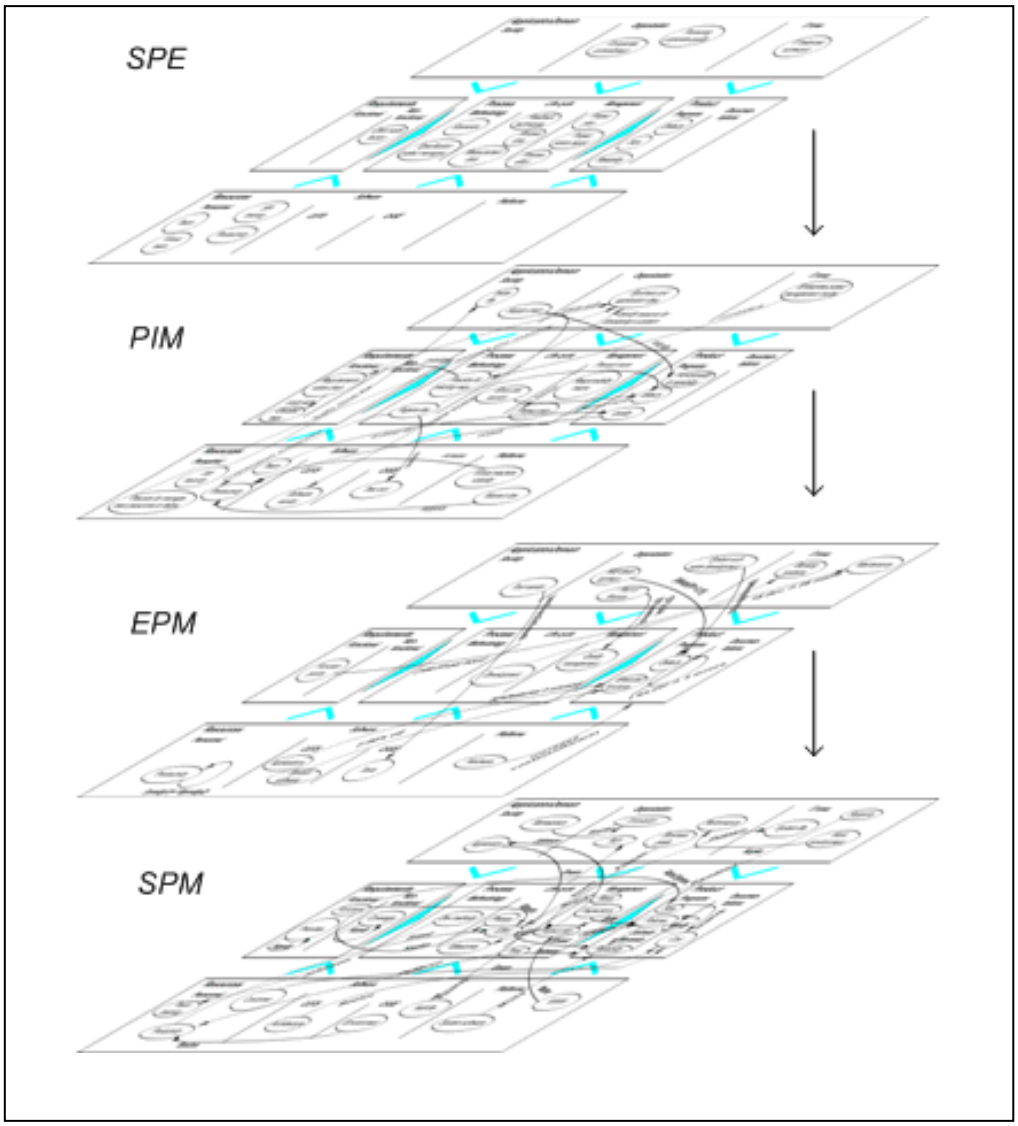


Figure 13: The different process semantic network levels

This visualization suggests that process management should consider all levels of process evaluation identified as *software process establishment (SPE)*, principles, laws and rules by *process improvement models (PIM)*, experimentation or rules of thumb by *empirical process models (EPM)* and process metrics by a *software process measurement model (SPM)*) in order to cover the process areas as much as possible.

On the other hand, our investigations and considerations have also clarified the current situation of process measurement using an appropriate method of structuring of (all) the software process involvements.

2 Causal Models and SE-based Causal Networks

2.1 Causal Model Definition by Pearl

We will start with a general description of causalities summarizing in a model view. A ***causal model*** is a triple $M = \langle U, V, F \rangle$, where [Pearl 2000]

- U is a set of background variables, (also called exogenous), that determined by factors outside the model;
- V is a set $\{V_1, V_2, \ldots, V_n\}$ of variables, called endogenous, that are determined by variables in the model – that is, variables in $U \cup V$; and
- F is a set of functions $\{f_1, f_2, \ldots, f_n\}$ such that each f_i is a mapping from (the respective domains of) $U \cup (V \backslash V_i)$ to V_i and such that the entire set F forms a mapping from U to V. In other words, each f_i tells us the value V_i given the values of all other variables in $U \cup V$, and the entire set F has a unique solution $V(u)$. Symbolically, the set of equations F can be represented by writing

  $$v_i = f_i(pa_i, u_i),\ i=1, \ldots, n,$$

  where pa_i is any realization of the unique minimal set of variables PA_i in $V \backslash V_i$ (connoting parents) sufficient for representing f_i. Likewise, $U_i \subseteq U$ stands for the unique minimal set of variables in U sufficient for representing f_i.

This causal model could be used in order to do the following considerations and further derivations [Pearl 2000]:

- *Submodel of M:* A submodel M_x of M is the causal model $M_x = \langle U, V, F \rangle$ where $F_x = \{ f_i : V_i \notin X \} \cup \{X = x\}$.
- *Effect of Action:* Considering X as a set of Variables in V of the causal model M and x as a particular realization of X, then the effect of action $do(X = x)$ on M is given y the submodel M_x.
- *Potential Response:* The potential response of Y to action $do(X=x)$, denoted $Y_x=(u)$, is the solution for Y of the set of equations F_x where $X, Y \subset V$.
- *Counterfactual:* The counterfactual sentence "The value that Y would be obtained, had X been x" is interpreted as denoting the potential response $Y_x=(u)$.
- *Probabilistic Causal Model:* A probabilistic causal model is a pair $\langle M, P(u) \rangle$ where M is a causal model and $P(u)$ is a probability function defined over the domain of U.
- *Causal Diagram/Network:* Causal network visualization could be based on a directed acyclic graph (DAG) where the nodes are the considered aspects U and V and the edges implements the causalities F.

The following figure 14 shows a simple example of a causal diagram or network for dynamic process controlling [Pearl 2000].

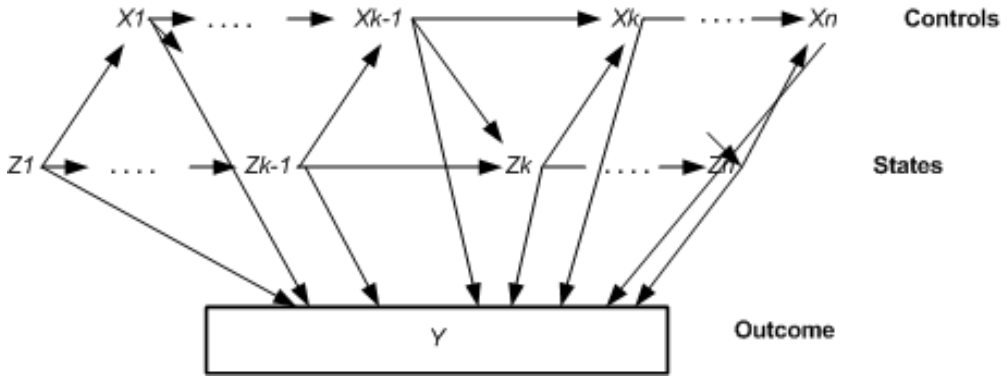


Figure 14: An example of a causal diagram for process controlling

We will use this formal background in order to define our causal process network approach described in the section below.

2.2 Software Engineering Causalities

We will give some examples of ***causalities in software engineering*** (cited from [Davis 1995]) in following:

- *General principles*: "Productivity and quality are inseparable." "Communicate with the customers/users." "Change during development is inevitable." "Technique before tools." "Most cost estimates tend to be low." "What applies to small systems does not apply to large ones." "A system that is used will be changed."

- *Software specification:"*Prototypes reduce risk in selecting user interfaces." "Requirement deficiencies are the prime source of project failures."

- *Software design:* "Design for change, maintenance, and errors." "Hierarchical structures reduce complexity." "Architecture wins over technology." "Software reuse reduces cycle time and increases productivity and quality."

- *Software coding and testing:* "Don't test your own software." "Don't integrate before unit testing." "Instrument your software." "Don't errors personally." "Online debugging is more efficient than offline debugging."

- *Software measurement:* "Measurements are always based on actually used models rather than on desired ones." "Empirical results are transferable only if abstracted and packaged with context." "Measurement requires both goals and models."

- The following cause-and-effect diagram shows the classification of some aspects of the software process [Florac 1999].

Another kind of causalities is described as ***software engineering laws.*** The following figure shows the variety of intentions of such laws addressed to the different parts of software development. The detailed content of these laws is described in [Endres 2003].

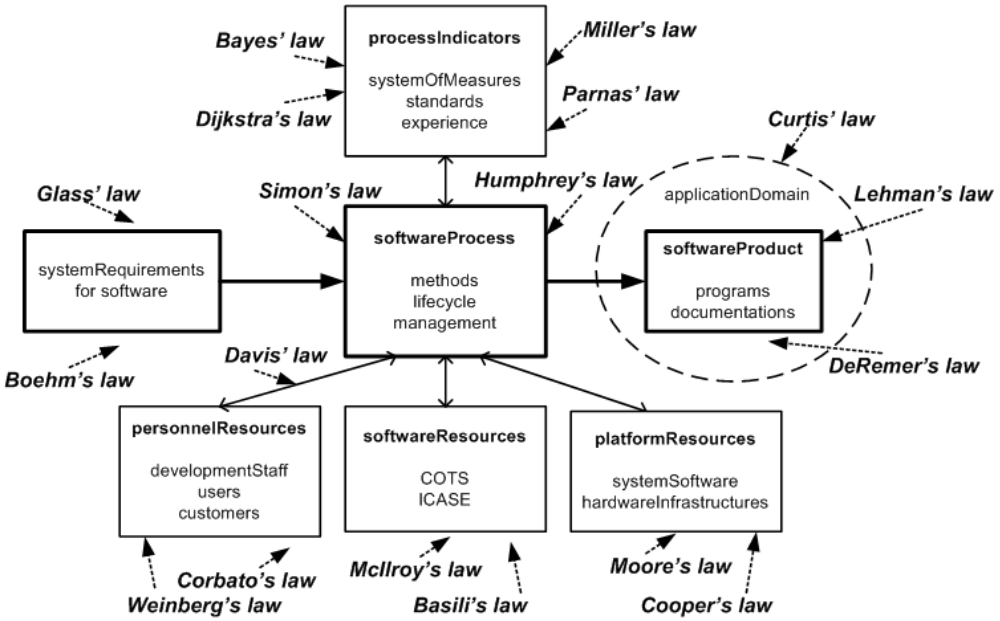


Figure 15: Intentions of chosen software engineering laws

2.3 Causal Networks in Software Engineering

Causal networks as a special kind of semantic networks are very expressive in order to see or analyze the relationships between process activities, areas and indicators in a logical manner. Typical results of such a modelling are ([Fenton 2001], [Pearl 2000])

- The consequence of process activities to other ones involving different quality characteristics like correctness, completeness etc.
- The repercussion of the chosen approaches for process evaluation and improvement
- The overview about strong and weak process connections in order to keep quality improvements
- The application of (causal) model-based principles in order to reduce the process complexity and involvements.

In a general manner a causal network "is a directed acyclic graph arising from an evolution of a substitution system, and representing its history" [Weisstein 2006]. The process evolution involves causal relationships between events, states, entities, objects, artefacts etc. which could be based on a special kind of empirical reasoning. In following we will consider special kinds of causal networks in process analysis, measurement, and evaluation. We discuss the following level of *causal process networks* including the kinds of causalities [Dumke 2006a]

(D) *"has an influence to"* (e. g. failure propagation)

(I) *"involve an improvement of" (*e. g. based on maturity models)

(Q) *"keep the quality of"* (e. g. quality reasoning for connected components).

Note other causalities exists between process components like *"leads to failures in", "decrease the", "has a feedback to", "has side effects from", "involves the ripple effect to"* etc. which are not considered in this paper. As helpful form of presentation we use the general schema of figure 6 (see above) of software process involvements derived from [Dumke 2006c] (see also Deek 2005], [Kenett 1999] and [Wang 2000]). These process-related areas are the context of our causal process network consideration and investigation explicitly.

Causal Process Networks of Dependencies: The dependencies in software processes could be related to anyone and anything. We will start with a first simple example considering the *five core metrics* of Putnam [Putnam 2003]. The definition of the relationships between these metrics leads us to the following simple *causal process network of dependencies*.

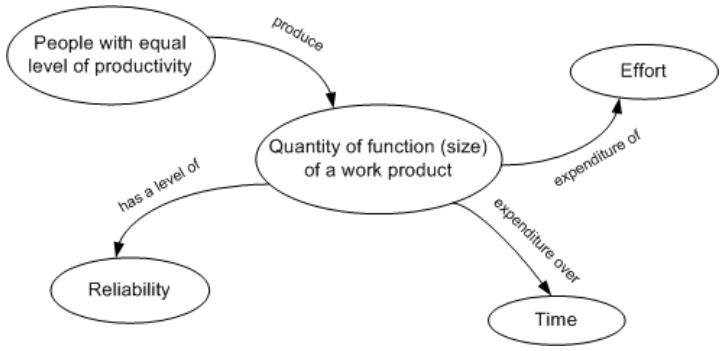


Figure 16: Causal process network of the Five Core Metrics of Putnam and Myers

Another kind of causality as an *impact trace* is defined in [Emmerich 2007] which considers the different relationships between any approaches, publications, standards and technologies. The following figure shows an example of this kind of causal network.

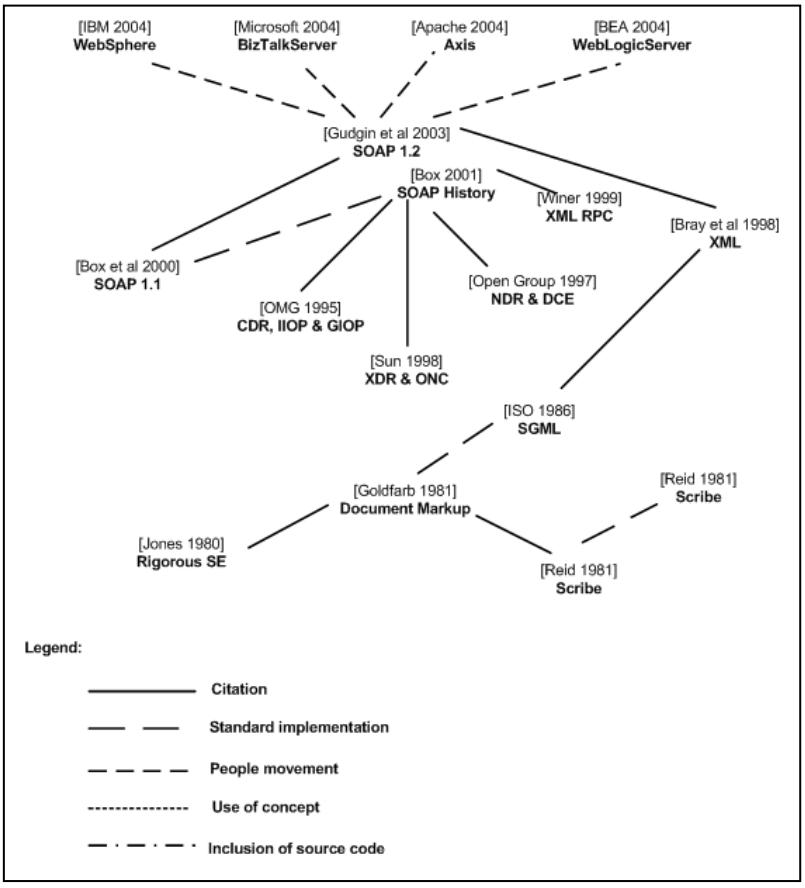


Figure 17: Impact trace of the definition of SOAP

The network of individual cause-effect relationships (so-called base mehanism) of the system dynamics in a generic process from Müller and Pfahl is given in following [Müller 2008].

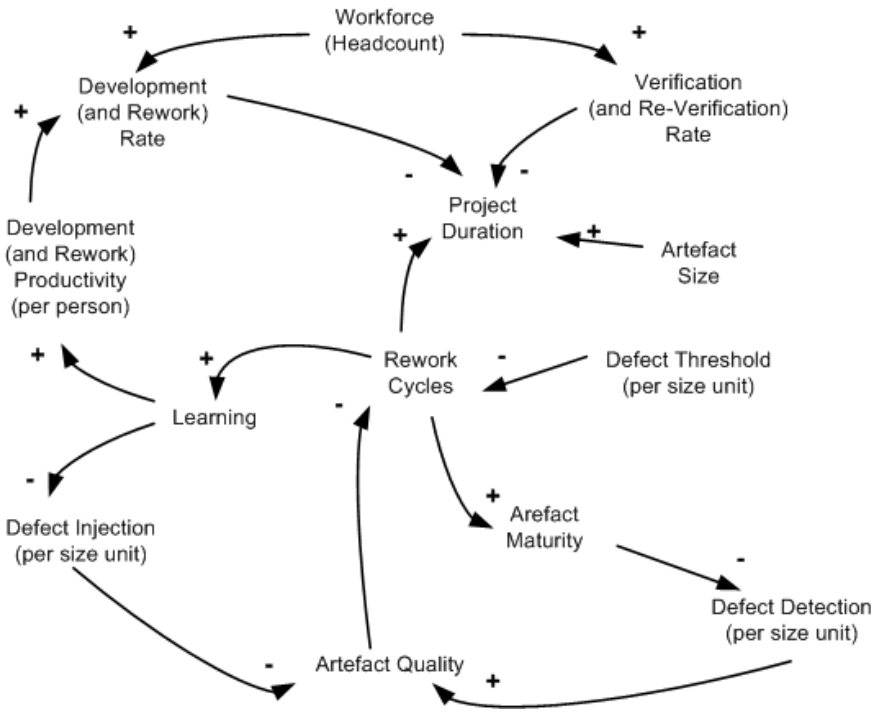


Figure 18: Base mechanism as causal network

Using our general process involvements schema, the dependencies will be defined on the basis of [Dumke 2005a], [Dumke 2005b], [Emam 1998], [Garcia 2005], [Haywood 1998], [Lecky-Thompson 2005], [Putnam 2003], [Richter 2005], [Royce 1998], [Wang 2000], and [Zettel 2001]. Adapting this experience can refine the different process aspects shown in the figure 19. We have chosen a rough description and only few of the (causal) relationships in order to keep the clearness of the principles and intentions. The causal process network of dependencies shows

(D1) The existing relationships which are necessary to consider in the different kinds of process aspects managing the changing, migration, upgrading, and evolution

(D2) The different kinds of relationships like *"determines", "requires", "motivates", "forms", "administrates", "performs"* etc. which lead to different kinds causal analysis and causal chain reasoning

(D3) The general network characteristics like direction of the nodes, singularities, and component/chain areas of software process involvements.

Note the shown background of the software process involvements keeps a holistic view of the process dependencies which can be lead o higher level causalities discuss in the following sections.

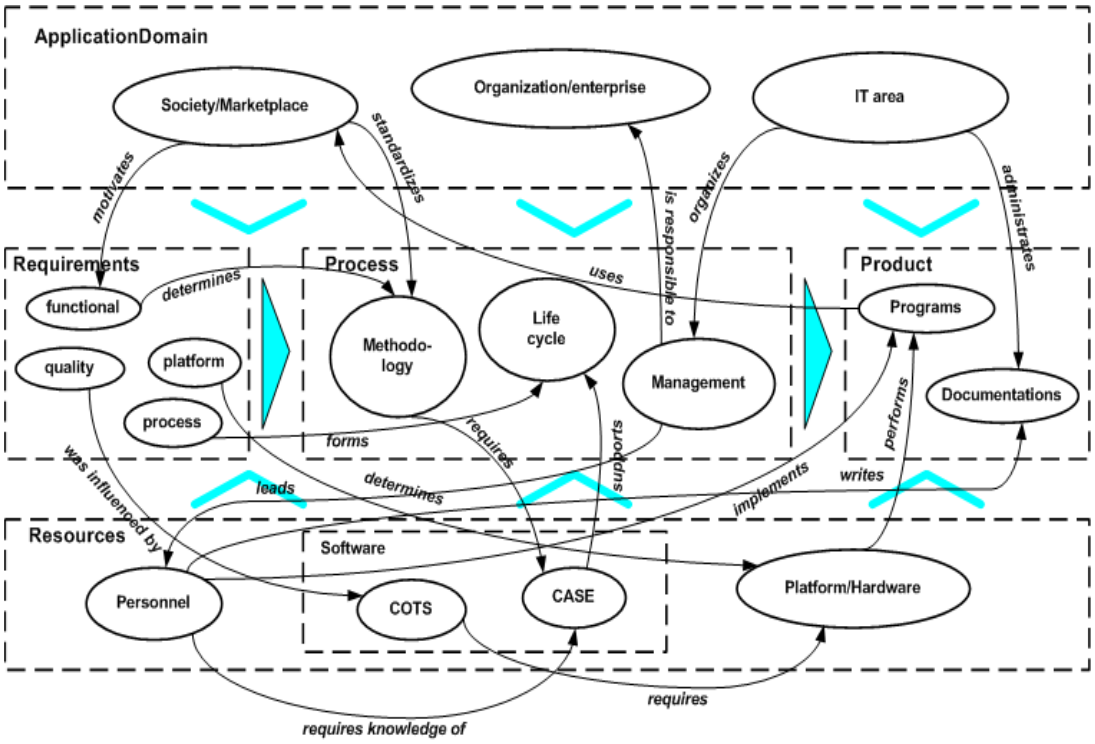


Figure 19: Causal process network of dependencies

Causal Process Networks of Improvements: In the next level of causalities in process related networks we will consider the characteristic of improving. We will start again with a very simple *causal process network of improvements* example. The *Software Process Improvement and Capability dEtermination (SPICE)* is defined as an ISO/IEC standard TR 15504 [Emam 1998]. The principles of the process assessment of SPICE are given in the following semantic network [SPICE 2006] achieving the improvement criteria.

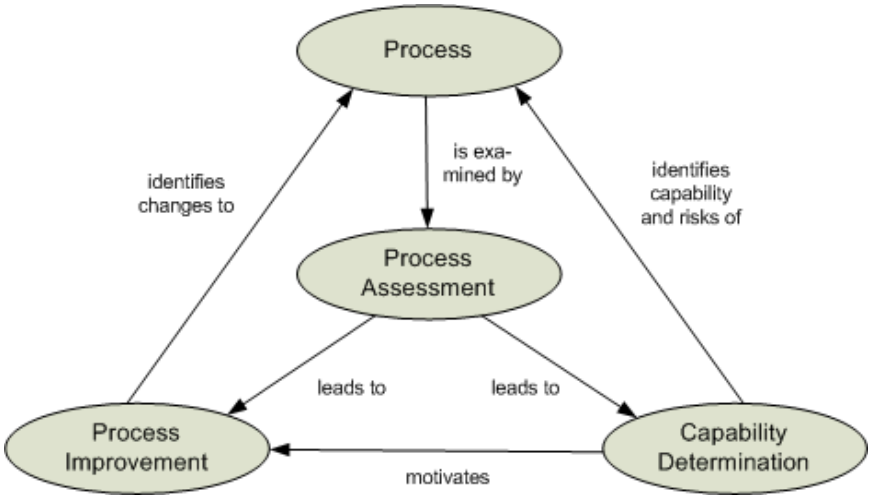


Figure 20: Causal process network of the SPICE approach

Based on the idea of process improvement, a lot of *maturity models (MM)* were defined and implemented in order to classify different aspects of software products, processes and

resources. Some of these maturity evaluation approaches are described in the following table. Their detailed description is given and/or referenced in [April 2005] and [Braungarten 2005].

Table 1: Chosen maturity models

Model	Description	Model	Description
PEMM	Performance Engineering MM	**CM3**	Configuration Management MM
TMM	Testing Maturity Model	**ACMM**	IT Architecture Capability MM
ITS-CMM	IT Service Capability MM	**OMMM**	Outsourcing Management MM
iCMM	Integrated CMM	**PM2**	Project Management Process Model
TCMM	Trusted CMM	**IMM**	Internet MM
SSE-CMM	System Security Engineering CMM	**IMM**	Information MM
OPM3	Organizational Project Management MM	**PMMM**	Program Management MM
OMM	Operations MM	**PMMM**	Project Management MM
M-CMM	Measurement MM	**IPMM**	Information Process MM
SAMM	Self-Assessment MM	**CPMM**	Change Proficiency MM
UMM	Usability MM	**ASTMM**	Automated Software Testing MM
ECM2	E-Learning CMM	**LM3**	Learning Management MM
WSMM	Web Services MM	**ISM3**	Information Security Management MM
eGMM	e-Government MM	**TMM**	Team MM
EVM3	Earned Value Management MM	**SRE-MM**	Software Reliability Engineering MM
WMM	Website MM	**EDMMM**	Enterprise Data Management MM
DMMM	Data Management MM	**S3MM**	Software Maintenance MM

In order to derive a causal process network of improvements keeping all the process involvements, we will consider the some of the MM's above that lead us to the following figure.

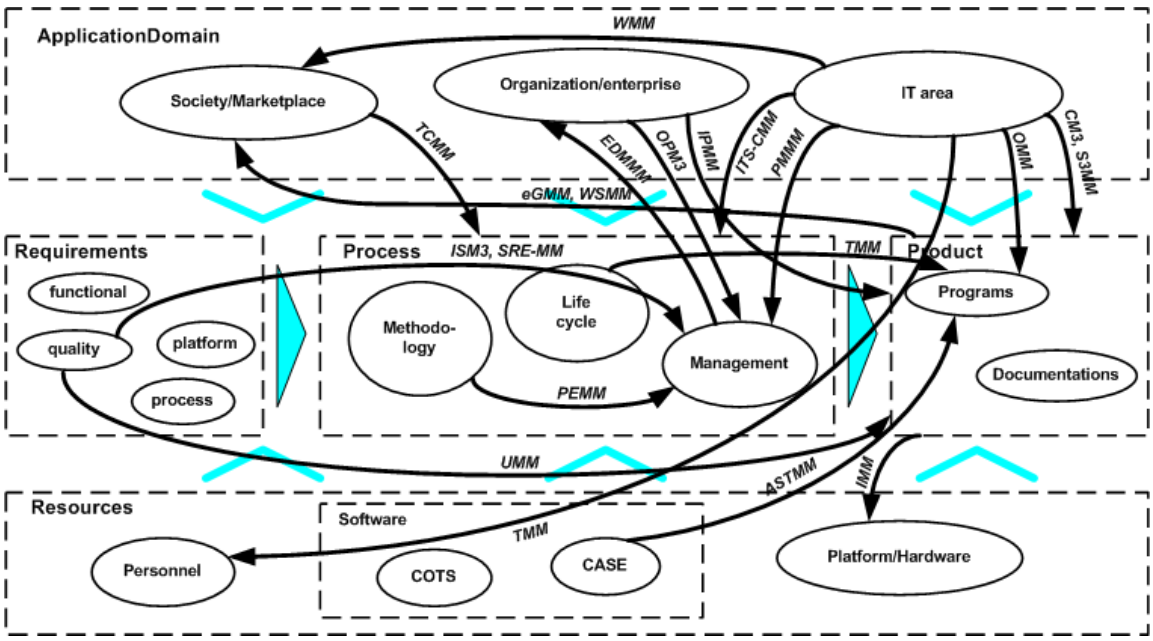


Figure 21: Causal process network of improvements

The bold arrows visualize the maturity-based improvement characteristics. The causal process network of improvements shows

(I1) The sources and goals of the different improvement models, methods and technologies which could be described in the context of the organizational level

(I2) The components which are not under any activities of (quality) improvements; they could be identified and are the basis for strategic evolvements

(I3) The possibility of adding some attributes which shows the current step of maturity or (quality) level achievements.

Furthermore, improvement activities could also be some techniques or technologies like structured programming, information hiding, coupling reduction, and aspect concerning which we don't haven consider in this paper.

Causal Process Networks of Quality Assurance: Finally, we will discuss the causal process network modelling considering the quality assurance aspects. We start again with a simple kind of *causal process network of quality assurance* based on the *Personal Software Process (PSP)* in the following figure 22 [Humphrey 2000].

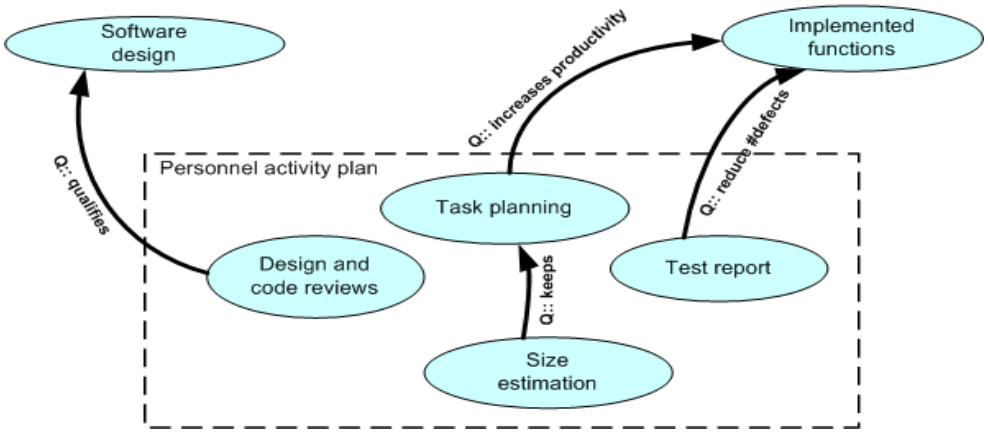


Figure 22: Causal process network considering PSP

In order to show general quality assurance connections we will select quality approaches like *Six Sigma* (6σ) and *ITIL* (described in [Dumke 2006b]) and some of quality principles summarized in [Keyes 2003] as

- Redmill's quality principles in the management of software-based development projects (RP)
- Corbin's methodology for establishing a software development environment (CM)
- Shetty's seven principles of quality leaders (SP)
- Zachmann's quality framework of development complex systems (ZP)
- Kemayel's controllable factors in programmer productivity (KF)

Therefore, we obtain one of the following versions of charts where the kind of quality assurance is given after the Q mark.

The causal process network of quality assurance shows

(Q1) The kinds of involved quality assurance methods and approaches in the current process analysis and reasoning

(Q2) The process involvements including under quality assessment and control in a given context of IT areas

(Q3) The possibilities of network analysis, evaluation, and transformation keeping intended quality levels.

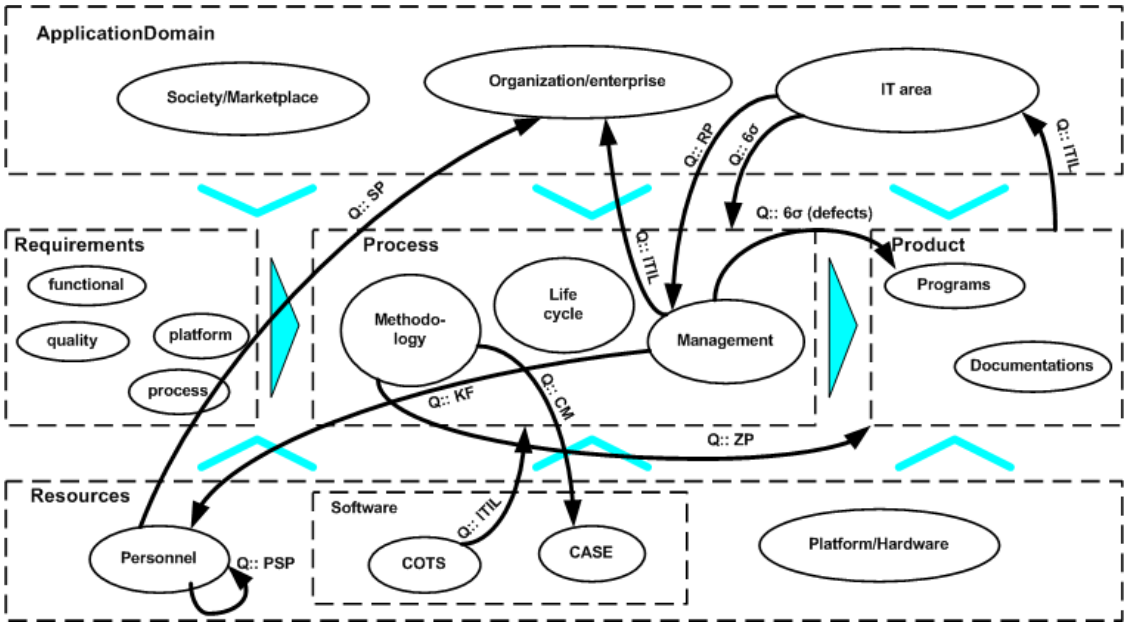


Figure 23: Causal process network of quality assurance

Some essential results and open problems could be identified as basics for future investigations in process measurement and evaluation. The causal network as a special kind of semantic networks is helpful in order to discuss the causal analysis and reasoning in a non formal manner of essential aspects in order to keep some of the requirements of the CMMI level five.

2.4 Reasoning in SE Causal Networks

The following approach was defined in [Dumke 2001] and is addressed to the possibilities of reasoning in semantic/causal networks. This leads us to the necessity to define the new/special contents of the semantic of the nodes and their links in a semantic network. Hence, we define the following kinds of nodes and links:

- A software measure or metric should be presented as an object/concept. The empirical evaluation of this metric are divided in the kinds of quantitative evaluation as *measured, estimated, predicted* or *delivered* (from measured) and in the kinds of qualitative evaluation as *nomination* and *classification.* On the other hand, the evaluations can be based on measured attributes only, can include the measurement unit or can be based on attribute evaluation only. Therefore, we will define the following notations as different nodes they are described in Figure 24.

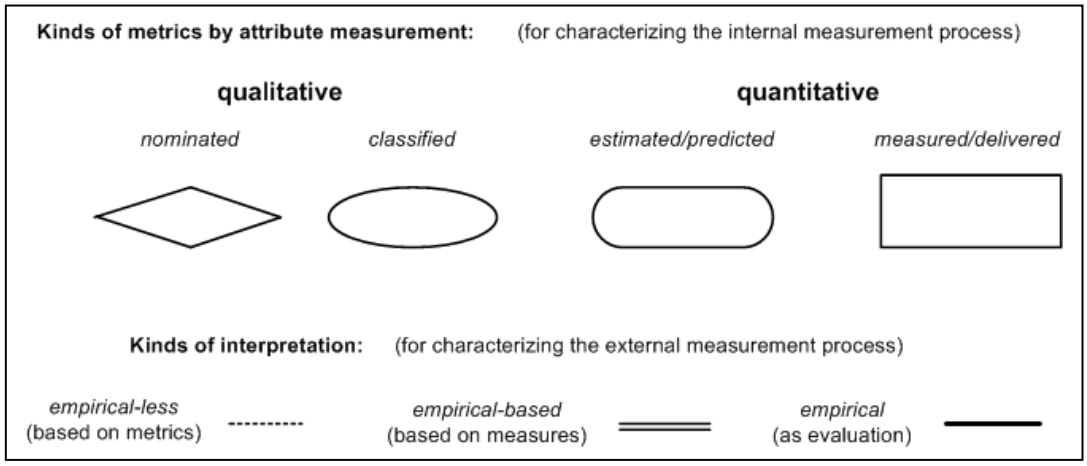


Figure 24: Nodes in the semantic network for describing the empirical criteria

- The links between the nodes are the assumed or proved semantic (empirical) relationships between the metrics. The symbols of these links are given in the Figure 25.

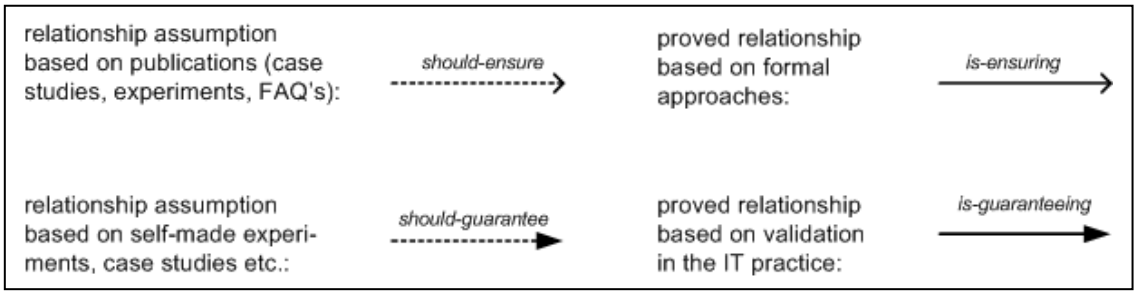


Figure 25: Kinds of links between empirical evaluated software metrics

As an example for using the symbolization above, we will give a simple metrics-based semantic network in Figure 26. This network includes two empirical-based measured components and assumed relations to other evaluated and considered aspects. We have assigned the customer as resource because of the role of the customer in the product application.

- The causal network can be *interpreted* to obtain a set of empirical criteria for software metrics applications. Valid interpretations include

  - ***Analysis:*** The semantic network can be analysed using the well-known graph interpretations such as connectivity, predecessors and successors, and singularity. Considering the graph in Figure 6, we obtain some of the analysis results as

    - *Connections:* the *CMM level* influences the *kind of process,* the *product functionality* and the *customer satisfaction* in the described manner,

- *Singularity:* the high quality of code relating to the *code readability* has no advantages in order to keep some other software development characteristics.

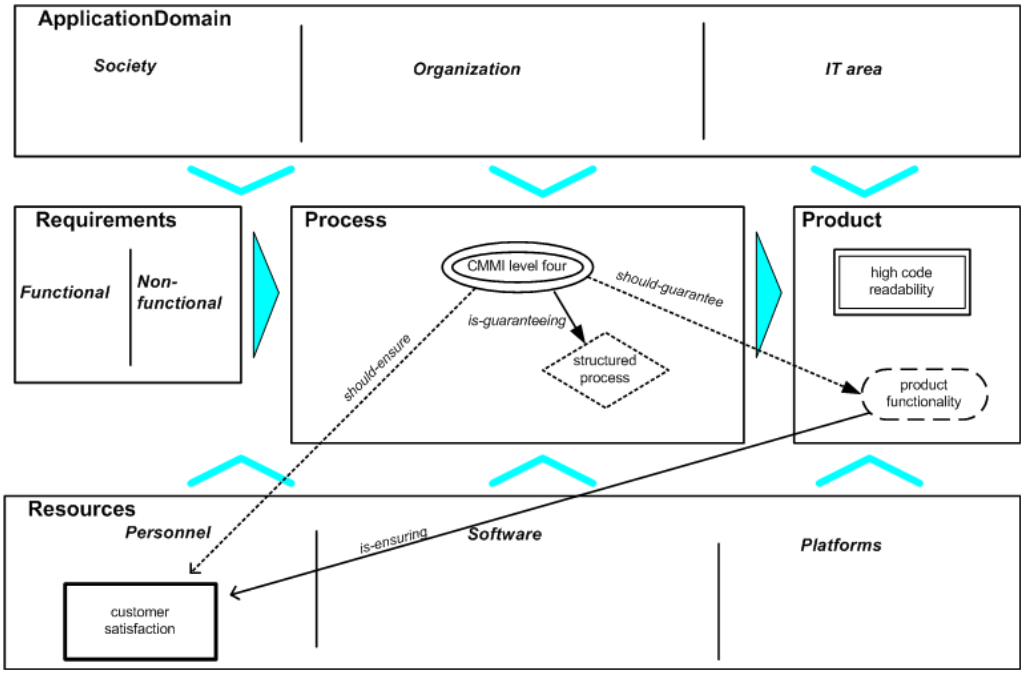


Figure 26: An example of a metrics-based semantic (causal) network

- ***Evaluation:*** The source of the evaluation of the semantic network should be the empirical characteristics and should allow define the empirical level of the measurement process relating to the different kinds of nodes and links. These characteristics are only related to the static aspects of the software measurement process both internal and external form of the process.

  Some of the special types of evaluations of the semantic network are the following:

  - the number of the components with a higher measurement level relating to the components with a lower measurement level for potential reduction of the semantic network,
  - the percentage distribution of the different nodes in the network in order to characterise the different parts of the measurement levels,
  - the number of the different kinds of arrows in the network which determine the possibility of the transformation of the empirical characteristics between the different components (nodes).

- ***Reduction:*** The interpretation of our semantic network includes also a kind of an operation. We define the following *rules for transformation* in Table 2.

Table 2: Rules of nodes transformation in the semantic network

Semantic connection *(it was assumed that the measurement level of A is higher than the level of B)*	***Rules of transformation***
A → B	Transform the component B to the level of the source A
A → B	Transform the component B to the next measurement level of A
A ⇢ B	Transform the component B to the level of A under special *process conditions*
A ⇢ B	Transform the component B to the next measurement level of A under *process conditions*

In the network of Figure 26, we can use the empirical-based aspect of the CMM level measurement to apply the defined transformations rules. The result of this reduction is given in Figure 27.

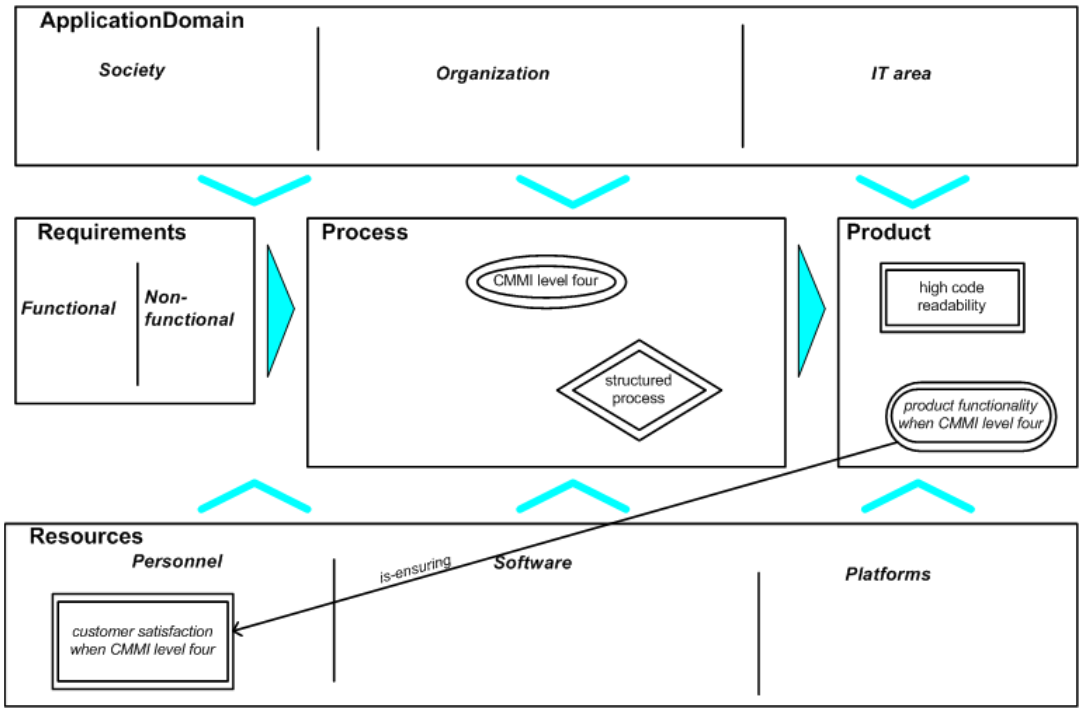


Figure 27: The use of transformations in the metrics-based semantic network

Note that we have 'consumed' the arrows between the components through the application of the transformation rules. On the other hand, the interpretation of the metrics-based empirical network must consider whether of nodes constitute a continuous measurement or a periodic assessment. This leads to a time-depended kind of these networks. In this manner, we obtain a reduced network from Figure

27 which is valid until the next CMM evaluation/certification is necessary. This temporary valid network is shown in Figure 28.

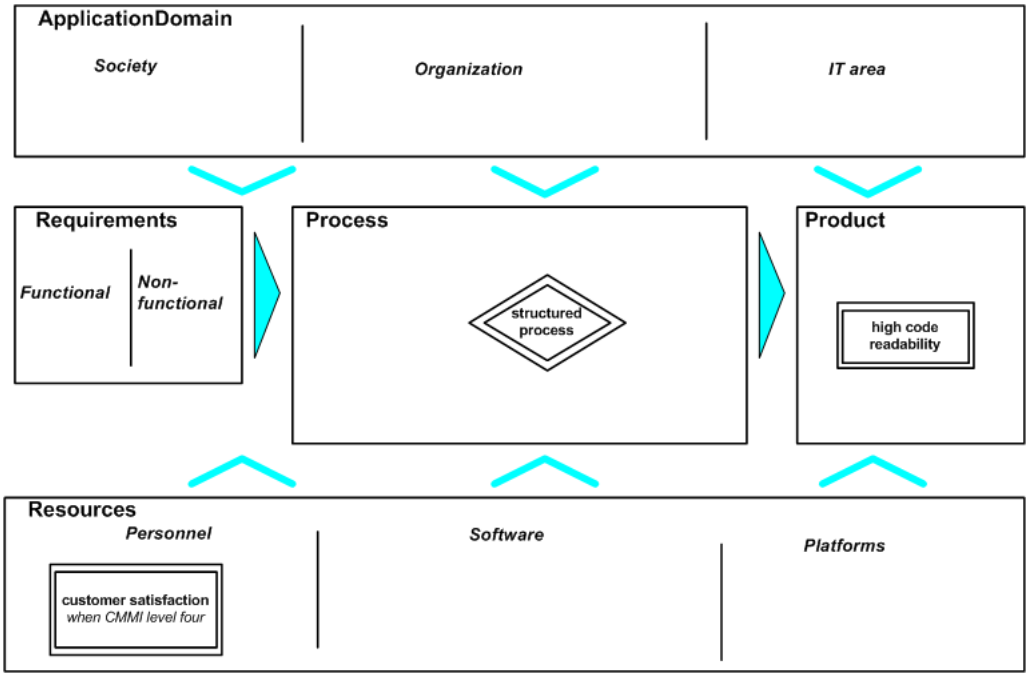


Figure 28: The reduction of the metrics-based semantic network

The semantic network for presenting the empirical characteristics in the software measurement is an appropriate tool for general ***description*** of the measurement process ***explicitly***. We will give some examples in this paper below. This step includes the *instrumentation* or the *automation* of the measurement process by tools. It requires analyse the algorithmic character of the software metrics and the possibility of the integration of tool-based 'control cycles' in the software development or maintenance process.

We will call the metrics tools as *CAME (Computer Assisted software Measurement and Evaluation) tools* [Dumke 1996]. In most cases, it is necessary to combine different metrics tools and *techniques related to the measurement phases*.

Techniques for the tool-based software measurement are the consideration of attributing, extension, composing and are addressed to the presentation model (as language, as parameterised component or verbal description). Therefore, we extend the symbolisation of the metrics or measures characterisation in the manner of Figure 29.

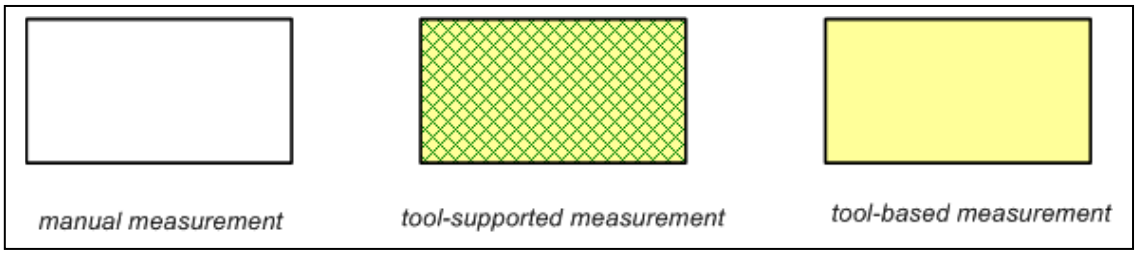


Figure 29: Tool-support characterisation of the empirical metrics description

Based on a special assumption, we can obtain a tool support related to our example described in Figure 28 in the kind of Figure 30.

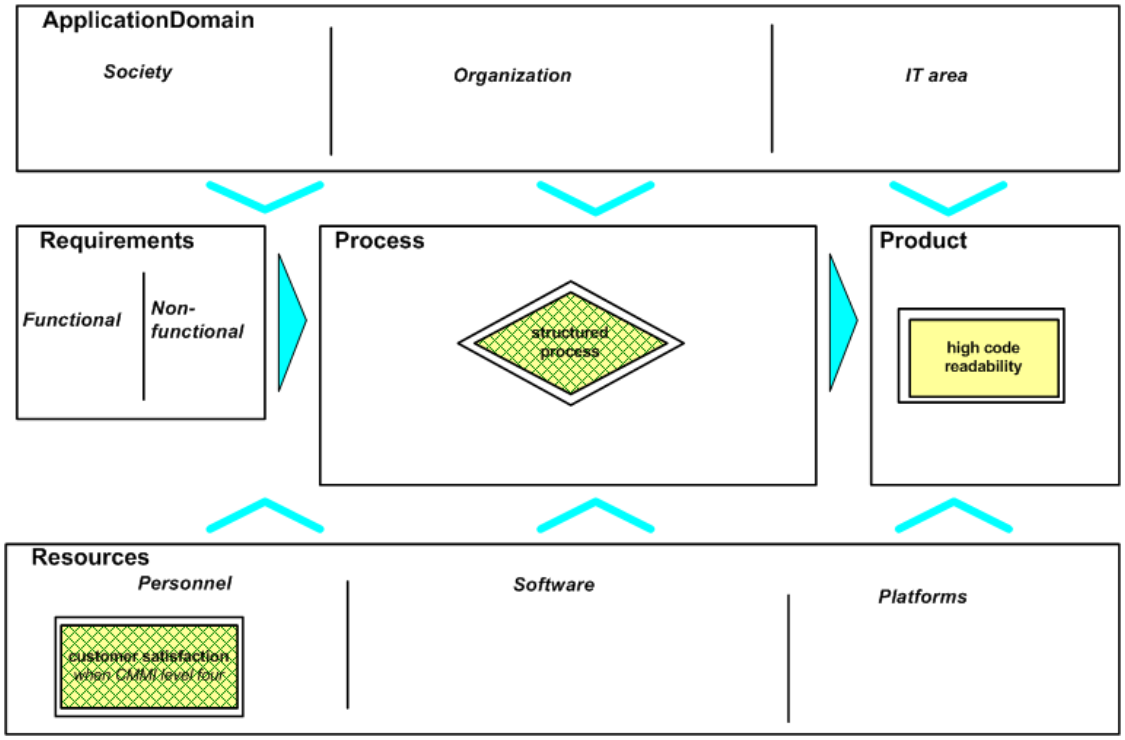


Figure 30: Tool-support example of software measurement

Therefore, we consider a method in order to improve the process level based on the semantic network of the empirical characteristics. The general steps of this method are the following:

Software process improvement

- *building of the semantic (causal) network* based on the empirical evaluation of all of the chosen metrics,
- *characterising of the semantic network* by analysing and evaluation in order to prepare the empirical-based semantic network for the following step of network reduction,
- *application of the reduction rules* of the semantic network in order to obtain a conditioned empirical description of the software measurement process.

The empirical characteristics of an ISO 9126 evaluation are shown in Figure 31. This causal network also shows that reductions are possible under specific conditions only.

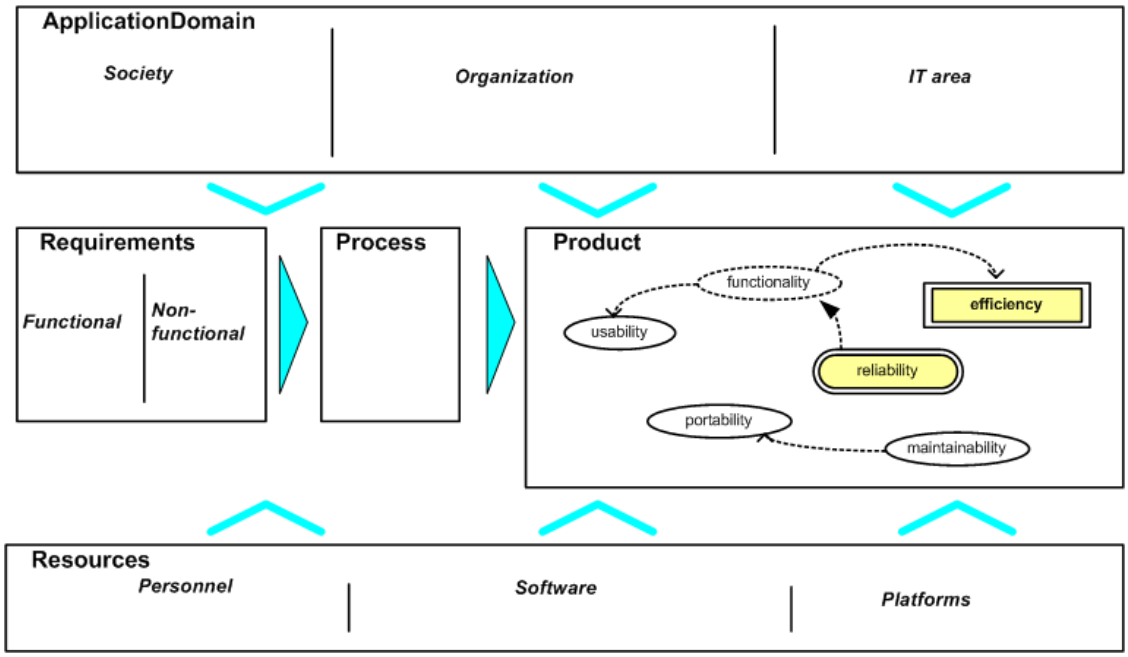


Figure 31: Empirical network based on the ISO 9126 standard

The high level of some quality aspects is based on the general characteristics of the CMM level four as "quantitative project management". Note, that the determination of the CMM level itself is based on an evaluation founding on questionnaires. The intention was to have a measurement process, which covers not only all the process aspects. They should also includes the product measurement and the resource evaluation. The special kind of CMM leads to influences in product and resource measurement. Hence, we can establish the situation which is shown in Figure 32.

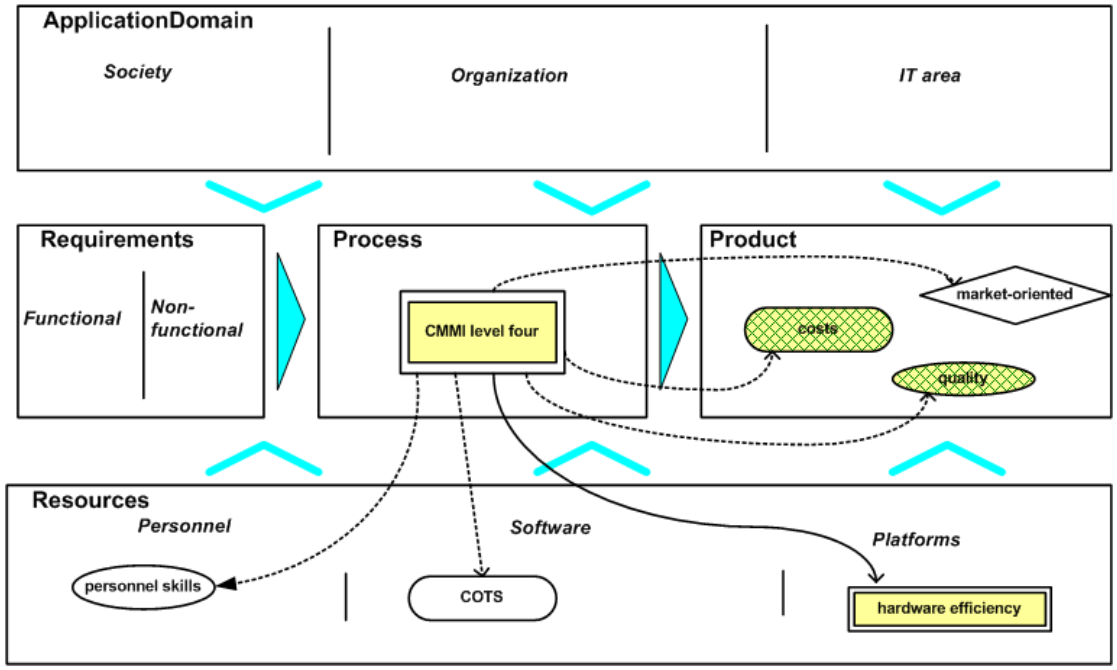


Figure 32: Simplified empirical network based on the CMMI

The connections between the empirical nodes in Figure 32 also do not allow application some of our reduction rules. On the other hand, the background of all the used metrics leads to investigations of the personal *process* aspects in order to observe improvement during the software development and maintenance in Figure 33.

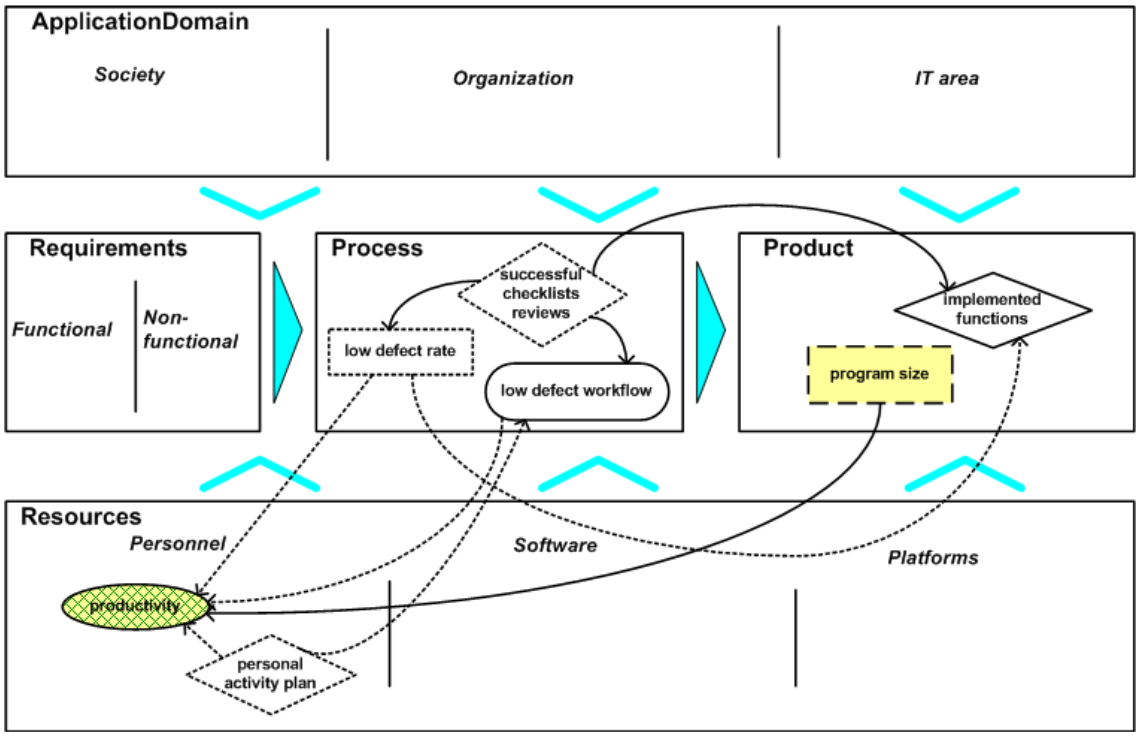


Figure 33: Empirical network based on the PSP approach

Note that the chosen examples are very simple in order to demonstrate the principles and possible applications of this kind of visualization and exploration.

3 Causal Network-Based Process Model (CNPM)

3.1 The Software Process Modelling

In order to define process cautions and their analysis for process improvement we must decide the kind of process description and visualization. A first simple kind of software process description can be used by the declarative model [Dumke 2006b]

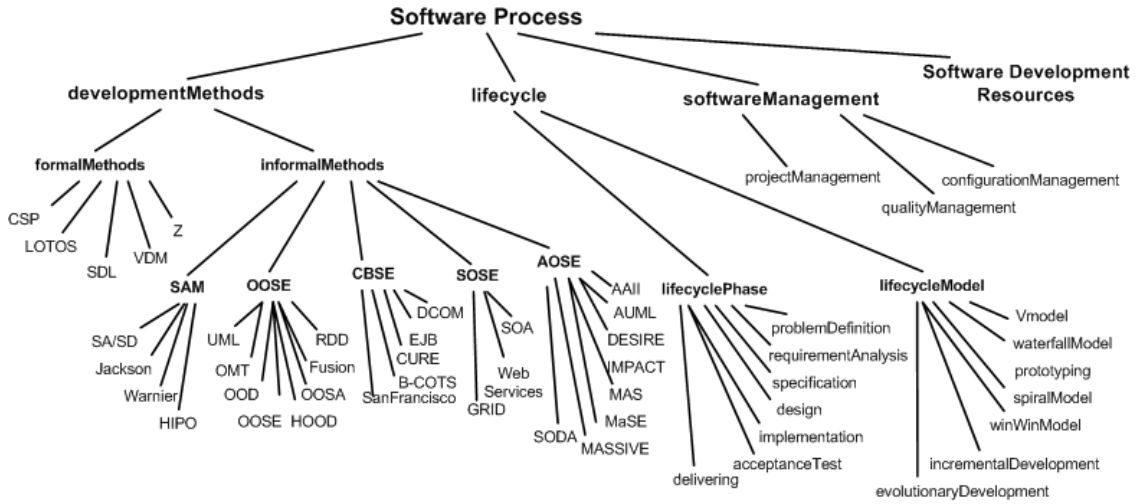


Figure 34: Declarative (general) software process description

Another kind of process description considers the different process components such as activities process structures etc. The following figure shows an example written in the *Business Process Modelling Notation (BPMN)* [Dumke 2006b].

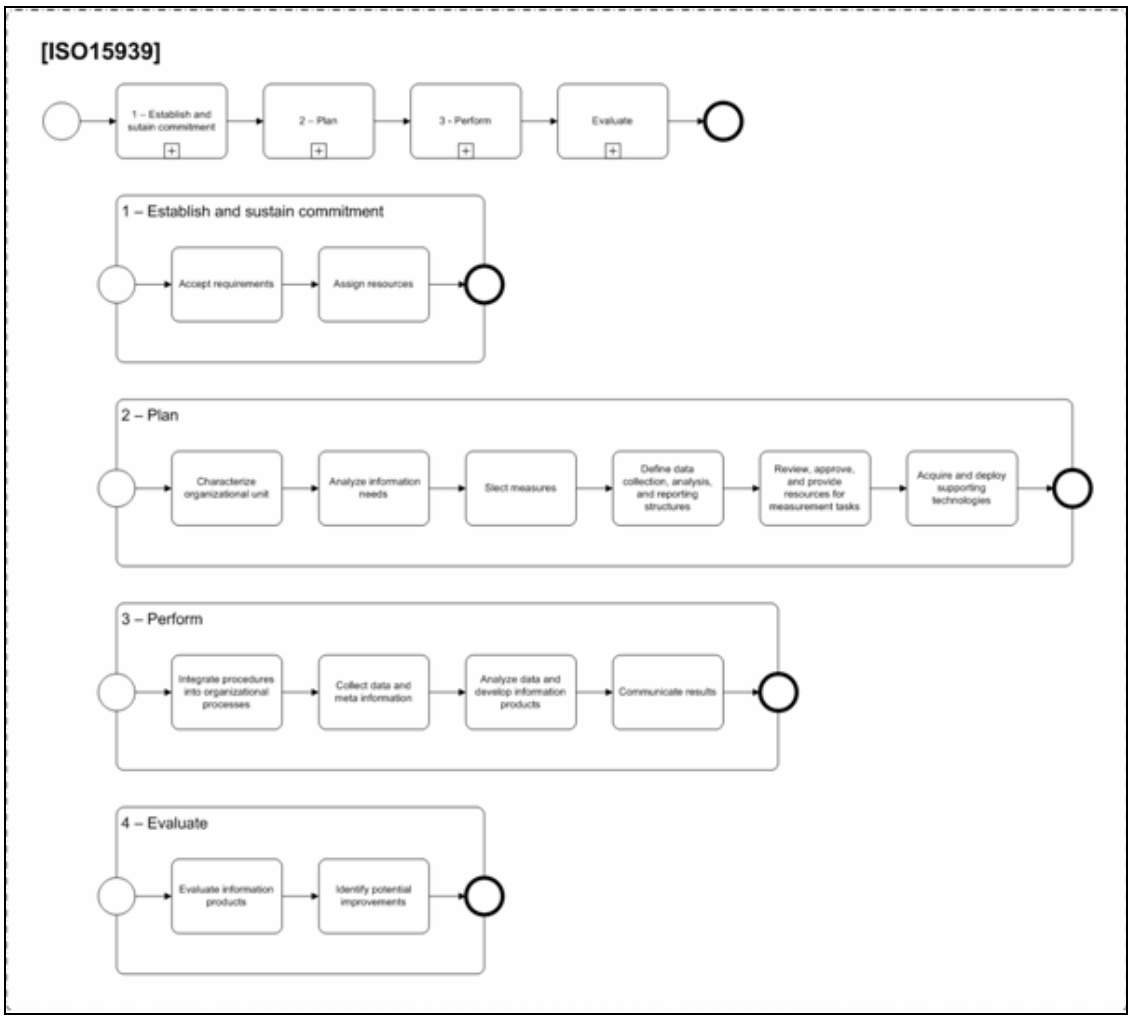


Figure 35: Software process description in the BPMN

Furthermore there are a lot of models and notations of process descriptions [Dumke 2006b]. But we will use the component model shown in 1.2 in order to address the essential software process aspects and involvements [Dumke 2006a].

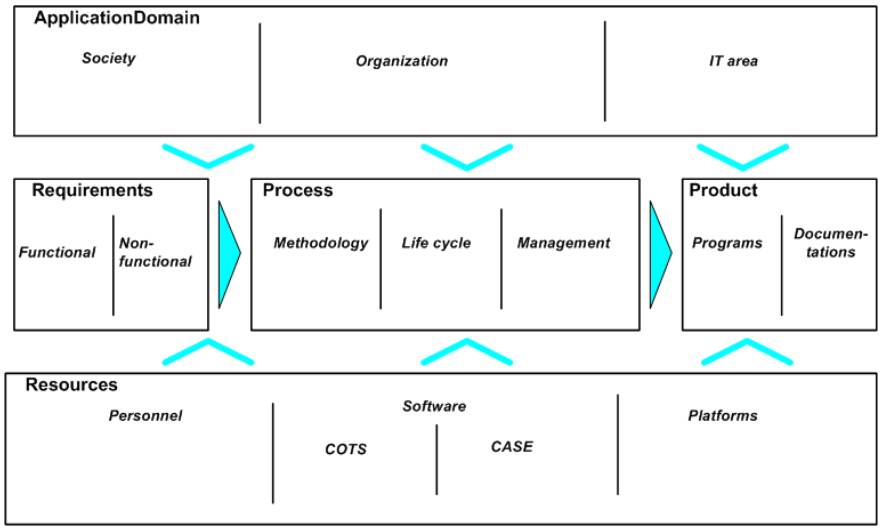


Figure 36: Software process component model

Formally the software process could be simply described based on this component model as following

- the software *process* is based on (product) *requirements* in order to create a software *product*
- *requirements* can be divided into *functional* and *non-functional* product requirements
- the *product* consists of programs and documentations
- the software *process* involves the *methodology*, the *lifecycle* and the *management* aspects
- the software *process* is based on the different *resources* as *personnel*, *software (COTS* and *CASE*) and *platforms* (hardware and system/basic software)
- the software *process* and the software *product* are involved in the *application domain* that consists of the *society, organization* and the *IT area*

On the other hand this component model can be mapped with the deterministic causal model approach of Pearl [Pearl 2000] in the following manner (see 2.1)

- the background (exogenous) variables U are

$$U = \{u^{(society)}_{applicationDomain}, u^{(society)}_{applicationDomain}, u^{(ITrea)}_{applicationDomain}, u^{(functional)}_{requirements}, u^{(non-functional)}_{requirements},$$
$$u^{(programs)}_{product}, u^{(documentations)}_{product}, u^{(personnel)}_{resources}, u^{(COTS)}_{resources}, u^{(CASE)}_{resources}, u^{(platform)}_{resources}\}$$

- the endogenous variable V are

  $V = \{v^{(methdology)}_{process}, v^{(lifecycle)}_{process}, v^{(management)}_{process}\}$

  that can be detailed in more (sub) components and elements

- the cautions $F=\{f_1, f_2, ..., f_n\}$ between the U und the V could be built as following examples

  $v^{(management)}_{process} = f_1 \ (v^{(methodology)}_{process}, u^{(CASE)}_{resources})$

  $v^{(lifecycle)}_{process} = f_2 \ (v^{(methodology)}_{process}, u^{(ITarea)}_{applicationDomain})$

  $v^{(methodology)}_{process} = f_3 \ ([v^{(lifecycle)}_{process}, v^{(management)}_{process}], u^{(programs)}_{product})$

  etc.

Note that we have introduced a very rough description at first in order to demonstrate the causal model/network in principle. The following figure shows a simplified characterization of this kind of causality description.

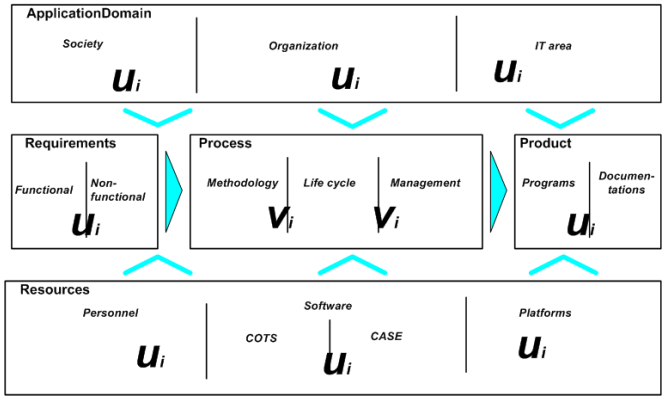


Figure 37: Causal model areas in the software process components

Note that there are possible other forms/boundaries of the considered causal model depending on the definition of U and V. Some other examples are show in the following figure.

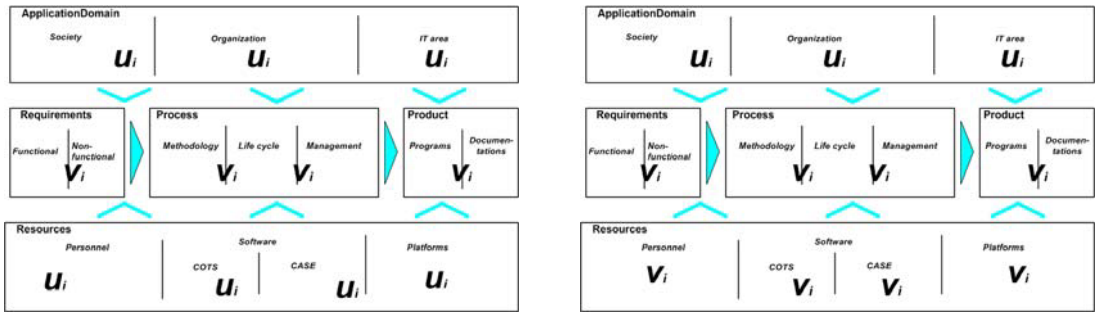


Figure 38: Variants of causal model areas

These kinds of causal network based descriptions could be considered as different parts of *empirical software engineering causal worlds.*

3.2 The CNPM Approach

The causal network based process model (CNPM) concept is defined in the following parts and components of this approach:

(M1) The causal network model M^{CNPM} is based on the following software process ingredients and involvements:

$$M^{CNPM} = \langle U^{CNPM}, V^{CNPM}, F^{CNPM} \rangle ,$$

where

- U^{CNPM} is a set of background variables that is determined by ***objects*** $o_{u,i}^{CNPM}$ ($i \in \{1,2,...,m\}$) as software process artefacts outside the considered model
- V^{CNPM} is a set $\{V_1^{CNPM}, V_2^{CNPM}, ..., V_n^{CNPM}\}$ of variables that are determined by ***objects*** $o_{v,i}^{CNPM}$ ($i \in \{1,2,...,n\}$) in the model – that is, variables or objects in $U^{CNPM} \cup V^{CNPM}$; and
- F^{CNPM} is a set of ***functions*** $\{f_1^{CNPM}, f_2^{CNPM}, ..., f_n^{CNPM}\}$ such that each f_i^{CNPM} ($i \in \{1,2,...,n\}$) is a mapping from (the respective domains of) $U^{CNPM} \cup (V^{CNPM} \backslash V_i^{CNPM})$ to V_i^{CNPM} and such that the entire set F_{CNPM} forms a mapping from U^{CNPM} to V^{CNPM}. In other words, each f_i^{CNPM} tells us the value V_i^{CNPM} given the values of all other variables in $U^{CNPM} \cup V^{CNPM}$, and the entire set F^{CNPM} has a unique solution V^{CNPM} *(o)*. Symbolically, the set of equations F^{CNPM} can be represented by writing

$$o_{v,i}^{CNPM} = f_i^{CNPM} (r_i^{CNPM}, o_{v,j}^{CNPM}, o_{u,j}^{CNPM}),\ i,j = 1, \ldots, n,\ i \neq j$$

  where r_i^{CNPM} is any realization of the unique minimal set of variables as ***roles***[1] R_i^{CNPM} in $V^{CNPM} \backslash V_i^{CNPM}$ sufficient for representing f_i^{CNPM}.

The following figure shows two examples of a simple CNPM model in different kinds of representation[2] in following.

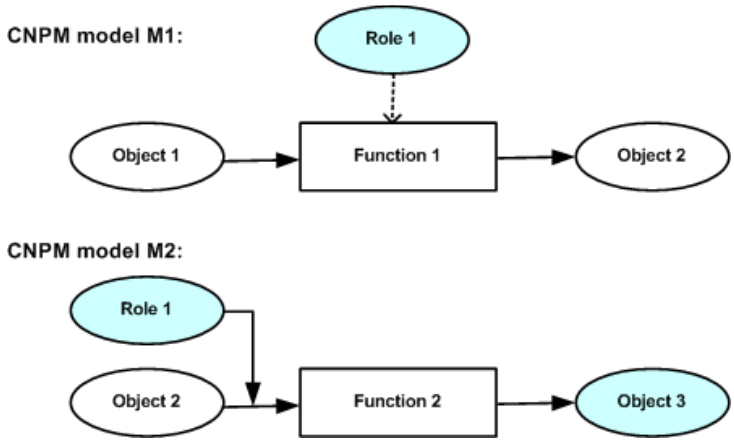


Figure 39: Simple examples of CNPM models

[1] Against the causality in natural science, software processes are based on activities of subjects. Therefore, we use a description of subjects as *roles*. Note that the roles define the causal heuristics addressed to the considered/presented function in the set of the software process artefacts.

[2] Note that the first kind of CNPM presentation is based on the typical cause-effect description. But we will use the second kind of presentation for our CNPM model visualization.

Considering the introduced levels of causality as dependencies, improvements and quality-based in the section 2.3 we can establish

$$f_i^{CNPM} = \{D \cup I \cup Q\}$$
$$= \{\text{“}determines\text{”}, \text{“}requires\text{”}, \ldots, \text{“}TMM\text{”}, \text{“}OMM\text{”}, \ldots, \text{“}ZP\text{”}, \text{“}KF\text{”}\}$$

(M2) The M^{CNPM} can be modified in the following manner considering the typical causal relationships between software process artefacts:

- ***Union*** or ***summarizing*** of CNPM models: this kind of modification exists in different variants based on the given situation between the considered CNPM models such as equivalent functions or equivalent (set of) roles, equivalent inputs or outputs. We will describe a simple example for CNPM model unification based on the equality of functions. The union of two CNPM models could be described in $f_{funcjoin}^{CNPM}$ as following

$$f_{funcjoin}^{CNPM} : M_x^{CNPM} \times M_y^{CNPM} \rightarrow M_{xy}^{CNPM}$$

where the models M_x^{CNPM} und M_y^{CNPM} are defined by

$$M_x^{CNPM} = < U_x^{CNPM}, V_x^{CNPM}, F_x^{CNPM} >$$

with the function $f_i^{CNPM} \in F_x^{CNPM}$, the outputs V_x^{CNPM} and the inputs U_x^{CNPM} and

$$M_y^{CNPM} = < U_y^{CNPM}, V_y^{CNPM}, F_y^{CNPM} >$$

with the function $f_i^{CNPM} \in F_y^{CNPM}$, the outputs V_y^{CNPM} and the inputs U_y^{CNPM}. The function f_i^{CNPM} represents the common function in both CNPM models with the following characteristics

$$M_x^{CNPM} : f_i^{CNPM} : U_x^{CNPM} \times V_x^{CNPM} \setminus V_i^{CNPM} \rightarrow V_i^{CNPM}$$

where $V_i^{CNPM} \in V_x^{CNPM}$

$$M_y^{CNPM} : f_i^{CNPM} : U_y^{CNPM} \times V_y^{CNPM} \setminus V_j^{CNPM} \rightarrow V_j^{CNPM}$$

where $V_j^{CNPM} \in V_y^{CNPM}$

Hence, the derived CNPM model M_{xy}^{CNPM} has the following characteristics:

$$M_{xy}^{CNPM} = < U_{xy}^{CNPM}, V_{xy}^{CNPM}, F_{xy}^{CNPM} >$$

with the function $f_i^{CNPM} \in F_{xy}^{CNPM}$, the inputs $V_{xy}^{CNPM} = V_x^{CNPM} \cup V_y^{CNPM}$ and $U_{xy}^{CNPM} = U_x^{CNPM} \cup U_y^{CNPM}$ and

$$f_i^{CNPM} : U_{xy}^{CNPM} \times V_{xy}^{CNPM} \setminus V_k^{CNPM} \rightarrow V_k^{CNPM}$$

with $U_{xy}^{CNPM} = U_{x}^{CNPM} \cup U_{y}^{CNPM}$, $V_{xy}^{CNPM} = V_{x}^{CNPM} \cup V_{y}^{CNPM}$, $V_{k}^{CNPM} = V_{i}^{CNPM} \cup V_{j}^{CNPM}$.

The following figure shows a simple example of the application of the function $f_{funcjoin}^{CNPM}$ using the both diagrams in figure 39.

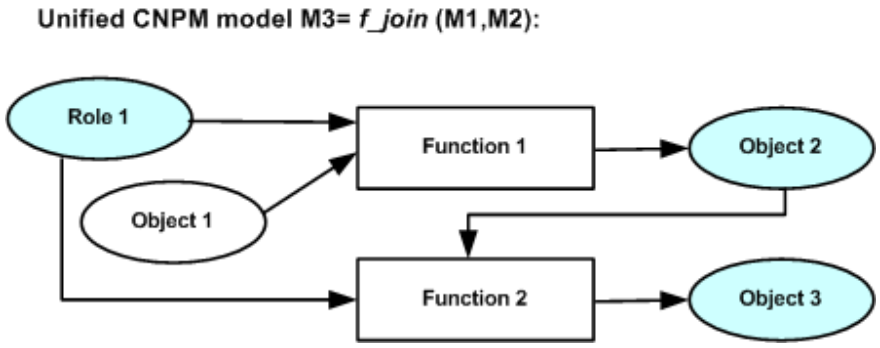


Figure 40: Simple example for $f_{funcjoin}^{CNPM}$ application

- ***Partitioning*** of a CNPM model consists of building sub models and special parts of models. The partitioning function f_{part}^{CNPM} builds new CNPM models using the existing elements or components and could be characterized as

$$f_{part}^{CNPM} : M_{xy}^{CNPM} \rightarrow M_{x}^{CNPM} \times M_{y}^{CNPM}$$

That means that the CNPM model $M_{xy}^{CNPM} = <U_{xy}^{CNPM}, V_{xy}^{CNPM}, F_{xy}^{CNPM}>$ with the inputs U_{xy}^{CNPM}, the outputs V_{xy}^{CNPM} and the causal functions $F_{k}^{CNPM} \subseteq F_{xy}^{CNPM}$: $U_{xy}^{CNPM} \times V_{xy}^{CNPM} \backslash V_{k}^{CNPM} \rightarrow V_{k}^{CNPM}$ would be divided in the two ***sub models*** $M_{x}^{CNPM} = <U_{x}^{CNPM}, V_{x}^{CNPM}, F_{x}^{CNPM}>$ with the function $F_{x}^{CNPM} \subseteq F_{xy}^{CNPM}$, the outputs $V_{x}^{CNPM} \subseteq V_{xy}^{CNPM}$, the inputs $U_{x}^{CNPM} \subseteq U_{xy}^{CNPM}$ and $F_{i}^{CNPM} \subseteq F_{x}^{CNPM}$: $U_{x}^{CNPM} \times V_{x}^{CNPM} \backslash V_{i}^{CNPM} \rightarrow V_{i}^{CNPM}$ and $M_{y}^{CNPM} = <U_{y}^{CNPM}, V_{y}^{CNPM}, F_{y}^{CNPM}>$ with the function $F_{y}^{CNPM} \subseteq F_{xy}^{CNPM}$, the outputs $V_{y}^{CNPM} \subseteq V_{xy}^{CNPM}$, the inputs $U_{y}^{CNPM} \subseteq U_{xy}^{CNPM}$ and $F_{j}^{CNPM} \subseteq F_{y}^{CNPM}$: $U_{y}^{CNPM} \times V_{y}^{CNPM} \backslash V_{j}^{CNPM} \rightarrow V_{j}^{CNPM}$. In practice it would be helpful to ensure that the sets of causal functions F_{x}^{CNPM} and F_{y}^{CNPM} are disjunctive. We will show a simple example again in the following figure which used the CNPM model from figure 40 in order to build any two sub models.

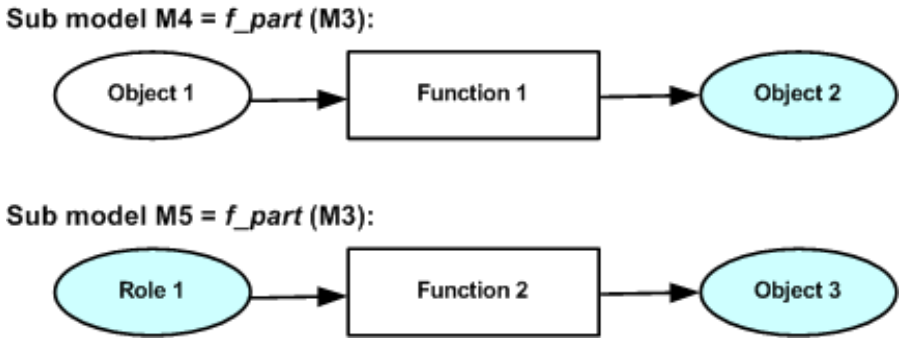


Figure 41: Simple example for f_{part}^{CNPM} application

- ***Restructuring*** of a CNPM model is reasonable in different practical situations. Formally, it could be an extension or reduction of the different sets of U^{CNPM}, V^{CNPM} and F^{CNPM}. That means an addition or extraction of any objects $o_{u,i}^{CNPM}$ or $o_{v,i}^{CNPM}$, roles r_i^{CNPM} or functions f_i^{CNPM}. Semantically, it is based on changing the functional background as a change of the causality. In following we will consider some special cases of restructuring using the CNPM model description $M_x^{CNPM} = < U_x^{CNPM}, V_x^{CNPM}, F_x^{CNPM} >$ with the inputs U_x^{CNPM}, the outputs V_x^{CNPM} and the causal functions F_x^{CNPM}.

  (1) Addition/extraction of a function: In principle, the addition or extraction of a causal function could be characterized as a summarizing or partitioning of two causal models. This is reason for the key aspect of the function as kernel of the CNPM models.

  (2) Addition/extraction of a role: The role as a causal indicator extends the causality characteristic of any function in the CNPM model. The addition or extraction of a role extends or reduces the input set U_x^{CNPM} and would be explained also in the next section.

  (3) Addition/extraction of an input: Considering the characterized CNPM model M_x^{CNPM} above, a new model was built $M_{x'}^{CNPM}$ in the following manner: in the case of addition the derived model $M_{x'}^{CNPM} = <U_{x'}^{CNPM}, V_x^{CNPM}, F_x^{CNPM}>$ includes the outputs V_x^{CNPM}, the modified inputs $U_{x'}^{CNPM}$ and the functions $F_{i'}^{CNPM} \subseteq F_x^{CNPM}$: $U_{x'}^{CNPM} \times V_x^{CNPM} \backslash V_i^{CNPM} \rightarrow V_i^{CNPM}$ where $U_{x'}^{CNPM} = U_x^{CNPM} \cup \{u_i^{CNPM}\}$. In the case of extraction we establish $U_{x'}^{CNPM} \times V_x^{CNPM} \backslash V_i^{CNPM} \rightarrow V_i^{CNPM}$ where $U_{x'}^{CNPM} = U_x^{CNPM} \setminus \{u_i^{CNPM}\}$. Note that in both cases the set of functions $F_{i'}^{CNPM}$ is addressed to u_i^{CNPM} only. The other functions in F_x^{CNPM} would be not changed using this kind of restructuring.

(4) Addition/extraction of an output: These kinds of operations of restructuring could be described in the same manner like (3) considering the output set V_x^{CNPM} modified to $V_{x'}^{CNPM}$. We obtain $M_{x'}^{CNPM} = <U_x^{CNPM}, V_{x'}^{CNPM}, F_x^{CNPM}>$ includes the inputs U_x^{CNPM}, the modified outputs $V_{x'}^{CNPM}$ and the functions $F_x^{CNPM}: U_x^{CNPM} \times V_{x'}^{CNPM} \setminus V_{i'}^{CNPM} \rightarrow V_{i'}^{CNPM}$ where $V_{x'}^{CNPM} = V_x^{CNPM} \cup \{v_i^{CNPM}\}$ and $V_{x'}^{CNPM} \subseteq V_x^{CNPM}$ and in the case of extraction $V_{x'}^{CNPM} = V_x^{CNPM} \setminus \{v_i^{CNPM}\}$.

A simple example of restructuring based on the CNPM model from figure 40 by addition of a role and extraction of an object for function 2 is given in the following figure.

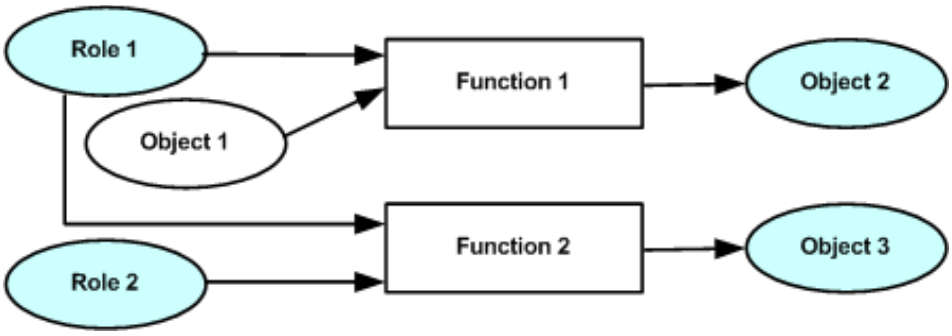


Figure 42: Simple example for restructuring of a CNPM model

(M3) The M^{CNPM} can be analyzed considering the typical causal relationships between software process artefacts in the following manner. The CNPM model could be considered as a directed graph where every node has some predecessors and any successors. Hence, it is possible to analyze or count these elements for a first level of CNPM analysis and evaluation. For instance, we obtain the number of all roles in the CNPM, the number of derived objects etc. Based on this idea, we can define the following function $f_{extract\_input}^{CNPM}$ of CNPM analysis as

$$f_{extract\_input}^{CNPM}: M_x^{CNPM} \times f_i^{CNPM} \rightarrow U_i^{CNPM}$$

where $M_x^{CNPM} = <U_x^{CNPM}, V_x^{CNPM}, F_x^{CNPM}>, f_i^{CNPM} \in F_x^{CNPM}, U_i^{CNPM} \subseteq U_x^{CNPM}$ and

$$U_i^{CNPM} = \{ u_i^{CNPM} : u_i^{CNPM} = \textbf{\textit{predecessor}}(f_i^{CNPM}) \}.$$

Extracting the roles r_i^{CNPM} could be based on the counting of their causal aspects because of their actor characteristic. That could be expressed by

$$U_i^{CNPM} = \{ u_i^{CNPM} : u_i^{CNPM} = predecessor(f_i^{CNPM}) \wedge actor(u_i^{CNPM}) = \text{“yes”} \}.$$

In the same manner we could analyze the expected results and outputs based on the function $f_{extract\_output}^{CNPM}$ as following:

$$f_{extract\_output}^{CNPM} : M_x^{CNPM} \times f_i^{CNPM} \rightarrow V_i^{CNPM}$$

where $M_x^{CNPM} = <U_x^{CNPM}, V_x^{CNPM}, F_x^{CNPM}>, f_i^{CNPM} \in F_x^{CNPM}, U_i^{CNPM} \subseteq U_x^{CNPM}$ and

$$V_i^{CNPM} = \{ v_i^{CNPM} : v_i^{CNPM} = \textbf{\textit{successor}}(f_i^{CNPM}) \}.$$

Applying these functions to our described examples of CNPM models we can derive the following characteristics:

- $predecessor(f_1^{CNPM\_M2})$ = {'Role 1', 'Object 2'}
- $predecessor(f_1^{CNPM\_M3})$ = {'Role 1', 'Object 1'}
- $successor(f_2^{CNPM\_M3})$ = {'Object 3'}

(M4) The M^{CNPM} can be evaluated in the following manner considering the typical causal relationships between software process artefacts. The CNPM model could be characterized as empirical evaluation that requires the identification of the empirical aspects explicitly. Such empirical characteristic for objects could be process artefact level, artefact quality or process artefact performance. From this point of view, the CNPM model evaluation could be performed as following:

- *causal coverage analysis* of the fulfilled requirements from a special software process point of view,
- *causal trace analysis* of the successful consideration of process flow based requirements,
- *causal achievement analysis* of the derived results and outputs in different parts on the CNPM model.

In order to explain some of these kinds of analysis we will consider the CPNM model M_x^{CNPM} describing the empirical-based process aspects mainly and the CPNM model M_y^{CNPM} describing the causal basics in general. On that we characterize a simple causal coverage analysis as

$$\textbf{\textit{coverage}}_{My}^{CNPM} = \sum(|F_y^{CNPM}|+|U_y^{CNPM}|+|V_y^{CNPM}|) / \sum(|F_x^{CNPM}|+|U_x^{CNPM}|+|V_x^{CNPM}|)$$

where $F_x^{CNPM} \subseteq F_y^{CNPM}$, $U_x^{CNPM} \subseteq U_y^{CNPM}$, $V_x^{CNPM} \subseteq V_y^{CNPM}$. Therefore we can establish that the best value of causal coverage would be achieved as $coverage_{My}^{CNPM}$=1. On the other hand, percentage-based kind of presentation would be obtained by

$$coverage_{[\%]} = coverage_{My}^{CNPM} * 100 .$$

Considering the different variables or objects and roles we can define

$$coverage^{CNPM}_{function\_My} = \sum |F^{CNPM}_{y}| / \sum |F^{CNPM}_{x}|$$

$$coverage^{CNPM}_{input\_My} = \sum |U^{CNPM}_{y}| / \sum |U^{CNPM}_{x}|$$

$$coverage^{CNPM}_{output\_My} = \sum |V^{CNPM}_{y}| / \sum |V^{CNPM}_{x}|$$

Furthermore, in the case of coverage lower 1 we have the situation of any missing objects. That could be characterized in the following manner

$$F^{CNPM}_{missing\_function} = \{ F^{CNPM}_{x} \setminus F^{CNPM}_{y} : F^{CNPM}_{y} \subseteq F^{CNPM}_{x} \}$$

$$F^{CNPM}_{missing\_input} = \{ U^{CNPM}_{x} \setminus U^{CNPM}_{y} : U^{CNPM}_{y} \subseteq U^{CNPM}_{x} \}$$

$$F^{CNPM}_{missing\_output} = \{ V^{CNPM}_{x} \setminus V^{CNPM}_{y} : V^{CNPM}_{y} \subseteq V^{CNPM}_{x} \}$$

For the causal trace analysis and achievement analysis the existing graph algorithm and methods of evaluation can be used that would not be considered here.

3.3 CNPM-Based Software Process Analysis

One of the possible uses for the CNPM model is the mapping of process standards. This shall be described by example of the key process area „Organizational Training“ (**OT**) of the CMMI. Also it will be considered that specific practices of this model give a hint for the implementation of a CMMI conformant process environment. The specific practice (**SP**) 1.1 will be used as an example for the implementation of a CNPM network.

SP 1.1 – Establish the Strategic Training Needs

This practice contains the following sub practices:

- *Analyze the organization's strategic business objectives and process improvement plan to identify potential future training needs.*
- *Document the strategic training needs of the organization.*
- *Determine the roles and skills needed to perform the organization's set of standard processes.*
- *Document the training needed to perform the roles in the organization's set of standard processes.*
- *Document the training needed to maintain the safe, secure and continued operation of the business.*
- *Revise the organization's strategic needs and required training as necessary.*

To create a network it is necessary to split the text into tasks, objects and roles. This decomposition leads to the following elements:

Objects:

- *Strategic business objectives*
- *Process improvement plan*
- *Set of standard processes*
- *Training needs for roles and skills*
- *Training needs for business*
- *Needed roles*
- *Needed skills*

Functions:

- *Analyse*
- *Document strategic training needs*
- *Determine roles and skills*
- *Document training needs to perform standard processes*
- *Document training needs for safe, secure, continued business*
- *Revise if necessary*

Roles:

The text of the CMMI contains no detailed information about the role executing the task. But it gives the general definition, that the management is responsible for all quality activities. So for the following networks the management will be used as executing instance of this task.

The resulting network is shown below:

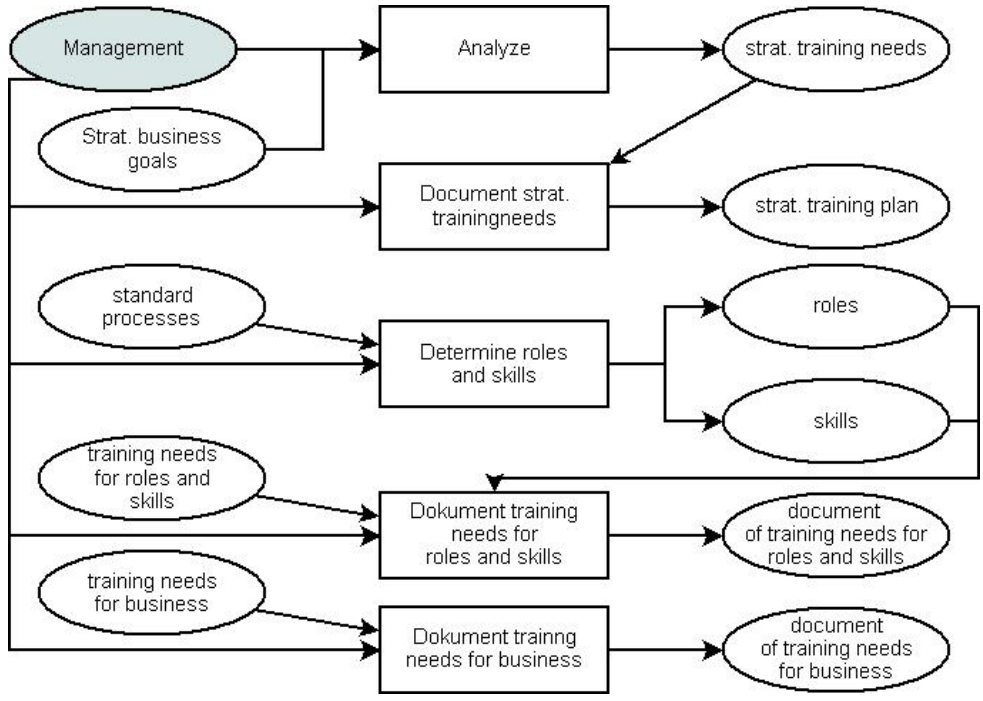


Figure 43: Organizational Training - SP 1.1 – first approach

A deeper analysis of the objects contained in this network shows, that there is no Task, creating the objects „training needs for roles and skills“ and „training needs for business“. This shows the incompleteness of the CMMI in some detailed views.

The inserted processes are the following:

- *Determine training needs for roles and skills*
- *Determine training needs for business*

Furthermore, it can be seen, that the network contains two functions for documenting two different types of training needs. Giving credit to the fact that the documentation of training needs doesn't depend on the type of the training need that is to be documented, both functions can be combined to a single one.

- *Document training needs*

The network constructed by these changes is shown in the following figure:

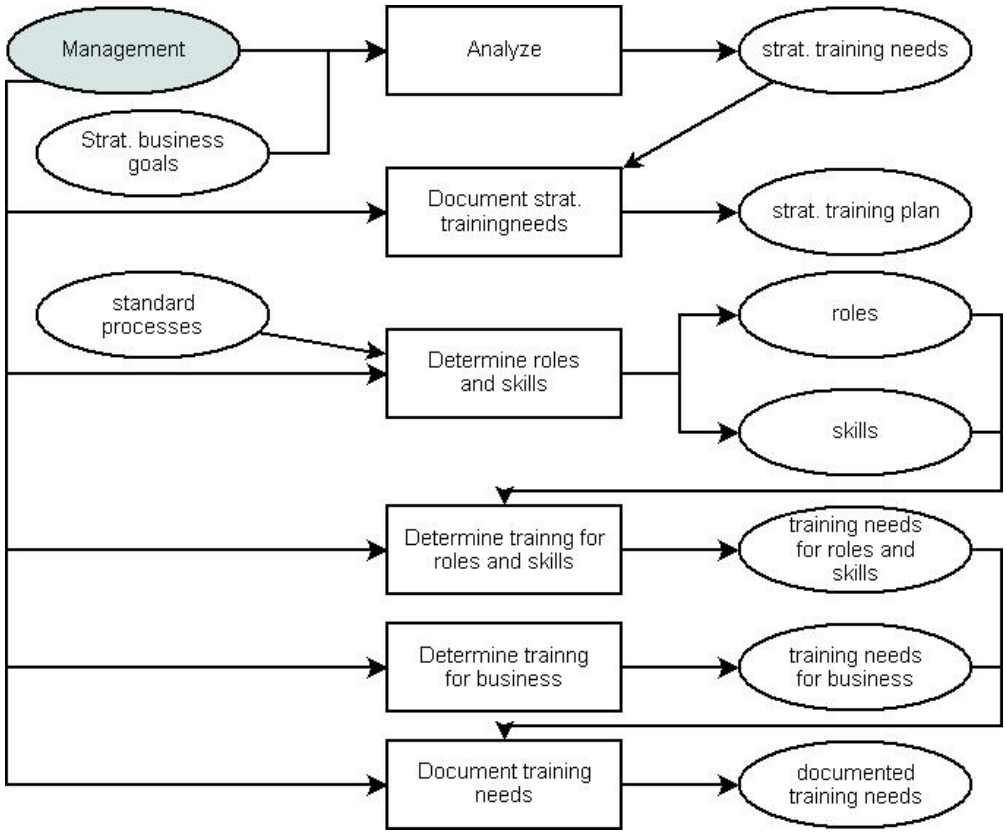


Figure 44: Organizational Training - SP 1.1 - restructured

To create the other specific practices some adaptations of the CMMI-text are necessary. For example, the fact, that the object "Satisfaction approaches" of OT-SP 1.4 is not used as an input object for any of the other contained sub processes. Furthermore it isn't a work product of the process area, what makes it necessary to insert the impact of this object in later functions.

Abstaining from showing the detailed description of the creation of the diagrams the following figures show the text and the final networks of the specific practices 1.2 to 2.3.

SP 1.2 - Determine which training needs are the responsibilities of the Organization

- *Analyze the training needs identified by the various projects and support groups.*
- *Negotiate with the various projects and support groups on how their specific training needs will be satisfied.*
- *Document the commitments for providing training support to the projects and support groups.*

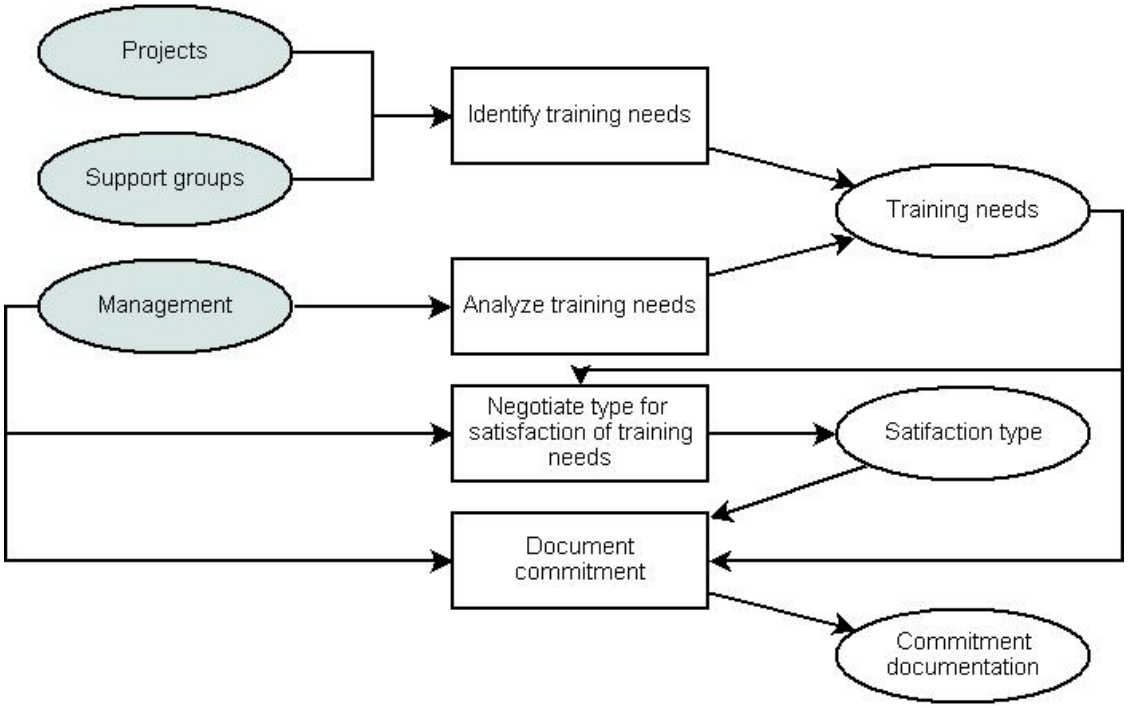


Figure 45: Organizational Training - SP 1.2

SP 1.3 – Establish an organizational training tactical plan

- *Establish plan content*
- *Establish commitments to the plan*

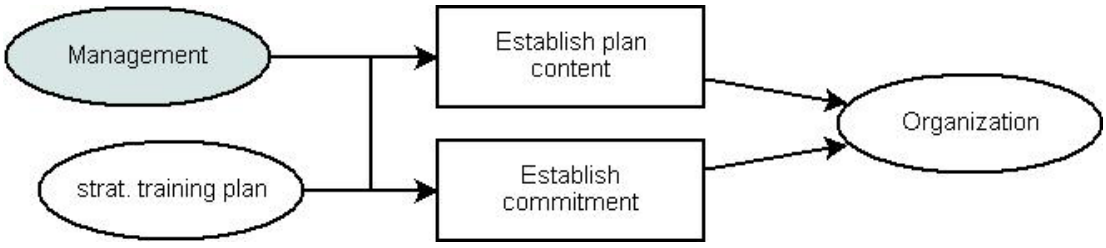


Figure 46: Organizational Training - SP 1.3

SP 1.4 - Establish training capability

- *Select the appropriate approaches to satisfy specific organizational training needs.*
- *Determine whether to develop training materials internally or acquire them externally*
- *Develop or obtain training materials*
- *Develop or obtain qualified instructors*
- *Describe the training in the organization's training curriculum*
- *Revise the training materials and supporting artifacts as necessary*

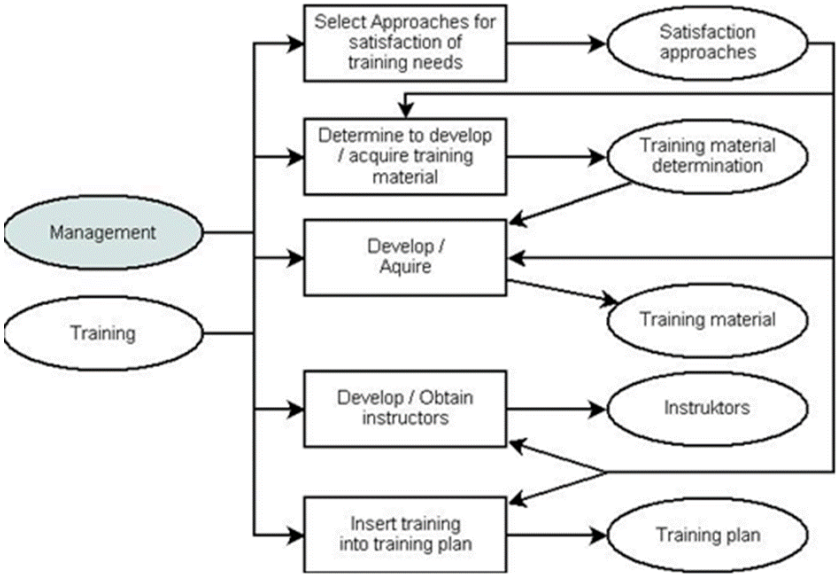
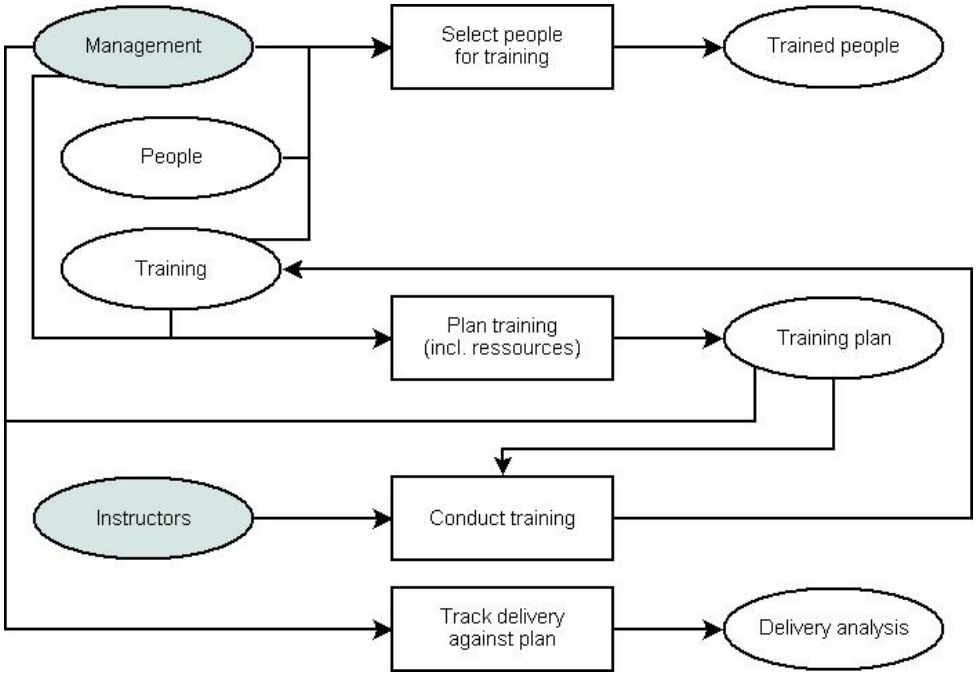


Figure 47: Organizational Training - SP 1.4

SP 2.1 – Deliver training

- *Select the people who will receive the training necessary to perform their roles effectively.*
- *Schedule the training, including any resources, as necessary (e.g., facilities and instructors)*
- *Conduct the training*
- *Track the delivery of training against the plan*



Fuigure 48: Organizational Training - SP 2.1

SP 2.2 – Establish training records

- *Keep records of all students who successfully complete each training course or other approved training activity as well as those who are unsuccessful*
- *Keep records of all staff who have been waived from specific training*
- *Keep records of all students who successfully complete their designated required training*
- *Make training records available to the appropriate people for consideration in assignments*

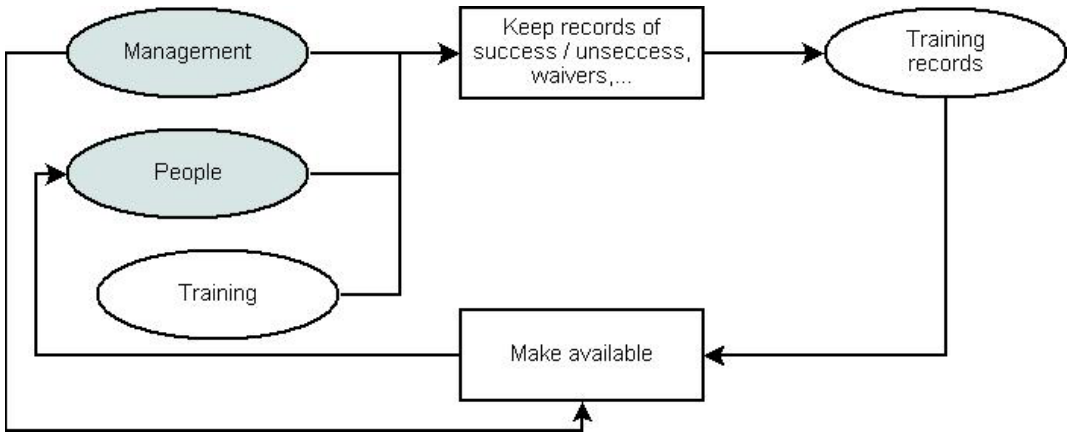


Figure 49: Organizational Training - SP 2.2

SP 2.3 – Assess training effectiveness

- *Assess in-progress or completed projects to determine whether staff knowledge is adequate for performing project tasks*

- *Provide a mechanism for assessing the effectiveness of each training course with respect to established organizational, project or individual learning (or performance) objectives*

- *Obtain student evaluations of how well training activities met their needs*

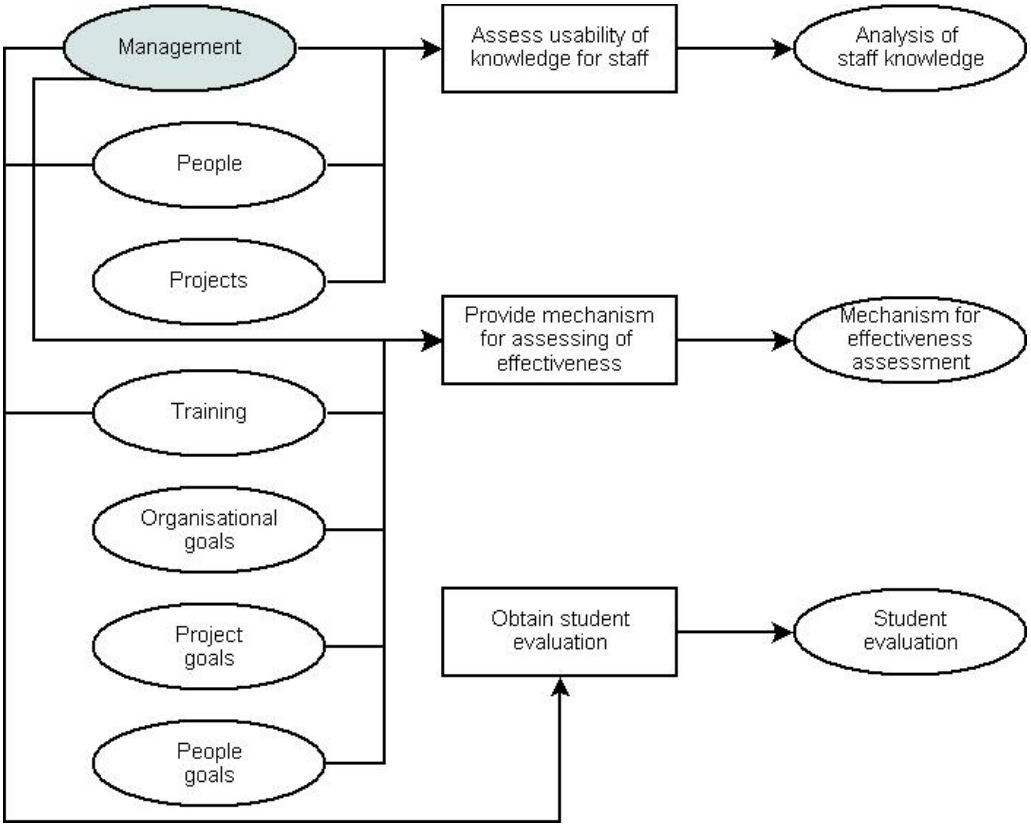


Figure 50: Organizational Training - SP 2.3

Using the methods described above, the shown networks can be combined to show the complete picture of the tasks fulfilling the requirements of key process area “organizational training”.

The combination is executed by a mapping over the role “management” and some output objects, being input objects in other CNPM networks. The following figure shows the overall process network of this process area.

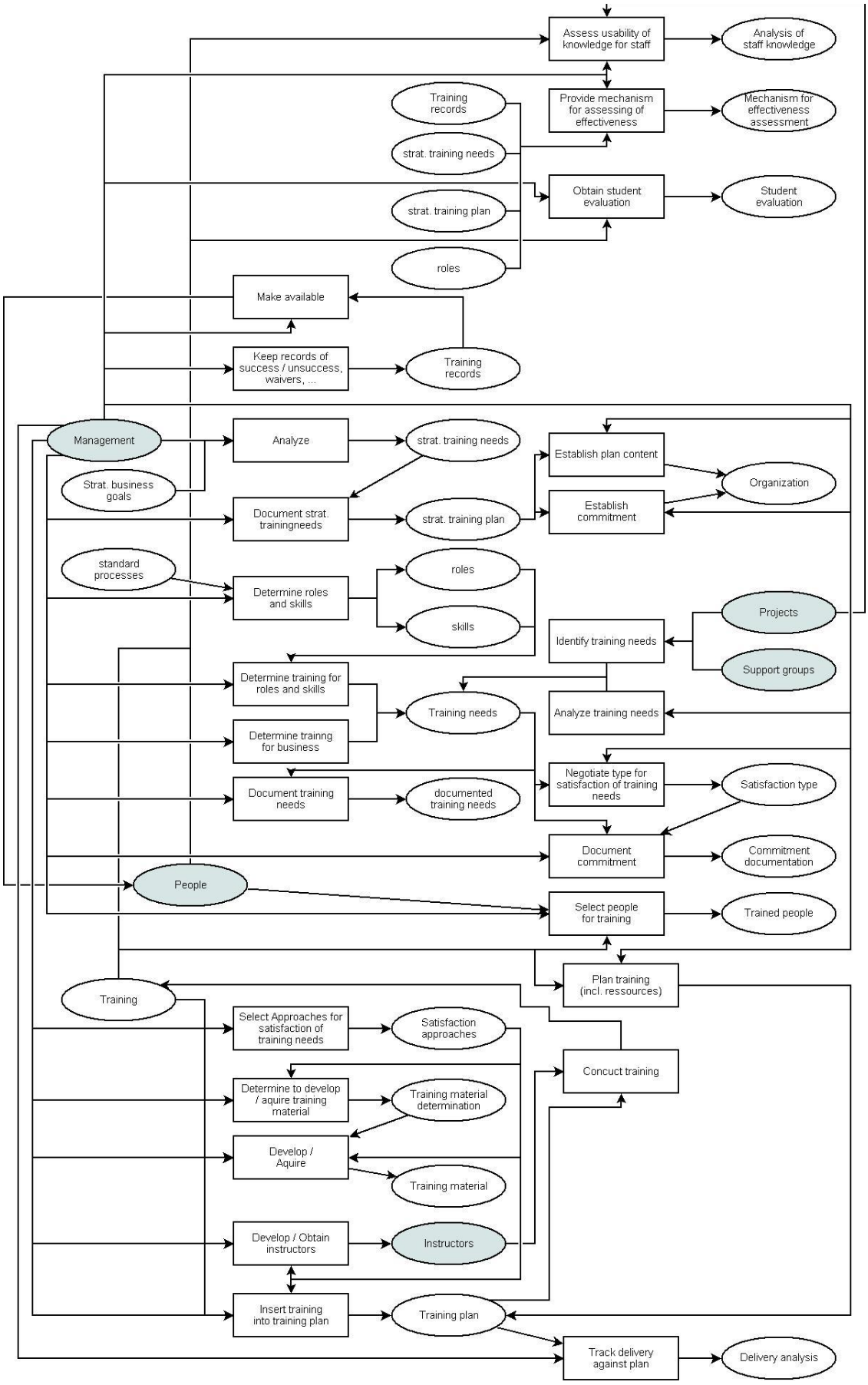


Figure 51: Organizational Training - complete network

4 Conclusion and Future Work

The presented CNPM-based approach was applied in practice in order to transform the textual CMMI standard in a causal network based form. This implies the chance of *explicit description* of the CMMI process evaluation from an *implicit* one. Otherwise it allows to consider other causalities and empirical relationships in the software process area depending on concrete industrial situations and methodologies.

In our further research different process evaluation approaches would be compared in order to understand, classify and improve the software process management level in the IT practice. The concrete application of our CNPM approach in practice was described in a separate PhD thesis in our software engineering group.

5 References

[April 2005] April, A.: *S^{3m}-Model to Evaluate and Improve the Quality of Software Maintenance Process*. Shaker Publ., Aachen, Germany 2005

[Braungarten 2005] Braungarten, R.; Kunz, M.; Dumke, R.: *An Approach to Classify Software Measurement Storage Facilities.* Preprint No 2, University of Magdeburg, Dept. of Computer Science, 2005

[Davis 1995] Davis, A. M.: *201 Principles of Software Development.* McGraw Hill Publ., 1995

[Deek 2005] Deek, F. P.; McHugh, J. A. M.; Eljabiri, O. M.: *Strategic Software Engineering – An Interdisciplinary Approach.* Auerbach Publications, Boca Raton London New York,2005

[Dumke 1996] Dumke, R.; Foltin, E.; Koeppe, R.; Winkler, A.: *Softwarequalität durch Meßtools – Assessment, Messung und instrumentierte ISO 9000.* Vieweg Publ., Braunschweig, Germany, 1996

[Dumke 2006a] Dumke, R.; R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Causalities in Software Process Measurement and Improvement.* Proc. of the MENSURA 2006, Nov. 6-8, 2006, Cardiz, Spain, pp.42-52

[Dumke 2006b] Dumke, R.; Braungarten, R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Software Process Measurement and Control – A Measurement-Based Point of View of Software Processes*. Preprint No 11, University of Magdeburg, 2006 (http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/froschung/Preprints. shtml)

[Dumke 2006c] Dumke, R.; Braungarten, R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Structuring Software Process Metrics - A holistic semantic network based overview.* Proc. of the IWSM 2006, Potsdam, Nov. 2006, pp. 483-498

[Dumke 2001] Dumke, R.; Lother, M.; Abran, A.: *An Approach for Integrated Software Measurement Processes in the IT Area.* Proc. of the FESMA 2001, Heidelberg, My 2001, pp. 15-29

[Dumke 2005a] Dumke, R.; Richter, K.; Fetcke, T.: *FSM influences and Requirements in CMMI-Based Software Processes.* In: A. Abran et al.: Innovations in Software Measurement, Shaker Verlag, Aachen, pp. 179-194

[Dumke 2005b] Dumke, R.; Schmietendorf, A.; Zuse, H.: *Formal Descriptions of Software Measurement and Evaluation - A Short Overview and Evaluation.* Preprint No. 4, Fakultät für Informatik, University of Magdeburg, 2005

[Emam 1998] Emam, K. E.; Drouin, J. N.; Melo, W.: *SPICE – The Theory and Practice of Software Process Improvement and Capability Determination.* IEEE Computer Society Press, 1998

[Emmerich 2007] Emmerich, W.; Aoyama, M.; Sventek, J.: *The Impact of Research on Middleware Technology.* Software Engineering Notes, January 2007, pp. 21-46

[Endres 2003] Endres, A.; Rombach, D.: *A Handbook of Software and System Engineering*. Pearson Education Limited, 2003

[Fenton 2001] Fenton, N,; Krause, P.; Neil, M.: *Probabilistic Modelling for Software Quality Control*. Proc. of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Toulouse 2001

[Ferguson 1998] Ferguson, J.; Sheard, S.: *Leveraging Your CMM Efforts for IEEE/EIA 12207.* IEEE Software, September/October 1998, pp. 23-28

[Florac 1999] Florac, W. A.; Carleton, A. D.: *Measuring the Software Process – Statistical Process Control for Software Process Improvement.* Pearson Education, 1999

[Garcia 2005] Garcia, S.: *How Standards Enable Adoption of Project Management Practice*. IEEE Software, Sept./Oct. 2005, pp. 22-29

[Haywood 1998] Haywood, M.: *Managing Virtual Teams – Practical Techniques for High-Technology Project Managers.* Artech House, Boston, London, 1998

[Humphrey 2000] Humphrey, W. S.: *The Personal Software Process: Status and Trends*. IEEE Software, Nov/Dec. 2000, pp. 71-75

[Kenett 1999] Kenett, R. S.; Baker, E. R.: *Software Process Quality – Management and Control*. Marcel Dekker Inc., 1999

[Keyes] Keyes, J.: *Software Engineering Handbook.* Auerbach Publ., 2003

[Laird 2006] Laird, L. M; Brennan, M. C.: *Software Measurement and Estimation: A Practical Approach.* John Wiley & Sons Publ., 2006

[Lecky-Thompson 2005] Lecky-Thompson, G. W.: *Corporate Software Project Management*. Charles River Media Inc., USA, 2005

[Müller 2008] Müller, M.; Pfahl, D.: *Simulation Methods.* In: Shull et al.: Guide to Advanced Empirical Software Engineering. Springer Publ., 2008, pp. 117-152

[Pearl 2000] Pearl, J.: *Causality – Models, Reasoning, and Inference.* Cambridge University Press, 2000

[Putnam 2003] Putnam, L. H.; Myers, W.: *Five Core Metrics – The Intelligence Behind Successful Software Management.* Dorset House Publishing, New York, 2003

[Richter 2005] Richter, K.: *Softwaregrößenmessung im Kontext von Software-Prozessbewertungsmodellen*. Diploma Thesis, University of Magdeburg, Faculty of Informatics, 2005

[Royce 1998] Royce, W.: *Software Project Management.* Addison-Wesley, 1998

[SEI 2002] SEI: *Capability Maturity Model Integration (CMMI^{SM}),* Version 1.1, Software Engineering Institute, Pittsburgh, March 2002, CMMI-SE/SW/IPPD/SS, V1.1

[SPICE 2006] *The SPICE Web Site*, http://www.sqi.gu.edu.au/spice/ (seen July 24, 2006)

[Wang 2000] Wang, Y.; King, G.: *Software Engineering Processes – Principles and Applications*. CRC Press, Boca Raton London New York, 2000

[Weisstein 2006] Weisstein, E.: *Causal Networks*. Script in Computer Science, http://mathworld.wolfram.com/CausalNetwork.html (August 1, 2006)

[Zettel 2001] Zettel, J.; Maurr, F.; Münch, J.; Wong, L.: *LIPE: A Lightweight Process for E-Business Startup Companies Based on Extreme Programming.* In: Bomarius/Komi-Sirviö: Product Focused Software Process Improvement. PROFES 2001, Kaiserslautern, Sept. 2001, LNCS 2188, Springer Publ., 2001, pp. 255-270