INTEGRATED MANAGEMENT PROCESS WORKBENCH

A Step to Successful Software Project Management
IMP WORKBENCH

Integrated Management Process Workbench

ESPRIT PROJECT P 938

Prime Contractor
INTERATIONAL COMPUTERS LTD
Applied Systems Division,
Westfields, West Avenue,
KIDSGROVE, STOKE-ON-TRENT ST7 1TL,
United Kingdom.
Tel: 0782 771000 / + 44 782 771000
Contact: Derek Creasy

Partners
IMPERIAL COLLEGE
School of Management,
LONDON SW7 2PG, United Kingdom.
Tel: 01 589 5111 / + 44 1 589 5111
Contact: John Jenkins

CETE MEDITERRANEE
Informatique et Gestion,
B.P. 39,
13762 LES MILLES CEDEX, France.
Tel: 42 59 99 10 / + 33 42 59 99 10
Contact: Leopold Boi

VERILOG
150, rue Vauquelin,
31081 TOULOUSE CEDEX, France.
Tel: 61 40 38 88 / + 33 61 40 38 88
Contact: Bernard Vial

NATIONAL INSTITUTE FOR HIGHER EDUCATION
School of Computing Applications,
GLASNEVIN, DUBLIN 9, Ireland.
Tel: Dublin 370077 / + 353 1 370077
Contact: Renaat Verbruggen
This workbench is being developed under a grant from the CEC's ESPRIT programme (Project P 938).

IMPW is a major step forward in the provision of computer-based support for the project manager of software developments. It permits integrated planning of tasks, staff allocation, quality management and risk assessment. It also provides extensive monitoring facilities to enable the project manager control a project.

By using a database containing records of previous projects coupled to an inference engine, the workbench can provide decision support and expert system facilities to managers.

The workbench is thus unusual in that, instead of requiring the project manager's development environment to be changed to work in accordance with the workbench's own conceptions of the development process, IMPW allows for considerable adjustment for development models, quality standards, planning methods, resource estimation, risk analysis techniques and progress monitoring. It is aimed at medium-sized projects, typically involving 15-20 staff years development effort.

AUTHOR: JOHN JENKINS
COMPANY: IMPERIAL COLLEGE
REFERENCE: IMPC - WP9 - A - 010
ISSUE: 1
DATE: 1 SEPTEMBER 1987
STATUS: WORKING PAPER
CIRCULATION: PUBLIC DOMAIN
1. OBJECTIVES

The original objectives of the project were:

* To produce a prototype, software project management workbench, using a Portable Common Tool Environment (PCTE)-based graphics workstation, as a base on which to integrate software management tools.

* To produce or adapt existing software management methods and tools concerned with project planning and control, including improved estimation and monitoring techniques, and to integrate them on the prototype workbench.

* To be compatible with and complement the software project management consultant/instructor system (PIMS) being developed by project P814.

* To evaluate the prototype workbench and tools by carrying out industrial trials in different types of industrial environments, analysing the results and assessing the effectiveness of the overall system and recommending further developments.

The current objectives are:

* Production of prototype Software Project Management Workbenches on three different UNIX based graphics workstations.

* Evaluation of these prototypes in different industrial environments.

* A degree of compatibility with PIMS.

* Validation of proposed IMP workbench architecture as a basis for integration of tools.

2. ARCHITECTURE

The workbench consists of three main software components:

* The INFORMATION SYSTEM, consisting of a relational database system, holding the details of the project and software engineering characteristics, linked to a Prolog system containing knowledge about the methods and tools of software management and a document storage system based on UNIX filestore.

* The MANAGER-WORKBENCH INTERFACE which comprises the I/O manager responsible for all interactions with the manager and the Workbench controller responsible for tool control.

* The COMPUTATIONAL SYSTEM, that includes a range of tools and utilities (such as a report generator).

JJ4ABA
3. RESULTS

The main outputs from the project will be:

- three instrumented, prototype workbenches in the three industrial partners sites
- user manuals
- an assessment of the effectiveness of the workbench concept
- a "Recommendations" report
- multiple code "classes" that can be easily reused elsewhere, in object-orientated programming environment
- a project management knowledge base.

For a list of public deliverables (documents) to date see paragraph 8.

4. TOOLS

The toolset supports both the Software Planning and Monitoring functions. They have been specified using a management process model developed by the project teams and are integrated around a common data model. All tools interact with the Manager via a common user interface (see also 6).

* Project Definition

This requires a description of the product to be developed, identified by characteristics such as size, target customer, life time, etc. It also requires a set of quality objectives such as correctness, reliability, efficiency, etc. The tool will allow a project profile to be defined or to be chosen from existing project profiles. The profile will include components for the structure of the product, the structure of the development process, and for the organisation of the project.

* Risk Analysis

This involves getting information on previous experience within the project team and within the organisation. It also requires financial information and historical data on previous projects and their outcomes. It provides guidelines for management action and for assessing the accuracy of workbench estimates.

* Quality Assurance Planning

This tool helps a manager generate a Software Quality Assurance Plan. Different software quality assurance plans can be proposed, eg. IEEE standard, a company standard or customer standard.
* Network Editor and Work Breakdown

This allows a logical network of project tasks to be built up and manipulated. This process can be automatic, manual or a mixture of both. Automatic construction is based on the principles that the hierarchy of the network corresponds to the hierarchy of the product, and that for each level of structure of the product there is a related life-cycle.

* Estimation

The tool provides both 'top-down' and 'bottom-up' estimation facilities. It allows model-based estimation using estimates of lines-of-code or function-points; it also allows a separate estimate by comparison with previous developments and adjusting for differences. In the cases of both top-down and bottom-up analogy the PM is presented with advice to help with the comparison between past and present projects; this advice is partly statistical and partly knowledge-based inference.

* PERT and Critical Path Analysis

Using the network of project tasks, the estimates of effort, and the resources allocated, the tool produces time and resource-based plans. Various views of the plan can be presented, e.g. barcharts, individual assignments and networks.

* Resource Allocation

Based on the network of project tasks and the project profile, deliverable due-dates and milestones are computed and added to the plan. Using information on resources available to the project, individual task allocations are made.

* Progress Monitoring

This tool allows progress information to be examined from different points of view. This involves both resource consumption and schedule information. This data is gathered using progress forms which are produced by the workbench and completed by the relevant project members. It allows the detection of progress problems and their resolution using both algorithmic methods and knowledge-based inference.

* Quality Monitoring

This involves the evaluation of technical output of the project, using metrics according to the strategy set out in the Software Quality Assurance Plan. The tool then provides information on those items that
have been evaluated, when the evaluation occurred, who was involved, etc. It therefore provides information on the level to which the product achieves its goals.

* Calendar

This tool provides the manager with reports of the current status of the project, and a diary which allow appointments, meetings and project related milestones to be recorded and displayed in a convenient form. It also provides automatic notices for project members when deliverables are due.

* Utilities

These exist for reporting at various levels and for producing different views of the project plan. As well there are facilities for allowing the descriptions of personnel and development environments to be updated. There are also simple text processing, communications and document management functions for those items produced by the workbench. A basic 'browsing' capability will be provided.

5. The IMP Information System is the implementation of a model of the project manager's environment on top of the software architecture of IMPISH. IMPISH is based on a Logical and Relational Knowledge System, the result of a RDBMS and a Prolog integration, and a Document Storage System. The RDBMS and Document Storage System ensure the basic storage and retrieval of information. Coherence, constraint verification and deduction process are allowed by use of Prolog rules. The most important and specific characteristics of IMPISH are the following:

   a) Query Language, SQL compatible, enriched with the logical capabilities provided by the inference engine,
   b) the capability to link external prolog programs to its data and document bases.
   c) the partial management of coherence between documents and modelized information.
   b) the flexibility of its interface.

The model implemented in IMP Workbench, on top of IMPISH, is the result of investigation and tool specifications. It ensures the coherence of the data shared by the tools.

6. USE OF PCTE

We intend to make the IMP Workbench specifications PCTE portable. Our current view of PCTE OMS is that the UNIX file system is the more mature technology and hence it is the most likely candidate for our document storage system.

The workbench I/O Manager component needs to use graphics based window
facilities. PCTE UI was developed specifically to provide these for software development tools.
We have investigated the PCTE UI and found it to be suitable for our needs at the specification level. Unfortunately there is no satisfactory implementation planned to be available in our timescales. Also we believe that the PCTE UI specifications have not yet stabilized. We are targeting the I/O Manager component to a portability layer that we can implement on X-Windows. We expect to be able to port to PCTE UI when it becomes stable and is available.

8. DELIVERABLES

D.1.1 (a,b,c,) General specification and Design of IMPW Project
Software Quality Assurance Plan.
D.3.1.a Specification of Workbench User Functions
D.5.1.a Survey of Cost Modelling
D.6.1.a Workbench Evaluation Factors and Criteria

9. MANAGEMENT

Prime contractor: ICL, Westfields, West Avenue, Kidsgrove,
Stoke-on-Trent, UK
Tel. +44. 782-771000 Telex 22971
Contact Person: Derek Creasy

Partners: Imperial College School of Management, London U.K.; CETE Mediterranee, Aix-en-Provence, France; Verilog SA, Toulouse, France;
National Institute for Higher Education, Dublin, Ireland
Project Objectives

The software development industry has experienced numerous major development projects that have run over budget and have not delivered systems on time. In addition many systems had less functionality than was originally planned.

The IMP Workbench plans to produce prototype management workbenches which will help project managers plan and monitor software projects from the requirements phase to implementation and handover to customers.

Key Features

- Integrated knowledge-based software project management tools.
- State-of-the-art manager-workbench interface.
- Configurable to different software development environments.
- Extensive historical project database.
- An advanced estimation facility.

Project launched: 26 February 1986

Effort planned: 42 man-years over 3 years

Further Information

A leaflet describing this project in more detail can be obtained from any of the partner organisations.
12 October 1987

Presentation for the Third Annual COCOMO's User Group meeting, Pittsburgh, Nov 1987.

COST-ESTIMATION AT THE I.C.S.M.

ADRIAN COWDEROY AND JOHN JENKINS

Imperial College School of Management
Exhibition Road
London SW7 2BX

This short paper summarises the work being done at the Imperial College School of Management (ICSM) in the development of methods for cost-estimation. The work falls into 4 categories:

- development of an application generator, for the calculation and inference processes used within estimation tools, to allow easy creation and modification of algorithms and knowledge
- study of potential uses of general and local knowledge within estimation tools
- statistical support for estimating costs by comparison with previous developments
- development of a prototype estimation tool using the above techniques and an icon-driven user interface

This illustrates some of the lines of development in which the art of cost-estimation may proceed in future years.

This paper assumes familiarity with the general concepts of parametric cost-estimation for software development.

The work is being done as part of the Esprit project P938, a 3-year project to develop a workbench of integrated tools suited to software development managers; it is partly funded by the Commission for the European Communities.
The Estimation Modelling Language Interpreter

The EML-I (Estimation Modelling Language Interpreter) [1] makes it possible to specify calculations by simple code-like statements within a text-file, for example, the Basic COCOMO model [2] might be expressed as:

```plaintext
@ Estimate effort (MM) and time (TDEV) in units of months, using code-size (KDSI) in units of k-lines of non-comment source code
IF (MODE = "organic") {
    MM = 2.4 * KDSI**1.05 ;
    TDEV = 2.5 * MM**0.38 ;
}
IF (MODE = "semidetached") {
    MM = 2.4 * KDSI**1.05 ;
    TDEV = 2.5 * MM**0.38 ;
}
IF (MODE = "embedded") {
    MM = 2.4 * KDSI**1.05 ;
    TDEV = 2.5 * MM**0.38 ;
}

@ Calculate proportion of effort & time for each phase & activity
IF (KDSI<4 AND MODE="organic") {
    EFFORT[plans_N_requirements] = 0.06 * MM ;
    EFFORT[product_design] = 0.16 * MM ;
    EFFORT[detailed_design] = 0.26 * MM ;
    EFFORT[coding_N_unit_test] = 0.42 * MM ;
    EFFORT[integration_N_test] = 0.16 * MM ;
    TIME[plans_N_requirements] = 0.10 * TDEV ;
    TIME[product_design] = 0.19 * TDEV ;
    TIME[programming] = 0.63 * TDEV ;
    TIME[integration_N_test] = 0.18 * TDEV ;
}

@ etc, for other code size and modes
@ The above could also be expressed (in less space)
@ by defining arrays of data
```

Such an EML-I is not a free-standing product for it has to exist within other tools which provide both a user-interface and a store for the data; the EML-I has to be configured to be aware of the names and format of data stored by the parent product. Such a parent product would typically be an estimation tool, which would no longer need to do calculations for itself.

The difficulty of porting an EML-I from one estimation tool to another is one of the major problems [3] of this approach: sophisticated tools such as IMFW use relational databases and the resulting data distribution is extremely complicated compared with conventional models such as COCOMO.
There are two main objectives for such an EML-I:

(1) Ease of change

Not only are the calculations easy to establish and debug, but making minor changes to them becomes easy enough for a sophisticated project-manager to attempt - it is no more difficult than a spreadsheet or the BASIC programming language. It is thus possible to offer estimation tools that customers can tailor to their own needs (encouraging them to use approaches such as the Bailey & Basili meta-model [4]), rather than being restrained by a particular set of assumptions.

(2) Variety of possibilities

With calculations being easy to write and debug it now becomes possible to provide a greater number of such calculations within an estimation tool. It is also possible to write conditionals and similar statements to express various forms of knowledge (see below). The range of potential calculations and problems that can be handled thus widens; for example:

- cost-model calculations
- selecting when a cost-model can or should be used
- provision of various statistical techniques for correcting estimates following observed project progress
- for estimating the final code-size or function-point count, assuming that the client will keep changing the requirements
- converting available metrics into forms suited to models such as COCOMO
- checking on the likely validity of the supplied data
- re-calibrating cost models and validating their accuracy

This also highlights the second major disadvantage of the EML-I approach: it is too sophisticated for a small cost-estimation tool, because people change their estimation facilities infrequently; only with sophisticated estimation tools, involving large number of different types of calculation, does its use become practical.
Inference engine

Consider a simple representation of one form of knowledge relating to the estimation process:

```c
IF ( product_size="small"
    OR estimate_requirements="rough"
    OR (est_risk="high" and product_size="medium")
)
    OBEY(Basic_COCOMO)
ELSE OBEY(Int_COCOMO);
```

In this simplistic example a few rules of thumb are used to describe when to use Basic COCOMO and when to use Intermediate COCOMO. But despite the simplicity of the statement, it does contain useful information and it is easy to understand. In a product that makes extensive use of EML-I command-files this provides a method of expressing the simpler forms of knowledge in a form that is easy to check.

Now consider the same example written in a different form, which could potentially also be handled by an EML-I if it were suitably defined:

```c
@ create object for logical statement
CREATE(Use_small_model, logic);

@ add all known knowledge to logical statement
I_KNOW(Use_small_model, product_size="small");
I_KNOW(Use_small_model, estimate_requirements="rough");
I_KNOW(Use_small_model, est_risk="high"
    AND product_size="medium");

@ use inference engine to deduce result
IF INFER(Use_small_model)
THEN OBEY(Basic_COCOMO)
ELSE OBEY(Int_COCOMO)
```

Using this approach it is possible to produce very much more complex logical statements, allowing people to represent their knowledge in a more "natural" fashion. This kind of ability represents a potential medium-term goal for the EML-I concept.
An analogy

Using an estimation tool that includes an EML-I it is also possible to encompass a formalised version of analogy-based cost-estimates. This would be used as a supplement to existing methods (such as use of a cost-model, or asking the local wiz-kid, or consulting an astrologer). Estimates are made by comparing the new development with a previous one of a similar nature, and using COCOMO-like attributes to estimate the relative change in cost; the process is best repeated for several previous projects and the results compared.

Example: consider a proposed new product being produced within the same development area where there is still some experience from a previous version produced several years earlier:

old_project   new_project
size = 57 kb   size = "probably 20-40% larger"
cmplx = high   cmplx = "similar?"
rely = norm    rely = "slightly higher than before"
aexp,vexp,lexp = norm    aexp,vexp,lexp = "gained experience"
modp/tool = "1979"       modp/tool = "1987 standards"
sched = norm   sched = similar management pressure
cost = 320 staff-months

Using Intermediate COCOMO cost-driver tables and (for the example) ignoring the slight exponential increase in cost with size, we might estimate for the new project:

\[
\text{expected}\_\text{cost} = 320 \times 1.30 \times 1.00 \times 1.15 \times (0.91 \times 0.90 \times 0.95) \times \\
(0.91 \times 0.91) \times 1.00 \\
= 308 \text{ staff months}
\]

\[
\text{low}\_\text{cost} = 320 \times 1.20 \times 0.93 \times 1.12 \times (0.88 \times 0.87 \times 0.92) \times \\
(0.88 \times 0.88) \times 1.00 \\
= 218 \text{ staff-months}
\]

\[
\text{high}\_\text{cost} = 320 \times 1.40 \times 1.07 \times 1.18 \times (0.94 \times 0.93 \times 0.98) \times \\
(0.94 \times 0.94) \times 1.00 \\
= 428 \text{ staff-months}
\]
This approach to estimation has several advantages:

- the estimation process can feel more natural to many project managers, who may consequently trust it more, making it less easy for them to claim that the new project is an exception to which cost-model do not apply (and that their blind-guess is consequently better)

- the analogy-approach is less dependent on direct estimates of codesize, since this is now being made by comparison with specific previous examples

- the exercise of encouraging people to review previous projects in planning a new one can have various advantages, such as the highlighting of potential organizational and technical problems

The analogy approach also has disadvantages:

* there is a lack of mathematical rigor

* the approach is open to misuse if the project-manager wants the estimated cost to meet a particular target

* the estimates are more time-consuming to produce

References


Copyright (c) Imperial College of Science and Technology 1987.