

An architecture for defining the processes of the software and systems life cycles

*Terence P. Rout and Peter Bernus
Software Quality Institute and
School of Computing and Information Technology
Griffith University
Queensland, Australia*

Terence P. Rout
Software Quality Institute
Griffith University
Queensland 4111 Australia
Phone: +61 7 3875 5046
Fax: +61 7 3875 5051
Email: T.Rout@sqi.gu.edu.au

Peter Bernus
School of Computing and Information Technology
Griffith University
Queensland 4111 Australia
Phone: +61 7 3875 5039
Fax: +61 7 3875 5051
Email: P.Bernus@cit.gu.edu.au

Abstract

Attempts to define consistent reference models for systems and software life cycle processes have been hindered by the lack of an acceptable underlying architecture for such models. The use of existing enterprise architecture models is put forward as an appropriate solution to this problem. By assigning processes to identifiable entities in the enterprise model, problems arising from the lack of definition of scope and of the relationship between different processes can be resolved. The use of appropriate enterprise models as the basis for process reference models is proposed as a resolution of the problems associated with the development and application of process reference models for the software engineering domain.

1. Introduction

Over the past five years there have been several attempts to define a set of standard definitions for the processes of the software and systems life cycle. These attempts have been hindered by lack of consensus over the structure and content of an appropriate reference model, but substantial progress has been made. This paper addresses one of the key remaining unresolved issues in this task.

1.1. Background

The first comprehensive set of definitions of software life cycle processes was contained within ISO 12207 - Software Life Cycle Processes [1]. During the development of ISO 12207, the SPICE Project developed a set of definitions in the form of a Baseline Practices Guide [2] having a different (though similar) architecture to that of ISO 12207; over the next few years, this model was modified and brought into strong alignment with ISO 12207, and finally approved as ISO/IEC TR 15504-2, a "Reference model for processes and process capability" [3].

The model originally defined in ISO 12207, while promising, had a number of deficiencies which detracted from its wide application. Principal among these was the fact that the standard was developed from the perspective of process implementation, rather than definition. The process descriptions in 12207 are based around prescriptions of tasks and activities that need to be performed in order to implement the process, and the intent of the process is described only in terms of the performance of these activities. The Baseline Practices Guide has a similar problem, though it included statements of basic purpose of each process in the model. In ISO/IEC 15504, there was the emergence of a consensus - at least within the standards community - as to how the standard descriptions of processes should be formulated: processes are defined by describing the purpose and

the outcomes of successful implementation. There is also general agreement on the nature of these descriptions; for example, ideally, each outcome would have the following characteristics:

- capability-neutral (“capability” is used here in the sense of 15504)
- role-independent
- succinct
- not a restatement or functional decomposition of the purpose
- phrased in terms of a continuing responsibility to:
 - (1) produce and maintain an artefact;
 - (2) achieve and maintain a state; or
 - (3) meet a constraint.

This approach leads to generally acceptable descriptions of processes of the software life cycle, and early experience indicates these are also applicable to other domains, including that of systems engineering. The development of ISO/IEC 15288 - Systems life cycle processes [4,5], however, has highlighted the lack of consensus on another critical aspect of this debate - the need for a common architecture for the process models. The lack of a common understanding of this issue is so pervading that a current proposal for resolution avoids the issue totally, putting forward a view that would result in a “repository” of process descriptions rather than a true reference model.

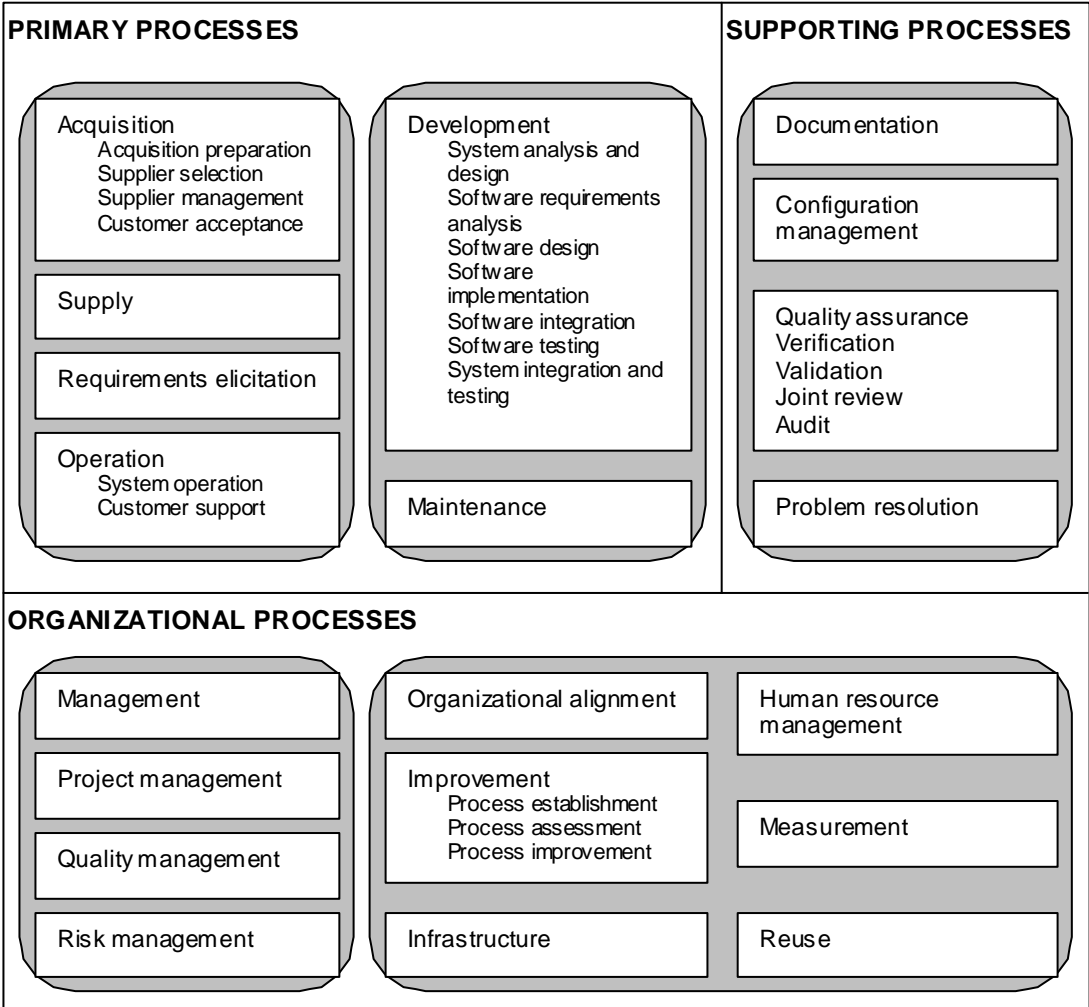


Figure 1 - Processes defined in ISO/IEC 15504-2

2. Enterprise Models and Life Cycle Processes

A possible solution to the conflict is to adopt an architecture based upon an existing framework for modelling that has shown its value and relevance in spheres beyond that of systems and software engineering. One strong candidate for a suitable framework is in the field of enterprise modelling, where a generic architecture for modelling enterprises and related entities exists and has been extensively evaluated.

2.1. GERAM - an overview

GERAM (Generalised Enterprise Reference Architecture and Methodology) [5] defines a tool-kit of concepts for designing and maintaining enterprises and their products for their entire life-history. GERAM is meant to organise existing enterprise integration knowledge. The GERAM framework unifies two distinct approaches of enterprise integration, those based on product models and those based on business process design.

The framework has the potential for application to all types of enterprise entities meaning all entities the life of which is worthy of consideration in connection with the enterprise. Several applications exist in the industry to date. The framework organises methods, models and tools needed to build and maintain the integrated enterprise [6] and its products. GERAM is the basis of the ISO 15704 - Requirements for enterprise-reference architectures and methodologies[7].

The GERAM framework identifies, in its most important component GERA (Generalised Enterprise Reference Architecture), the basic concepts to be used in enterprise engineering and integration (for example, enterprise entities, life-cycles and life histories of enterprise entities). GERAM distinguishes between the methodologies for enterprise engineering (EEMs) and the modelling languages (EMLs) which are used by the methodologies to describe and model, the structure, content and behaviour of the enterprise entities in question. These languages will enable the modelling of the human part in the enterprise operation as well as the parts of business processes and their supporting technologies. The modelling process produces enterprise models (EMs) which represent all or part of the enterprise operations, including its manufacturing or service tasks, its organisation and management, and its control and information systems. These models can be used to guide the implementation of the operational system of the enterprise (EOSs) as well as to improve the ability of the enterprise to evaluate operational or organisational alternatives (for example, by simulation), and thereby enhance its current and future performance.

The methodology and the languages used for enterprise modelling are supported by enterprise engineering tools (EETs). The semantics of the modelling languages may be defined by ontologies, meta models and glossaries which are collectively called generic enterprise modelling concepts (GEMCs). The modelling process is enhanced by using partial models (PEMs) which are reusable models of human roles, processes and technologies.

The operational use of enterprise models is supported by specific modules (EMOs) which provide prefabricated products like human skill profiles for specific professions, common business procedures (e.g. banking and tax rules) or IT infrastructure services, or any other product which can be used as a component in the implementation of the operational system (EOSs).

A key component of GERAM is GERA, the Generic Enterprise Reference Architecture, which defines the generic concepts recommended for use in enterprise engineering and integration projects. These concepts can be classified as human oriented; process oriented; or technology oriented concepts.

The process-oriented concepts defined in GERA are: *enterprise entity life-cycle* and *life-cycle phases*; *life history*; *enterprise entity types*; and *enterprise modelling with integrated model representation and model views*. Life-cycle activities encompass all activities from inception to decommissioning (or end of life) of the enterprise or entity. The different life-cycle phases define types of activities which are pertinent during the life of the entity. As one enterprise entity operates, it implements life-cycle functions of another entity - eg. as an engineering project operates, it may support the design and implementation activities of the product life-cycle. The different classes of entity recognised in GERA include:

Repetitive Service- and Manufacturing Enterprise Entity which are enterprises supporting one or more types or a families of products, produced in a repetitive or sustained mode. Examples are service enterprises, manufacturing plants, engineering firms, infrastructure enterprises, etc.

The products of the repetitive enterprise may be diverse, ordinary products or products which are enterprises themselves, e.g. a plant, or a project.

Project Enterprise Entity (often with a short life history) created for the one-off production of another enterprise entity. (E.g. one of a kind manufacturing projects, engineering projects, etc.)

The project enterprise are normally created by repetitive service and manufacturing enterprises. and are closely linked with the life-cycle of a single product or service. The products of project enterprises may be diverse, such as large equipment, buildings, systems etc., or an enterprise in its own right (e.g. a plant, or an infrastructure enterprise).

Product Entity - a very large class of entities including any artificial product, such as customer goods, services, hardware equipment, computer software, etc. These entities are not enterprises themselves, but their life-cycles are described by GERAM.

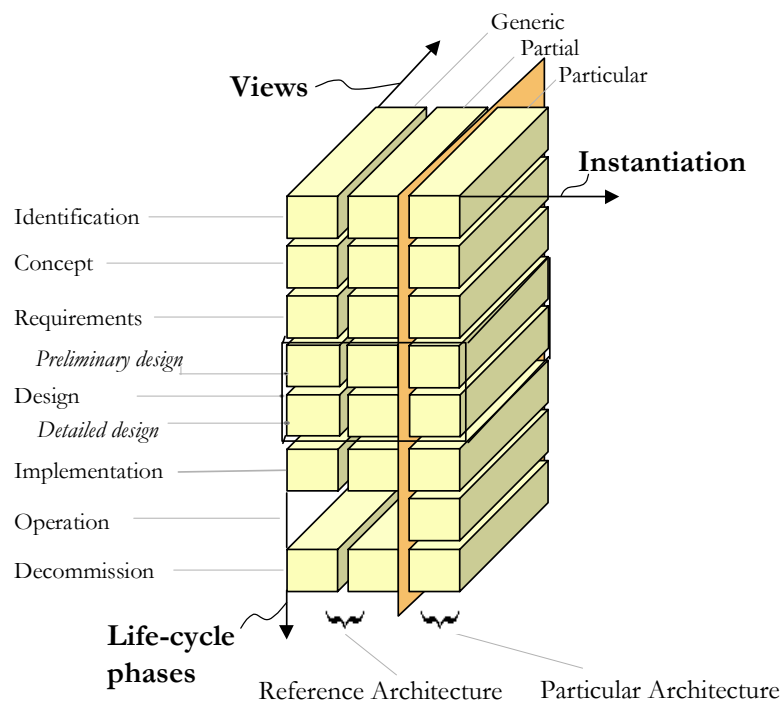


Figure 3 - GERA Modelling Framework

GERA provides an analysis and modelling framework which is based on the life-cycle concept and identifies three dimensions for defining the scope and content of enterprise modelling.

- *Life-Cycle Dimension*: providing for the controlled modelling process of enterprise entities according to the life-cycle activities.
- *Genericity Dimension*: providing for the controlled particularisation (instantiation) process from generic and partial to particular.
- *View Dimension*: providing for the controlled visualisation of specific views of the enterprise entity.

Figure 3 shows the three dimensional structure identified above which represents this modelling framework.

Within this framework, process modelling is the activity that results in various models of the management & control as well as the service and production *processes*, and their relationships to the resources, organisation, products etc. of the enterprise. Process modelling allows to represent the operation of enterprise entities and entity types in all their aspects: functional, behaviour, information, resources and organisation. This provides for operational use of the models for decision support by evaluating operational alternatives and for model driven operation control and monitoring. It is this feature of GERAM that establishes its suitability for supporting the process reference models for the software and systems life cycles.

2.2. An enterprise-based process model

The nature of the enterprise-related entities associated with an undertaking - in this instance, a software engineering project - are shown in Figure 4. The architecture of the processes associated with an undertaking of this type is determined by associating the processes each to their relevant entity. Thus, the basic processes of the software product life cycle - from Software Requirements Analysis through to Software Operation - are associated with the product entity. The processes classed in ISO 12207 and ISO 15504 as “supporting” are associated with the Project entity, while other, organization-wide processes can be associated with the “Company” entity.

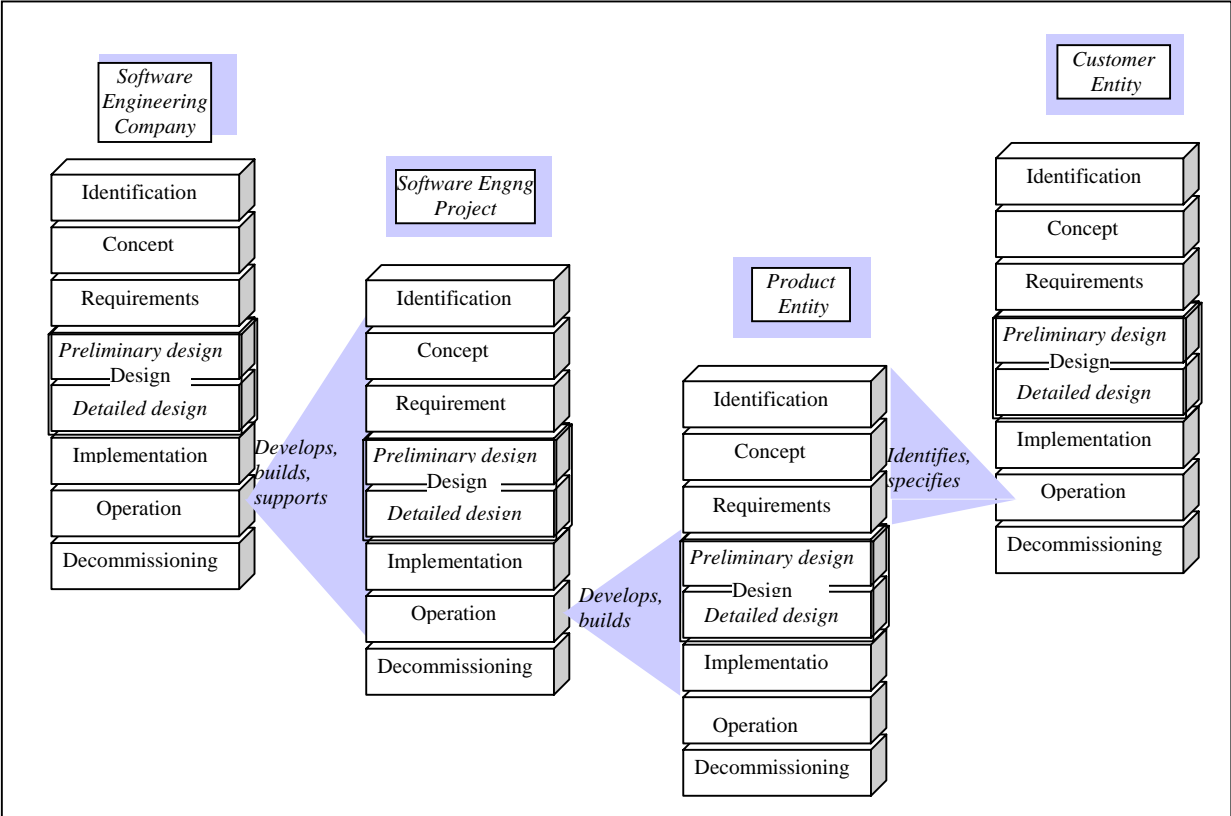


Figure 4 - Enterprise-based model for the software life cycle

The example in Figure 4 shows a relatively small sample of what would, if fully described, be a very complex architecture. Some of the strengths of this approach can be seen if the issues of acquisition and supply of the “product” entity are considered; Figure 5 displays some of the entities that would be associated with these functions, and show a possible classification of life cycle processes consistent with the architecture. In this expanded model, enterprise entities are identified for the Supplier; the Acquirer; and the Developer. Separate entities can be identified for the various projects - the Acquisition Project, Supply Project, and Development Project - with interactions between the various primary entities.

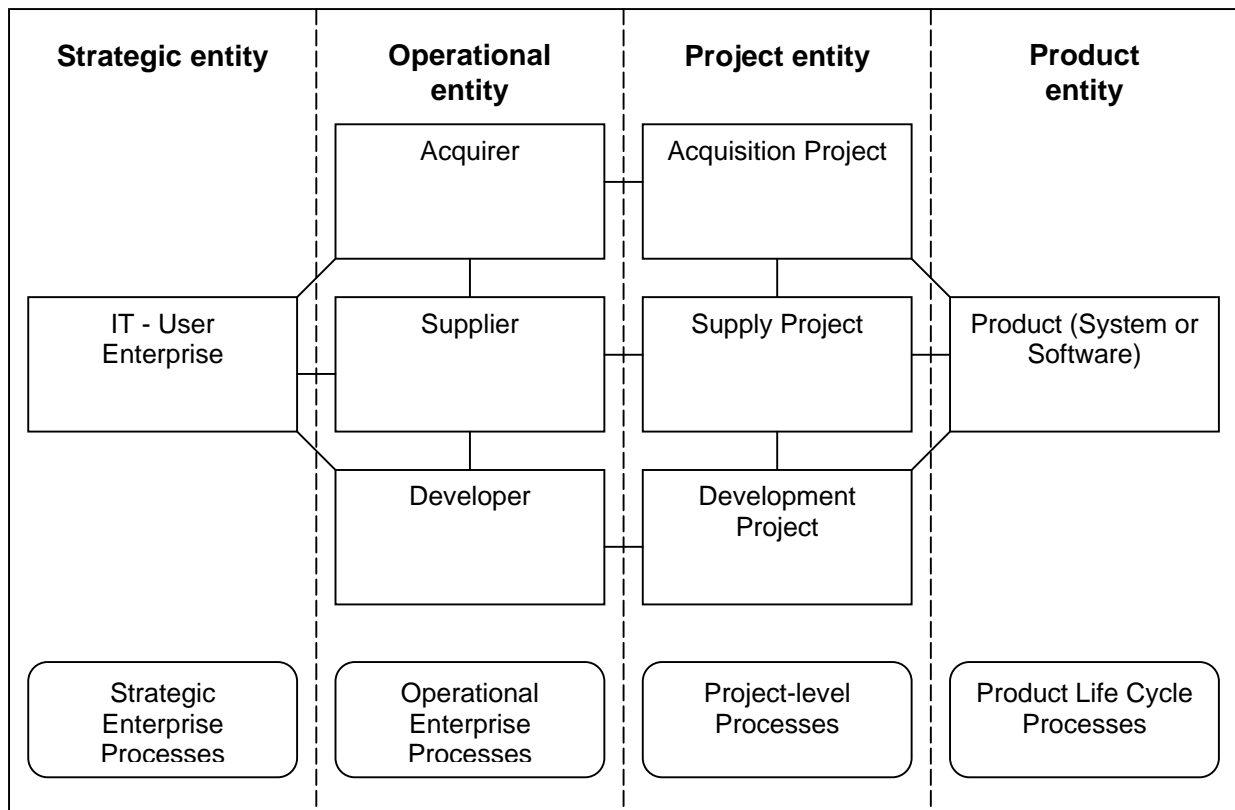


Figure 5 - Enterprise Entities and Life Cycle Processes

Each of the Project Entities is associated with the Product in different ways. The model shows how a common set of processes can be associated with all of these various entities, enabling a common understanding of the operations required to achieve the joint undertaking to be shared between all of the parties involved.

We are not attempting in this paper to describe a complete architecture for the set of life cycle processes; our intent is to demonstrate the value of an architecture based upon existing enterprise architecture models, and to perhaps lay the foundations for more detailed work to fully enunciate such a model.

3. Benefits of the proposed architecture

In examining the benefits of adopting a process model such as outlined above, it is necessary to first explore the reasons for the development and adoption of reference models for the life cycle processes. The primary justification for such a model is that of establishing a common vocabulary and understanding of the processes associated with the product life cycle. The existence and widespread acceptance of a set of standard descriptions of processes associated with a specified domain - be it software engineering, systems engineering, or (for that matter) conventional manufacturing is that it provides all of the stakeholders in an undertaking with a common vocabulary for expressing their understanding of the processes to be employed and controlled to achieve the goals of the undertaking.

The use of an established architecture as a framework for the process descriptions provides several benefits. In developing the reference model itself, the architecture provides structure and helps to “anchor” the definitions against accepted entities. The architecture helps also to provide a scope within which the process is intended to function; the lack of an evident scope for existing definitions can cause difficulties in interpreting the models.

In the application and use of the reference model, the existence of a well-structured architecture also has demonstrable benefits. It permits, for example, the specification of an appropriate set of implementable processes for a virtual enterprise - a relevant context for systems engineering. It also

provides a bridge between the modelling concepts and techniques applied in integrated manufacturing approaches, and the sophisticated (and primarily personnel-based) life cycle concepts used in software engineering.

4. Summary and conclusions

There are inconsistencies and unresolved problems in the existing and proposed reference models for systems and software life cycle processes. In order to provide a firm foundation for integrated approaches to systems and software engineering, it is important that these differences are resolved. The problems reflect in large part the absence of a firm underlying architecture for the process models.

We propose that the existing and well defined architectures of enterprise modelling can provide the basis for a suitable architecture. In our proposal, processes are related by their association with established entities in enterprise models - product, project or enterprise entities. An examination of some examples of appropriate indicates that the use of such an architecture can resolve at least some of the existing problems and inconsistencies.

It is concluded that the lack of a sound underlying architecture is the basic cause of the problems and inconsistencies in existing process reference models for the software and system life cycles. The use of enterprise modelling principles to establish suitable architectures is proposed as a resolution to the issue.

5. References

1. ISO/IEC 12207: 1995, Information Technology – Life Cycle Processes
2. SPICE Project (1995), Software Process Assessment – Part 2: A model for process management, Version 1.00.
3. ISO/IEC TR 15504: 1998, Information Technology – Software Process Assessment - Part 2: A Reference Model for Processes and Process Capability
4. J.G. Lake (1997), Report on Development of ISO Standard 15288 - System Life Cycle Processes, INCOSE '97, Los Angeles.
5. ISO/IEC JTC1/SC7 WG7 (1998), Life-Cycle Management — System Life Cycle Processes, Working Draft 3.
6. IFIP / IFAC Task Force (1997), Generalised Enterprise Reference Architecture and Methodology Version 1.5.
7. P. Bernus, and L. Nemes, A Framework to Define a Generic Enterprise Reference Architecture and Methodology, Proceedings of the International Conference on Automation, Robotics and Computer Vision (ICARCV'94), Singapore, November 10–12, (1994). also in *Computer Integrated Manufacturing Systems* 9,3 (July 1996) pp 179-191.
8. ISO 15704: 1998, Requirements for enterprise reference architectures and methodologies.