The complementary use of IDEF and UML modelling approaches

Cheol-Han Kim\textsuperscript{a}, R.H. Weston\textsuperscript{b,*}, A. Hodgson\textsuperscript{b}, Kyung-Huy Lee\textsuperscript{a}

\textsuperscript{a}Information System Engineering, Daejon University, Daejon, South Korea

\textsuperscript{b}Department of Mechanical and Manufacturing Engineering, MSI Research Institute, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK

Received 22 November 2001; accepted 4 October 2002

Abstract

The IDEF and Unified Modelling Language (UML) modelling approaches have become popular in industrial and academic circles. IDEF comprises a suite of graphical modelling techniques designed to formally specify and communicate important aspects of enterprise engineering projects, whereas UML is a modelling language that can be used to generate computer-executable models that encode key aspects of software engineering projects.

This paper considers similarities and differences between IDEF and UML modelling approaches. It is observed that the combined development and reuse of IDEF and UML models has the potential to place information technology (IT) systems engineering projects into a wider context of enterprise engineering. An electronics industry case study is described to illustrate this observation. Particularly, this study illustrates how semantic information encoded by different types of IDEF diagramming technique can be re-represented and reused as models expressed in alternative notations. The study illustrates benefits gained from using IDEF as a business front end to UML. Further, it indicates how consistency can be maintained between multi-perspective models expressed in terms of general purpose IDEF and UML modelling constructs.

Keywords: Enterprise modelling; Systems integration; Modelling methods

1. Introduction

In recent decades, advances in information technology (IT) have provided essential tools for business, commerce and government [1–3]. IT systems also provide “organisational glue” that binds strategic, tactical and operational activities carried out within and between organisations [4]. A shift in the industrial usage of IT systems has been identified as moving from the application of “data-driven environments” to that of “co-operative information and knowledge-driven environments” [5,6]. Increasingly frequently collaborative groupings of business partners have sought collectively to configure, resource and optimise the performance of integrated sets of business processes [3,7]. The growing requirement for collaborative working has necessitated the development of new concepts, methods and techniques to conceive, design, implement and develop systems of confined scope that can be readily configured and integrated with other confined scope IT systems into interoperating IT systems of wider scope [8]. It follows that considerable research and development effort world-wide has
led to interconnected advances in (1) enterprise mod-
elling and (2) enterprise integration (EI).

Enterprise modelling methods, architectures and
tools can be used in support of the life cycle engi-
neering of large scale, complex and changing sys-
tems [9,10]. The IDEF suite of enterprise modelling
approaches, which comprises IDEF0, IDEF1,
IDEF1x, IDEF3 and other graphically based modelling
notations [11–14] have been applied extensively
in support of large industrial engineering projects.
Individually, these notations are designed to
model an enterprise from a defined viewpoint, such
as a “function viewpoint” or an “information view-
point”. This is both a strength and a weakness of
IDEF enterprise modelling approaches. However,
possibly because IDEF modelling concepts and tools
have been incrementally developed over a number of
decades, there is no overarching modelling frame-
work that has been formally defined to interconnect
individual IDEF notations. Each can be individually
applied and reapplied, in a variety of ways and its use
can be supported by a selection of proprietary sys-
tems engineering tools. However, the downside of
not having an overarching IDEF modelling frame-
work is that, when carrying out multi-perspective
modelling of any given complex domain, difficulties
will arise in communicating various domain con-
cerns between different domain experts, because
the models will not naturally be coherent one with
another. Essentially, therefore the problem of achiev-
ing an integrated use of multiple IDEF notations and
their supporting systems engineering tools is left to
users to solve. Generally, where large scale, complex
systems require the use of IDEF notations then
model integration is achieved in an ad hoc manner.
This can be expected to incur significant project costs
and, probably worse still, it can inhibit model reuse
and therefore it is unlikely to yield solutions that can
readily be reapplied.

Several recent technological advances in EI con-
cepts, methods and tools have been associated with the
ongoing development of distributed-object technol-
gy. The development of object-based methods that
enable IT systems to be reused, reconfigured and
scaled up and down is made possible by deploying
a structured approach to software system analysis and
design [15]. Such an approach can provide a frame-
work for interoperation between software objects that
may be referred to as software components of IT
systems.1 Furthermore, well-defined and standard sys-
tems integration mechanisms and services have
become available, often as an integral part of network
systems such as the Internet. These integration
mechanisms underpin interoperation between soft-
ware objects that comprise different types of IT-based
application system or tool [16].

The Unified Modelling Language (UML) has fast
become a de facto software engineering standard [17].
It provides a set of modelling notations designed to
support various domain specialisms and life phases
involved in the engineering of object-oriented soft-
ware systems. It follows that UML provides suitable
and widely-used modelling constructs for developing
structured, configurable, reusable and readily-distrib-
uted multi-perspective models of IT systems.

Normally “enterprise engineering” projects are
of broader scope then “software engineering” pro-
jects. Enterprise engineering projects are conceived
and resourced to seek to deliver new or modified
solutions to business, human and IT problems. IDEF
modelling notations were designed to provide means
of modelling enterprises in their entirety, so as to
systematically deliver abstract representations of
different enterprise views that can be used by con-
cerned parties in different ways. It follows that one
possible use of IDEF model views is to formally
specify different aspects of IT system requirements.
Enterprise engineering requirements specified by mul-
tiple IDEF models can provide a semantically rich
source of reference on business, human and IT con-
cerns. Such an information source can be referenced
by persons with software and associated IT system
engineering responsibilities. What is more, most if not
all software engineering projects are of confined
scope, with many software engineering projects car-
rried out in an essentially piecemeal, incremental
manner during the lifetime of an enterprise. Therefore,
it can be argued that a rich source of reference about
enterprise requirements, that can be updated periodi-
cally or episodically, is a necessity to enable the IT
needs of an enterprise to continue to be satisfied.

1 The term software component has been used differently by
many authors. One significant body of international research effort
has investigated ways of producing reusable software components
that can readily be configured into wide-scale systems (OMG,
1995).
The above thread of argument was the motivation for this paper which seeks to highlight theoretical and practical observations made when applying IDEF and UML. Those observations support the notion that a coherent use of IDEF and UML models can offer significant potential to develop: (i) large scale IT systems, comprising multiple limited scope IT systems that interoperate as part of a co-operative information-driven environment or (ii) IT systems of limited scope (such as IT system components) that can be readily integrated into large scale co-operative information-driven environments.

2. Background information on IDEF modelling constructs

The following sub-sections provide a brief overview of some commonly used IDEF notations. The reader should access the referenced material to gain detailed understandings of their individual capabilities and limitations.

2.1. IDEF0 function modelling method

IDEF0 was developed in order to represent activities or processes (comprising partially ordered sets of activities) that typically are carried out in an organised and standard manner [12,18]. The IDEF0 definition of a function is "a set of activities that takes certain inputs and, by means of some mechanism, and subject to certain controls, transforms the inputs into outputs". These inputs, controls, outputs and mechanisms (ICOMs) can be used to model relationships between different activities.

Generally IDEF0 modelling starts by defining a context diagram. This represents the overall purpose of the system and its interfaces with an external environment. Normally, IDEF0 models comprise a hierarchy of related diagrams that are hierarchically decomposed thereby encoding semantic information at so-called lower levels of modelling. This hierarchical decomposition results in both wide-scope and detailed representations of environmental or system activities.

2.2. IDEF1x data modelling method

The IDEF1 [19] notation was developed to describe information models that represent both the structure and semantics of information found within a target environment or system. The IDEF1x [13] notation is an extended version of the IDEF1 notation and provides a semantic data modelling technique capable of supporting the development of conceptual schemas. IDEF1x shares similar concepts and modelling constructs to those found in entity-relationship (E-R) models. It facilitates formal representation of entities, attributes, and relationships between entities. Each attribute of IDEF1x models can only have a single owner. An important notion in IDEF1x relates to the use of candidate keys. A key of an entity is the smallest set of entity attributes needed to uniquely identify an entity instance. Each entity must have at least one key. IDEF1x modelling constructs encode three kinds of relationships, namely: specific connection relationships; non-specific relationships; and categorisation relationships. Specific connection relationships describe associations between entities. Connection relationships may be further specified in terms of their cardinality. Non-specific relationships are used as part of a so-called high-level entity-relationship view to represent many-to-many associations between entities.

Categorisation relationships describe relationships between generic entities and category entities. Each instance of a generic entity must have attributes that uniquely identify which category the instance belongs to.

2.3. IDEF3 process modelling method

Section 2.1 explained that collections of IDEF0 activities can graphically specify and communicate requirements of systems and processes in terms of static representations of functional flows. By contrast, IDEF3 notation was developed as a means of describing the time-based behaviour of systems [14] and provides means of representing sequence, timing and reachable states [20]. IDEF3 provides two main groups of modelling mechanism, namely: Process Flow Network (PFN) modelling constructs and Object State Transition Network (OSTN) modelling constructs. Process Flow Networks represent the order in which, and conditions under which, activities are performed by a system. The Object State Transition Network describes the "transition states" that an object can pass through during the execution of a specific process.
2.3.1. Process Flow Network (PFN)

PFN notation is based on the use of the four elements, namely: units of behaviour (UOB); junction; precedence link; and referent. As IDEF3 process building blocks, UOBs are characterised in terms of the objects they contain, the interval of time over which they occur, and the temporal relations that they hold with respect to other process building blocks. Each UOB box has an associated elaboration, i.e. a set of logical conditions and/or constraints. If a UOB appears at the start of a precedence link in a PFN graphical representation it is called as source, whereas, if it appears at the end of a precedence link it is referred to as a destination. To accord with PFN notation UOB boxes can be the source or destination of no more than one precedence link. Junctions are mechanisms used to specify the logic of process branching by encoding time and logical relationships. Referents are used to represent additional information related to UOBs.

2.3.2. Object State Transition Network (OSTN)

The OSTN is used to represent object-centred views of a process. It provides a summary of allowable transitions for all objects within the entire scope of a modelled process. OSTNs are constructed and represented graphically using state symbol, transition junction, transition link and related UOB modelling constructs. Here a transition only involves a change from one object state to another object state.

3. Background information on UML modelling constructs

UML is a widely used de facto standard object-oriented visual modelling language [21–26]. UML is also being used as a base object description language for the emerging UEML (Unified Enterprise Modelling Language) proposed by IFAC/IFIP [26]. In UML, there are three main modelling viewpoints, namely: “use case” models, “static” models and “dynamic” models. Use case models describe system requirements from user viewpoints. Static models are essentially class diagrams that describe system elements and their relationships (including generalisation, aggregation and association relationships). Dynamic models describe system behaviour over time and support Kruchten’s “4 + 1” view model [27].

3.1. UML use cases and use case diagrams

Use cases define generic processes the system must be able to handle. The building blocks of use case models are use cases, actors and the system modelled. The use case, represented graphically as a named oval, specifies the functionality provided by the system or a task that has to be done with support from the system. An actor, usually shown as a stick person, is not part of the system, but an external entity that must interact with the system. An actor is a type (a class), not an instance. Each type of actor represents a role, not an individual user of the system. Actors communicate with the system by sending and receiving messages. When an actor sends a message to a system this will initiate a use case.

3.2. UML class programs and static modelling

The term “class” is a descriptor used in UML notation to refer to a set of objects with similar data structure, behaviour and relationships. A class diagram can be used to provide a static view of a system in terms of its object classes and the relationships among those classes. However, class diagrams do not encode temporal information. Four kinds of UML modelling construct (association, composition, generalisation and dependency) are used to describe static relationships within class diagrams.

3.3. Dynamic modelling

Generally, system objects need to communicate with other system objects and objects external to that system. In UML notation, object interactions are described using a dynamic model. Modified system behaviour is normally initiated following the transmission of a message from one object to another. In UML, modelling constructs are provided to describe the four types of message namely synchronous, synchronous with immediate return, asynchronous and simple.

UML also provides modelling constructs to construct four types of dynamic diagram, namely state diagrams, sequence diagrams, collaboration diagrams and activity diagrams. State diagrams and activity diagrams can be used to encode structural descriptions, whereas sequence diagrams and collaboration
diagrams are designed to describe how behavioural descriptions are executed.

3.3.1. State diagram

State and change of state will be encoded by values attributed to an object. State diagrams are typically deployed to complement the description of a class. A state diagram can show all possible states that can be reached by objects of the class, and which events trigger/cause state changes. An event can be encoded by an object initiating a message.

3.3.2. Activity diagram

Activity diagrams are used to model flows of activities, such as activities comprising a procedure. In UML, it is assumed that states will be changed by activities, while transitions will be related to actions. Consequently, activity diagrams are inherently linked to state diagrams. In an activity diagram, events are generally referred to as transitions. Unlike state diagrams, focus here is on describing events that occur within a single object as it responds to message instances. An activity diagram can be used to model an entire business process. Further, it can provide a high level view of what is going on inside a use case. Modelled elements in an activity diagram are activities, transitions, synchronisation bars, decision diamonds and start and stop markers.

3.3.3. Sequence diagram

A sequence diagram is used to show how objects and actors take part in a collaboration. It defines and illustrates the sequence in which objects pass messages to each other. Thereby the sequence diagram focuses on message sequences over time. A sequence diagram has two dimensions, namely: a vertical dimension used to represent time and a horizontal dimension used to represent aspects involved in a collaboration. The modelling constructs of a sequence diagram comprise objects, links and messages.

3.3.4. Collaboration diagram

In UML notation, collaboration defines relationships between a set of objects in a particular context. Collaboration diagrams can be used to show the sequence in which events occur by describing the structural organisation of objects that exchange messages. Here message arrows are drawn between objects to define and illustrate message flows between objects. Modelled elements of a sequence diagram are: objects, links and messages. Collaboration diagrams and sequence diagrams encode similar information. Which is best to deploy will depend upon what aspect of a given interaction the user needs to focus on [21].

4. Analogy drawn between the use of IDEF and UML for modelling business systems

Today, two distinct modelling communities have developed across academic and government research groupings world-wide. One such community, referred to in this paper as “enterprise modellers”, has a prime concern centred on common business issues in organisations or groupings of organisations and on how these concerns impact on the life cycle of business processes and their underpinning enterprise systems, which include human and IT systems.

From the above, it can be seen that the scope of concern of enterprise modellers is very broad and even encompasses issues related to economic influences, consumer and product market conditions, political influences, social and human factors and environmental conditions. However, a unifying theme has been to develop multi-perspective modelling formalisms that enable very complex problems to be understood and both model-based and physical solutions to these problems to be engineered and reused harmoniously as instances of change impact on the problem domains analysed. Much of the work of enterprise modellers and their developed methods has a top-down life cycle engineering emphasis, even though their modelling concepts can be deployed in bottom-up and middle-up-down ways. Often IDEF and CIMOSA modelling is begun by a process of functional decomposition in which abstract groupings of essentially self-standing enterprise activities are defined and progressively detailed. This offers a systematic way of dealing with levels of complexity. IDEF and CIMOSA approaches are representative of the state of play for many enterprise modellers. Like other enterprise modelling approaches, IDEF and CIMOSA can represent business and social aspects of problems and solutions as well as technical aspects. However, because of their emphasis on formalism and method their underlying
concepts and methods are better developed with respect to encoding the semantic of technical issues in a top-down manner.

The second and more sharply focused community of modellers, which will be referred to as the “distributed object system modellers”, has been concerned primarily with software systems engineering and reuse. UML is representative of the state of play for this second modelling community where their interests have an even sharper technology focus. However, the distributed object system modeller community is well aware that the configuration, reuse and scalability properties of object-oriented methods can beneficially be applied in social and business domains, as well as in technical domains. Hence, the primarily bottom-up development from distributed object system modellers of so-called “application components” and “business components” as building blocks of business systems.

The two modelling communities have much to offer each other as is evident by cross-community membership on relevant ISO and national standards committees [28]. Bearing this point in mind, the following sub-section illustrates distributed object system modellers’ viewpoints on modelling enterprises.

4.1. A UML view of enterprise engineering issues

An enterprise is commonly considered to be a complex system that has a specific purpose or goal. Individual businesses have different goals and internal structures but many businesses use similar concepts to describe their structures and operations. Within this context the distributed object community has emphasised the importance of defining “objects” (which typically are referred to as “enterprise objects” or “business objects”) which are the “stuff” or “things” from which complex enterprise systems are made [22,29]. In this context, therefore “objects” are essentially a classification of “things”, real instances of which can be reused and bound by appropriate structures to achieve specific enterprise goals. UML notations can be used explicitly to describe some of these concepts [6]. For example, “business objects” can be viewed as being common parts of business systems such as an Enterprise Resource Planning System or a Computer Aided Process Planning System, real instances of which comprise both software and people who operate in a semi-structured way. Thereby, business systems can be modelled as collections of business objects, including models of relationships that structure dynamic interactions between business objects in various situations [22,29]. Eriksson and Penker [22] refer in UML terms to the following types of “enterprise building block”.

**Resources**: Business objects (such as people, material, information and products) are used or produced by a business system. Resources are arranged into systems via the imposition of structure. Resources are used, consumed, refined or produced through processes and can be categorised as physical, abstract and informational.

**Processes**: Business activities performed to change the state of resources. Processes describe what is done within a business and are governed by rules.

**Goals**: The purposes of a business or what the business as a whole is trying to achieve. Goals often comprise sub-goals allocated to sub-businesses, such as individual processes and objects. Goals can express the desired states of resources.

**Rules**: Statements that define or constrain some aspect of the business and represent business knowledge. Rules govern how the business should run, i.e. how processes should execute and how resources may be structured and related to each other and to processes. Rules can be enforced externally (with respect to the business) or internally and categorised as functional, behavioural and structural.

**Metamodels**: These are a model of the basic business concepts and their relationships used in a given domain. The concepts are used to create other models. For example, metamodels can show how processes result in the achievement of goals, how processes interact with resources, etc.

Based on the above or similar notions, many authors have used UML symbols and diagrams to create different types of business model. Static diagrams can be expressed as class or object diagrams and dynamic diagrams (such as sequence state-chart or activity diagrams) can be used to describe a software system function as part of a wider business system. However, if the aim of UML-based business modelling is to capture reality and facilitate business thinking then it has been argued that it is not appropriate to
think of translating the business models directly into the syntax of a programming language [22]. However, a key focus of UML use in software engineering is to translate design concepts directly into code. This shift in focus of use of UML has been a prime motivation for the UML community developing so-called “built-in extensions” to improve the utility of UML in a business context.

This paper questions the practicality and advisability of the “distributed object system modelling” community developing business extensions to UML in isolation. Rather it is proposed that the unified use of the best results of both modelling communities should be sought. To provide one starting point for such an initiative, this paper explains how despite differences in their viewpoints and emphasis, IDEF and UML modelling approaches complement each other. Potentially, IDEF modelling offers a standard and widely used approach that can be used to develop business “front ends” for UML, whereas UML can provide a standard and widely used software engineering “back end” for IDEF modellers.

Evidence to support the above argument is presented in Table 1 and in the domain study described in Section 5. Table 1 is drawn from observations made in Section 3 of this paper and from ongoing practical experiences of the authors and their colleagues in the MSI Research Institute. Here IDEF, CIMOSA, UML and various other modelling approaches have been developed and used in different application domains and industrial sectors in support of both enterprise engineering and software engineering projects. The table summarises similarities in coverage of concepts, context and constructs developed and used, whereas the domain study illustrates the practical and beneficial use of IDEF modelling as a business front end to UML modelling. Hence, at least in principle, there arise opportunities to determine and deploy formal mappings between IDEF and UML modelling constructs.

Detailed discussion on this point is included into a previous paper of the authors [30]. Section 5 shows how example IDEF model elements are converted into UML model elements according to tabulated mapping rules. For example, 0-level functions of a context diagram are converted into use case model elements that describe things that a system does. This is enabled because the context diagram in IDEF0 notation encodes functional requirements about a system. The IDEF “mechanism” modelling construct indicates interaction between the external environment and plays the role of an actor. IDEF0 activity diagrams encode co-ordination aspects of activities in terms of transitions, synchronisation bars, decision diamonds, and start and stop markers. Whereas IDEF3 notation can be used to create activity diagrams in terms of alternative transition modelling constructs. The decision point of an IDEF0 activity diagram is analogous to the use of XOR junctions in IDEF3. Whilst representation of concurrency issues in IDEF0 is analogous to the use of AND junctions in IDEF3.

In general, one to one mappings between model elements cannot be explicitly defined. The view of the
modeller will be different dependent on the purpose of the model, therefore the semantics will not necessarily be the same. However, mappings between some pairs of specific modelling construct may be explicit and consistently represented. In such cases, some guidance rules for mapping can be defined. Further discussion in Section 5 of this paper and by Kim et al. [30] illustrates this point.

Model views developed using IDEF modelling notations are essentially conceptual models and therefore they have not been designed to generate implementation schemata. Despite this constraint, value can be gained if explicit mappings can be established between equivalent elements of IDEF conceptual models and object-oriented models, because the latter model types of a structure and facilitate the generation of computer-executable systems.

5. Generating UML models from IDEF models—a domain study

To illustrate the complementary nature of IDEF and UML modelling approaches, this section considers an example domain study where it has proven beneficial to use IDEF modelling as a business front-end for UML modelling. The study represents an abstraction of problems tackled and solutions developed when creating an engineering process for a new consumer electronics product in a multi-national company with a base in Korea. Following this study in 1998, production of this consumer product commenced. Fig. 1 shows an overview of the product development process. First the engineering activities and their requirements were analysed using IDEF notations. Specific aspects of the IDEF models describing the engineering activities involved were then converted, using mapping algorithms developed by the authors, into equivalent UML models. Subsequently, the UML models were used to structure and inform software engineering activities when developing the factory systems required to manufacture the consumer electronic products.

5.1. IDEF modelling during the domain study

5.1.1. IDEF0 model

The product development process encompassed “product planning” through to “prototype product development”. Fig. 2 shows the top-level IDEF0 context diagram developed to represent the scope of this process. In this domain study, the context diagram was decomposed into three main sub-functions, namely: “analyse market”, “plan a product” and “design a product”. Fig. 3 is a IDEF0 diagram which encodes the functional decomposition corresponding to the “develop a new product” process. Fig. 4 shows a further level of decomposition at the level of the “design a part” sub-process. In total, four levels of functional decomposition were required during IDEF case study modelling, which included a functional decomposition of the “record a new part” sub-subprocess, see Fig. 5.

5.1.2. IDEF1x model

Like many products in this domain, the particular electronic consumer product is composed of both mechanical parts and electronic-electrical-mechanical parts. It also has a product family (that can be characterised by product models) built from common elemental building blocks or modules. Characteristic properties of this product structure can be usefully described as a hierarchical structure expressed using IDEF1x modelling constructs. Information about the product encodes properties of the product family and of the product itself. Family information developed as part of this case study included: family definition, module, kits and components information. Specific product information included product documentation, BOM, model information, etc. Figs. 6 and 7 show IDEF1x models captured at the product level and assembly level, respectively.
5.1.3. IDEF3 model

IDEF0 models are designed to model functions and relationships between functions that comprise an enterprise. These functional units can be successively decomposed until, at the lowest level, a primitive unit of function is defined that equates to a unit activity in the enterprise being modelled. These units of function can be re-represented as units of behaviour using the IDEF3 notation. Here ICOMs of a function are converted into equivalent IDEF3 objects. Essentially these IDEF3 units of behaviour (or UOBs) are reference objects that can describe process flows, conditions and events. For example, they can encode additional information to that represented earlier by IDEF1x models, such as in the form of IDEF3 PFNs. Furthermore, object states that will be governed by the result of the activation of functions, can be described using IDEF3 OSTNs. Fig. 8 shows an example PFN which defines a standard activity flow for the object “part”, in terms of activities (specified earlier by Figs. 3 and 4) that should be carried out on that object and relationships between activities that need to be maintained. Fig. 9 shows a related OSTN for the object “part” which concerns the activity “design process”. In fact Figs. 8 and 9 were semi-automatically generated during case study work by using mapping algorithms and rules previously developed by the authors, that were designed to transform instances of IDEF0 modelling constructs into equivalent instances of IDEF3 PFN and OSTN modelling constructs [31].

5.2. UML model

5.2.1. Use case diagrams

We have seen that UML use cases can be used to model generic processes and to define the way in which different classes of user should interface with
generic processes. It follows that information captured in the form of IDEF0 models, such as information about input data sources and mechanisms, can be recoded in the form of UML use cases. Therefore, alternative user-centred views on context diagrams and functional decompositions (previously expressed in IDEF0) can be developed. Fig. 10 shows a UML use case-centred view which has been drawn from IDEF0 models shown in Figs. 2 and 3.

5.2.2. UML class diagrams

It was explained that UML class diagrams can be used to classify reusable enterprise objects. Therefore, it can prove useful to define and model UML classes by recoding information previously encoded by IDEF0 and IDEF1x models. UML associations can be derived from semantic information made available about connection relationships (both specific and non-specific). The semantic used may concern conceptual or physical “connections” between entities (making them an entity class) or generalisations used within IDEF1x. Because IDEF1x diagrams include “is-part-of” relationships, then complex product structures can be derived from known relationships connecting entity classes. Fig. 11 shows a class diagram developed to represent the case study product structure in UML notation. This class diagram was derived from enterprise knowledge previously encoded in and represented graphically by Figs. 5 and 6. Dependency relationships exist between the function class and its related entity classes because the attributes of the function class are derived from the entity classes. Fig. 12 shows dependency relationships derived for the function class “register-a-new-part” and its related classes.

5.2.3. Dynamic modelling

Dynamic modelling can prove useful by specifying how communication should be achieved between sets
5.2.3.1. State diagram. Because state diagrams describe possible conditions of objects, the IDEF3 OSTN may be used to derive inputs to a UML state diagram. Here OSTN descriptions of object states and related behaviours can provide such an input. Object state in OSTN can be directly translated into an equivalent object state in a UML state diagram, whereas IDEF0 UOBs can be converted into equivalent events that cause a state transition. Fig. 13 shows a UML state diagram derived from semantic information previously encoded by Fig. 9.

5.2.3.2. Activity diagram. Information contained in an IDEF3 model can be used to develop equivalent UML activity diagrams. UML activity and transition model elements can be derived from equivalent IDEF3 UOBs and transitions, where the decision point of UML activity diagrams is analogous with IDEF3 XOR junction modelling constructs. Concurrency aspects of UML activity diagrams are in part analogous to the use of AND junctions in IDEF3. Fig. 14 shows a UML activity diagram derived from the IDEF3 OSTN shown in Fig. 8.

5.2.3.3. Sequence diagram. UML sequence diagrams encode interactions between objects from a temporal standpoint. Some of the semantic information required to capture UML sequence diagrams can be derived
Fig. 5. Functional diagram for the "record a new part" sub-sub-process.

Fig. 6. Case study IDEF1x model developed at the product level.
Fig. 7. Case study IDEF1x model developed at the assembly level.

Fig. 8. Example IDEF3 PFN for the object “part”.

---

**Fig. 7. Case study IDEF1x model developed at the assembly level.**

**Fig. 8. Example IDEF3 PFN for the object “part.”**
from information contained in equivalent IDEF0 or IDEF3 models. From semantics encoded by IDEF0 models, UML objects can be defined from a knowledge of ICOMs data and translated into an equivalent entity class. Alternatively, objects can be derived from IDEF3 referent modelling constructs that are designed to model organisation and resource aspects of entities. As functions can be decomposed into unit functions, it is possible to estimate and attribute processing times for the operation of unit functions. This can be useful when encoding the time span of a sequence diagram. Accordingly, a vertical
dimension representing time can be determined from timings associated with the functional diagram, while objects derived from the functional diagram are arranged on the horizontal axis. Properties of flows, encoded by arrows in IDEF0 functional diagrams and by the order of UOBs in IDEF3 models, can also be interpreted to determine the sequence of UML messages. Indeed, arrows representing IDEF0 links and IDEF3 junction descriptions can also be interpreted to determine attributes of equivalent links represented in UML collaboration diagrams. Fig. 15 shows a sequence diagram derived from Fig. 5 which represents the sequence involved in registering a new part. During the development of UML sequence diagrams, attributes about objects (such as “part”, “R&D team”, and “purchasing team”) are analysed and recoded.

5.2.3.4. Collaboration diagram. UML collaboration diagrams can also be derived from semantic informa-

![Fig. 11. Class diagram for the case study product structure.](image1)

![Fig. 12. An example of dependency relationship between classes.](image2)
tion previously encoded within IDEF0 or IDEF3 models. The recoding of semantics in a UML collaboration is similar to that when deriving UML sequence diagrams because essentially collaboration and sequence diagrams provide alternative representations of similar information. Fig. 16 shows an example collaboration diagram, which represents similar information to that contained in Fig. 15.

5.3. The benefits arising from the combined use of IDEF and UML

During the case study, specific benefits arose from using the combined UML/IDEF modelling approach. The case typified product realisation in a multinational company producing many kinds of electronic consumer product, such as TVs, micro-ovens, mobile
phones, PCs, printers, etc. There is a need for frequent engineering change in that company arising from ongoing customer requirements change, product technology change, business and manufacturing process change and less directly because of changes in political, economic and social conditions. Hence, there arose a need to document and model change processes in such a way that reasoning can be facilitated about what should change, when changes should be made, who should make the changes, etc. Also evident as markets globalise was a need for the case study company to begin to document and model interactions between many types of enterprise system used by business partners. Modelling in IDEF alone is unlikely to lead on to system descriptions that can be readily implemented in the form of suitable teams of people, machines and related software systems. In general, IDEF notations are suited best to describing what a very complex system (like a manufacturing enterprise) is required to do from many points of view. It is less well suited to detailed modelling of the behaviours of enterprise objects (i.e. people, software systems and machines) so that responsibilities for activities can be attributed...
effectively and so that system structures, components and software can be created and reused rapidly and effectively. Whereas when beginning modelling in UML, generally it is difficult to establish links to well defined business, people and IT concerns. This is because there is no explicit way in UML of linking abstract descriptions of process and resource requirements and structure to detailed descriptions of object behaviours and structure. For the case study example of product engineering, the combined use of IDEF and UML much improved this situation and ensured that downstream project engineering activities were related to business concerns, requirements and benefits.

As in many enterprise situations different groupings of personnel (with requisite knowledge and skills) were involved in deciding (1) what the enterprise should do, (2) how the what might best be achieved and (3) doing what was necessary when it had been decided how the what should be done. In the case study, it was observed that groupings of personnel concerned with (1) were best served by IDEF notations. While groupings of personnel concerned with software system aspects of (2) were best served by UML. Further, it was observed that persons concerned with (2) could be guided by semantics encoded into various IDEF models generated by persons concerned with (1) so that their system design work is facilitated and better targeted. Therefore, it was observed that the adoption of dual (IDEF and UML) modelling streams naturally generated multiple views of “enterprise requirements”, “candidate enterprise system designs” and “actual enterprise system descriptions” that can be used by various concerned parties in alternative ways that facilitated their enterprise. On the negative side though the adoption of a dual approach will lead to modelling and model redundancy, as different groupings of concerned personnel (and groupings of groupings of concerned personnel) develop and use models with a degree of independence. However, from their experience the authors believe that the quality and flexibility of associated modelling and engineering processes can be improved by adopting a dual approach and this benefit outweighs any additional cost arising from doing duplicate modelling work and from devising procedures to protect the integrity of developed models (and associated versions of model fragments).

Some general observations can be drawn from the case study about the quality of the enterprise models developed using the dual approach. Firstly, semantic information previously captured and coded in the form of IDEF models can be recoded and re-represented. We have seen that this enables key understandings about an enterprise to be communicated and used effectively between concerned parties. A greater number of special purpose of customised enterprise views can be generated that potentially are consistent, one with another. The generation of these different views can be consistent in the sense that they are derived from common parent views. This is particularly important as it can inform and help to target a series of software engineering projects that typically over significant time periods incrementally improve the operation and organisational structures of an enterprise. Secondly, the availability of special-purpose, multi-perspective model views can prove beneficial through different life phases of enterprise systems, such as when developing “system requirements”, “conceptual descriptions of systems”, “implementation descriptions of systems”, and “runtime models”, possibly leading to “model-driven execution of enterprises” and “model-driven system change”. It can support model reuse and thereby (at least in principle) can lead to better, faster and cheaper enterprise engineering. Essentially enriched descriptions of objects, object relationships and object behaviours can lead to improved ways of resourcing (and organising the resourcing) of activities and processes.

6. Conclusion

As the boundary of an enterprise is extended, it becomes increasingly important that a common “big picture” of needed business, people and IT system activities and relationships between these activities is developed and shared by all the parties involved. Because “things” must be integrated, co-ordinated and changed they need to be modelled. Modelling methods typified by IDEF provide useful means of describing enterprise entities and their relationships and behaviours. However, practical experience exemplified by the case study models shows restrictions on the use of the various IDEF modelling notations. In IDEF0, no effective means is provided to model
objects such as supplier, customer, product and manufacturer. Therefore, within this enterprise view alone some key constituents of a process cannot be represented. Neither do IDEF0 modelling mechanisms model organisational structures in detail, except in terms of ICOMs activities. Because IDEF0 and IDEF3 models exclude means of encoding aspects of organizational views of the enterprise their notations are not suited to describing interactions between organizations. The lack of any clear distinction between material flow and information flow also leads to semantic constraints that limit the use of all IDEF models in support of detailed human and IT system design, implementation and change. Mapping semantics from IDEF0 into IDEF1x is not easy and IDEF1x models do not contain sufficient semantics to automate the design of a database. This may well give rise to inconsistencies between IDEF0 and IDEF1x models. It follows that few IDEF model types can be directly transformed into computer-executable models that can be executed. These kinds of restrictions on the use of IDEF notations motivated the authors to investigate how complementary modelling methods such as UML should be utilised.

Because UML is a modelling language, not a modelling methodology, there is no pre-set procedure (or process to be followed) when developing enterprise systems. This can increase the flexibility and utility of the approach but does mean that UML modelling is an expert task because little inherent semantic information about any given application domain is “imported” by the modelling language. The situation is similar when developing and using some types of IDEF model. On the other hand, this means that UML and IDEF notations can be widely applied and they do not implicitly (and unbeknown to the modellers) import specific domain knowledge and constraints.

Because of natural constraints associated with individual and collective IDEF and UML notations, benefit was derived from combining their use when tackling large scale enterprise engineering problems associated with the case study example. By drawing on relative strengths of IDEF and UML, a “semantically rich picture” of enterprise systems was developed and reused effectively by different types of expert user. It was observed however that this can lead to high cost when producing multi-purpose views.

---

![Diagram](image)

**Fig. 17.** Modelling concepts considered in this paper and examples of other candidate technologies.
and may lead to data integrity problems when the different views need to be consolidated (i.e. used in an integrated fashion). Hence, the authors have current research interest centred on (1) characterising “mappings” between UML and IDEF modelling concepts and constructs and (2) developing algorithms and rules that semi-automate the creation of alternative views from semantic information contained in a parent view. Underlying these interests is a general aim to promote model reuse during the lifetime of enterprise systems and thereby to drive down the high costs and long lead-times currently associated with large-scale projects. Recent sister publications of the authors describe progress made in respect to specifying algorithms and mapping techniques capable of generating UML models from collections of pre-existing IDEF models [30,31].

In this paper, IDEF modelling concepts and constructs, embodied into IDEF0, IDEFLx, and IDEF3 notations are compared with counterpart UML modelling concepts and constructs. Although there is no direct one-to-one mapping of modelling constructs, because of differences of emphasis and modelling aspect, semantic information contained in IDEF models can be interpreted and analysed by domain experts (such as IT and human system designers and implementers) and recoded into the form of UML models that provide complementary perspectives about an enterprise. Fig. 17 has been constructed to complement Table 1 in Section 4. Together the Table and Figure indicate how multiple views of activities and activity flows can be fleshed out by using IDEF0, IDEF1 and IDE3 notations. The outcome is many related IDEF models of each type that encode semantics, mainly about enterprise requirements from the different perspectives of groups of enterprise personnel responsible for business, people and IT concerns. This therefore can constitute a semantically rich picture of requirements that potentially can be reused by system designers, builders and developers on a piecemeal and incremental basis. Subsequently, therefore persons concerned with the life time of IT systems can develop and reuse multi-perspective UML models within their domains of concern. When doing so, they can extract semantic information from the IDEF requirements models, possibly on a selective basis as required. Modelling notations other than IDEF and UML could have been selected to populate Fig. 17. For example, CIMOSA also provides multi-perspective diagramming templates and well-defined and interconnected sets of modelling constructs to encode business, people and IT aspects of enterprise requirements. Indeed, CIMOSA also defines coherent templates and constructs that can be used to structure and facilitate detailed system design, system build and system change. This paper steers clear of discussion about relative pros and cons of IDEF and CIMOSA. Both provide worthy candidate front end modelling capabilities for UML. Various classes of Petri net model can also be used to complement IDEF/CIMOSA and UML modelling [32] Petri nets are highly expressive and provide good formalisms when modelling concurrency, causality, synchronisation and non-determinism of concurrent systems. Therefore, Petri nets can be usefully deployed at the boundary of IDEF and UML modelling, particularly to exercise alternative system design. However, they cannot provide multiple views of entire enterprise systems.

In conclusion, it is observed that the engineering of distributed object-oriented systems can be positioned within a wider context of enterprise engineering. Therefore, in the near future component-based enterprise systems should be designed, constructed, distributed, integrated and changed with a common “big picture” in mind. However, this requires “enterprise modellers” and “distributed object system modellers” to more actively seek improved synergy between their concepts, methods and tools. This paper illustrates example steps towards this goal.

References


[28] D.N. Shorter, CEN and ISO work on architectures, frameworks, constructs for manufacturing enterprise modelling. A personal overview from David Shorter (mail to: david@itfocus.demon.co.uk), Convener CEN TC310 WG1, 2000.


Cheol-Han Kim is an associate professor in the Division of Computer and Communications Engineering at Daejon University, Korea. His main research areas are in enterprise engineering, e-business and virtual enterprise. He has a BS and a MS in Precision Mechanical Engineering from Hanyang University and PhD in Industrial Engineering from Pohang University of Science and Technology, Korea. In 1999/2000, he was an academic visitor (Post Dr. Program) at Loughborough University, UK. His industrial career includes Factory Automation Department, LG Software Ltd. (1987–89) and Information Strategic Planning Department, Samsung SDS (1993–95).

Richard Weston, PhD, BSc is head of the MSI Research Institute and professor of flexible automation at Loughborough University. He has carried out personal research and supervised over 50 postgraduate staff and students in areas of enterprise modelling, enterprise integration, software systems engineering and flexible automation. He is author of around 300 publications. He is vice chair of the IFIP Working Group 5.12 on architectures for enterprise integration; he is a member of various national (EPSRC, DTI, LINK and BS) committees and on the
editorial boards of five journals. He is a visiting fellow for the EPSRC Business Process Resource Centre at Warwick University and consultant professor at Harbin Institute of Technology in China.

Allan Hodgson is currently a research associate at the University of Nottingham. Prior to this he was a senior lecturer at Loughborough University, having joined academia in 1983 with 12 years experience in industry, mainly in engineering, manufacturing organisation and information technology (the latter included 3 years in a software house). He has research experience in production planning and control, manufacturing systems modelling and simulation, business process re-engineering, systems integration, semantic modelling of information systems, user/task modelling and human-computer interfaces. He is author or joint author of approximately forty publications in the above areas.

Kyung-Huy Lee is employed as assistant professor at Division of Computer and Communications Engineering, Daejon University, Korea. His main research areas are in virtual enterprise, XML applications, Business process/workflow management. He has a BA in Industrial Engineering, Seoul National University (1983), MS in 5 Industrial Engineering, KAIST (1985), and PhD at the Pohang University of Science and Technology, Korea (1995). His previous career has been as a researcher, Operations Analysis Department, Agency for Defence Development (1985–89); Full-time Instructor, Industrial Engineering, KyungBuk Industrial University (1995–97).