PERGAMON

International Journal of Project Management 20 (2002) 461–468

INTERNATIONAL JOURNAL OF PROJECT MANAGEMENT

www.elsevier.com/locate/ijproman

Integrating system analysis and project management tools

Roy Gelbard[a], Nava Pliskin[b,*], Israel Spiegler[c]

[a]Department of Industrial Engineering, Tel-Aviv University, Israel
[b]Department of Industrial Engineering and Management, Ben-Gurion University, Israel
[c]Department of Information Systems, the Recanati Graduate School of Business Administration, Tel-Aviv University, Israel

Received 23 August 2000; received in revised form 28 November 2000; accepted 1 May 2001

Abstract

Currently, computer-aided tools for system analysis are distinct from project management tools. This study proposes and prototypes a model that integrates these two aspects of the Information System Life Cycle (ISLC) by automatically mapping system analysis objects into project management objects. To validate the feasibility of our model and without loss of generality, the conversion of Data Flow Diagrams (DFD) objects into Gantt and Pert diagrams is demonstrated in this study. Experiments with the prototype confirm that integrating common tools for system analysis and standard tools for project management, during system development, helps improve system building tasks and their management. In addition, project managers using the proposed mapping approach can better assess project duration and system performance parameters such as response time and data traffic. We address implications of our work to both academics and practitioners, discussing directions future research might take as well as opportunities and prospects for commercialization of the proposed approach.

Keywords: Software engineering; System analysis and design; Project management; Data Flow Diagram (DFD); Gantt diagram; Pert diagram; Information System Life Cycle (ISLC)

1. Introduction

Computer-Aided Software Engineering (CASE) tools support the analysis, design, construction, and implementation stages of the Information System Life Cycle (ISLC) [1,2,26]. Although Project Management (PM) tools support management and control of ISLC development tasks [4,5], there is hardly any integration to date between CASE and PM tools. Thus, ISLC modeling approaches, such as Data Flow Diagrams (DFD) or Unified Modeling Language (UML) [6], even when automated, are used in the early analysis stage primarily for visual documentation. The "database of specifications", laboriously elicited and gathered during the creation of modeling diagrams, is hardly ever applied again for project management purposes, even though this information is valuable for project managers who are involved in the construction and implementation stages. In fact, due to lack of integration along the ISLC, the specifications database is often either overlooked altogether or collected again as if their creation earlier never took place. Moreover, standard methods for system analysis and development usually make no reference to methods for project management [7].

Given the current object orientation of both system analysis and project management methodologies, this paper contends that a model integrating these two aspects by automatically mapping system analysis objects into project management objects is not only desirable but also feasible. The need for such mapping emerges from the next section that contains a theoretical and practical review of DFDs, Functional Hierarchies (FH), Pert and Gantt diagrams, and the corresponding computer-aided tools. The focus of this study on dataflow, Gantt, and Pert diagrams are done, without loss of generality, for demonstration purposes of our model to validate the model and its feasibility.

Based on Section 2, it becomes clear that a major problem in system development is the gap between the analysis phase, which deals with the required *functionality* (i.e. the Requirements Engineering phase [8]), and the

* Corresponding author. Tel.: +972-8-647-2203; fax: +972-8-647-2958.

E-mail addresses: gelbard@post.tau.ac.il (R. Gelbard), pliskinn@bgumail.bgu.ac.il (N. Pliskin), spiegler@post.tau.ac.il (I. Spiegler).

0263-7863/02/$22.00
PII: S0263-7863(01)00044-8

construction phase, which deals with the actual program *coding*. Sections 3 and 4, respectively, describe and prototype a model that maps system analysis objects, based on DFDs or FHs, into project management and control objects, in the form of Gantt and Pert diagrams. On the basis of this mapping, it is also possible to run a Critical Path Method (CPM) analysis with respect to "DFD component chains" paths. Section 4 shows that bridging analysis and construction based on our approach is more objective than without our approach because raw DFD components are stored in the specifications database. Because of its automation, the proposed mapping can shorten project duration, contribute to efficient use of resources, and help estimate the required development effort and cost as well as the expected system performance, complexity, and quality. In the concluding section, we sum up the study and address the implications of our work to academics, in terms of directions for future research, and to practitioners, in terms of opportunities and prospects for commercialization.

2. Background

CASE tools are trying to automate the entire ISLC and commercial tools, such as Rational-Suites [www.rational.com/products], are covering main stages of ISLC. The "Rational Requisite-Pro" tool, for instance, is designated to the stage of requirement definition and analysis, "Rational Rose" to the analysis and design stage, "Rational Test-Studio" to the testing stage, and "Rational Clear-Case" to configuration management usage throughout the entire ISLC (including the production and maintenance stages). Since a complete review of ISLC tools is beyond the scope of this paper, the reader is referred to such studies as Barker [9], Pressman [2], Roberson [8], and Sommerville [10]. Similarly, this section does not attempt a complete review of the project management area, other than for the purpose of comparing the area of software engineering, and the reader is referred to Agnus [11], and Kerzner [12].

One conclusion that emerges from a thorough review of both areas is that tools for systems modeling and software engineering are much more heterogeneous than project management tools. According to Fox and Spence [13] and Pollack-Johnnson and Liberator [14], Gantt and Pert diagrams have become dominant project management modeling tools [15] and are currently included in standard PM software such as MS Project. A survey of 1000 project managers has found that 48.4% use MS Project, 8.5% use MS Excel, and the rest use Gantt/Pert-based tools from other vendors [13]. The average satisfaction from PM tools in this survey was 3.7 on a scale of 1–5. Another survey reveals that only 10% of 240 project managers do not use PM tools at all, down from 33% in 1996 [14]. Moreover, more than 50% use Gantt/Pert-based project management software to manage every project, independent of its application domain and characteristics.

In contrast, the following three commercial CASE software packages demonstrate the heterogeneity of tools in the area of software engineering. PowerSoft (by Sybase) offers *Power-Designer*, that supports DFDs, for procedural system modeling, and Entity Relationship Diagrams (ERD), for database modeling. Oracle's *Designer 2000*, supports Functional Hierarchy Analysis and ERD [9]. Finally, Rational offers *Rose*, a UML modeling tool. Project management tools thus seem more standardized and mature than CASE tools. This could be the reason why 71% of 397 software engineers surveyed in 20 European countries employ PM tools while only about 26% utilize CASE tools, despite similar levels of training [16].

Although the leading CASE tools mentioned earlier support teamwork, none contain elements for time planning and control that take into consideration teamwork, dependencies, and resources. Moreover, none include Gantt or Pert models or offer built-in interfaces to PM tools such as MS Project.

According to Reel [3], methodologies and models for managing software projects have yet to make it from the idea to the product phase, despite persistent improvements in automated tools for requirement definition, systems modeling, and software engineering. The failure to transform project management theory to practice in the context of software development is especially troubling since more than 50% of such projects do not succeed. In addition to the lack of PM tools for software development, Reel also observes that managers in charge of software projects usually refrain from basing managerial judgement on data about requirements and functional characteristics of the specific development project.

With decades of systems development behind us, there is quite a consensus today with respect to the Critical Success Factors (CSF) of system development projects (Reel, [3]). Sawyer et al. [17] and Roberson and Roberson [8], among others, acknowledge that assembling a set of clear requirements, which is critical to success, is both expensive and time consuming. In a review of 18 non-automated tools for managing corporate IT, Ahituv et al. [18] classify the tools by their functionality: periodic training (7 tools), project feasibility analysis (12 tools), and project progress control (1 tool). They note, however, that in all but one of the approaches, utilization of CASE and PM tools is recommended. Boegh et al. [19] and Jiang and Klein [20] join many other authors in describing the need to introduce *effective* measures for better control of software projects. They mention, for example, that lines of code (LOC) or cost estimators are rather general and related to the specific

project or the specific organization. Another important capability for assuring product quality is *traceability* — the need to deal with implications of changes (Change Control and Configuration Management) and be informed at all times about project progress in terms of completed function points [21,22]. According to Kautz [23], tools capable of forecasting, tracing, and controlling product quality are rare. All these observations lead one to conclude that assembling a repository of system requirements and system components, complete as it might be, does not guarantee effective planning of team work, scheduling of tasks, and controlling deviations between planned milestones and actual progress.

Against this background, the questions to consider are:

1. Is the gap between CASE and PM tools bridgeable?
2. What can be done to smooth the transition between CASE and PM tools?
3. How can analysis and design data, collected by CASE tools, become directly available to planning and control processes, supported by PM tools, without being subjectively interpreted or biased?
4. Is there a way to improve software modeling and engineering by introducing a managerial perspective in addition to the technical perspectives?

Our preliminary answers to those questions are as follows:

1. Yes, it is possible to bridge the gap between CASE and PM tools using mapping models as the required bridges. DFD mapping into Gantt–Pert Diagram, for example, could be regarded as validity and feasibility proof of this concept.
2. Transition between CASE and PM can be achieved using existing commercial tools. There is no need to develop new tools, which will enable analysis tasks as well as management tasks. It is sufficient to look for solutions to bridge tools from both areas.
3. Analysis and design data, collected by CASE tools can become directly available to PM tools. This can be achieved if and only if repositories of both kind of tools are shared.
4. A managerial perspective is essential, as evident from many case studies reflecting the need to introduce a managerial perspective in addition to the technical perspectives.

Creating bridges between the components of system analysis as defined by Pressman [2], ERD, DFD, and State Diagram (SD), is beyond the scope of this paper and we must choose among the three, one that would best serve to demonstrate the feasibility of bridging.

- ERD describes the logical functional database (DB) schema. Since it is relatively simple to manage this activity, ERD can not be regarded as a suitable candidate for demonstrating how to bridge the gap between technical and managerial tasks.
- An SD can be decomposed into Activity Diagram and, according to Fowler [6], an Activity Diagram can be decomposed into an SD. Since an Activity Diagram looks quite similar to a Gantt Diagram, mapping SD into Gantt–Pert Diagram cannot be regarded as a suitable candidate for demonstrating how to bridge the gap between technical and managerial tasks.
- Considering these limitations of ERD and SD as candidates for mapping demonstration and aspiring to discuss a higher-level integration of analysis and management, DFD modeling seems most suitable for validating bridging feasibility, especially since conversion from DFDs should be more difficult and more challenging than from ERD or SD.

We have engaged in symmetry–isomorphism research with respect to distinct methodologies for software engineering and project management. Since Pert and Gantt diagrams are techniques for visual description and charting networks, the ability to convert the DFD model to a network format is at the basis of our symmetry–isomorphism research. It is our intention in this paper to demonstrate, based on this research, a possible integration scheme and provide more robust answers to the earlier questions. Given the wealth of CASE and PM tools, this work refrains from developing yet another one. Instead, we prototype a platform for integrating a well known CASE tool, *PowerDesigner* from PowerSoft-Sybase, and a widely used PM tool, *MS Project* from Microsoft. We show that combining the two sets of capabilities can create the desired synergy where the whole is greater than the sum of its parts.

3. Mapping model

The problem that we face is that the lengthy and costly ISLC is fostered by disparate tools without effective consolidation of data generated throughout the overall ISLC. Hence, the present study focuses on mapping DFD and FH systems modeling objects into Gantt and Pert project management and control diagrams, thereby integrating these two ISLC aspects. For the sake of brevity, we present below DFD representation examples, arguing that mapping from a DFD to a Gantt diagram is more complicated, and thus more general, than from a Functional Hierarchy to a Gantt diagram. Our model, general in scope, can handle both DFD and FH methods.

3.1. Mapping DFD objects into Gantt objects

Basic mapping of DFD objects into Gantt diagram objects is based on the following conversions:

1. Each of the external entities are represented only once for input (if they produce input) and only once for output (if they produce output).
2. Each Read Only (RO) data store and each Read/Write (R/W) data store are represented only once for input and for output.
3. Each basic flow appears only once in every Gantt diagram.
4. Each basic process appears only once in every Gantt diagram.
5. OR connections between flows are not represented in the absence of parallels in Gantt diagrams. Logical connection traits between flows can, however, be included within basic process characteristics, thus maintaining mapping completeness.
6. A general process is represented by means of a summary task, i.e. a grouping of activities and flows under a general name.
7. General flows are those that connect between summary tasks.

3.1.1. Example: mapping of hierarchical DFD

The DFD methodology enables hierarchical analysis of systems. The hierarchical description is achieved by "blowing-up" General Processes into dedicated diagrams. Those dedicated diagrams represent lower level descriptions and can be composed of Basic Processes as well as General Processes. The hierarchical description can be halted when there are no more General Processes at the lowest-level diagrams. Except for the root level, identified as "DFD-0", each diagram in the DFD hierarchy is identified by the respective General Function.

For the sake of simplicity, the DFD used in the examples below contains only the following objects: basic processes, general processes, basic flows, external entities, and data stores. Processes are symbolized by *ellipses* and denoted by **P**#, entities by *rectangles* and **E**#, and flows by *arrows* and #.

Figs. 1 and 2 demonstrate a hierarchical DFD. Fig. 1 describes the root level with Basic Process P2 and General Process P1 (a General Process is depicted by concentric ellipses). Fig. 2 describes a lower level description of the General Process P1.

A composite DFD, made up from both DFD-0 (Fig. 1) and DFD-1 (Fig. 2), is shown in Fig. 3. A composite DFD, not a typical or common representation of an hierarchical DFD, is included here because of its similarity to Gantt diagram representation, where Summary Tasks and Subtasks can be displayed on the same diagram. A Summary Task represents, in the mapping model, a General Process, while each a Subtask represents a component at the respective hierarchical DFD level.

As can be seen in Section 4, Fig. 4 depicts the Gantt diagram corresponding to DFD-0 in Fig. 1, and Fig. 5 depicts the mapping of the composite DFD in Fig. 3 into a Gantt diagram.

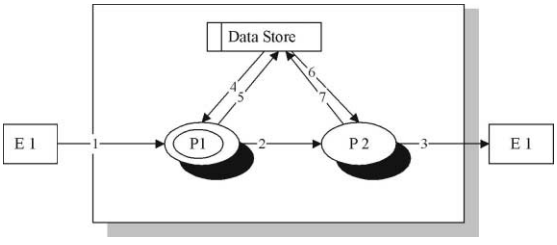

Fig. 1. DFD-0.

3.2. Improving the basic mapping

The basic mapping model, presented earlier, can be modified to improve the output Gantt diagram along the following lines:

1. Enabling Gantt's activity dependencies to be different from DFD dependencies. In the basic mapping model, Gantt activities are in the same sequence order as in the DFD. For managerial purposes, however, Gantt activities have to fit a project's building blocks and their dependence relationships, not just DFD dependencies.
2. Decomposing feed flows from and to external entities into the following types: User interface

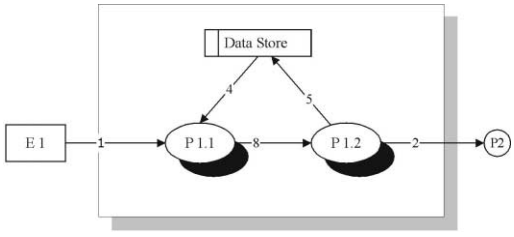

Fig. 2. DFD-1 (General Process P1 "blown-up").

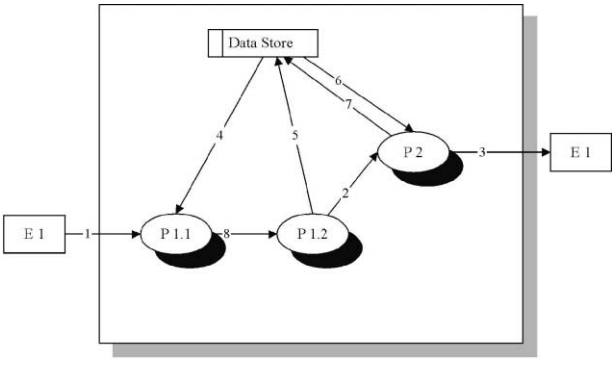

Fig. 3. A composite DFD representing both DFD-0 and DFD-1.

(screen), request for a report, and fixed-time or real-time activity alerts.
3. Decomposing data-writing flows into the following types: Insert, Update, and Delete.
4. Applying the additional possible attributes of DFD components listed in Table 1.

3.3. The integrated repository

As mentioned earlier, customization of the database structure (DB schema), which stores the DFD data components, is required to enable mapping DFD activities and dependencies as needed for Gantt diagrams. Tailoring the DB schema, as required, is illustrated in Fig. 6, which presents the DB schema that supports both DFD and Gantt objects. A repository of a commercial tool containing such a schema will enable integrated managerial and engineering data interchange.

The repository contains three main components:

1. <**DFD Components Dictionary**> = Tables: [DFD COMPONENTS], [COMPONENT TYPES], and [PRECEDENCES].
2. <**Data Items Dictionary**> = Tables: [DATA ITEMS], [ITEM SYNONYMS], [DOMAINS], and [RELATION GROUPS].
3. <**Ascription of Data Items to Basic Flows**> = Table: [DATA ITEMS in FLOWS].

While these components enable representing and storing of DFD objects, they also enable representation of Gantt objects. The [PRECEDENCES] Table stands for many-to-many network relationships as required for a Gantt representation.

In Fig. 6, rectangles represent database tables, with the table name is contained in the black header and the primary key bolded. Lines between rectangles represent the database constraints (foreign key) and indicate the cardinality (one-to-many) of the relation. To distinguish between the various DFD components, each component, in the [DFD COMPONENTS] Table, is attributed to a component type, defined in the [COMPONENT TYPES] Table. Table 1 presents possible additional attributes for each DFD component. Fields <FEATURE #>, in the [DFD COMPONENTS] Table, are used for storing those attributes. The [DFD

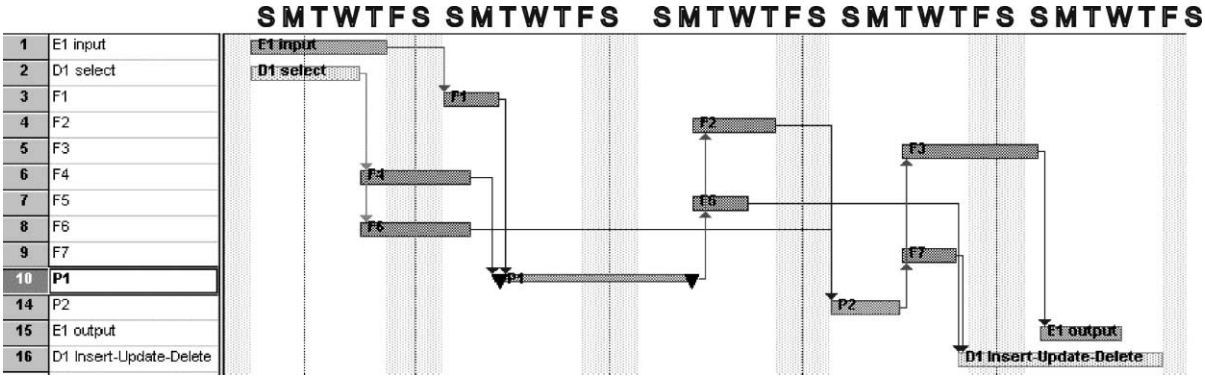

Fig. 4. Ms Project Gantt diagram representation of Fig. 1 DFD's objects.

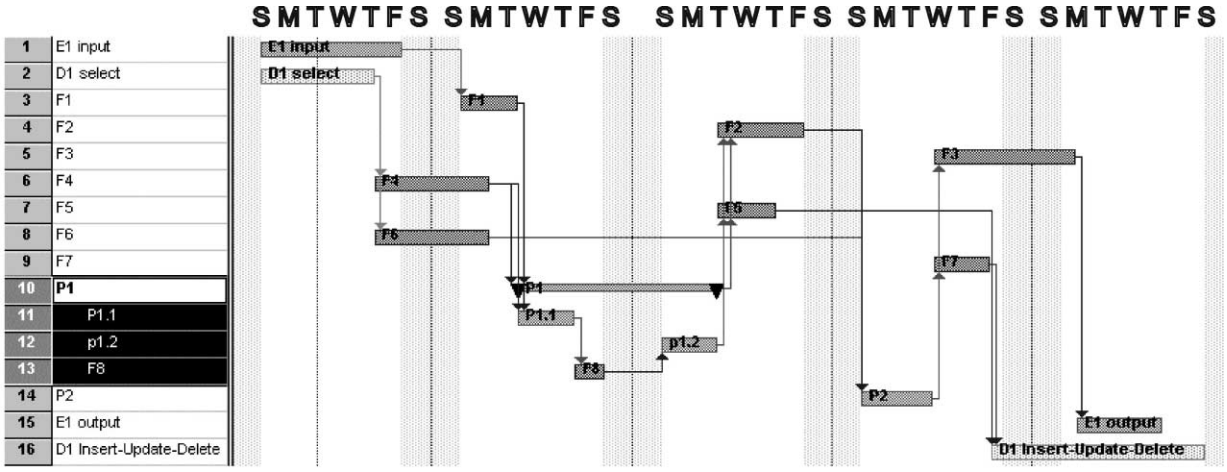

Fig. 5. Ms Project Gantt diagram representation of Fig. 3 DFD's objects.

Table 1
Additional attributes of DFD components

Attribute	Process	Flow	Entity	Store
1	And/or alert	Arrival manner	Internal/external	Internal/external
2	And/or output	Frequency	On/off line	Access protocol
3	Duration	Data volume	Real/Non real time	Authorization
4	Memory usage	Channel rate	Authorization	Historic depth

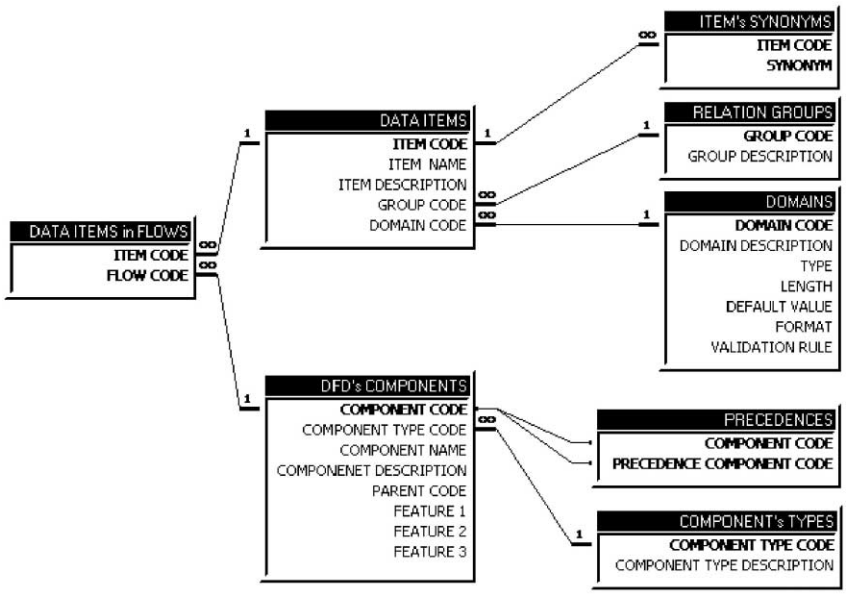

Fig. 6. Database schema for the DFD-Gantt mapping.

COMPONENTS] Table is not normalized because of the generality of those suggested improvements.

4. Mapping prototype

For the sake of demonstration, our mapping approach was prototyped and this section presents the results of using the prototype for mapping the DFD models shown in Figs. 1, 2, and 3 into Microsoft's MS Project Gantt diagrams. Fig. 4 displays the Gantt diagram corresponding to DFD-0 in Figs. 1 and 5 displays the mapping of the composite DFD in Fig. 4 into a Gantt diagram. The Summary Task P1 in Figs. 5 and 6 (line 10) corresponds to General Process P1, while Subtasks of P1 are represented only in Fig. 5. The P1 Summary Task has a distinct symbol with emphases at the edges.

Double clicking on line 10 "blows up" the Summary Task and displays its Subtasks. Lines 11, 12 and 13, in the Gantt diagram of Fig. 4 are hidden because they are related to a lower level in the DFD hierarchy, DFD-1 (Fig. 2).

Lines 11, 12 and 13, in Fig. 5 represent the DFD objects that are shown in DFD-1 (Fig. 2).

5. Discussion and conclusions

The current research demonstrates the derivation of project management objects directly on the basis of raw system analysis and design objects. By so doing we provide an integrating layer, which combines standard project management and control tools with common system analysis and design tools. Applying such an integrating layer in software development projects enables, due to the possibility of executing the CPM on paths constructed of "software component chains", improved assessment of such essential elements as development duration, development cost, response time, and data traffic volumes. CPM execution can enable monitoring and control over various topics uniquely related to software projects.

Without such an integrating model, estimating development cost and time is difficult, no matter which methodology is applied, for the following three reasons:

1. First, estimation must be based on *quantitative* data, which are available only after providing the results of analysis to members of the development team who, for the most part, are not involved in the analysis stage [1,2,24–26].

2. Second, methods for evaluating development time and cost are hardly an integral part of system analysis and design tools. The same holds for methods for estimating the response time of the system under development.
3. Third, accepted models such as COCOMO [27], are based on a presumptive assessment of archaic factors such as program size and LOC. Today's fourth generation software provides little in terms of certainty or accuracy. Estimation of project cost and time is, therefore, based usually on subjective judgment of a veteran development manager or consultant on the basis of comparisons with previous similar projects, or breaking down the project into sub-components, and evaluating each separately [28].

Other than these drawbacks of conducting development in the absence of an integrating layer, the mapping model presented in this study has the following advantages:

1. The mapping model enables effort and cost estimation for information system development to be an integral part of conventional analysis and design methodologies, i.e. DFD and FH. This is due to the possibility of **deriving assessments directly from system analysis and design raw data**, as opposed to relying on aggregates or other secondary sources.
2. The mapping model enables **extension of the development effort and cost assessment process**, beyond the mere aspects of $<Time>$ and $<Resources\ and\ Budget>$, to include also $<Complexity\ and\ Quality>$, yielding expected system **response times**, and **data volume traffic**.
3. The mapping model bases the estimation process on the **general** project management and control tools Gantt and Pert, which are widely used and supported (e.g. MS-Project).
4. Use of Gantt and Pert diagrams enables **dynamic** control of the estimation, based on reports regarding *actual* progress. This provides for *projected-to-actual* comparisons of cost, and *moment-to-moment* updates of CPM calculations, as opposed to the **static** control methods customary in the field of system building.
5. Use of Gantt and Pert charts allows "**drilling down**", into the **system code design**, including master routines and system major service routines. This differs from current methods used for information systems development, which are limited primarily to the area of **functionality**.
6. The potential of a detailed drill-down concerning system code design provides for engaging and **integrating** the technical team (development managers) as early as the analysis stage. This results in a more **reliable** and **accurate** estimation base for the entire system development project.

It is noteworthy that conversion of a DFD model to Pert/Gantt diagrams is actually a representation of a knowledge model as a semantic network. Even though DFD models have been used for demonstration purposes in this paper, the current research suggests that the demonstrated conversion capability is applicable to any other model, which can be represented in a network format. Stated differently, we have reason to believe that the integration of system analysis and project management tools, modeled and prototyped in this study, is not limited to the DFD approach, but can be applied to any "network-based" modeling such as the UML. Therefore, a promising direction for further research may focus on integrating UML diagrams with project management tools.

The fact that conversion of a DFD model to Pert/Gantt diagrams is actually a representation of a knowledge model as a semantic network opens opportunities for commercialization for practitioners. Gaines and Shaw [29] present a formal visual language that enables such a representation and show how this language can represent a knowledge model in a document, either within a word processor or within a Web browser. Based on their work, the prospects are good for commercially expanding our mapping model, in the future, to support virtual development teams on the Web, regardless of geography and time constraints.

In sum, this study showed the feasibility and validity of translating system analysis objects into project management objects. Our mapping, which aims to integrate common information system analysis and design methodologies with standard project management and control tools, can potentially improve estimation, planning, and control of software development projects, in terms of cost, time, and management. In such projects, where so many things may go out of control, any raw information gained during the analysis phase will make a contribution in later phases of system building and project monitoring.

References

[1] Becker SA, Bostelman ML. Aligning strategic and project measurement systems. IEEE Software 1999;May/June:46–51.
[2] Pressman RS. Software engineering. 4th ed. McGraw Hill, New York, 1997.
[3] Reel JS. Critical success factors in software projects. IEEE Software 1999;May/June:18–23.
[4] Gilb T. Principles of software project management. Addison Wesley, New York, 1988.
[5] Grady RB. Practical software metrics for project management and process improvement. Prentice Hall, New York, 1992.
[6] Fowler M, Scott K. UML distilled. Addison-Wesley, New York, 1997.
[7] Wood B. SC21 contribution to the join workshop on standards for the use of models that define the data and processes of information systems. Proceedings of the SC21 Meeting, May 1996.

[8] Roberson S, Robertson J. Mastering the requirements process. Addison-Wesley, New York, 2000.
[9] Barker R, Longman, C. CASE method: function and process modelling. Addison Wesley, 1992.
[10] Sommerville I. Software engineering. 5th ed. Addison-Wesley, New York, 1997.
[11] Agnus RB, Gunderen NA, Cullinane TP. Planning, performing, and controlling projects: principles and applications. 2nd ed. Prentice Hall, New York, 1999.
[12] Kerzner H. Project management: a systems approach to planning, scheduling, and controlling. 7th ed. John Wiley & Sons, New York, 2000.
[13] Fox TL, Spence JW. Tools of the trade: a survey of project management tools. Project Management Journal 1998;September:20–7.
[14] Pollack-Johnson B, Liberator MJ. Project management software usage patterns and suggested research directions for future developments. Project Management Journal 1998;June:19–25.
[15] Kaindle H, Carroll JM. Symbolic modeling in practice. Communication of the ACM January 1999;42(1):28–30.
[16] Dutta S, Lee M, Van Wassenhove L. Software engineering in Europe: a study of best practices. IEEE Software 1999;May/June:82–9.
[17] Sawyer P, Sommerville I, Viller S. Capturing the benefits of requirements engineering. IEEE Software 1999;March/April:78–85.
[18] Ahituv N, Zviran M, Glezer C. Top management toolbox for managing corporate IT. Communication of the ACM April 1999;42(4):93–9.
[19] Boegh J, Depanfilis S, Kitchenham B, Pasquini A. A method for software quality planning, control, and evaluation. IEEE Software 1999;March/April:69–77.
[20] Jiang JJ, Klein G. Information system project-selection criteria variations within strategic classes. IEEE Transactions on Engineering Management May 1999;46(2):171–6.
[21] Jarke M. Scenarios for modelling. Communication of the ACM January 1999;42(1):47–8.
[22] Domges G, Pohl K. Adapting traceability environments to project-specific needs. Communication of the ACM December 1998;41(12):54–62.
[23] Kautz K. Making sense of measurement for small organizations. IEEE Software 1999;March/April:14–20.
[24] Grady RB. Successfully applying software metrics. Computer 1994;27(9):18–25.
[25] Paulish J. Case studies of software process-improvement measurement. IEEE Computer 1994:50–7.
[26] Whitten JL, Bentley LD, Barlow VM. System analysis and design methods. 2nd ed. Irwin, New York, 1989.
[27] Boehm B. Software engineering economics. Prentice-Hall, New York, 1981.
[28] Weller F. Using metrics to manage software projects. IEEE Computer 1994:27–33.
[29] Gaines BR, Shaw MLG. Embedding formal knowledge models in active documents. Communication of the ACM January 1999;42(1):57–63.