A framework to define a generic enterprise reference architecture and methodology

Peter Bernus* and Laszlo Nemes†

*School of Computing and Information Technology, Griffith University, Nathan, 4111 Australia
†CSIRO, Division of Manufacturing Technology, Preston, 3072 Australia

The generic enterprise reference architecture and methodology is about those methods, models and tools which are needed to build the integrated enterprise. The architecture is generic because it applies to most, potentially all types of enterprise. The coverage of the framework spans products, enterprises, enterprise integration and strategic enterprise management. The proposal for the architecture follows the architecture itself improving the quality of the presentation and of the outcome. Definitions of generic enterprise reference architecture, enterprise engineering/integration methodology, enterprise modelling languages, enterprise models, and enterprise modules are given. It is proposed how the above could be developed on the basis of previously analysed architectures (and other results too), such as the Purdue enterprise reference architecture (PERA), the GRAI integrated methodology, CIM-OSA, and TOVE. GERAM is meant to unify existing architectures rather than intending to replace them. This is achieved by ensuring that the scope of GERAM spans all areas of consideration in an enterprise. Copyright © 1996 Elsevier Science Ltd.

Keywords: enterprise integration, reference architecture, enterprise engineering methodology, enterprise modelling

Introduction

With the advent of globalisation of economies, enterprises are more viewed as 'products' themselves. If they are given field companies, they have to be designed, built and put into operation. If the enterprises already exist, changes have to be specified, designed and carried out.

To design enterprises for their entire life-cycle we need fundamental principles, complex tools and methodologies. Those who build machines and electric devices have been developing and using design tools for a long time. The big difference is in complexity by a number of magnitudes and in the fact that enterprises are socio-technical/socio-economic systems rather than technical systems alone. The tools and methods needed for enterprise integration therefore are not just more complex, but substantially different.

Enterprises keep changing. Nothing is permanent in the business, manufacturing practice, organisational structure or information technology infrastructure of an enterprise. Not even for months. The design of an enterprise may take a long time and involve many people. Contrasting to this, enterprises change more often than the design of any other product. So their models and descriptions have to be changed too. As a consequence for enterprise integration methodologies to work effectively they must keep up with this pace of change.

Previous research carried out by the AMICE Consortium on CIM-OSA, by the Purdue Consortium on PERA, and by the GRAI Laboratory on GIM, (and similar methodologies by others) have produced many fine results. Analysis by the IFIP/IFAC Task Force has shown that none of the existing reference architectures subsumes the other; each of these has something unique to offer. The recognition of the need to define a generic architecture that does not suppress these individual results was a result of that work. Gerem is a new framework which encapsulates and orders the previously mentioned systems providing an overall structure to use those methods and modelling techniques — made available by others.
GERAM is thus not another new proposal for enterprise reference architectures, but a framework* which is meant to organise existing enterprise integration knowledge instead of re-defining it.

The developers of previously published reference architectures are thus able to retain their own identity, while identifying through GERAM the overlaps and complementing benefits with others.

GERAM is about those methods, models and tools which are needed to build the integrated enterprise. The architecture is generic because it applies to most, potentially all types of a real or virtual enterprise.

The scope of the architecture is therefore the union of domains which need the attention of enterprise engineering and development. Thus the scope is defined through a pragmatic need, the need to design and redesign as well as continuously improve the functioning of enterprises.

Requirements of a generic enterprise reference architecture and methodology

These are requirements that any enterprise reference architecture and methodology should satisfy. References to strong features of example architectures are also given. The overview is primarily based on the analysis which was carried out by the IFIP/IFAC Task Force† on enterprise reference architectures§.

It is proposed that a published definition of the generic reference architecture and methodology should contain its own detailed requirements definition and design decisions. The following list forms the basis of this requirements definition.

- The best treatment of the enterprise scope from the system theoretic point of view

It is necessary that all activities which are involved either directly or indirectly in designing and operating or improving the enterprise should be covered by the architecture. Given that the goal is to provide methodologies (not an architecture alone) the architecture should be the backbone of the methodology. Thus the architecture should be based on the modelling of the enterprise engineering process, or lifecycle.

A prominent example of this is the PERA¹ architecture which most fully captures the enterprise lifecycle.

- The provision of a consistent modelling environment leading to executable code

The modelling views offered should cover a minimal set (such as in CIM-OSA or TOVE), but this set should be expandable with new related views. Modelling views should be based on a common

---

*Adopted by the IFIP/IFAC Task Force on enterprise reference architectures.
†Formerly called IFIP/IFAC Task Force on integrating manufacturing entities and enterprises.

theory, or meta-model, through which views can be related.

The ideal modelling environment should be modular so that alternative methodologies could be based on it, i.e. there should not be any prejudice built into the modelling languages as to what the methodology will be.

The modelling environment should be extensible rather than be a closed set of models, and permissive — leaving space for alternative modelling methods⁵.

CIM-OSA (after adopting GRAI’s modelling of the decision aspect) has created a consistent set of modelling languages (with a common meta-model) which can support an enterprise engineering methodology. Since the architecture also covers the operation of the enterprise the requirement is strongly connected with the functionality of information integration infrastructures.

Other suites of modelling methods also exist (e.g. the IDEF set of languages⁶). The TOVE⁷ system of generic enterprise models also defines a consistent set of modelling classes (as an enterprise ontology). The generic enterprise modelling language and tool set should amalgamate the advantages of these (see design considerations as to how this can be achieved).

- The existence of a detailed methodology which enterprises can follow

In addition to a methodology being technically correct it must be understandable and usable by the communities targeted. Thus the methodology should be executable by real (as opposed to hypothetical) teams, guaranteeing a high quality result within acceptable cost, time, and resource constraints.

The methodology should identify the application circumstances which must hold, with particular attention to the size and maturity of the enterprise wishing to apply it, as for example has been established in the capability maturity model (CMM) for software enterprises⁸. Also the ISO 9000 series of relevant quality standards should be treated as a guideline for both the enterprise and the enterprise engineering process that follows from the methodology.

The GIM methodology⁹ of GRAI has been demonstrated on numerous industrial projects as effective while the Purdue Implementation Procedures Manual” has the widest coverage of the enterprise life-cycle.

An ideal enterprise engineering methodology should also be expandable, since new engineering methods will always come into existence and the framework needs to anticipate and accommodate those developments.

It should be possible for the methodology to be presented both in a generic fashion and in specialised forms. These specialised forms could better serve particular areas of industry. A generic enterprise engineering methodology should allow an entire family of methodologies to be defined on its basis.

- The adoption of good engineering practice for building reusable, tested, and standard models.

---

⁴Adopted by the IFIP/IFAC Task Force on enterprise reference architectures.
⁵Formerly called IFIP/IFAC Task Force on integrating manufacturing entities and enterprises.
⁶Adopted by the IFIP/IFAC Task Force on enterprise reference architectures.
⁷Formerly called IFIP/IFAC Task Force on integrating manufacturing entities and enterprises.
⁸Adopted by the IFIP/IFAC Task Force on enterprise reference architectures.
⁹Formerly called IFIP/IFAC Task Force on integrating manufacturing entities and enterprises.
"Adopted by the IFIP/IFAC Task Force on enterprise reference architectures.
It is important that the apparent complexity of the enterprise engineering process should be kept low. Intricate details of models should be encapsulated in reusable building blocks.

Enterprise integration and enterprise engineering is a complex design process carried out by a group of people. As such, the methodology should ideally be based on a sound design theory, treating design as a collaborative activity of multiple agents. Such a theoretical basis is meant to ensure that the methodology is not only practical but also is formulated in a durable way.

- The provision of a unifying perspective for products, processes, management, enterprise development, and strategic management

The development of the enterprise should be considered as only one of the activities in the enterprise. Thus the architecture should tie and relate enterprise integration and enterprise engineering to the rest of activities in the enterprise. Especially important is the wish to support evolutionary paths to enterprise integration, although revolutionary processes can (and should) not be excluded.

The generic enterprise reference architecture and methodology must clearly define how other efforts to integration relate to it (e.g. at the time of writing concurrent engineering, business process re-engineering, computer supported collaborative work, total quality management, virtual enterprise, workflows etc.). The architecture should make it simpler to grasp integration than it was before.

It is permissible, indeed desirable, to have multiple presentations of the architecture to facilitate understanding and acceptance by a variety of user and developer groups and to ease identification with it.

Our earlier work on the SATT methodology developed a recursive view of enterprise architecture where the end product of one architecture is the process creating the second. Using this principle the complexity of the methodology can be tackled by applying the reference architecture and methodology to its own development — in a kind of bootstrapping method. The present article attempts to start this by a presentation which follows the anticipated architectural design.

The above list of requirements can be completed and organised as functional, information, organisational and resource requirements, etc.

Design decisions on the generic enterprise reference architecture and methodology

Proposed design decisions to satisfy the requirements

From the requirements follow that design decisions on the generic enterprise reference architecture and methodology must address all components and their combination: architecture, methodology, modelling language and generic models/models. It also follows that it should be permitted for a multitude of methodologies to exist — some competing — while the reference framework allows this competition to occur on common grounds.

Through the combination of the dimensions along which the PERA, GIM, and CIM-OSA architectures divide concerns of the enterprise life-cycle, the authors developed and contributed to a matrix representation of the enterprise life-cycle model. This model was further improved by expanding it with the identification, concept, build and operate phases of the enterprise life-cycle.

The matrix there developed described the life-cycle of the enterprise and mapped on it the three investigated architectures (PERA, GIM, and CIM-OSA) to compare their coverage. This matrix can be thought of as a generalisation of the CIM-OSA cube or as a CIM-OSA-cube-like presentation of the PERA two-tiered diagram or of the structured method of GIM.

However, as shown below, the matrix is capable of describing the life-cycle of any other entity covering generic, partial and particular models/descriptions of that entity. These models/descriptions are the ones that should (or may) be produced during the entity’s life-cycle.

Before presenting how the idea of the matrix can be utilised for the design of the generic enterprise reference architecture and methodology a brief presentation of some technical details is necessary.

The life-cycle matrix

The matrix (see Figure 1), first of all, is a specification of the generic architecture of an entity, such as the enterprise. Each area in the matrix has:

- Subject matter specification — collating these specifications one can produce the terms of reference for the entity, the life-cycle of which is under consideration.

For example the ‘generic concept’ area contains the concept definitions necessary to develop the concept of the entity. If the entity is the enterprise, then the ‘generic concept’ area contains the definitions of the terms: enterprise mission, vision and values, strategy, objectives, goals, policies — all the terms that are to be used when a particular enterprise’s concept (or a particular type of enterprise’s concept) is formulated. Note that the terminology is divided into two parts, that part which concerns the control/management of the enterprise and that part which concerns the production of services or products for the customer. Some concepts may be shared between these two domains. The terminology definition can be given in various forms; such as descriptive, semi-formal (e.g. structured English) or formal (e.g. some form of logic theory — see Generic Models below). It is acknowledged that not all areas in the matrix are amenable to strict formalisation. e.g. the

*The improvement was suggested by T. J. Williams.
concept of ‘activity’ and ‘event’ is easier to formalise in a logic language than the concept of ‘management values’.

- Modelling language specification — such that the language is suitable for expressing descriptions/models that belong to the subject matter. The modelling language specifications together can be also called a Modelling Framework.

The various modelling views in Figure 1 (derived from CIM-OSA and GIM), and the corresponding classes of models, may have to be extended with additional ones, such as cost or time. The architecture does not take a strict stance on what is a ‘compulsory’ set of model views, what is important is that all relevant fact types that relate to the function, information, decision, resource, cost, time aspect of the enterprise get represented in some modelling view. e.g. it may be that a modelling framework includes cost in the functional model (through activity based costing) or it may be that process costs are represented in another, separate view. It is expected that at least the four main views listed in Figure 1 will be available. However, these views should be (as they are appropriately called) only projections of an underlying integrated model. In other words, adding for example a view of temporal behaviour should be done by incorporating time in the underlying metamodel and then expressing the temporal aspect through a view mechanism. A system that implements the requisite CIMOSA modelling languages is CIMBIOSYS.

- Candidate architecture — a ‘candidate architecture’ is an alternative presentation of a generic architecture. Such candidates are the PERA two tiered diagram (Figure 2), the CIM-OSA cube (Figure 3) or the IMPACS cube (Figure 4) of GIM [Figures 2-4 show the original life-cycle coverage of the

Figure 1 Matrix representation of the life-cycle of any entity
A framework to define a generic enterprise reference architecture and methodology: P Bernus and L Nemes

Identification of the CIM business entity

Concept layer (mission, vision and values)

Definition layer (functional requirements)

Specification layer

Detailed design layer

Manifestation layer

Operations layer

Figure 2 The Purdue Enterprise Reference Architecture is one candidate GERAM compatible architecture

CIM-OSA and GRAI-GIM architectures; the coverage of these has been recently extended to match the GERAM life-cycle definitions. How that is achieved is beyond the scope of this paper.

A candidate architecture can be mapped onto the matrix of Figure 1 by graphically presenting it as a collection of ellipsoids drawn on the matrix diagram. Each ellipsoid corresponds to a description category of the candidate architecture, and an ellipsoid normally covers more than one subject matter of the matrix. If not all subject matters are covered by ellipsoids representing a candidate architecture, then that candidate architecture is not yet complete (that area is missing from its coverage).

- **Methodology** — given a matrix that represents the life-cycle of a particular entity it is then possible to define a methodology as a set of procedures and methods which enable the user to produce the set of necessary subject matter specifications (see above). The graphical representation on the matrix of a methodology’s coverage is by shading the ellipsoids that correspond to the components of the candidate architecture. If not all ellipsoids are shaded for a methodology that means that the methodology is not yet complete relative to its own architecture definition.

- **Generic models** — models in the left hand column of the matrix are semantic definitions of the concepts that can be used in the given subject matter’s description. These ‘models’ are in fact ontological theories describing the meaning of common concepts of the subject matter. If such a model is available, then the graphical representation on the matrix of this fact is a filled triangle in the corresponding area.

Proposed four matrices of the generic enterprise reference architecture

A life-cycle matrix describes a life-cycle of some entity. Any such entity has a ‘service to the customer’ (purpose, mission, or service) which is the mere reason for its existence. The ‘operation’ phase of the

Figure 3 The CIMOSA architecture-covering the R/D/I phases of GERAM (how further coverage is achieved in CIMOSA is beyond the scope of this article)

Figure 4 The IMPACS modelling framework of GRAI-GIM-covering the R/D/I phases of GERAM (how further coverage is achieved in GRAI-GIM is beyond the scope of this article)
given entity's life-cycle is responsible for carrying out that mission.

The life-cycle of the enterprise as an entity was presented in matrix form. Adopting the recursive view of integration* we now define the life-cycle of three more entities: that of strategic enterprise management, the enterprise engineering/integration processes, and the life-cycle of the Product. The definitions of the life-cycles would then consist of the concept, identification, requirements definition, design, implementation (detailed design) etc. of these entities in turn. The present document is a first attempt to do this.

Altogether the four entities and the corresponding matrices are:

1. The strategic enterprise management process life-cycle
2. The enterprise engineering/integration process life-cycle (*methodology life-cycle* in short)
3. The enterprise life-cycle
4. The product life-cycle

The enterprise life-cycle matrix #3

The entities of which the life cycle is described are of two kinds: product or process. A process type entity is typically short-lived and its life is expected to terminate after its once-only goal is achieved. A product type entity is expected to live on, perhaps undergo changes, but retain its identity through a longer life span. First it may seem unusual for the reader to talk about life-cycles of processes in the same way as talking about life-cycles of products or that of the enterprise. However, it should be born in mind that many enterprises themselves are best thought of as processes themselves, e.g. the project of building a bridge is best considered as a one-off process enterprise. In this sense the enterprise engineering project is an enterprise also, and therefore the life-cycle matrix describing any enterprise should also be applicable to the enterprise engineering project. (If this was not true then our framework for the reference architecture would not be generic — see the life-cycle of the enterprise engineering process below). Figure 1 is the representation of enterprise life-cycle (but also the representation of the life-cycle of any entity). Figure 2 is an alternative representation of a particular enterprise's life-cycle as shown by the PERA reference architecture.

The service to the 'customer' of the enterprise is embodied in the operation phase, in which the enterprise produces products or services needed by its customers. Therefore

- the customer service side is preoccupied by doing all those activities that produce the product or services of the enterprise, and
- the control information side of the Enterprise is preoccupied with all those activities that the enterprise does for itself, e.g. enterprise management is an activity that the enterprise does for its own benefit.

The division of enterprise activity in GERAM into customer service and control information sides is similar to the PERA architecture.

The methodology life-cycle matrix #2

The service to the customer of the enterprise engineering process is that it implements the life-cycle phases of the enterprise. In other words, when the enterprise engineering process operates it will carry out all those activities that are needed to identify, conceptually define, specify, design, and build (rebuild) the enterprise. Matrix #2 can be divided into two broad areas, namely the areas that are preoccupied with the 'customer service' of the enterprise engineering process and the 'control information' needed to control/manage the enterprise engineering process:

---

*Developed in SATT (Bernus, 1983).
The definition of the ‘customer service’ side of enterprise engineering projects

These areas of the matrix together cover the implementation level design (i.e. detailed design), the building, and the operation of an enterprise integration project. Recall that the matrix notation populates the matrix with ‘models’ (denoted by the ▲ symbol), and models in the generic and partial areas of the matrix can be used to create the necessary models of the individual entity in question (here the enterprise engineering process). The model of a generic ‘customer service’ of the enterprise engineering process is the description of how, in general, enterprise engineering processes should look; in fact, the descriptions/models of this area should collectively make up a complete implementation procedures manual!

For the above reason a methodology handbook (such as the Purdue Implementation Procedures Manual) is, in this matrix, a generic functional model of the customer service side of the enterprise integration processes. (See ‘PERA IPM’ in Figure 5). The representation of this fact on matrix #2 is a triangle in the ‘generic requirements’ area. A detailed investigation of the Purdue Implementation Procedures Manual and of the corresponding GIM documents should be undertaken to establish which other areas of the matrix are also covered and which ones could be further detailed to the benefit of the generic methodology. It is important to observe that although a methodology handbook is a complete description of the enterprise engineering project the detail of the project description is added on the individual level. Most probably the project manager will generate, from the generic model of the methodology, a project model. The generic model of the methodology need not be described in terms of time and resources, while an individual enterprise engineering project needs proper timing and a resource allocation. The connection between the methodology’s model and the project model is that the functions scheduled in the project model should all be instances of the functions described in the methodology’s model. e.g. if a project manager produces a Gantt chart of the individual project then the activities in the chart should all be activities defined in the methodology.

The definition of generic project management concepts, requirements, and designs — along with generic models (and modules such as project management software).

Similarly to the customer service side of the lifecycle the management of the entity (in this case the management of the enterprise engineering project) should be based on genetic models of project management. These generic models would then be specialised for the purposes of enterprise engineering projects and even further specialised for any particular enterprise integration project.

The generic models of this level would formulate a project management ontology and specialised models would be available for different types of projects. The best practice of project management in general, and enterprise-engineering-project management in particular, would be documented as models in this matrix, including the definition of these models on all levels from the concept to the implementation and operation (see Figure 6). The more models are available (i.e. not only requirements but also design and

![Figure 6 Management Requirements of Integration Projects in the Methodology Matrix](https://example.com/figure6.png)
implementation level models) the easier the task of the manager of the enterprise engineering project. Models of former projects may be used as ‘go-by’ models, in which case repeated projects will use the former project’s individual models as typical and modify these to suit the next project. This is a very efficient (and in industry often used) method.

The ‘operation’ phase of this matrix is the actual enterprise engineering activity, while the ‘build’ phase is preoccupied with creating the enterprise engineering project team (including education of team members). Phases before the build phase are carried out by management (as part of the strategic enterprise management operations) and define the mandate of the enterprise integration/enterprise engineering project, organise its project management and plan the project’s execution.

It immediately follows that a generic enterprise engineering/integration methodology is a pattern (in the left hand side) of matrix #2; and there can be specialisations of the methodology. (See Figure 7.)

A few useful specialisations would be:

- Methodology for integrating enterprises of a given sort, (e.g. a methodology for the integration of government departments etc.);
- Methodology for innovative re-engineering of an enterprise as opposed to a methodology for evolutionary enterprise engineering projects,
- Mix-and-match methodology, which is a combination of available partial methodologies for the individual task at hand.

Specialised methodologies have the advantage that they better suit industry (filling in the gap between abstract methodological instructions and practical steps). Furthermore, specialised methodologies are more efficient because there is less need to spend time on adopting them to the special circumstances of the individual project. (The specialisation activity is represented in Figure 7 as an arrow labelled ‘develop methodology for target industry’.)

The practical advantage of defining, in detail, the subject matters of matrix #2 would then be that the connection between enterprise integration/enterprise engineering and project management would be apparent in such an extended reference architecture and this would provide a common ground for two separate — although in the practice mutually reliant — disciplines (enterprise engineering and project management).

The strategic enterprise management life-cycle matrix #1

As the process of matrix #2 (enterprise engineering) results in all deliverables of matrix #3 (the enterprise) it is relevant to ask: what is the process that brings about the enterprise engineering process and is it of interest to the enterprise integration community?

It is proposed that the subject of matrix #1 is the life-cycle of strategic enterprise management (SEM). As SEM operates it brings about (among other things) enterprise engineering projects.

Although not all subject matters in that matrix are directly relevant to enterprise engineering, some parts are. Namely, the methods for the identification of the necessity and the starting of enterprise engineering efforts belong to the operation of strategic enterprise management, and these methods should also be part

![Figure 7 Specialising the Generic Enterprise Engineering Methodology](image-url)
of the generic enterprise integration methodology. Similarly, the selection of a suitable methodology (and the methods to create a specialised methodology) belong to this matrix.

Matrix #1 should be defined to the extent that the above functions can be identified. The requisite methods need to be specified and (if possible) paradigmatic models should be incorporated.

The product life-cycle matrix #4

Similarly to the way matrix #1 was discovered it is possible to ask: which entity’s life-cycle is implemented by the enterprise? The obvious answer is that it is the life-cycle of the products (services) produced by the enterprise.

In fact many attempts to implement the integrated information flow in the enterprise have been based on integrated product models. The question arises, then, how does enterprise engineering — as an integrating factor — relate to product-model-based integration?

With the introduction of the product life-cycle matrix it is now clear that the enterprise model should be such (including the requirements, design, detailed design and building and operating the enterprise) that it implements the full product life-cycle, or a part of it, as determined by the mission of the given company (enterprise).

By looking at the product life-cycle matrix the enterprise engineer can derive a full set of requirements for the enterprise designed. The knowledge of the product life-cycle is therefore a prerequisite of the successful enterprise engineering process.

The product life-cycle matrix (matrix #4) should be defined and disseminated together with the generic enterprise reference architecture and methodology. Furthermore, matrix #4 would allow to compare product modelling languages and their scopes (e.g. STEP[^1]), showing the connection with the product life-cycle, and with the enterprise life-cycle.

As an important result, this approach unifies product model-based and enterprise engineering-based integration.

Are all the matrices identified?

To find out whether there is a matrix below matrix #4 we would need to know the outcome of the ‘operation’ of the product. Since matrix #4 describes all types of products there is no common engineering concept that grasps the outcome. In this sense the product model is fundamental to enterprise modelling.

To find out if there is a matrix above #1 we need to identify what operation brings about strategic enterprise management (SEM). We believe that processes which bring about SEM are beyond the realm of engineering and the enterprise — they belong to politics and education. Therefore the four matrices do cover the entire scope of enterprise integration.

Further connections between the life-cycles identified

When developing the present form of the matrix representation it was the target of considerable debate whether to include the ‘build’ phase in the life-cycle models or not? CIM-OSA did not include this in the CIM-OSA cube, nor did the first version of the matrix presentation as contributed to[^2]. This controversy can be resolved now:

- The life-cycle model includes all major phases of building the enterprise thus building it can not be left out (‘build’ should be in the life-cycle matrix).
- The descriptions of each phase are about the entity under consideration. e.g. if looking at matrix #4 then in each phase there should be some description of the product, because the matrix describes the product’s life-cycle.

This is not true of the build phase. The description given there is not what the product is but how the product is built. However, the ‘build’ phase is actually a part of the ‘operate’ phase of the matrix above. That is: by operating the enterprise the products are built.

It therefore does not harm incorporating the ‘build’ models in the life-cycle of any entity with the understanding that the build process is part of the operation one level above.

The components of GERAM

Identification of deliverables in the development of the generic reference architecture and methodology

The functional components of a fully developed generic enterprise reference architecture and methodology would have to be the deliverables below:

Generic enterprise reference architecture (GERA)

This is the definition of enterprise related concepts, with the primary focus on the life-cycle of the enterprise. Since the life-cycle can be considered as a design process the architecture will also have to identify the results and the intermediary components of this design process.

Generic enterprise engineering methodology (GEEM)

This is the description, on a generic level, of the processes of enterprise integration. In other words the methodology is a detailed process-model, with instructions for each step^ of the integration project. (Note, that it is possible to define more than one viable generic methodology.)

^'Step' is to be understood as a functional component not as an element of a temporal sequence of actions.
Generic enterprise modelling tools and languages (GEMT&L)

The engineering of the integrated enterprise is a highly sophisticated, multidisciplinary management, design and implementation exercise during which various forms of descriptions and models of the target enterprise need to be created. To express these models potentially more than one modelling language is needed.

Generic enterprise models (GEMs)

Generic enterprise models capture concepts which are common to all enterprises. Therefore the enterprise engineering process can use them as tested components for building any specific enterprise model. It is possible to distinguish two levels of models.

Ontological theories (OTs) — these theories describe the most generic aspects of enterprise-related concepts. (They can also be considered to be 'meta-models' because the facts and rules in them are about facts and rules of enterprise models). Ontological theories play similar role that 'data models' play in database design. Ontological theories capture the most basic properties of enterprise-related concepts (function, structure, dynamics, cost, etc.), and define the semantics of the modelling language used.

Reusable enterprise models — these are of the following types:

- Models which capture some common part of a class of enterprises. This type of reusable enterprise model can be used as a building block of a complete set of models.
- Paradigmatic (prototypical) models which describe a typical enterprise of a class. Prototype models can be subsequently modified to fit a particular case.
- Generic (abstract) model of a part of a class of enterprise which captures the commonalities but leaves out specific details. This type of model is of the 'fill-in-the-blank' type.

Generic enterprise modules (GMs)

Modules are products, which are standard implementations of components that are likely to be used in enterprise integration — either by the enterprise integration project or by the enterprise itself. Generic modules can be configured to form more complex modules for the use of an individual enterprise. Two typical generic modules are:

- Enterprise engineering tool (EET)
- Enterprise integration platform or integration infrastructure (IIS)

Strategy to fully develop the components of GERA

In its report and subsequent article the IFIP/IFAC Task Force identified two possibilities to develop the required complete architecture and methodology:

- develop a new generic architecture and methodology, or
- combine existing ones.

Both technically and strategically the combination of existing architectures and methodologies has great advantages and this line is followed from here on. However, the strategy to develop GERA is not necessarily the same as the strategy to develop GEEM or the other deliverables.

The respective strategies should be acceptable to the community of users and developers which in turn depends on the following criteria:

- The quality and adequacy of the proposed deliverables for the task at hand;
- The apparent complexity: the proposed development must clarify and not obscure the area;
- The degree to which to capitalise on the value of investment in the respective (partially incomplete) reference architectures and to contain the cost of this further development;
- The retention of the advantages which result from having accumulated knowledge and intellectual property in the respective teams;
- The acknowledgement of the intellectual contributions of those who were the main engineers of the respective methodologies and avoidance of trying to create a Procrustean bed;
- The simultaneous maximisation of the benefit to the developers and users (to the extent possible).

It is possible to give separate consideration to each of the deliverables respecting in each case the above strategic requirements.

The development of the generic enterprise reference architecture (GERA)

The building of GERA should be similar to a car-race track and defining the rules of the game rather than building a Formula-1 T-model of enterprise engineering. GERA is an open system framework.

GERA must be a public domain document (possibly considered for standardisation) to allow meaningful and comparable competition to occur in the area of enterprise engineering. Similar goals were adopted by projects as CIM-OSA and the existing links to standards bodies (CEN-CENELEC, ISO TC184 etc.) must be built upon as well as links to other groups involved to enterprise modelling (ICFMT SIG).

Examples of candidate competitive products which could arise out of GERA, or could use GERA as a leverage are: computer-aided enterprise engineering, enterprise modelling and project management tools;
integration platforms or infrastructures; enterprise (re)engineering know-how; educational material etc. Milestones to develop GERA:

1. Take the Purdue enterprise reference architecture as complete in scope and add to it the areas discovered in GIM, CIM-OSA, and in the four matrices. This will include the development of the complete specification of GERA (the present article can be built upon), i.e. including the concept, identification and requirements and design of GERA. The specification will also have to include the definitions of the other deliverables (i.e. what are GEEM, GEMS, G1s and GMs).

2. Develop two presentations:
   Presentation of GERA for the user community — for the user community, preferably use the two tiered diagram format (now presenting four such diagrams according to the four matrices). The two tiered diagram format will group together those ‘subject matter’ areas which will belong together in the methodology. This presentation can be prepared on the basis of the PERA document by adding further chapters. Such presentation is not methodology neutral (not independent of a corresponding GEEM), therefore there may be several corresponding (compatible) GERA presentations for the user community.
   Presentation of GERA for the developer community — the second presentation is for the methodology, modelling, and standardisation community, prefer ably in the form of the four matrices. The matrix presentation will be seen as a further development of the CIM-OSA cube. The matrix presentation will use a refined subject matter presentation drawing on the corresponding CIM-OSA definitions and on the above subject matter groups (as developed for the user community).
   This presentation is methodology neutral and as such may be unique and common to all candidate architectures and associated methodologies (GEEMs).

The development of a generic enterprise modelling technology (GEMT&Ls) and generic enterprise models (GEMS)

The development of GEMT&Ls is not independent of the most generic enterprise models (GTs) because these are the meta-models for the language design. The ontological theories (OTs) define the semantics of the enterprise modelling languages and for this reason the two are dealt with together. In this article we not not propose any agenda for the development of more specific (in CIM-OSA terminology ‘partial’) models, although obviously such models will be of great practical significance.

The GEMT&L is a collection of modelling tools (languages/templates) which cover the needs of GERA. The definition can be done on two levels.

The first level is the separate definition of suitable modelling languages for each area. This is expected to be done for each area in GERA, at least for those subject matters which require model development in the application phase of the generic methodology (minimum requirement).

The second level is where the meta-models of these modelling languages are expressed as views of a common meta-model. If this is done then far more possibilities exist for enterprise engineering tools to analyse, execute (e.g. simulate), cross check, and validate models than in the first case. This integration of languages is also necessary for a smooth transition from the design to the building and operating of enterprise models.

The second case also allows for the adoption of the general modelling system and permissive modelling concept, which are based on the recognition that designers need to be able to use a wide variety of languages to express themselves. Once a core set of languages (e.g. CIM-OSA) is expressed as views of the underlying meta-model, it will be possible for other views to be developed as add-ons. Information contributed to the enterprise design database, using either the core languages or the add-on languages, will all be added to the underlying design database and thus becomes visible through all the defined views.

A major part of the necessary languages was developed in the CIM-OSA project and in GIM (organisational/decisional modelling), and some additions will have to be considered (e.g. temporal behaviour, cost/economies). There is no evidence that a closed language would ever be possible to serve all present and future modelling needs, and so we believe that the modelling technology of GERA should be an open system of languages. To share language definitions and to ensure the evolution toward a more complete set of modelling languages we propose to use an extensible language with meta-modelling capability (e.g. Telos*).

The granularity of modelling is a problem at the moment. CIM-OSA strives at model executability down to the level of enterprise operation. This requirement limits CIM-OSA’s ability to incorporate in the modelling language anything more than a simple ontological theory of events/actions and data (for efficiency reasons).

TOVE also strives at executability but is less concerned with the efficiency of execution. For this reason a complex ontological theory of actions, time, cost etc. can also be part of the meta-model.

The tension between the two approaches is apparent and a possible solution is proposed here, however, further investigation is necessary in this regard. It is proposed to express CIM-OSA modelling languages* as views of TOVE, a more complex ontology than the CIM-OSA meta-model. In this way

*And the additional ones which might not at the moment be part of CIM-OSA but required by GERA.
a design system based on CIM-OSA models could take advantage of the added semantics of CIM-OSA concepts, and the improved analysis capability in the design phase. However, for efficiency, the simpler but compatible CIM-OSA meta-model could form the basis for building and operating enterprise control software.

For the user of GEMT&L there would be no difference seen, but for the developer of enterprise engineering tools and integration platforms the differentiation would be visible.

Milestones to develop GEMT&L and GEM:
1. Determine which modelling tasks are not currently addressed by CIM-OSA or GIM.
2. Identify the complementing set of languages required. Draw on the IDEF set of languages.
3. Feed the result into the definition of the generic enterprise engineering methodology (GEEM) project. This ensures that the existing set of modelling constructs will be met by a compatible presentation of the enterprise engineering methodology.
Up to this point this is a feasible engineering project. The following tasks, however, need to be addressed by separate research projects.
4. Integrate these languages on the meta level — possibly using ontological theories developed in TOVE or IDEF. Since the core CIM-OSA languages are already based on one integrated meta-model this would not have to be worked out from scratch.
5. Express the languages as views of an underlying ontology.

One suitable expression of the modelling concepts of the languages so defined is the definition of the basic language concepts as modelling classes in a single object-oriented language. This step need not necessarily lead to the design of any new language (although some extensions may have to be done both on the language and the meta-model levels.)
6. Extend the underlying ontology as required.

The development of a generic enterprise integration and engineering methodology (GEEM)

It is acknowledged that more than one methodology may exist to cover the GERA framework. The present proposal points out how one such methodology can be developed with relatively small expenditure, drawing upon the Purdue and GIM methodologies. Other, equally viable proposals could be presented on the basis of only one of them or based on other combinations with proprietary methodologies.*

The first goal of developing GEEM is to provide one complete methodology which covers the entire GERA architecture. The coverage of the methodology may be achieved by adding more information to the Purdue Implementation Procedures Manual or adopting compatible methodologies which complement the Purdue Implementation Procedures Manual with methods in the areas not addressed (or not addressed in sufficient detail). In any case the presentation of GEEM in the form of a process model would considerably improve the common understanding, and ultimately the usability, of any such generic methodology (see the filled triangle representing this model on the left hand side of Figure 7).

The second goal of developing a GEEM is to detail the methodology such that a consistent set of modelling languages is proposed along with the presentation of procedures. This will allow the addition of examples more easily. If the methodology is in turn specialised to an area of industry (Figure 7, lower middle part), then GEMS can also be associated with it (i.e. reusable models, cookbooks, etc.).

It is supposed that the GEMT&L project provides the languages to detail GEMS in this way. However, because the basis of that is the CIM-OSA modelling languages, this is not a heavy constraint.

Milestones to develop GEEM:
1. Determine, given GERA, the areas which should be but are not yet covered by the Purdue Implementation Procedures Manual (IPM) — since this methodology is the closest to being ready for publication; work with further methodologies may be pursued in a similar manner.
2. Extend the methodology regarding the specification and design of the decision system of the enterprise, based on GIM where these details are highly developed and tried.
3. Based on the extensions to the CIM-OSA modelling framework (GEMT&L) illustrate, with examples and specific instructions, the steps of the methodology.

Conclusion

We presented GERAM, a proposal of a framework that covers the enterprise integration area and can serve as a neutral taxonomy of enterprise integration related methods, products etc. The main feature of GERAM is that it is not a separate new proposal which tries to compete with existing reference architectures, but one that harmonises the contribution of each to the EI field.

The article proposed a road map to the development of each of the components of GERAM, for use by user communities, standardisation, and last but not least the developer community.

One advantage of GERAM is its enabling nature: through the definitions of GERAM it is possible to meaningfully relate (and possibly combine) areas
such as 'business process re-engineering', 'total quality management', 'concurrent engineering' etc.

At the same time each of the reference architectures (life-cycle models for enterprises) continues to be developed (and undoubtedly others will join). The role of GERAM will then be to allow the comparison of the advantages of particular reference architectures and the selection of the best one that matches a particular organisation's requirements. It is equally possible, that a particular organisation selects parts of existing architectures and develops a best match architecture for in-house use. This would of course not be possible unless a commonly acceptable framework of concepts — which GERAM is meant to be — is available.

GERAM shows how reference architectures can be applied to various subclasses of enterprise, e.g. product oriented vs project enterprise (e.g. manufacturing firm vs a large engineering project as an enterprise), or real vs virtual enterprise (e.g. company vs consortium). As an example, the life-cycle of an enterprise engineering project is described in GERAM the same way as the life-cycle of the enterprise.

Through the definition of the GERAM matrix types a road-map of substantial scope has been created that allows better communication between disciplines contributing to enterprise integration.

References

1 Williams, TJ The Purdue Enterprise Reference Architecture, Instrument Society of America, Research Triangle Park, NC, USA (1992)


4 Fox, MS 'The TOVE project, towards a common sense model of the enterprise' in Enterprise Integration Modelling (C Petrie) MIT Press, Cambridge, MA (1992)


7 Bemus, P 'Rigour and permissiveness in the design of CAD/CAM systems' in Integration of CAD/CAM (D Kochan) North Holland, Amsterdam (1983) pp 161–178


10 Purdue Laboratory for Applied Industrial Control An Implementation Procedures Manual for Developing Master Plans for Computer Integrated Manufacturing TR 155, Purdue Lab., Purdue University, USA (1992)


12 Li, H and Williams, TJ A formalisation and extension of the Purdue Enterprise Reference Architecture and the Purdue Methodology TR 158, Purdue Lab. of Applied Industrial Control, Purdue University, USA (1994)


