AN ANALYSIS OF THE ZACHMAN FRAMEWORK FOR ENTERPRISE ARCHITECTURE FROM THE GERAM PERSPECTIVE

Ovidiu Noran

Griffith University (Brisbane, Australia)
E-Mail: noran@cit.gu.edu.au

ABSTRACT: This Article presents the analysis of the Zachman Framework for enterprise Architecture and its mapping onto the Generalised Enterprise Reference Architecture and Methodology (GERAM) framework / ISO 15704:2000 requirements. Aspects covered concern the ability of the Zachman Framework to cover the complete scope of the GERAM meta-mode, such as life cycle / life history concepts, modelling framework, enterprise entities and entity recursion, associated enterprise engineering methodologies, modelling languages and reference models.

1 Introduction

This analysis aims to assist a prospective user of the Zachman framework in using the ISO15704:2000 / GERAM requirements to check how it meets, or can be used to meet, the needs of an enterprise architecture project or program

1.1 A Short Introduction and History of GERAM

The main aim of the Generalised Enterprise Reference Architecture and Methodology (Bernus, '01), (ISO/TC184/SC5/WG1, '00a) is to generalise the contributions of various existing and emerging Enterprise Architecture Frameworks and Enterprise Reference Architectures in order to define a complete collection of tools, methods and models to be employed by any enterprise engineering and integration effort. As such, GERAM assists in the choice of tools and methodologies by providing criteria that need to be satisfied, rather than trying to enforce particular options. Used as a generalisation of frameworks, GERAM may also assist in establishing the completeness and suitability of frameworks proposed to form the basis to a particular change process.

There have been several notable attempts to map the existing life cycle architectures and their associated artefacts against each other (e.g. (Williams et al., '96)), which have highlighted some of the difficulties encountered in the mapping process.

The result of the mappings of existing architectures against a fixed reference was a matrix-like structure of requirements (Bernus, Nemes, and Williams, '96) – a two dimensional form of the GERA modelling framework.

Figure 1: A possible GERA metamodel (based on ISO/TC184/SC5/WG1, '00a).

The modelling framework was also supplemented with specific concepts such as entity type, recursion, life history, etc. The architecture framework was then built by putting together the generalised modelling framework with all essential generic concepts of enterprise engineering / modelling - such as enterprise models, modelling languages, generic enterprise modelling concepts, partial models, etc. In addition, a three-dimensional representation of the modelling...
framework, proposed by Williams and Li,\(^6\) has improved the readability of the previously used two-dimensional matrix.

The current outcome of these efforts is reflected in the GERAM document contained in (IFIP-IFAC Task Force, '03) and (ISO/TC184/SC5/WG1, '00a).

GERAM, as a generalised architecture framework, aims to categorize life cycle architectures and their associated artefact types (methodologies, reference models, ontologies, etc).\(^7\) GERAM alone cannot be used to engineer an enterprise; however, it should be used to assess what is needed for a given enterprise integration task (or task type).

The following analysis will focus on the Zachman framework and demonstrates how it may be used to cover the GERAM metamodel shown in **Figure 1**, hence discussing:

- use of Zachman reference architecture to cover the GERAM life cycle and life history concepts;
- modelling languages proposed in Zachman's modelling framework;
- enterprise engineering methodologies;
- reference models;
- other important elements in the GERAM framework (generic modelling concepts, enterprise modules, modelling tools, etc) and how they apply to the Zachman framework.

### 2 Life Cycle Phases

Mainstream system engineering and enterprise integration literature acknowledges the existence of two main types of architectures: architectures that represent the structure of a system at a given point in time (snapshot) and life cycle architectures,\(^6\) which describe the possible phases and artefacts involved in the life of a system.\(^8\)

An essential part of the requirements for a life cycle architecture is to include the concept of *life cycle phase*, understood as a set of possible processes or activities\(^5\) which may be - once, several times, or not at all - be performed in the enterprise during its existence (i.e. throughout its *life history*).

Common misconceptions resulting from a lack of consistent terminology across the field\(^9\) may lead to the flawed conclusion that life cycle phases imply succession (i.e. temporality). The life cycle dimension in the GERA sense abstracts from time and it is simply a repository of possible processes performed during the life history of a system, categorised according to the level of abstraction that these processes use to consider the enterprise entity in question.\(^10\) Some of these processes may be performed repeatedly and in different succession and some may not occur at all during the existence of the system.

#### 2.1 The Zachman Life Cycle Aspect

The Zachman framework takes a somewhat original approach towards life cycle, presenting life cycle phases as perspectives of the various stakeholders involved in the enterprise engineering effort. Although the concepts of life cycle and life cycle phases are not explicitly present, a mapping is still possible. This is because various stakeholders use different levels of abstraction to consider the enterprise entity in question, which match GERAM life-cycle phases.\(^11\) A direct mapping is not possible since GERA contains *types* of activities, while Zachman describes *deliverables* that given stakeholders produce.

The Zachman framework does not include a stakeholder perspective that can explicitly and completely match the GERA Identification phase. However, in analyzing the contents of the Zachman cells in the Objectives / Scope row (refer **Figure 2**) one realizes that these cells contain deliverables that describe the business\(^12\) (very high-level functional / organisational structure, boundaries, etc - which match deliverables created in the GERA Identification phase) but also business goals and strategy (which are developed in GERAs Concept phase). Hence, the Objectives Scope perspective maps\(^13\) onto the GERA Identification and Concept life cycle phases.

As shown above, a single GERA phase may map onto several Zachman perspectives and several GERA phases may converge to a single Zachman perspective. As an example: the Requirements phase maps to both the Business Owner's perspective and the Architect's perspective. This is because the architect should be able to gather user requirements and translate them into system requirements (which must be satisfied by the Preliminary Design). Bridging across user- and system requirements is optimal if e.g. the architect is involved both in the GERA Requirements and Preliminary Design phases.

---

\(^8\) the proposal was presented at meetings of the IFIP-IFAC Task Force.

\(^7\) as such, GERAM may resemble an empty bookcase providing shelves for content in the field of enterprise integration, and it may provide a to-do list for a given enterprise integration task.

\(^5\) such as conception, development, build, operation, dissolution etc.

\(^6\) a process may be a "partially ordered set of enterprise activities" (ISO/TC184/SC5/WG1, '00b).

\(^9\) several such problems were described and solutions proposed in (Noran, '00) - e.g. phase vs. stage, life cycle vs. life history, etc.

\(^10\) e.g. the design process may consider the enterprise entity in question on a different level of abstraction than the identification process

\(^11\) in the following 'business' and 'enterprise' will be interchangeably used.

\(^12\) although a direct mapping is not possible, the term 'mapping' is used in this section with the meaning of association of *deliverables and stakeholders perspectives* (Zachman) with *types of activities within a life cycle phase* (GERA).
The meanings attached by the subcontractor may however only be involved in the Implementation phase. The current involvements vary considerably (depending on the user's background and current involvements). Users of the Zachman Framework may find it useful to add another column to the Zachman framework, labelling it with GERA life cycle phases (potentially with even finer grain subdivision of phases than used by GERA), and then assigning concrete human roles to these (according to the problem at hand) as per the first column of the original Zachman Framework. Thus, the practitioner has a map of 'who does what'. This would also mean that the perspective of one human role (e.g. the architect) would include more then one model (such as a functional requirements model and an architectural model) if necessary.

In conclusion, the Zachman framework represents the perspectives of the human roles involved in the life cycle phases described by GERA.

2.2 Conclusion

The life cycle concept as understood in GERA is implicitly present in the Zachman framework. Any entity being conceived, designed, implemented and possibly operated will eventually reach the end of its useful life. The Zachman framework does not explicitly cover this aspect, although this is the very phase when (a) a decision is made to either retire or revamp the entity and (b) irrespective of the decision taken in (a), the relevant enterprise knowledge and other resources are transformed, either for their preservation or their reuse. A 'complete' (in GERA sense) reference architecture should attempt to cover this life cycle phase to a depth adequate to the purpose of the modelling task.

3 Life history: The Timeline Aspect

Any business entity aspiring to a lasting presence in today's ever changing environment must adapt to its surroundings, via change processes of its own. An agile enterprise not only copes with, but also thrives on (and possibly pre-empts) the environment changes. Most of the change processes occurring during the life of an enterprise are concurrent and interact with one another. This character of change processes within an enterprise may be suitably modelled using the GERA life history concept.

As previously shown, the concept of life history has often been mistakenly identified with that of life cycle. GERA makes a difference between life history - 'the actual sequence of steps a system has gone...'(or will most likely go), "...through during its lifetime" and life cycle - seen as "...the finite set of generic phases and steps a system may go through over its entire life history." (ISO/TC184/SC5/WG1, 2000b).

During their life histories, enterprises interact. This is explained in (ISO/TC184/SC5/WG1, '00a) using the concept of recursion; the same concept exists (under various designations) in several architectures, including in the Zachman framework. Recursion is explained in more detail in Section 8.2.

3.1 Life History Aspects in Zachman

The Zachman framework does not have an explicit life history concept, and thus only an implicit mapping is possible. For example, life history may be represented here via the When column (as in Figure 4 and Figure 5), which implies temporality and succession. Such an approach works for successive / non-parallel change processes. Alternatively, one could simply omit the 'When' column from the Zachman Framework and represent it on a separate 'time' axis (as represented in Figure 3). For each change process there may be...

---

14 other Zachman and GERA frameworks' dimensions omitted for clarity.
15 i.e. allowing for constraints such as tool / technology/ material availability, relevant standards / laws, etc (Zachman, '87)
16 refer (Hay, '00) for an interpretation of the Zachman framework cells from the Oracle life cycle perspective.
17 it is possible for example to identify the Objectives/ Scope view with a 'Manager's (or CEO's) View' and the 'Functioning System' view with an 'Operator's View'.
18 e.g. resources / knowledge must be properly archived and human resources retrenched / retired in the case of obsolescence. In the case of artefact revamping, existing resources may need converting / aggregating / refining etc, and human resources may need retraining.
19 hereafter also referred to as an "enterprise" or "enterprise entity"
20 agility: "capability to respond" (Goranson, '98).
21 note that an important part of these change processes involves the life histories of other enterprise entities.
22 it is preferable to use a time axis orthogonal to other dimensions since time is usually an independent aspect (e.g. processes (function), data, resources, locations (network), motivations may all change / evolve with time).
separate deliverables (as accommodated in a Zachman framework created for that change project), while the occurrences of the life cycle activities for multiple change processes may be simultaneously shown on a combined time diagram. To preserve the applicability of tools that support the Zachman Framework, the temporal aspects of each separate change process could be combined on an aggregate time diagram, thus showing their possible interactions. Figure 3 shows a possible interpretation of life history in the Zachman framework according to the GERA life history concept. In such a life history, there would be a purpose associated with each deliverable (characterised in the Zachman framework’s ‘Why’ column). This purpose can refer to the role of the deliverables in the life history of the entity.

Another temporal aspect in the Zachman framework is the concept of versioning, described in (Sowa and Zachman, '92) as one of the three aspects of recursion applicable to the Zachman framework. Versioning may be considered a temporal concept since it identifies stages in the evolution (hence in the life history) of an artefact. NB in this sense, versioning is present in an implicit form in all reviewed frameworks. The reader should note that irrespective of the specific way in which a given architecture’s modelling framework decides to subdivide views, these subdivisions must be views of an integrated metamodel. Thus, the way views are subdivided is less important, as long as the combined scope of the modelling framework is complete. For example, it should be possible (using e.g. an enterprise engineering tool) to look at the collection of integrated enterprise models through any of the views defined in the various existing modelling frameworks.

4 The Modelling Framework

The GERA Reference Architecture includes the definition of the GERA Modelling Framework. The significance of the modelling framework is in the fact that it defines the scope of potential models that change processes may need to use, or to produce. Therefore, the explicit (and implicit) scope of the Zachman framework will be investigated here. There are two questions to be asked:

- what explicit guidance is provided by the Zachman framework about the potential list of models that might need to be created during the course of various change processes;
- to what extent is the Zachman framework able to accommodate the complete scope of modelling as defined in the GERA Modelling Framework? This represents the implicit ability of the framework to accommodate any model that might be required, even though the framework gives no specific help for the user to identify that model as a potential candidate deliverable.

The reader should note that irrespective of the specific way in which a given architecture’s modelling framework decides to subdivide views, these subdivisions must be views of an integrated metamodel. Thus, the way views are subdivided is less important, as long as the combined scope of the modelling framework is complete. For example, it should be possible (using e.g. an enterprise engineering tool) to look at the collection of integrated enterprise models through any of the views defined in the various existing modelling frameworks.

4.1 Zachman’s Modelling Framework

The Zachman Framework initially addressed the information subsystem of an enterprise (Zachman, '87). It also appeared to be restricted to the modelling framework component of the architecture framework. Subsequently, the scope of the framework has been extended to cover aspects informally defined in the entity's history (past succession of life cycle activities), one may (and should) make appropriate selections from the pool of life cycle phases and thus influence the future life history of the entity. For this purpose, possible future life histories may be represented on alternative life history diagrams.

---

23 versioning may be considered a temporal concept since it identifies stages in the evolution (hence in the life history) of an artefact. NB in this sense, versioning is present in an implicit form in all reviewed frameworks.

24 refer Section 8.2.1 for details.

25 e.g., a higher granularity modelling framework gives more specific advice on the kinds of models compared to a low granularity modelling framework.

26 this is presently not possible because of the lack of a commonly agreed metamodel and ontology for integrated enterprise models.

27 the information subsystem (or information system) has always been part of enterprises (and has been enabled by the information technologies historically available) but has recently dramatically increased in importance.
The previous version (Sowa and Zachman, '92). This extension and formalisation, together with the emergence of modelling methodologies and partial models have provided for the evolution of the original Zachman modelling framework into the Zachman architecture framework.

A mapping of the Zachman framework abstractions onto the GERA Function / Information / Resource / Organisation views is shown in Figure 4. The Zachman framework clearly acknowledges the Information aspect via the Data abstraction - onto which the GERA Information view may be mapped.

The remaining GERA views however are not 'one-to-one' mappable onto the Zachman framework:

- the People abstraction in Zachman is mappable to the GERA Organisation view, since it comprises a similar scope. However, people are also a (human) resource - hence the People abstraction maps onto the Resource view in GERA as well;
- Zachman's Function abstraction covers the domain type defined by the GERA Function view. The Zachman Motivation abstraction however, provides for rule modelling - which also belongs to the Functional view in GERA;
- The Network abstraction in Zachman covers the non-human resources - therefore, together with the People abstraction, it maps onto the GERA Resource view.

The mapping of the Motivation abstraction on the GERA Function view deserves an extended explanation. Rule modelling is part of behaviour modelling, which in its turn is part of functional modelling. When modelling the functional aspects the following degrees of detail may be observed (not necessarily in the order shown):

- activities are shown with their inputs and outputs (e.g. using IDEF0) which are objects (but no details of these objects are shown);
- events are added (e.g. using Petri Nets, Harel State Charts, State Transition Diagrams, etc) so as to model system behaviour / dynamics (reaction to events).

---

28 the initial name of the Zachman framework, i.e. 'Framework for Information Systems Architecture' (Zachman, '87), was subsequently changed to 'Framework for Enterprise Architecture' (Zachman, '90a), possibly in order to reflect and support this evolution.

29 a direct mapping is again not possible since generally, information is data, interpreted by the user.

30 note that the GERA modelling framework considers the Organisation as the relation between human resources (people) and functions (roles that humans take). Resources (including human resources) are modelled in the Resource view (i.e. human resources are identified and their capabilities are described);

31 modelling languages used for Function modelling usually allow (or require) an up-front statement of the modelling context, describing why and for what purpose the model was constructed;

32 ICAM (USAf Integrated Computer Aided Manufacturing) DEFinition Function Modelling Language (ICAM, '81)

---

33 a specialisation of Petri nets where tokens own attributes called colours. Refer (Jensen, '92) for details.

34 for a general overview of the most important members of the IDEF family of languages refer to (Menzel and Mayer, '98)

35 a specialisation of Petri nets including temporal aspects (i.e. a duration is associated to each transition or each place).
covered and (b) which of those possible aspects are indeed covered at each perspective. A partial solution is to explicitly specify this information for each cell by attaching separate descriptions to the cells in question. One other solution is to use the GERA multi-view approach and create subdivisions inside each cell of the Zachman framework corresponding to each of the applicable GERA views. A filled cell subdivision would then indicate that the matching GERA aspect is covered within that cell.

In Figure 5, M, H denote the Machine and Human aspects in GERA; Hw, Sw are the GERA Hardware and Software aspects; and CS, MC are the Customer Service and Management / Control views of GERA. Figure 6 presents an extract (a single cell) from the right-hand side of Figure 5, covering the Architect's view of the Network abstraction and using the descriptions provided in (Zachman, '87). The represented cell explicitly covers the Software and Hardware aspects of the Non-Human (Machine) side of the Management and Control of the enterprise. This extract is presented primarily to illustrate the potential use of the GERA views to further scope the Zachman framework cells and to identify potential coverage gaps and overlaps.

Note. Although the Zachman framework considers the Network abstraction to cover location and structure, in this example the Hardware description of the Human side of both Management / Control and Customer Service in Figure 6 has been limited to Location. This is mainly due to the presence of the People abstraction, which may be more appropriately used to represent the structure of the enterprise personnel (aggregation into organisational entities). The Zachman framework does not explicitly contain a genericity dimension. However, separate frameworks may be used for the three main areas of genericity (generic - partial - particular) as defined in GERA.

36 the software and hardware non-human side of the Customer Service (manufacturing) may also be considered as covered, although this is not explicitly stated in the reference used.

In doing so, however, the 'Functioning System' perspective should be omitted from the generic and partial frameworks (the shaded areas in Figure 7). The particular Zachman framework would contain fully developed models, while the partial framework would contain reference models. The generic level framework would then contain metamodels - modelling constructs and their relationships, ontologies and glossaries.

Referring to Figure 7, there is another (implicit) genericity dimension along the Perspective axis. The views of the people involved in the life cycle phases of the artefact may be "of a different nature" (Zachman, '87) but they do refer to the same artefact (since they belong to the same framework) on a different level of

37 these may be models of prototypes, 'class models' or models of the common parts of a set of enterprises, according to the GERA concept of partial model.

38 shaded areas mark the non-applicable views; the instantiation shown only applies to the particular Zachman framework.
detail. For example, a high level user requirement in
the owner's view may translate into several system
requirements in the architect's view, become
implementable by using a combination of technologies
/ materials / tools described in the builder's view and
physically be fulfilled by putting together the
components built in the subcontractor's view.
Each representation may use specific means of
expression, but in essence they refer to the same
artefact in various degrees of detail. The degree of
specialisation of the views (models) will gradually
increase as one moves closer to the functioning system
view (downwards, in the Zachman framework). This
happens because decisions have to be made regarding
various system parameters (e.g. the kind of architecture
to be used, processes / technology / human skills to be
employed etc). Instantiation then occurs when abstract
representations become reality. This only occurs in the
Builder / Subcontractor View of the particular level
Zachman framework (mapped onto the GERA
Implementation phase (refer Figure 2).
The Zachman framework also appears to include a
third, implicit genericity dimension derived from its
recursiveness, as subsequently shown in Section 5.1.

4.2 Conclusion
A modelling framework is a structure containing
placeholders for artefacts needed in the modelling
process. Depending on the structure of the framework,
the type of these artefacts may be limited to models, or
it may extend to other construct types such as partial
models, metamodels, glossaries, etc. For example, the
GERA modelling framework contains placeholders for
artefacts whose type is described in the GERA
metamodel (Figure 1).
Some of the existing modelling frameworks are
'complete' in the sense that they can accommodate the
complete GERA scope, but each goes into detail from
different aspects. Therefore, the user of any of the
architectures should consider the GERA scope
definition as a guidance or checklist for potential
omissions of deliverables that may be needed but are
not covered in the respective architecture.
Note. It is often the case that several of the possible
subdivisions shown in Figure 6 apply only to some
modelling tasks. GERA therefore includes subdivisions
into views wherever the models used in the relevant
life cycle phase do make a difference between the
contents of these views.

The reader should also note that, ultimately, it is the
time enterprise engineering methodology that needs to
identify the models that are necessary to support an
time enterprise architecture viewpoint; therefore, the
mapping provided in this article is primarily useful for
methodology developers and would not necessarily be
utilised directly by end users. The methodology
developer should consider, based on the objectives of
the given change process, which artefacts are necessary
and useful – taking into account that practicing
enterprise architecture is likely to have multiple
objectives, and that artefacts produced should support
every one of these.

5 Modelling Languages
The practice of Enterprise modelling needs to produce
a set of models aimed at an intended audience in order
to communicate the models' meanings. In order to
achieve this purpose, the models must be constructed
using languages that a) are appropriate to the
enterprise aspect modelled and b) can be understood by
the target audience. For this purpose, existing
languages may be adopted (as-is, or customised) or
new languages may be created. Any extensions to
chosen existing languages must be fully explained and
justified, as they in effect change the structure (the
metamodel and therefore, implicitly the meaning) of
the language itself. Newly designed modelling
languages should be based on sound ontologies
(metamodels with semantic rules), which will ensure
the required consistency and determine the expressive
power of the modelling as required by the desired
time enterprise view.
GERAM contains several requirements regarding
modelling languages definition (constructs and
semantics), expressiveness, required domain coverage
of the chosen set of modelling languages and the
requirement that models produced should be able to be
integrated across various subject areas of enterprise
modelling.

5.1 Modelling Languages for the Zachman
Framework
In today's ever faster changing business environment, it
is understandable that reference architectures aim to
remain as generic as possible. A balance must be
reached though between the degree of genericity and
the level of usability of an architecture framework.

e.g., the GERA Identification phase does not separate
Human/Machine or Function/Information/etc aspects. This
is because in GERA the Identification phase abstracts
from decisions that would need the differentiation
between these aspects.

Note. The integration in question may be based on a common
ontology of the languages and/or using integrity
constraints. The need for common (integrated) ontology
and metamodel is particularly true in the case of
some sets of languages (e.g. IDEF), which are not based on,
or even define a common ontology, despite claiming to
belong to the same 'family'.

e.g. too general an architecture may not provide any useful
guidance as to what deliverables may be needed and how
to obtain them, while an excessively detailed architecture
An architecture framework may avoid prescribing specific instances of artefacts (such as a particular modelling language or modelling tool) but still make recommendations or at least describe the essential requirements for such artefacts.

In this respect, the Zachman framework does not attempt to define dedicated language constructs, nor does it try to enforce any particular third-party modelling languages. The Zachman framework does however suggest several possible modelling languages in its framework cells (refer Figure 8). Modelling tool and / or methodologies developers must employ various proprietary or third-party modelling languages in order to populate their tools and/or methodologies. The lack of guidance in choosing appropriate modelling languages for the Zachman framework requires users / developers to choose a language they feel appropriate for a specific modelling task. It does not, however, offer a mechanism to ensure the consistency or interoperability of the models created using the chosen languages - unless this task is accomplished by choosing an integrated family of languages (based on common metamodel(s)).

Figure 9 shows one of the many possible selections of languages. This particular selection has been compiled using languages proposed by Popkin Software (Popkin Software, 2001) and additional languages such as First Order Logic (FOL), Structured English and Decision Tables.

Figure 9 Possible Modelling Languages to populate Zachman's Modelling Framework.

In Figure 9, RP denotes Rich Pictures - using symbols to express various concepts in a semi-formal manner. ER(M) denotes ER models allowing M:N relationships. IDEF1 (rather than IDEF1x) is proposed by the author at this level in order to maintain a high degree of independence from the method of implementation. GRAI Nets have been included in the How abstraction since they may further detail the activities inside the decision centres.

5.2 Conclusion

Given the diversity of objectives that underlies enterprise integration efforts, practitioners of enterprise architecture need to be able to select modelling languages that allow the expression of their intent and design as needed for the particular task. Architectures such as Zachman aim not to prescribe a complete set of languages, instead arguing that the given engineering domain already has adequate means to describe any deliverable, and the architecture is there just to create a checklist of deliverables and to make their relationships explicit. Such architecture frameworks only give examples of selected typical modelling languages.

For some areas of modelling, typical modelling languages may be recommended without restricting the use of the framework to a specific application domain or change objective (e.g. one can generally propose the Entity relationship Data Model for information modelling, IDEF0 for activity modelling, and IDEF3 for process modelling). On a generic level this restraint may allow the Modelling Framework to gain longevity by being non-prescriptive.

However, most enterprises today share some common objectives likely to persist for a long time. While these objectives are quite generic in nature, it is possible to identify common needs of change processes regarding types of models or descriptions, such as to express

---

44 by means of graphical representation (i.e. typical entity and relationship symbols for an ER diagram, map for locations, typical Input, Control, Output, Mechanism (ICOM) symbols for IDEF0, etc. Some recommendations also exist in (Sowa and Zachman, 1992).

45 this is a common problem among the architecture frameworks that offer little or no guidance and language requirements.

46 may be used in combination with text
47 refer (Menzel and Mayer, '98) for IDEF languages
48 or alternatively IDEF1x, UML Class diagram.
49 or alternatively Data Flow Diagram, Functional Decomposition.
50 or alternatively Event Driven Process Chain, Petri Nets, CIMOSA process model, or UML Sequence / Collaboration diagrams.
ideas, capture requirements and describe designs. Hence, some Architecture Frameworks propose a concrete set of modelling languages to (fully or partially) populate their Modelling Frameworks. The advantage of setting a standard set of modelling languages is that people can be trained in their use, tools can be developed, and in general the communication of the models and descriptions becomes easier.

Today, no single existing modelling language by itself is capable of modelling all necessary aspects of an enterprise. Therefore, in order to completely and meaningfully model an enterprise, one has to use several languages. (Vernadat, '01) has found an overwhelming number of modelling languages used in many different non-interoperable tools, having an inconsistent vocabulary and based on a weak ontology, or no ontology at all. The solution currently in development is the proposed Unified Enterprise Modelling Language (UEML52), which is to be based on a metamodel and ontology and composed of a set of core and additional constructs. In order to succeed, however, such a language must have a large acceptance and appeal to both business users and modelling tool developers. A major benefit of UEML would be a unification of the terminology used in enterprise modelling languages.

One should not expect that there will ever be a complete, closed set of modelling languages suitable for all projects, once and for all. One can, however, expect that there will be a reasonably complete integrated set of constructs that support e.g. three quarters of modelling needs for all change processes, with the remaining quarter being selected as needed by the project.

6 Methodologies

According to Figure 1, GERAM specifies the need for Enterprise Engineering Methodologies (EEMs), which "describe the process of enterprise engineering" (ISO/TC184/SC5/WG1, '00a). GERAM sets several requirements for modelling methodologies, such as:

- the need to cover human role (the need to involve users in the analysis and design phases);
- the necessity to distinguish between user oriented- and technology oriented design;
- the requirement for the methodologies to use project management techniques;
- the need for an economic aspect (e.g. allow cost and performance evaluations and comparisons).

Few methodologies cover the full set of requirements in the GERA sense. Therefore, it is often necessary to adopt a set of methodologies, rather than one single methodology. Overlapping coverage of the selected methodologies may then be used towards the purpose of triangulation.

6.1 Zachman Methodologies

The Zachman framework aims to be generic by not prescribing a particular implementation or modelling methodology. However, some high-level guidance is available. Similar to other frameworks, Zachman identifies53 two main stages in enterprise modelling:

- model the existing enterprise so as to improve existing operational processes;
- change the enterprise using a generalisation of the models developed.

In the first stage, the framework is populated with particular models of the existing enterprise. The second phase employs a formalisation and generalisation of the particular models in a bottom-up approach, in order to effectively obtain partial models54. Various proprietary modelling methodologies relating to the Zachman modelling framework have emerged. Among these are ForeSight, developed by the Zachman modelling framework's authors, the Popkin Process (Popkin Software, '01), the Visible methodology and Ptech's Causal Architecture55.

The Popkin process offers a methodology to identify content for the Zachman framework cells, to relate the cells and to choose the appropriate languages / tools to achieve these goals. The Popkin process methodology also provides high-level guidance for choosing a suitable, out-of-the-box 'established' methodology (Business-, Data-, Structured- or Object oriented modelling).

The Ptech Causal Architecture approach uses Causal Loop Diagrams56 (CLD) models to identify relevant content for the Zachman framework cells. CLDs are used in combination with value- and causal mappings (Vail, '01) to identify key values and high leverage factors for the enterprise. The Causal Architecture methodology then provides a guidance to map these values to the Zachman cells and assign priorities to them.

A complete review of all significant methodologies that can be used in combination with the Zachman framework is beyond the scope of this paper.

51 for aspects of the meaning of enterprise models refer (Bernus, Nemes, and Morris, '96).
52 UEML is a specialised language dedicated to enterprise modelling. UML (as a somewhat general purpose set of modelling languages) may be used to model UEML constructs (Vernadat, '01).
53 on the Zachman Institute for Framework Advancement (ZIFA) web site, at the time of this writing (Zachman, '00b).
54 refer Section 7 for more information on partial (or reference) models.
55 these methodologies are not described here in more detail due to their proprietary nature. More information may be obtained from the authors’ web sites - at the time of this writing www.zifa.com, www.popkin.com, www.visible.com and www.ptech.com.
56 a Systems Engineering approach used to describe cause / effect narratives.
6.2 Conclusion

Enterprise Engineering Methodologies would often prescribe a certain selection and/or succession of life cycle phases (i.e. suggest a typical life history) for a given (or a given type of) change process and, as part of this, a modelling methodology that selects the types of languages and tools in support of the selected life cycle activities.

The modelling methodologies described in Section 6.1 mainly concentrate on the technical tasks of model and/or deliverable development (with little- or no emphasis on e.g. the organisational aspect), which may have the effect of creating market dependency on particular methodologies. In addition, proprietary methodologies are usually difficult to assess due to the lack of publicly available details - which in the long term may also affect their general acceptance.

Companies need to develop in-house capability to exercise enterprise architecture at least to the extent that they should be capable of making informed decisions about the time when external help is needed in the process and about the type of help required.

7 Reference Models

Presently, knowledge management and its preservation appear to be the main drivers of enterprise modelling. A 'good' enterprise model is able to encapsulate (and therefore preserve) knowledge that may otherwise be lost. Knowledge preservation makes most sense in view of its (complete or partial) reuse. In enterprise modelling, reuse may be achieved by means of Reference- or Partial Models\(^{57}\), displaying various levels of specialisation (and hence various degrees of applicability\(^{58}\)) describing solutions for a particular type of modelling problem. They are potential resource savers for the enterprise modeller.

Partial models in the GERA sense may be prototypes, abstract models or models of classes of enterprises which must be specialised and ultimately instantiated in order to obtain the model of a particular enterprise. GERA specifies human role (organisation, responsibilities, etc), process and technology as possible domains for the partial models. Emphasis is being laid on partial models covering technology-oriented IT systems and Integrating Services, which greatly assist in the enterprise engineering effort\(^{59}\). The reader should also be aware at this point that architectures describe structure at a given level of abstraction - e.g. partial models may exist on the policy level (Concept life cycle phase), requirements level, architecture level, etc. – even including the detailed design level (where partial models might describe various views of an often used actual product). Therefore, in a broader context architectures themselves may also be considered partial models. This is in line with the finding that the components of architecture frameworks may play multiple roles, resulting in the distinction between such components often being blurred.

7.1 A Sample Zachman Framework Partial Model

Zachman does not explicitly\(^{60}\) define the notion of reference models. However, as it has been shown in Figure 7, it is possible to define Zachman frameworks at the partial level. A partial level framework may subsequently be either further specialised (in effect producing another, more specific reference model) or instantiated into a particular framework, applicable to the particular modelled entity.

On the other hand, various applications of the Zachman framework to specific industry types have been produced by third parties\(^{61}\). They may be considered partial models at various levels of specialisation (although they are not specifically included in the Zachman architecture framework).

The following example originates from a high-level 'business process-driven OO development' example presented in (Popkin Software, '01). It is the model of a typical small-scale Object-Oriented software development project covering the first phase of a change process in an enterprise, i.e. modelling of the existing processes in order to better understand (and subsequently improve) the business operations.

The example assumes that the need for the new artefact (in this case a software system) has already been identified and the decision has been taken to respond to the need by building it.

The project does not attempt to improve or re-engineer, but rather just document the existing processes and produce a software system to reflect them. It is assumed that the business already has one or more databases containing the information, which will be used by the new application.

As one can see from Figure 10, the Where and When columns have not been modelled since the locations were considered fixed and known, and the temporal aspect's complexity was seen as too low to justify modelling. The Who abstraction, although also known, has been modelled from the application's point of view (in order to produce the user interface specifications and its code). The What abstraction does not include the Contextual and Conceptual perspectives since they are beyond the extent of the proposed modelling task. The 'What' (data) Logical and Physical perspectives

\(^{57}\) 'reference models' and 'partial models' are used interchangeably hereafter.

\(^{58}\) the more specialised a partial model is, the narrower the area of its potential application becomes.

\(^{59}\) this type of partial models are similar for most enterprises and thus may be widely reused

\(^{60}\) in the GERA sense - i.e. an explicit component in the architecture framework.

\(^{61}\) similar to modelling methodologies, most third-party (and Zachman proprietary) partial solutions are not publicly available and therefore not included in this review. The reader is directed towards the web sites of such developers, previously listed in Section 6.1.
have to be 'reverse engineered', based on the existing database system. In Figure 10, the names used for the Zachman perspectives vary widely according to the user's domain. This is reflected in the notations used in this article for the Zachman framework. The equivalence of the perspectives' designations is as follows: Contextual = Objectives, Scope; Conceptual = Business, Enterprise; Logical = Architect, System; Physical = Builder, Technology; Out of Context = Subcontractor, Components.

Detailed Design on the Management and Control side, excluding humans (refer Figure 10).

The GERA Function view is represented in all life cycle phases relevant to this example. The GERA Information view is represented in its Preliminary Design, Detailed Design and Implementation life cycle phases. Zachman framework's People column may map onto the GERA Organisation and Resource views as shown in Figure 4. In this case however, the Organisation view is only represented at the GERA Requirements phase (using a process chart with roles and swimlanes for the matching Zachman cell content).

The GERA Resource view is covered via hardware / people aspects in the GERA Requirements- and Architectural Design phases (called in Zachman Conceptual and Logical respectively), and via Software aspects (e.g. diagrams and code) at the GERA Detailed Design- and Implementation phases (called in Zachman Physical and Out of Context).

7.2 Conclusion about partial models

Reference architectures attempt to model the structure and life cycle of complex artefacts. In their bid to tackle this complexity, architectures often employ a 'divide and conquer' approach, resulting in more than one model (each reflecting a particular view of the artefact modelled). Therefore, GERAM-compliant reference architectures must provide constructs and frameworks expressive enough to construct and hold these models, implying a certain degree of complexity. This complexity makes reference architectures less attractive to business users, since they face the challenge of learning several architectures (each with a framework, modelling methodologies, proprietary languages, etc).

Partial models play an important role in alleviating these difficulties, since they are essentially templates, which may potentially accelerate the learning process (learn by example) and save resources by means of reuse. Therefore (also according to GERA), the more partial models a reference architecture encompasses, the easier to learn and use by the users it becomes. Intuitively, building a partial model is a two way process: on one hand they are produced by specialising generic models, but on the other hand they need to be validated via several good particular

---

62 the existing database (i.e. the logical model and schema) may subsequently be altered as a result of the modelling process.

63 cell content shown in italics is optional.

64 in the original (Popkin Software, '01) example, the Why abstraction was modelled by means of Requirements / Test plans, due to the specific approach taken by the Popkin software tool

---

Figure 10 Zachman partial model example (based on (Popkin Software, '01)) and its mapping onto GERAS Partial Model level

Modelling the 'How' (functional) abstraction is identified as the backbone of the entire modelling effort - therefore all perspectives (i.e. GERA life cycle phases) are covered. As shown in Section 4.1, Figure 4, the Why abstraction belongs to the functional aspect (rule modelling) therefore it is also fully covered. The model presented belongs to a partial level (using a process chart with roles and swimlanes for the matching Zachman cell content). It can be mapped onto the Partial level of GERA, covering the GERA life cycle phases from Concept to
models obtained by instantiating the partial model. In addition, the validation process is almost certain to modify and enrich the partial model.

8 Other Relevant Constructs

This section attempts to cover the rest of the GERAM components and other relevant components of the GERA, such as enterprise entity types and their recursivity. Very often, the concepts described in this section are only present in an implicit form. If the concepts cannot be identified, an attempt is made to determine whether the respective framework is compatible with the concept in question - and therefore could potentially accommodate it.

8.1 Enterprise Entity types

The identification of the target Enterprise Entity Types is an essential activity in enterprise architecture. The concept of entity type provides the means for entity classification and hence allows a more structured approach towards enterprise integration. (Bernus and Nemes, '94) and (Bernus and Nemes, '96) initially defined four types of enterprises entities: Strategic management, Engineering, Manufacturing, and Product. GERA has added a fifth type of entity, the Methodology. For details on entity types refer (ISO/TC184/SC5/WG1, '00a).

8.1.1 Entity Types in Zachman

Zachman does not provide a specialised enterprise entity type concept. That said, (Sowa and Zachman, '92) does identify separate frameworks describing various kinds of entities, such as product, manufacturer, information system or CASE tool manufacturer (refer Figure 11). This denotes that the Zachman architecture framework does recognize entity types in the GERA sense. Also in Zachman (like in GERA), not all of the entity types are necessarily physical entities.

8.2 Recursivity of Entity Types

The recursivity of entity types as defined in (Bernus and Nemes, '94) and (ISO/TC184/SC5/WG1, '00a), acknowledges the existence of relations of a recursive nature between life cycles of various enterprise entity types. It is however emphasized that, according to GERA, only the Operation life cycle phase of an entity may influence other entities' life cycle phases. This section aims to acknowledge attempts of the Zachman framework to define relationships of a recursive nature between (types of) enterprises.

Note that, in the following, recursiveness is understood as repetition and refers by default to an active and direct influence of one entity in the development of another entity. An alternative to this direct influence is the use of one or more deliverables obtained during the creation of an entity into the development of another, by a third, operating entity.

8.2.1 The Recursion Concept in the Zachman framework

Zachman identifies three dimensions to the recursivity:

- relation between frameworks associated to entity types;
- applying the logic of the Zachman framework to the framework itself (applicable, but in the author's opinion relevant only in a first recursion step, mainly due to the rapidly increasing complexity); and
- versioning of the Zachman frameworks. Because of reflecting the AS-IS, TO-BE and possible intermediate states, versioning is also a life history concept. Versioning based on recursion may imply that a framework at a given point in time is expressed in terms of all of the previous versions of that particular framework. This may not always be true (it is often the case that a current version of a particular product is obtained from the most recent version only (i.e. only the first recursion step)).

Out of these three aspects of recursivity, only the first type applies to entity types via their modelling frameworks, and hence matches the scope of this Section. The second and third recursivity aspects are common to all of the reviewed frameworks, even if it is not always explicitly stated.

It has been previously shown that the GERA concept of enterprise type recursion assumes that it is always the Operational phase of an entity life cycle that may influence another entity (ISO/TC184/SC5/WG1, '00a).

---

69 instantiation seen here as making constant all the variables.

A significant exception constitute models containing run-time variables (i.e. meant to be set and re-set during operation). Such models would in essence constitute partial executable models, producing a new particular model at run-time with each new reconfiguration.

70 which is merely called 'enterprise' in (Sowa and Zachman, '92).

71 e.g. GERA defines a methodology entity ('type 5'), which is not an actual enterprise. Similarly, Zachman defines a 'knowledge management' framework.

72 recursivity example: an enterprise of a particular type X may support another enterprise of the same type X - therefore the 'support for enterprise type X' function may be defined in terms of itself (and possibly, an additional invariant used to stop the recursion).

73 and implicitly between the modelling frameworks relating to the enterprises in question

74 an alternative meaning could be e.g. decomposition.

75 NB this type of recursion applies to any architecture framework (including GERA), because if an architecture deliverable needs to be developed the life-cycle of this deliverable should (and can) be described by the framework itself – often considered as a 'sub-project', such as customary in systems engineering;
The recursive relation in GERA explicitly refers to 'defining, creating, developing and building' the influenced entity. The Zachman approach to entity type recursivity is somewhat different. The relation between entities is restricted primarily to a descriptive type. Figure 11 represents this descriptive relation between frameworks - identifying products, enterprises, information systems and CASE tool manufacturers. The descriptive character of the relation is revealed in (Sowa and Zachman, '92), which argues that Row 2 of the manufacturing entity's framework (containing the owner/business view of that entity) contains in fact the complete description of the manufacturing entity's product(s). Hence, the manufacturing entity framework's row number two must in fact be a metamodel of the product framework.

While the concept shown in Figure 11 is theoretically valid, it may not be able to cover some situations such as e.g. dynamic self-reconfiguration of a manufacturing entity at run-time (Row 6). In the GERA sense, Row 2 represents the gathering of user requirements for the manufacturing entity. The manufacturing entity is not yet operating and thus it cannot actually perform any action. In the case of re-engineering, the enterprise is operating in parallel with the gathering of new requirements, however any future products are just being described as part of the requirements gathered.

If other actions, such as develop, build etc are to be considered within the recursive relation, then it is not necessarily the case that a framework should cover the whole of another, but rather only the relevant parts. For example, the manufacturing entity may have its Context and Business Owner's views defined by a management entity and its Architect's and Builder views defined by an engineering implementation entity (refer (ISO/TC184/SC5/WG1, '00a) for a typical example).

It is therefore suggested that a recursive relation involving definition, development, building, etc should originate from Row 6 of the 'parent' (supporting/influencing) framework(s) towards the relevant parts (and not necessarily the whole as shown in Figure 11) of the 'child' (supported/influenced) framework. Row 2 may also be considered in defining recursion, in the sense that a third-party, operating entity may be involved. In this case, the descriptions contained in Row 2 of the first entity are used by the operating phase (Row 6) of a third entity to influence/define relevant phases (Rows) of the second entity.

8.3 Generic Enterprise Modelling Concepts

The Generic Enterprise Modelling Concepts (GEMCs) of a reference architecture and methodology are the glue that holds it together and ensures the consistency and compatibility of its components. (ISO/TC184/SC5/WG1, '00a) sets the requirements that should be met by generic modelling concepts and briefly describes their components: glossaries (an explanatory collection of terms used), metamodels (used to describe the meaning of the modelling constructs of the modelling languages) and ontologies (most formal descriptions of the theories on which the architecture and methodology are based).

8.3.1 Generic Enterprise Modelling Concepts in the Zachman Framework

Zachman's framework was initially aimed at the enterprise's information sub-system. It is based on building and manufacturing industry approaches translated to the information systems, as described in (Zachman, '87). John Zachman found that the manufacturing and building industries in fact asked the same questions as the information systems discipline. He subsequently observed that information systems did not follow the industries' way of organising the answers to the questions by ordered roles (Martin and Robertson, '00). This has led to the idea of using the industry approach in information systems, which has ultimately resulted in a framework grid, organised by roles and questions (or abstractions).

Later on, in (Sowa & Zachman, 1992) the framework has been extended and its scope widened in order to cover the enterprise itself. In addition, the framework

---

76 or manufacturing entities, in GERA terms.
77 where the first cell from the left (Owner's View intersected with the Who column) contains all data descriptions relating to the product(s), second cell (Owner's View intersected with the How column) contains all the business processes necessary to manufacture the product(s), and so on.
78 e.g. a robot, or entire manufacturing cell may change its configuration (physical or control) during operation.
79 metamodels are often expressed as an 'information' (ER / IDEF1x) model or UML class diagram describing the modelling language in question.
has been partially defined (formalised) in terms of conceptual graphs. The underlying metamodel of the Zachman framework is partially described in (Sowa & Zachman, 1992). The representation uses similar notations to the Entity Relationship modelling language, albeit less rigorous (e.g. lack of basic constraints, such as cardinalities). Notwithstanding this, the Zachman framework metamodel (Sowa and Zachman, '92) still describes the structure of several perspectives and abstractions of the Zachman framework. From the metamodel representation (partially shown in Figure 12) it can be seen that the various cell contents are connected along the same row. This means that, similar to other modelling frameworks, the various abstractions (views) referring to the same perspective (life cycle phase) are complementary.

According to GERAM, modules are products that are standard implementations of components, which are likely to be used in the enterprise integration project or by the enterprise itself. Enterprise modules may be configured to form more complex modules, for the use of an individual enterprise. The enterprise architect / modeller may use trusted modules in order to simplify, speed up and even standardise the modelling task at hand.

8.4 Enterprise Modules
Libraries of modules are commonly used in several engineering disciplines in order to build complex systems. Similar to the Object-Oriented encapsulation and information hiding principles, the users only need to know the public variables / interfaces (specifications) of such a module in order to be able to use it. In enterprise integration terms, the business management would be able to plug together trusted process components in order to create a business (Benus and Nemes, '94). Enterprise modules in the GERA sense represent instantiated partial models, where e.g. all build-time variables are fixed (in value or range) and all run-time variables are publicly known. GERA makes a special mention of the Integrating Infrastructures as a particularly important (and therefore essential within an architecture framework) enterprise module.

According to the high-level GERAM metamodel presented in Figure 1, modelling tools which are 'compliant' with a specific architecture (such as those described in Section 9.2) may also be considered enterprise modules, which implement languages and methodologies defined by the architecture framework in question.

8.4.1 Enterprise Modules in the Zachman Framework
To the knowledge of the author, there are no in-house Zachman's Framework enterprise modules published at the time of this writing. As previously stated, one may however consider third-party software packages which support the Zachman framework as being specialised Enterprise Modules.

8.4.2 Conclusion
According to GERAM, modules are products that are standard implementations of components, which are likely to be used in the enterprise integration project or by the enterprise itself. Enterprise modules may be configured to form more complex modules, for the use of an individual enterprise. The enterprise architect / modeller may use trusted modules in order to simplify, speed up and even standardise the modelling task at hand.

9 Enterprise Engineering Tools
As previously shown, enterprise modelling frameworks and methodologies employ various approaches (such as using views) in order to limit the size and complexity of the models produced. Commonly however, these approaches do not alleviate the need to use specialised tools to create / manipulate such models. There are many possible requirements for enterprise engineering tools. They relate to the tools being able to construct, store, analyse and communicate models. Furthermore, enterprises are moving targets and therefore the tools should be able to a) promptly build models that reflect the AS-IS state of the modelled entity (before its AS-IS state changes considerably) and b) update the models as the modelled target evolves. (Vernadat, '96) lists additional requirements, such as model archival and management and model execution capabilities. A 'good' modelling tool should also support the storage and administration of partial models (e.g. in the form of templates and/or libraries). Very few existing tools address most of the above requirements. Therefore, the enterprise modeller must

---

80 for example the inputs / outputs of the functions represented in the How abstraction are entities which may be further described in the What abstraction.

81 there may be however cases where the internal structure of the module has to be known - either for trust building (by visibility) or customisation / optimisation.

82 according to GERA, this type of module may be obtained by specialising and instantiating the integrating services technology partial model.

83 many of the existing tools are not 'aware' of the model they are creating i.e. i.e. they are not based on a metamodel describing the modelling language used (if one is at all
make an informed selection of tools, appropriate for the specific modelling task.

The GERAM metamodel shown in Figure 1 may be used to establish the degree to which a particular tool is compliant with the architecture framework it supports. For example, the tool should support methodologies (i.e. be prepared to manage views and deliverables that the methodology proposes to utilise), implement modelling constructs (language), support generic modelling concepts (i.e. the tool’s repository should be based on an integrated metamodel) and be able to store / customise partial models of the supported architecture framework (which may need additional language constructs).

A typical enterprise engineering task requires a combination of languages. The use of several modelling tools requires model interchange and cross-consistency checking capabilities.

9.1 Generic Modelling Tools

These tools implement one or more of the modelling languages usable for various architectures, such as Entity Relationship, the IDEF family of languages, UML, Graphs, etc (refer Section 5 for usable modelling languages). Better tools from this category have mechanisms for checking/enforcing the syntax of the language, library maintenance (e.g. for storing partial models), semantics and meta-modelling84 capabilities.

The selection of tools suitable for a modelling task essentially depends on the intended life cycle phase coverage of the architectural products. Usually, modelling tools feature one or more collections of constructs, which do not necessarily form a proper language. Therefore one should identify the most likely necessary constructs / language and based on those needs then select a preliminary set of tools.

For example, in the Identification phase one may use hand- or computer drawn graphical symbols85, while in the Implementation phase use of formal languages may be desirable, e.g. if the end products are executable models. The use of a word processor, together with a well-defined and consistent glossary and optionally a partial model (e.g. standard / policies manual) may be enough to 'implement' instructions for a human.

In conclusion, a large number of modelling tools support a set of modelling languages in preference to a specific architecture framework. These tools are usable whenever it is possible to use one or more languages they support.

A complete review or list of this type of tools, available or emerging, is beyond the scope of this paper. Some such tools are: Knowledge Based Systems' (KBSI) suite of IDEF-based tools (AI0Win, SmartER, ProSim, ProCost, ProABC, etc), Meta Software's Design/IDEF and Design/CPN (for Colored Petri Nets), Computer Associates’ ERWin / BPWin tools (for IDEF0/IDEF1x/IDEF3) and Rational Corporation's Rational Rose UML-based modelling tool. Modelling tools supporting particular architecture frameworks are described in Section 9.2.

9.2 Zachman 'Compliant' Tools

These tools provide limited support for the Zachman methodology and languages. Third-party developers have either built their Zachman-compliant tools around the Zachman framework, or offer plug-ins for it. Two such developers have been reviewed: Popkin Software and Ptech, Inc.

9.2.1 Popkin Software's System Architect

System Architect is developed around a shared 'corporate' repository. Various modelling techniques may be used with this repository, enabling model sharing across projects (potentially using different methodologies).

As can be seen from Figure 13, System Architect's repository is customisable, i.e. the user may alter the predefined meta-data, including integrity constraints.86 This is a powerful, but hazardous feature (altering the metamodel essentially changes the meaning of the modelling language, with expressiveness and complexity consequences).

System Architect supports most mainstream structured- and object-oriented modelling methods (as shown in Figure 13), by using an Enterprise Architecture Framework (modelled after the Zachman

---

84 whereby the user may alter the structure of the existing modelling language used by the tool (effectively creating a new modelling languages).

85 for example Rich Pictures. Note use of predefined symbols may be good in improving the communication but may also stifle creativity in the early Identification phase, especially when graphical editors with predefined, fixed and limited graphical libraries are employed.

86 e.g., the user may change the definition of what is a 'legal' model.
System Architect provides the capability to view relationships between several Zachman framework cell contents. A matrix editor allows cross-referencing and correlating the design artefacts across several models. Add-ons (modules) are provided for business analysis, such as Activity-Based Costing (ABC) or model simulation (based on process charts and IDEF3).

Figure 9. Therefore, attention must be also paid to the proper modelling of the decisional, human and social aspects of the enterprise.

### 9.2.2 Ptech's FrameWork

Ptech, Inc has developed a methodology-independent integrated modelling environment called FrameWork. It is based on an object-oriented data structure which has its semantics fully defined via metamodels. Being methodology-neutral, FrameWork is claimed to have wide applicability (e.g. be able to capture, analyse and design data and link it to organisations, activities, locations, etc.). The downside to its genericity is that it has to be specialised for particular modelling tasks via specific extensions called 'accelerators' (John, '01) 87.

FrameWork offers full support for the Zachman framework (via the Zachman framework Control Panel, containing a template and most of the model types required). Ptech also offers a modelling methodology associated with their tool, called Causal Architecture.

FrameWork is based on a shared knowledge base designed as a semantic network, which is also a rule-based inference tool. This design enforces consistency of existing and newly added objects.

Owing to this structure (refer Figure 14), FrameWork may represent the same data in a Zachman framework-based view (based on Zachman perspectives and abstractions) or e.g. from a C4ISR (C4ISR Architectures Working Group, '97) viewpoint 88.

### 9.3 Conclusion

Enterprise modelling tools provide capabilities that can be employed to create enterprise models, which in turn may be used to implement Enterprise Operational Systems. Thus, modelling tools (or plug-in modules for them) are in fact a special type of enterprise module, as defined in GERAM. Similar to the modelling languages, there is no 'complete' enterprise modelling tool suitable for modelling all aspects of an enterprise.

At the time of this writing, some common trends in modelling tool development are web-enabling, the use of a central repository and modularity (plug-in modules). Most of the tools strive to provide metamodelling - an advanced feature, which may however lead to an undesirable modelling language proliferation. Plug-in modules may be provided for further processing (e.g. simulation) of the models produced, for modelling additional aspects of the enterprise, or for interfacing with other established (e.g. Resource Planning, etc) tools.

The answer to an 'all rounder' modelling tool may be a meta-tool, describing the rules of how to combine the available modelling tools for a type of modelling task. 89 Such a tool would sit on top of existing modelling tools (rather than coercing these tools to comply with it) and implement constructs contained in a meta-language 90.

A comprehensive review of commercial third party and proprietary modelling tools is beyond the scope of this article. Enterprise modelling tools reflect the rapid

---

87 this is resolved by use of plug-in modules for specific architectures

88 the validity of the mapping thus obtained depends essentially on the underlying metamodels. Therefore, the user is advised to test such mappings prior to employing this tool for e.g. conversion of models.

89 such a meta-tool may e.g. be an expert system that designs a modelling tool / language selection solution on the fly for a particular modelling task.

90 although not explicitly defined as a meta-language as such at the time of this writing, this may possibly be the UEML. UEML attempts to provide an unifying umbrella (based on clear ontologies and metamodel) over existing languages rather than enforcing yet another modelling language (Vernadat, '01).

---

![Figure 14 Ptech's FrameWork structure (based on (John, '01))](attachment:image.png)
evolution of software and hardware platforms and the increasingly shorter software / hardware product update cycles.

10 Final Conclusions

This analysis has shown that the difference between the architecture framework's components is often blurred. Just consider the fact that an architecture may also be considered a reference model or that a modelling framework populated with content (models) is potentially a language in itself.

The views contained in GERA (such as Functional, Information, etc) appear to display various degrees of dependence on one another and may (implicitly or explicitly) contain other aspects. For example, the Organisation view in GERA is understood to represent 'who does what', i.e. the mapping of the Resources to the Functions 91. Similarly, the Zachman metamodel reveals that the cells within the same perspective (row) of the Zachman framework may be interconnected.

The concept of versioning is covered in Zachman as a form of life cycle. It is argued that recursion between frameworks associated to entity types as defined in Zachman should be limited to the operational phase of the modelled entities unless a third entity is involved. Proprietary methodologies (such as those based on the Zachman framework) may provide commercial advantages, but their closed nature hinders advancing the cause of enterprise modelling as such and inciting public interest for the reference architecture or architectural deliverables they support. A solution may be a mixture of publicly available white papers (laying the foundations (e.g. metamodels, ontologies) for the methodologies and proprietary detailed methodologies for commercial use.

The Zachman framework, like many other architecture frameworks, came into existence in an initially partial form and has (subsequently) evolved into a more complete framework. A generic enterprise reference architecture such as Zachman may assist the evolution of life cycle architectures by identifying potential gaps and thus helping define their desired development areas.

Historically, the architecture frameworks involved in the development of GERAM have benefited from their participation by achieving a better understanding of their own structure and their contribution to the overall enterprise integration endeavour. The dialog and exchange of ideas within the IFIP-IFAC Task Force on Architectures has promoted a synergy towards advancing the cause of enterprise architecture. A continuation of this effort (involving all major architecture frameworks) is necessary in order to ensure that GERAM stays up-to-date as the enterprise integration domain evolves.

11 References


---

91 This perception of the Organisation view holds true for both human and machine Resources - therefore implying that Organisation may refer to both human and non-human (e.g. software agent) aspects. Moreover, the Decisional aspect may be considered a specialisation of the Functional view.

92 (IFIP-IFAC Task Force, ’93)
Schmidt (Eds.), Handbook of Enterprise Architecture (pp. 40-82). Heidelberg: Springer Verlag.


