An Information Systems Design Product Theory for Software Project Estimation and Measurement Systems

Pekka Forselius
4SUM Partners
Tekniikantie 14, Espoo
02150 Finland
Pekka.Forselius@4sumpartners.com

Timo Käkölä
Claremont Graduate University &
University of Jyväskylä
40014 University of Jyväskylä, Finland
timokk@jyu.fi

Abstract: There is relatively little research on software Project Estimation and Measurement Systems (PEMS). Commercial PEMS vary in functionality and effectiveness. Their intended users thus do not know what to expect from PEMS and how to evaluate them. This paper creates an information system design product theory for the class of PEMS that prescribes the meta-requirements, the meta-design, and applicable theories for all products within the class. Meta-requirements and the meta-design are derived from the project estimation and measurement literature, experiences obtained during more than ten years of empirical work in Finnish Software Measurement Association, and a commercially available PEMS.

Keywords: Functional size measurement, Knowledge management, Organizational learning, Outsourcing, Software process improvement, Software project estimation and benchmarking

1. Introduction

Software project estimation and measurement (PEM) has been researched extensively due to negative organizational and financial consequences of projects that fail to deliver desired functionalities and to meet nonfunctional quality requirements and that run late and out of budget. PEM aims at predicting the size, productivity, total effort and/or cost, and schedule of a project any time and as often as necessary before the end of the project. PEM is critical in complex software business networks where consumers acquire and integrate software from numerous providers through eSourcing, a business practice of looking for domestic or foreign providers capable of performing or subcontracting services previously performed in-house. PEM is deployed during six phases of the seven-phased eSourcing process [26].

There are numerous software estimation approaches [24; 44]. The most important ones rely on parametric estimation models [20; 43] that predict effort and/or cost based on historical project data and parameters such as the estimated size of software typically measured in lines of code or function points (fp) (e.g., COCOMO II [2], SLIM [36]). Measurement processes are crucial for managing system and software life cycle activities, assessing the feasibility of project plans, monitoring the adherence of project activities to those plans, and improving processes and products [18]. Estimation and measurement need to be closely linked to bridge project planning and execution throughout the project life-cycles. Yet, concepts combining both estimation and measurement process areas are scarcely available in the literature [29].

A comprehensive PEM concept and a commercially available software product supporting the concept have been developed by the Finnish Software Measurement Association (FiSMA). Several member organizations of FiSMA have been able to leverage the concept and the product in order to deliver their all projects consistently in time and in budget with the agreed upon functionality over the period of several years. In one member organization, the actual cost, time, and delivered functionality have systematically varied less than three per cent from the original estimates for several years. To our knowledge, similar results have not been obtained before.

The PEM concept consists of functional size measurement, delivery rate analysis, situation analysis, and reuse analysis. Functional sizes of the pieces of software to be produced are measured in function points by using standardized functional size measurement methods. Function points express the amount of business functionality an information system provides to users, independent of the technology used to implement the information system [17; 19]. The delivery rate is assessed in terms of the average number of development hours required in similar past development projects to deliver a function point. Situation analysis makes the estimates more precise by analyzing the factors that affect the development productivity or characterize the development circumstances. Reuse analysis further perfects the estimates by determining the rate of reuse of available domain artifacts and the reusability requirements of the pieces of software to be developed. The higher the rate of reuse, the lower the total effort needed in software development. The higher the reusability requirements of the software to be developed, the higher the total effort needed.

PEM is a set of knowledge-intensive business processes and thus critically dependent on effective information systems support. For example, internal project databases are needed to calculate the internal delivery rates.

978-0-7695-3450-3/09 $25.00 © 2009 IEEE

International software project benchmarking and estimation systems and databases [12; 32; 33; 38] can then be used cost-effectively to benchmark the internal delivery rates with those of best-in-class local and international eSourcing service providers and see whether the development can be done most cost effectively in-house or by the local or international providers [26].

There is relatively little research on software Project Estimation and Measurement Systems (PEMS) and how to design and use PEMS effectively for redesigning and enacting the PEM and eSourcing processes. Commercial PEMS [4; 5; 10; 12; 34; 37; 41; 42] vary greatly in functionality and effectiveness. The intended users for such systems thus do not know what to expect from the systems and how to evaluate them.

This paper creates an information system design product theory (hereafter, design product theory) for the class of PEMS. A complete information system design theory (ISDT) prescribes both the product and process aspects of a class of IS, that is, what are the meta-requirements, the meta-design, and applicable theories for all products within the class and how the products should be built [45; 46]. The paper focuses on prescribing the product aspects for the class of PEMS because the existing literature does not provide such a theory. Yet, the theory must be an integral part of the PEM concept because the PEM business process cannot be enacted effectively without using a PEMS instance. Meta-requirements and a meta-design are derived from the relevant literature, the PEM process, experiences obtained during more than ten years of empirical work in FiSMA, and a commercially available PEMS. It is beyond the scope of this paper to study the class of project management systems used for coordinating and scheduling projects with numerous partially overlapping and interdependent tasks.

2. Research background and method

The first author started the development of the PEM concept in early 1990s. Software industry was growing and the number, size, and complexity of projects were increasing in Finland. The industry needed better ways of scoping and estimating projects. Commercially available estimation systems and existing models for functional size (e.g., IFPUG 3.0) and cost estimation (e.g., COCOMO and SLIM) were studied but found inadequate. For example, the cost models required extensive calibration to suit the Finnish context which was not possible because historical project data was missing. The development of new estimation models and systems together with systematic but light-weight data collection and analysis methods to establish and continually grow the project data set were thus deemed necessary.

For this purpose, LATURI project was started in 1990 with 16 industrial partners, including the major banks, insurance companies, a nationwide trader and retailer, and an oil company, all of which had large internal software development departments. In addition, two large software providers were involved. Laturi followed the principles of design science [11], aiming at maximum practical utility for both consumers and providers while leveraging rigorous research methods to construct and validate the Laturi software and its key artifacts for functional size measurement, situation analysis, reuse analysis, and delivery rate determination.

In 1991, the first author initiated the development of a new FiSMA FSM method for functional size measurement (FSM) in a permanent work group of FiSMA. Versions of the method have been used in numerous organizations and further developed based on the lessons learnt. The first author has participated in the global standardization of FSM methods since late 1990s, enabling FiSMA to crystallize FiSMA FSM 1.1 into an international standard [19] that combines the best practices of the other leading methods COSMIC [13], IFPUG [14], Mk II [15], and NESMA [16] while eliminating their weaknesses. FiSMA FSM also benefited from the ideas of Capers Jones [21, pp. 118-124] who recognized the need to measure the complexity of mathematical algorithms in order to accurately determine functional size in computationally intensive application domains. FiSMA FSM 1.1 represents the most comprehensive, widely applicable (across business sectors and application domains), easy-to-use, and accurate FSM method internationally.

Initial productivity factors for situation analysis were created in 1991 through a Delphi study where each of the 16 partners identified ten most important productivity factors. As a result, 55 factors were identified. The partners then voted about the factors, resulting in 15 factors [3]. By 1996, a total of 40 consumers and providers had actively participated in research; a total of 182 projects with a median size of 521 function points had been collected to the Laturi database, and extensive statistical analysis of the project database had been conducted. The analysis revealed that the use of only 15 factors lead the situation analysis model to become too sensitive for mistakes in interpretations and/or valuations of the factors. As a result, the most sensitive factors related to functional and non-functional requirements and the skills of project team members were split in several new factors and the important *pressure on schedule* factor (i.e., the higher the pressure on schedule, the longer the project is estimated to last) from COCOMO was incorporated, resulting in 21 factors [6].

The Laturi software was instrumental for establishing the initial database and creating and validating the associated methods. However, it was too cumbersome for large-scale industrial use. Therefore, the first author led two productization projects in 1997 and 2000 further developing the concepts of Laturi into an easy-to-use commercial product called Experience®. A new project has been launched in 2008 to further improve the product.

Organizations buy the product incorporating the project database and pay an annual maintenance fee to receive new product versions and updated data sets. They can add their own data to the database and receive reduced maintenance fees for each project contributed when they donate their data to the shared database. Organizations collect data using the same product, the value of every project variable is precisely defined, and the algorithms and rules codified into the product are transparent to users and publicly known, ensuring the validity and comparability of the data. The first author has also contacted the organizations providing the data regularly over the years to verify and check their submissions and to understand the evolutions of their needs and work contexts holistically and longitudinally.

The rapidly growing project database enabled the use of analogy-based estimation for determining delivery rates by identifying the most similar completed projects to the project being estimated. The first author participated actively in developing the estimating algorithms based on the database together with international research collaborators [3; 22; 23; 25; 31; 32; 33; 40; 47].

3. Meta-requirements of the Design Product Theory for Software Project Estimation and Measurement Systems

The PEM concept involves project-, organizational-, and inter-organizational (network) level of learning (c.f., [1]). Figure 1 identifies the main parties and defines the tasks involved in PEM. Top management, project office, and project management are parties from the provider organization responsible for facilitating project-level and organizational learning. The measurement network consists of (1) member companies, government organizations, and universities; (2) stable and long-term working groups developing and maintaining the methods and PEMS; and (3) administration supporting the utilization of PEMS and enabling communication, remembering, and learning at the inter-organizational level.

The process structure of the project level of the PEM Concept (Figure 3) is aligned with the process model for managing project scope defined by the Guide to the Project Management Body of Knowledge (PMBOK) [35]. The Guide has defined five groups of project management processes (Figure 2) including five scope management processes and 34 other project management processes. The project scope management processes of PMBOK within their respective process groups are explained next:

- Initiation process (Initiating process group) specifies the preliminary project scope statement, that is, the high-level product requirements, project boundaries, acceptance criteria, and methods for controlling project scope that are commonly understood and agreed upon by all stakeholders.
- Scope planning process (Planning process group) develops the project scope management plan that states partly based on the preliminary project scope statement how the scope will be specified, controlled, and verified throughout the project life-cycle.
- Scope definition process (Planning process group) refines the preliminary project scope statement into a detailed one enabling the project team to decompose the statement into smaller, manageable work packages that can be reliably estimated in terms of required resources and duration.
- Scope verification (Controlling process group) obtains the stakeholders' formal acceptance of the project scope and associated deliverables throughout the project life-cycle.
- Scope change control process (Controlling process group) deals with factors that create project scope changes (e.g., deliverables unacceptable during scope verification) and controls the impacts of the changes.

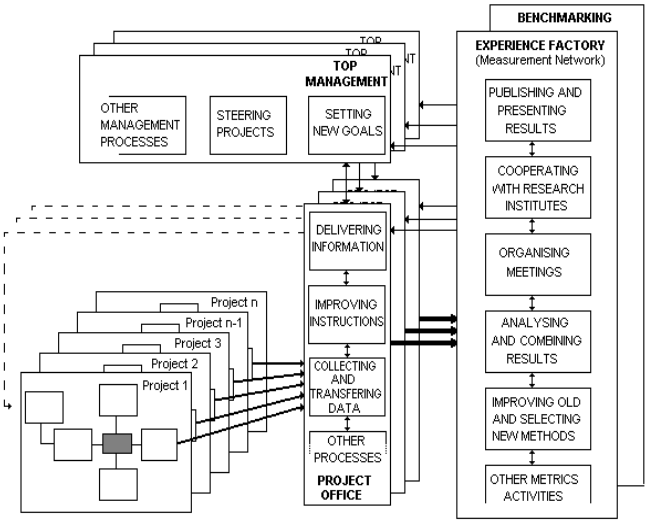

Figure 1. Project-, organizational-, and inter-organizational (network) levels of learning of the PEM concept

The project level of the PEM concept consists of five processes covered in more detail in Chapter 4:

- Initiating the project and the software to be developed. This process corresponds to the initiation process of PMBOK.
- Estimating Cost and Duration. This process corresponds to the scope planning and definition processes.
- Progress Controlling. This process corresponds to the scope verification process.
- Managing Changes. This process corresponds to the scope change control process.
- Closing the Development Project documents, analyzes, and measures the project, releases its results, and suggests actions that may improve the success of future projects. It not only measures the realized total effort, time, cost, and number of defects but links the

measures with the realized software size of the project in function points, enabling productivity evaluation, benchmarking, and the different levels of learning involved in PEM. There is no corresponding scope management process in PMBOK.

Figures 1, 2, and 3 pay little attention to information systems needed to institutionalize the PEM concept within the inter-organizational networks. Yet, PEMS are critical in all the levels of working and learning. To study the extant literature relevant to designing and using PEMS and to crystallize prescriptive meta-requirements for the class of PEMS on the bases of the review and extensive experiences obtained in FiSMA, the authors co-advised the M.Sc. thesis project of Matti Matikainen. In the following, a summary of the meta-requirements (MR) is presented mostly based on the FiSMA experiences. Matikainen [30] provides a review of the literature related to the meta-requirements.

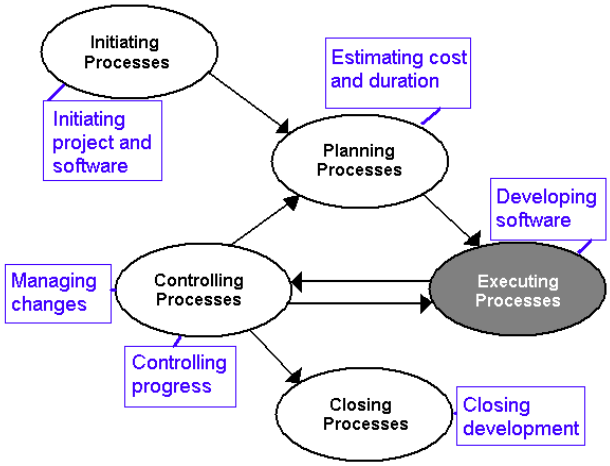

Figure 2. The PMBOK process groups (ovals) and their mapping against the scope management oriented processes (squares) of the PEM concept

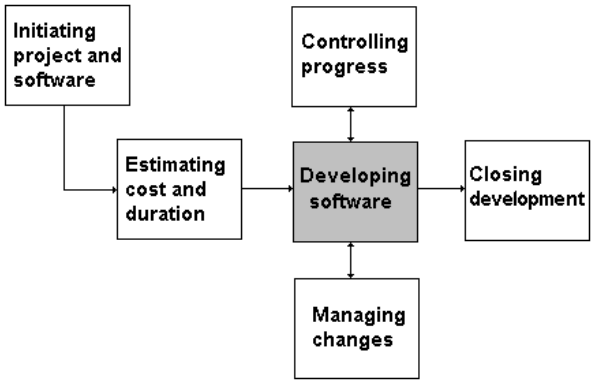

Figure 3. The PEM concept: scope management oriented project estimation and measurement processes

MR 1: PEMS shall support the project manager and acquisition management in effort estimation through

- functional size measurement of software
- reuse analysis
- situation analysis
- determining the delivery rate based on an experience database
- retrieving methodical knowledge from the experience database to enable functional size measurement, reuse analysis, and situation analysis
- performing the estimations and analyses and
- storing the estimates and other results of the analyses in the experience database.

MR 2: PEMS shall support the project manager in

- automated project-level macro-estimating (i.e., rapid and easy estimation of the entire project on a coarse level) and
- task-level micro-estimating (i.e., time consuming and laborious but accurate estimation of each activity and deliverable of the project).

MR 3: PEMS artifacts (i.e., methods and models codified in PEMS) shall support the initiation of the project and the storing of project data in a standard format by enabling the project manager to

- select the most relevant methods and models for functional size measurement, reuse analysis, situation analysis, and delivery rate determination
- utilize classification questions that enable the determination of delivery rates
- adjust available methods and
- define the target software.

MR 4: PEMS artifacts shall support project manager in cost and duration estimation by

- storing and collecting data related to project size, cost and duration in a standardized format
- identifying the characteristics of the development environment
- collecting functional user requirements
- determining non-functional requirements
- estimating reusability requirements
- calculating the estimated size, cost and duration, and
- determining the target delivery rate based on similar internal or external projects.

MR 5: PEMS artifacts shall support delivery rate determination through internal and external experience databases.

MR 6: PEMS artifacts shall support acquisition management, project manager, and project office in progress controlling through

- change control and estimate version control
- defining a readiness rate of the project deliverables and calculating a new readiness rate
- calculating a new project estimate
- comparing the previous and the new versions of the estimate
- storing updated measurement data, and
- providing comparison reports about progress.

MR 7: PEMS artifacts shall support change management and its linkage to the estimation process by

- analyzing the impacts of changed or new requirements on the project
- calculating a new version of the estimate based on changed functional size, situation multiplier, reuse rate, and delivery rate
- comparing previous and changed versions of the estimate, and
- providing comparison reports.

MR 8: PEMS artifacts shall support development closure by

- enabling the project manager to close the realized functionality and to define the final version of the project data including the actual duration and effort
- determining the actual productivity (e.g., function points/h) and delivery rate (e.g., h/fp) of the project
- providing and delivering the final reports to project manager, project office, and acquisition management.

MR 9: PEMS artifacts shall support new development, enhancement, maintenance, and modification projects.

MR 10: PEMS artifacts shall require

- effective requirements process throughout the project life-cycle (specifically during initial scoping and analysis and whenever new change requirements or constraints require change management and impact analysis) to yield high quality specifications for calculating functional size of the software
- evaluation of reusability requirements
- evaluation of resources and non-functional requirements
- information about changes
- information about project deliverables, and
- identification of target delivery rate.

MR 11: PEMS artifacts shall support the benchmarking of software and systems development and maintenance processes with the help of different databases

MR 12: PEMS artifacts shall support top-management decision-making, new goal setting and project steering through

- benchmarking and
- deriving product and process development goals from the business goals.

MR 13: PEMS artifacts shall support software process and organizational improvement by enabling the project manager, project office, and acquisition management to

- control progress and manage changes during the development project by continuously analyzing the project, process, product and people related situational factors
- benchmark the current situation to past performance within the organization and the measurement network
- analyze the productivity implications of IT tools and software engineering environments used in the projects
- improve development methodologies and tools and
- develop the competencies of the people involved and reassign responsibilities respectively.

MR 14: PEMS artifacts shall support data quality assurance to ensure excellent data quality and reliable support for decision-making and organizational learning.

MR 15: PEMS artifacts shall support classification, categorization, and sample selection of project data (e.g., business sector, project type) for

- delivery rate determination and
- benchmarking.

MR 16: PEMS artifacts shall support domain-specific and organization-specific benchmarking.

4. Meta-design of the Design Product Theory for Software Project Estimation and Measurement Systems

Participants and Systems Involved in Scope Management Processes at Project Level

Several actors are needed to make the scope management processes reliable and fluent. This chapter introduces all different groups and systems involved in the five processes of the PEM concept. Three main actors participating in all processes are acquisition management including both provider and consumer representatives, the project manager, and PEMS. Other actors are project team members, the project office of the organization, the measurement network (FiSMA), and project databases and other project management systems interfacing with PEMS.

Consumers are responsible for acquisition management and broader scope, business management, and development program steering. When a software product line provider is developing assets for strategic reuse across the product line, consumers typically include the top management and other internal stakeholders within the provider organization.

Initiating the Project and the Software to be Developed (MR 1, 2, 3, 9, 10, 15)

The purpose of initiating the project is to ensure that the project and software to be developed are manageable and measurable. The active participants of initiating process are acquisition management, project manager, and PEMS. Complementary information may be needed from project team members, project office, and sometimes from the measurement network.

The initiating process starts when the acquisition management asks the project manager to estimate the cost and duration of a new software development project. Project manager gathers all background information needed from the requirements and other documents and by interviewing stakeholders as necessary. Then he or she uses the PEMS to establish basic information about the project

(e.g., expected deliverables) and one or more pieces of software to be developed in the PEMS. The experience database contains several different classifiers (e.g., business sector of the customer, development type, and development tools and programming languages used) needed for project and software classifications. Project manager answers all questions asked by the software and then selects the appropriate estimation methods. The PEMS consists of several alternative methods that may be needed during estimating. Project manager makes her or his selection depending on the project type (e.g., new development or corrective maintenance) and recommendations of project office. Finally, the experience database is updated if there is new data available. Project manager may ask for the most recent data from the company project office or with its help from the measurement network. There are several project databases available for FiSMA members using PEMS.

The initiating process is usually immediately followed by the estimating process. Acquisition management typically doesn't need to formally accept the initiation. But the project manager may ask for advice from the management whenever there is substantial uncertainty about the measurability of the software or the manageability of the project.

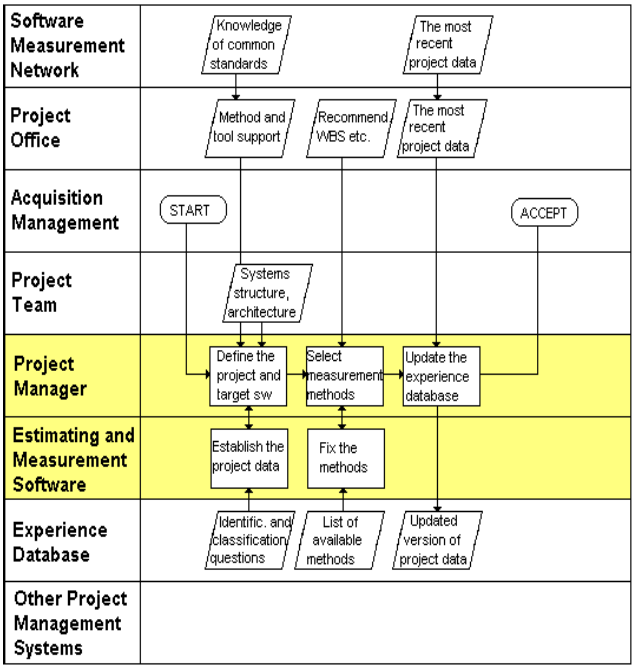

Figure 4. Initiating software development

The initiating process has become an increasingly important part of the PEM concept during this decade. It used to be the first part of the estimating process but this design lowered its perceived importance, particularly hampering the estimation and measurement of large projects. Too large projects need to be divided into multiple sub-projects preferably by the initiation phase and an appropriate work-breakdown-structure and project life-cycle model need to be selected to reduce project risk. Project initiation has thus been made an independent process. The process grouping of PMBOK [35] also helped to make this decision: the only process in the initiating processes group involves scope management.

Estimating Cost and Duration (MR 1, 4, 5, 10, 12, 15)

Estimating starts when acquisition management asks the project manager to estimate the effort and duration of the initiated project. The purpose of estimating is to collect and analyze all information affecting the effort and duration of the project. The deliverables of the process will be appendixes of the project contract and the project plan.

Project manager starts estimating by collecting information of all functional user requirements and entering data of functional components and their characteristics to the PEMS. The PEMS supports several alternative functional size measurement methods. The exact list of questions asked during this step depends on the FSM method selected, but the information needed is based solely on functional user requirements. Again, the project manager may find the information from requirements specifications or by interviewing the user representatives pointed by the acquisition management. The result of this step is the functional size of the software expressed in function points.

The second step of estimating is reuse analysis [7]. Reuse may remarkably increase or decrease the effort needed. If there are lots of reusable components available for the project, the effort estimate is reduced from the average level. If new reusable components need to be developed during the project, the total effort estimate will increase. Result of this step is the coefficient of different types of reuse, varying on both sides of 1. The PEMS calculates both the coefficient multiplier and reuse rate of the project.

The third step of estimating is situation analysis. Project manager collects information about circumstances of the project. There are two different sets of productivity factors implemented in the PEMS. The method selection is made based on the development project type, that is, new development and enhancement [6] or maintenance. In both sets there are questions about the organization, the development process maturity, the quality requirements of the target software and the skills and experience of the development project team. The result of situation analysis is a situation multiplier. The better the circumstances are the smaller is the multiplier. If everything is average, the value of the multiplier is 1.

Next, the project manager shall determine the delivery rate (expressed in hours per functional size unit, h/fp) to be applied in the estimate. The database of PEMS contains data from several project databases organized to support the searching of past projects analogous to the current project. Projects have been categorized by development type, business sector, target platform, and dep-

loyed software development tool. If the project manager's organization has systematically collected data from projects similar to the current project by all four criteria, the delivery rate is determined on this basis because the data best reflects the organization specific engineering and business practices likely to be applied to the project (c.f., [22; 23]).

When own applicable data is not available for finding analogies to determine the delivery rate, other databases need to be used. For the FiSMA network, the Experience® database (almost 900 projects from Finland by 2008) and the international ISBSG [12] database (more than 4000 projects from 28 countries in 2008) are available. Almost 10 per cent of all projects in ISBSG are Finnish projects, thus partially overlapping with Experience® database but the ISBSG database has a set of attributes slightly different from Experience®.

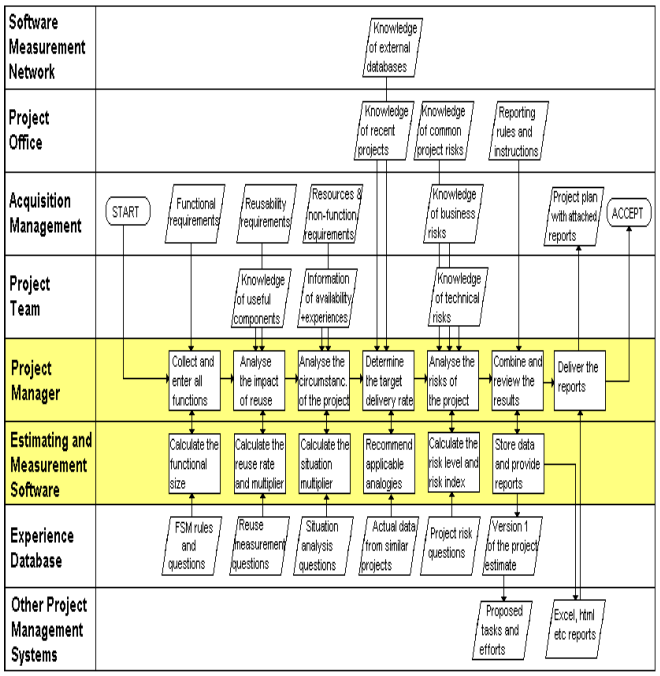

Figure 5. Estimating software development cost and duration

The fifth step of estimating is risk analysis. However, it is not a part of the design product theory. Based on the analysis of accumulated project data in the experience database of FiSMA, risk analysis does not affect the accuracy and other quality properties of cost and duration estimates. Moreover, there are no validated kernel theories for refining software project estimates based on risk analysis. It has been included in the estimating process because (1) members of the FiSMA network have explicitly required it to meet their practical needs and (2) it is possible that risk analysis can be developed in future research to enhance estimation and measurement to the extent that it can be incorporated in the design product theory.

Progress Controlling Process (MR 6, 13)

PEMS supports periodical controlling of the progress of the project. Acquisition management typically wants the progress reports bimonthly or monthly. In agile development projects, progress can be assessed after every sprint (i.e., weekly or even daily). Project manager retrieves the most recent estimate from PEMS, consults the project team as necessary to evaluate the minor changes in functional and reusability requirements and reuse of components that have occurred after the previous estimate was created, and updates the changes in PEMS. PEMS then calculates the functional size, reuse rate and multiplier, and situation multiplier (referred to as "variables" in Figure 6). Delivery rate can also be refined if it has proven unrealistic. PEMS then produces new versions of the effort and cost estimates. Next, project manager updates information about progressed deliverables (e.g., functional user requirements that have changed status from initiation to specified, designed, constructed, tested, or ready for installation). PEMS then calculates the readiness rate of the project in percentage (100 * the total functional size of the finished deliverables divided by the total functional size of all expected deliverables of the project). Readiness rate is similar to the concept of earned value, but earned value is typically expressed in Dollars or Euros rather than function points or percentages. The functions and phases matrix refers to a table, where the rows and columns represent, respectively, all functional user requirements and the phases of the applied work breakdown structure. Each phase has been allocated a certain percentage of the total effort during the initiating and estimating processes. For example, if specification accounts for 30 per cent of the effort and the size of a report is 5 fp, the specified report deliverable accounts for 1.5 fp in readiness rate calculation. The matrix is a crucial artefact of the design product theory because functional user requirements must be developed and tracked following a common life-cycle model, that is, each requirement must be specified, designed, constructed, tested, and prepared for installation. Finally, the project manager analyzes and delivers appropriate progress reports provided by PEMS to the project team, acquisition management, and project office and, when needed, has the possible changes in the project plan approved by acquisition management.

Change Management Process (MR 1, 7, 9, 10, 13, 15)

The desired outcomes change during the progress of the project. Existing requirements are clarified and new ones are introduced as the work progresses, and particularly when the first results are implemented, in both custom software development projects and in projects that result in software products. Changes in functional and nonfunctional user requirements affect project scope and thus need to be systematically managed from the early requirements elicitation phase throughout planning and executing [28; 39].

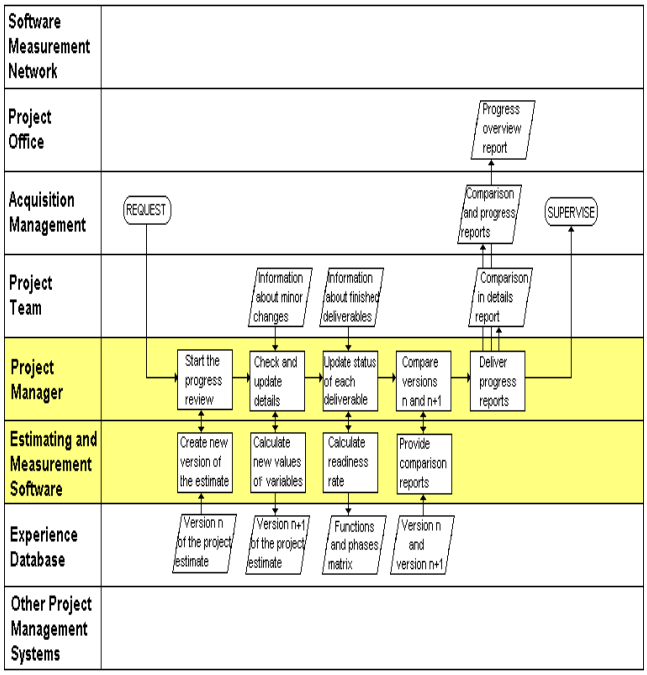

Figure 6. Process of controlling software development progress

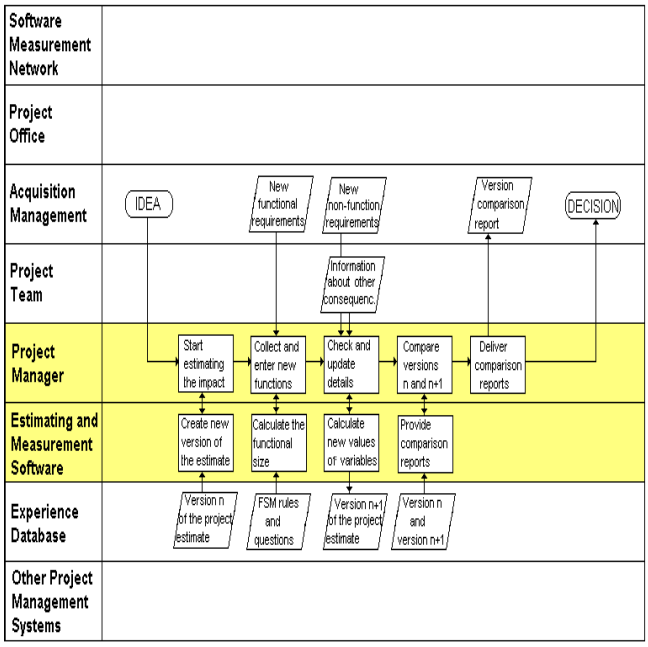

Figure 7. Process of managing software development changes

Change management process is started when acquisition management has a *set* of significant needs (referred to as *idea* in Figure 7) that have not been anticipated before and that will substantially change the scope if accepted for development. Such needs should be collected in the form of explicit change requests and analyzed throughout the project life-cycle [28]. Whenever possible, related requests (e.g., all requests related to security requirements) should be prioritized and batched together instead of dealing with them individually [27], their scope impact should be analyzed in a single instantiation of the change management process, and subprojects should be established to deal with the batches in order to improve efficiency and effectiveness of analysis and development work. If such a background process is not in place, numerous heterogeneous requirements may make change management very complex for the project manager.

The project manager collects the new functional requirements and uses PEMS to estimate their functional sizes and the total functional size of the software. PEMS typically helps the project manager refine and improve the quality of the requirements in collaboration with acquisition management because the functional sizing methods afforded by PEMS do not work well when requirements are unclear. Next, changes in non-functional requirements and other consequences of the changed scope (that is, reusability requirements, project situation, and delivery rate) are assessed and updated in the experience database and a new version of the effort estimate is calculated and stored in the database. Finally, a comprehensive report about the results of the impact analysis is presented to acquisition management who then makes the decision about the possible scope change.

Closing the Development Project (MR 8, 11, 13, 14, 16)

Normally, a project is closed when acquisition management decides that the desired results have been realized and the project tasks have been completed. In this case, the project manager (1) creates and stores the final version of the project information in PEMS, including the realized size of software, the effort spent in every phase of the project life-cycle, the duration of each phase, and the methods and tools used, (2) delivers the project reports provided by PEMS for acquisition management and project office, and (3) extracts project data for project office who may forward (parts of) it for use in external benchmark databases. PEMS calculates the realized delivery rate for use in estimating analogical projects in the future. To facilitate organizational and inter-organizational learning, project office is very active in analyzing and benchmarking the results internally with other projects, documenting the lessons learnt, and preparing and delivering data to external benchmark databases.

When the project does not reach the desired results but has to be ended anyway, the process is usually followed normally. For example, the realized functional size will simply be lower than anticipated. However, in the rare case of a complete project failure where there is no meaningful software deliverable to measure, no updates will be made to the project database to ensure high data quality.

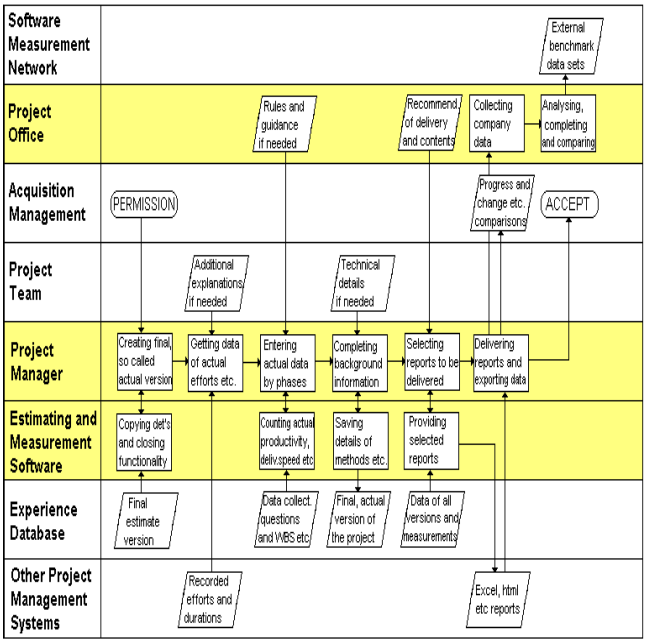

Figure 8. Process of closing software development

5. Conclusions and Future Research

The contributions of this paper are the holistic presentation of the PEM concept and the meta-requirements and meta-design of the design product theory for PEMS. They help practitioners develop their estimation and measurement processes and develop or acquire systems and associated databases to enable the processes. The referenced FiSMA sources facilitate detailed implementation of PEMS instances because even the detailed algorithms are available. The PEM concept has been registered in 2008 and published with the name northernSCOPE™ on the FiSMA web-site [8].

Due to space limitations, the design product effectiveness hypotheses of the theory (clarifying the expected organizational benefits from using a PEMS instance derived from the class of PEMS) and the data model of the experience database have been beyond the scope of the paper. The hypotheses are needed for the empirical validation of the theory in future research. The data model is a crucial part of the design product theory. Software providers typically need their own experience databases and those of the software measurement network and even other external service providers to yield reliable estimates and benchmark their development practices against best-in-class providers. Currently the databases are incompatible and relatively small with respect to the number of projects partly because no international standard based on a validated design product theory exists for the data model and partly because no standardized policies and procedures exist for data collection and analysis of the databases and for calibration and validation of the estimation models. Future research and international standardization efforts are thus needed to establish effective and agreed upon data collection and analysis practices and a data model for the databases associated with PEMS. Such a standardization project has been initiated in 2008 by International Organization for Standardization and both authors of this paper are involved in it.

To further validate the theory, our future research will conduct well-representative case studies in a few member organizations of FiSMA. The studies will analyze how and why the PEM concept and the commercially available PEMS aligned with the theory have helped (1) a few consumer organizations systematically acquire software successfully through domestic eSourcing and (2) a few software vendors systematically deliver what the consumers had wanted in time and in budget. We will also investigate a new job role of a professional Scope Manager, who can assist and advise project management groups in scope management throughout the development lifecycle. That role has recently been suggested to be instrumental in further improving the effectiveness of the PEM concept [9]. To increase the generalizability of the design product theory, we will analyze during the validation and refinement process to what extent the PEM concept and the design product theory are applicable in the U.S. context. The validated theory helps (1) system vendors to develop productized PEM systems and (2) the markets to know what to expect from the systems and how to evaluate them.

6. References

1 Basili, V.R., Caldiera, G. and Rombach, H.D. (1994). The Experience Factory. Encyclopedia of Software Engineering, Volume 1, John Wiley & Sons.

2 Boehm, B.W., Abts, C., Winsor Brown, A., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., and Steece, B. (2000). Software Cost Estimation with COCOMO II, Prentice-Hall.

3 Briand, L.C., El Emam, K., Surmann, D., Wieczorek, I., and Maxwell, K.D. (1999). An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. Proceedings of the 21st International Conference on Software Engineering, Los Angeles, CA, United States, 313-322.

4 Cost Xpert (2008). Cost Xpert Suite. http://www.costxpert.com/en/index.html.

5 DCG, David Consulting Group (2008). The DCG Industry Benchmark Database Query Tool. http://www.davidconsultinggroup.com/resource/industry/.

6 FiSMA (2001). Experience ND21 Situation Analysis Method. http://www.fisma.fi/wp-content/uploads/2006/09/fisma_situation_analysis_method_nd21.pdf.

7 FiSMA 2002. FiSMA Reuse Measurement Method, FiSMA RMM version 2002. http://www.fisma.fi/wp-content/uploads/2006/09/fisma_reuse_analysis_method_10.pdf.

8. FiSMA (2008). www.fisma.fi.

9 Forselius, P., Dekkers, C., Karvinen, M., and Kosonen, M., (2008). Program Management Toolkit for Software and Systems Development, Talentum Media, Helsinki, Finland.

10 Galorath (2008). SEER Project Management Tool. http://www.galorath.com/index.php/products/.
11 Hevner, A. R., March, S. T., Park J., and Ram S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75-105.
12 ISBSG (2008). International Software Benchmarking Standards Group, *http://www.isbsg.org/*.
13 ISO (2003a). ISO/IEC 19761: Software Engineering -- COSMIC-FFP -- A Functional Size Measurement Method. International Organization for Standardization, http://www.iso.org.
14 ISO (2003b). ISO/IEC 20926: Software Engineering -- IFPUG 4.1 Unadjusted Functional Size Measurement Method -- Counting Practices Manual. International Organization for Standardization, http://www.iso.org.
15 ISO (2003c). ISO/IEC 20968: Software Engineering -- Mk II Function Point Analysis -- Counting Practices Manual. International Organization for Standardization, http://www.iso.org.
16 ISO (2005). ISO/IEC 24570: Software Engineering -- NESMA Functional Size Measurement Method, version 2.1 -- Definitions and Counting Guidelines for the Application of Function Point Analysis. International Organization for Standardization, http://www.iso.org.
17 ISO (2006). ISO/IEC 14143 Information Technology -- Software Measurement -- Functional Size Measurement-Part 6. International Organization for Standardization, http://www.iso.org.
18 ISO (2007). ISO/IEC 15939, Information Technology -- Software and Systems Engineering – Measurement Process. International Organization for Standardization, http://www.iso.org.
19 ISO (2008). ISO/IEC 29881, Information Technology -- Software and Systems Engineering -- FiSMA 1.1 Functional Size Measurement Method. International Organization for Standardization, http://www.iso.org.
20 ISPA (2008). Parametric Estimating Handbook. International Society of Parametric Analysts. http://www.ispa-cost.org/ISPA_PEH_4th_ed_Final.pdf.
21 Jones, C. (2008). Applied Software Measurement, Third Edition. McGraw-Hill.
22 Jeffery, R., Ruhe, M., and Wieczorek, I. (2000). A Comparative Study of two Software Development Cost Modeling Techniques Using Multi-organizational and Company-specific Data. Information and Software Technology 42, 1009-1016.
23 Jeffery, R., Ruhe, M., and Wieczorek, I. (2001). Using Public Domain Metrics to Estimate Software Development Effort. Proceedings.of METRICS 2001: Seventh International Software Metrics Symposium.
24 Jørgensen, M. and Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering, 33(1), 33-53.
25 Kitchenham, B.A. (1992). Empirical Studies of Assumptions that Underlie Software Cost-estimation Models. Information and Software Technology, 34(4), 211-218.
26 Käkölä, T. (2008). Best Practices for International eSourcing of Software Products and Services. Proceedings of HICSS-41. IEEE.
27 Käkölä, T., Koivulahti-Ojala, M., and Liimatainen, J. (2009). An Information Systems Design Theory for Integrated Requirements and Release Management Systems. Proceedings of HICSS-42. IEEE.
28 Käkölä, T. and Taalas, A. (2008). Validating the Information Systems Design Theory for Dual Information Systems. Proceedings of the 29th International Conference on Information Systems, Paris (in press).
29 Laird, L.M. and Brennan, M.C. (2006). Software Measurement and Estimation: A Practical Approach. Wiley-IEEE Computer Society Press.
30 Matikainen, M. (2006). Information System Supported Project Estimation and Measurement. M.Sc. thesis. University of Jyväskylä, Finland.
31 Maxwell, K.D. (2002). Applied Statistics for Software Managers. Software Quality Institute Series, Prentice-Hall.
32 Maxwell, K.D. and Forselius, P. (2000). Benchmarking Software Development Productivity. IEEE Software, 17(1), 80-88.
33 Premraj, R., Shepperd, M.J., Kitchenham, B.A., and Forselius, P. (2005). An Empirical Analysis of Software Productivity over Time. *11th IEEE International Symposium on Software Metrics, IEEE Computer Society.*
34 PRICE Systems (2008). True S: Software Acquisition and Development.http://www.pricesystems.com/products/true_s.asp.
35 Project Management Institute (2000). A Guide to the Project Management Body of Knowledge (PMBOK).
36 Putnam, L.H. and Myers, W. (1991). Measures for Excellence: Reliable Software on Time, within Budget. Prentice Hall.
37 QSM, Quantitative Software Management (2008). Slim Tools: a Total Life-cycle Solution. http://www.qsm.com/products.html.
38 Rollo, A., Morris, P., Wasylkowski, E., Dekkers, C., and Forselius, P. (2008). ISBSG Benchmarking Standard, Version 1.0. http://www.isbsg.org/.
39. Salo, A. and Käkölä, T. (2005). Groupware Support for Requirements Management in New Product Development. Journal of Organizational Computing and Electronic Commerce, 15(4), 253-284.
40 Shepperd, M. and Schofield, C. (1997). Estimating Software Project Effort Using Analogies, IEEE Transactions on Software Engineering, 23(11), 736-743.
41 Softstar (2008). Costar 7.0 Software Estimation Tool. http://www.softstarsystems.com/.
42 SPR, Software Productivity Research (2008). SPR KnowledgePlan. http://www.spr.com/products/knowledge.shtm.
43 STSC (2003). Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems - Condensed Version 4.0. Software Technology Support Center, the U.S. Air Force. http://www.stsc.hill.af.mil/resources/tech_docs/.
44 Stutzke, R. D. (2005). Estimating Software-Intensive Systems: Projects, Products, and Processes. Addison-Wesley.
45 Walls, J.G., Widmeyer, G.R., and El Sawy, O. (1992). Building an Information System Design Theory for Vigilant EIS. Information Systems Research, 3(1), 36-59.
46 Walls, J.G., Widmayer, G.R., and El Sawy, O. (2004). Assessing Information System Design Theory in Perspective: How Useful was our 1992 Initial Rendition? Journal of Information Technology Theory and Application, 6(2), 44-58.
47 Wieczorek, I. (2001). Improved Software Cost Estimation: A Robust and Interpretable Modelling Method and a Comprehensive Empirical Investigation. Ph.D. Theses in Experimental Software Engineering, vol.7. Fraunhofer IRB Verlag.