# Steps in the Ontological Modelling of the Software Verification and Validation Activities

**Javier Dolado**
University of the Basque Country,
Spain
dolado@si.ehu.es

**Javier Tuya**
University of Oviedo,
Spain
tuya@lsi.uniovi.es

**Daniel Rodríguez**
University of Reading,
UK
drg@ieee.org

## Abstract

This paper describes the initial stages of building an ontology of the activities of software Verification and Validation (V&V) as key elements in the management of a software project. There is a need for information in the V&V phases, as many decisions have to be taken with information and data which varies from project to project. The current work has its roots in the issues related to the integration of information in software project management. Among the tasks, data and decisions that the project manager has to deal with, those related to V&V of the software system are critical for a successful outcome. The goal in this work is to model the data, concepts, terms and relations used in the phases, processes and activities of software V&V from the project manager point of view. To do so, we are looking into ontologies in order to model such concepts and as a way to integrate the different sources of information.

## 1 Introduction

Software verification and validation (V&V) activities are key elements in the management of a software project. There is a need for information in the V&V phases, as many decisions have to be taken with information and data which varies from project to project. The current work has its roots in the issues related to the integration of information in software project management [22]. Among the tasks, data and decisions that the project manager has to deal with, those related to V&V of the software system are critical for a successful outcome, i.e., complete a project in time and within the budget. The criteria for acceptance of the testing and validation phases and products include many sources and types of information. Moreover, project managers have to deal with information at different level of interest. For example, the needs for metrics in testing are different in case of vendor-developed software [3] or if an automated framework is used [18].

The goal in this work is to model the data, concepts, terms and relations used in the phases, processes and activities of the software V&V phase. It is assumed that the "prevalent practices in the industry are still immature…" and that "tools are not ready for large-scale commercial use" [8].

The need for a framework in which to integrate different measurements has already been recognized in [14, 16]. However, the proposals are too general and do not intend to model the domains of software management, i.e., those approaches only try to define the measures correctly. Other researches [26] have also highlighted the importance of developing measures without clarifying concepts and definitions of concepts and measures. Other researches are also looking into other measurement fields and standards to clarify concepts and improve the way metrics are collected [1]

The use of *ontologies* [24] has been a recourse used in other fields in order to *integrate* the information, to *communicate* what people has achieved, to *adapt* the goals of the organization and to *support* the efficiency of the processes. The creation of ontologies is not straight forward; there are no standard modelling methodologies but a mix of guidelines that are combined with techniques from the database modelling and object oriented modelling. Figure 1 shows a high level view of the process steps to be considered when developing ontologies.

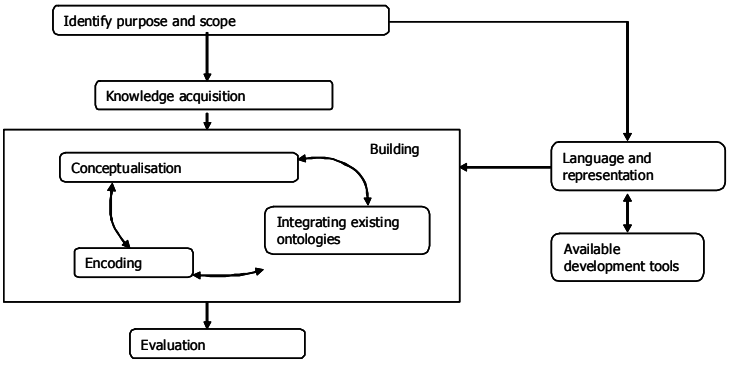


**Figure 1 Ontology Building Process**

## 2 Ontologies in the Context of Software Project Management

Ontology, in the philosophical sense, deals with models of the reality and with the nature of the real world. In the field of information systems, an ontology is an explicit representation of domain concepts and provide the basic structure of the system. The ontology "defines the vocabulary of a problem domain and a set of constraints on how terms can be combined to model the domain" [7]. The common uses of ontologies include communication between people and organizations and interoperability between systems, i.e., translation of modelling methods, paradigms, languages and software tools [23]. In the software engineering arena, it means that ontologies help achieving some desirable qualities such as reusability providing formal representations, searchability providing meta-data as an index into information, reliability performing consistency checking, etc. Therefore, ontologies allow us to add semantics to data so that different software components can share information in a homogeneous way.

Usually it is impossible to achieve an exhaustive ontology in a domain. As a consequence, the general approach is to define fit-for-purpose ontology, i.e. a good enough shared vocabulary. One of the main issues in defining an ontology is therefore the selection of the terms. They have to be defined after a careful examination of the field so that, a small set of central concepts is identified. Moreover, we cannot consider the data and concepts of V&V in isolation, as they have an intrinsic connection with the rest of software development and management. For example, ontologies allow us to link measubrement programs with software development processes, including V&V activities.

From the perspective of reusing concepts we take of the concepts already developed in the Enterprise Ontology and in the set of TOVE ontologies. Other ontologies have also been created in the scope of software engineering such as REFSENO [20] (Representation Formalisms for Software Engineering Ontologies) that has been applied for modelling experience factories [2] using the Goal-Question-Metric paradigm.

For the basic ontological framework, we resort to the works of Wand and Weber [25], which are rooted in the concepts of the philosopher M. Bunge. This framework has been made operational in different ways and it may integrate the rest of the ontological constructs referenced here.

From a more technical point of view, a standard way of defining ontologies is through the Resource Description Framework (RDF) [15] defined by the World Wide Web Consortium (W3C). Although the RDF was originally developed for the so-called Semantic Web [4], it allows us to integrate a variety of applications using XML as an interchange syntax without semantics. The RDF specifications provide an ontology system to support the exchange of knowledge but only modelling data with some support for semantics. On top of RDF, RDF Schema allows us to model concepts, adding more semantics and basic inference. Currently, further extensions of RDF Schema such as DAML+OIL [9] are being developed to support features missing from RDF (e.g. frame-based systems, description logics) providing a more expressive ontology language with more complex inferences capabilities.

## 3 Ontological Concepts in Software V&V

The sources for information of the concepts used in software V&V are diverse. Although there is no accepted ontology, there are some standards such as [10] and the guide to the software engineering body of knowledge [5] that can provided a foundation. Moreover, there are basic concepts which are presented in references such as [8, 17].

The activities under the term V&V may have varied forms, and the features may be different according to the uses in the organization. The basic concepts are [8]:

*debugging*: a process for locating and fixing code,

*verification*: a process for verifying that the code implements the specifications,

*testing*: the varied forms of testing try to find cases where a program or system does not meet its specifications,

*validation*: evaluating software to ensure compliance with requirements.

Also, these activities may be ordered or organized in a specific software development process. For the *validation tasks*, a reference to the requirements established should be made. The concept of *inspection* [17] is one of the ways of articulating the processes of verification.

A term that describes at a higher level all what is carried out under V&V is COMPLIANCE or CONFORMANCE, in the sense that the goal of each activity takes "some product" as a referent.

In software engineering it is usual to relate RESOURCES, PROCESSES and PRODUCTS. A product is developed by a process that uses a set of resources. These three categories have been also been modelled as ontologies in other works. The PRODUCTS involved in V&V are, in the broad sense: unit, component, system and solution. The RESOURCES can be broadly categorized in: a) *human* and b) *technological*. The PROCESSES in V&V are defined in the context of other software processes.

In a more lower level, it is necessary to define a wide range of terms and metrics that project manager will make use of such as *error*, *fault*, *failure* which are often used in an interchangeable way but meaning different things. Moreover, as stated by Kitchenham et al. [14], it is necessary to define every label or point for nominal and restricted ordinal scales, e.g., severity as catastrophic, serious or trivial.

Therefore, we also need to model the processes, including the testing processes. By a software process we will mean any software activity associated with the development and maintenance of software: from requirement analysis through to maintenance. Following Satpathy *et al.* Typed Generic Process Model (TGPM) [19], a product is an entity, which a process (e.g. any software activity) produces as output. Products may also be fed to processes as inputs. For example, product *rd* belongs to the set of *requirement documents*, etc. TGPM defines a process as a relation from a set of products to another set of products. A process may have many type definitions. For example, if an organization develops formal specifications (FS) from requirements, then the type of the formal specification process (FS process) is: RD?FS. A process may be

composed of subprocesses. When considered from a process point of view, the inputs and outputs of subprocesses are called intermediate products. A subprocess which is not decomposed further is called an atomic process. For example, Figure 2 shows how the testing process takes an implementation and a set of test cases as input and produces a tested implementation and Test results as output. The testing process consists of two subprocesses: unit testing and integration testing. Unit testing consists of two atomic processes: black box testing and white box testing. Integration testing is an atomic process. After the unit testing, the program modules are integrated by the integration process. Since the process of integration is not a part of the testing process, it has been shown outside the scope of the testing process. The integrated modules and the test cases are then fed to the subprocess integration testing which produces a tested implementation along with test results.

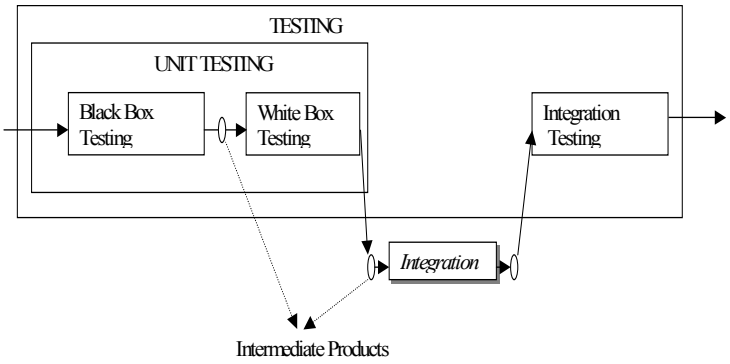


**Figure 2 Example of Testing Process**

Languages that allow such processes descriptions in an ontological context are described below in section 3.3.

### 3.1 Reusing ontologies

We want to take advantage of the works already published. The *Enterprise Ontology* [24] provides a first-order logic representation, and it provides the concepts of objects, relations and functions. The levels which are of interest here are those of: a) *activities and processes*: activity, resource, plan, and capability; b) *strategy*: purpose, strategy, help to achieve, and assumption.

The terms ACTIVITY and STRATEGY play an essential role in V&V activities. The ACTIVITY captures "the notion of anything that involves actual doing, in particular including action". It also includes important terms such as activity specification, resources and other terms of immediate application to V&V. The STRATEGY is defined as "a plan to achieve a strategic purpose". The two concepts have to be extended for V&V.

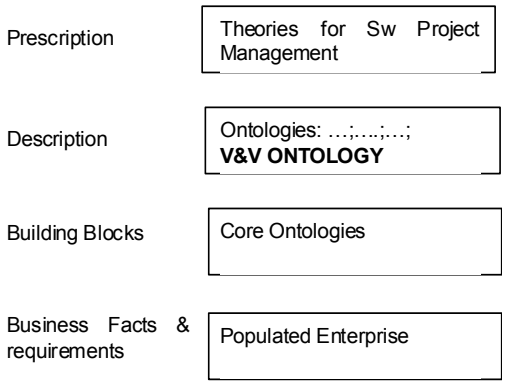


**Figure 3 V&V Ontologies Definition**

The set of TOVE ontologies for Quality Modelling provide representations for the following:

- Core Ontologies
- Measurement
- Traceability
- Quality Management System
- Activity-Process Mapping Ontology.

The *core ontologies* are based, in the works of [12], upon a first-order language (situation calculus); however, we think we can resort to Wand and Weber [25] for a more fundamental approach.

For the *measurement ontology* we can also reuse the results of Kitchenham *et al.* [14] and those of Paul et al. [16].

The *traceability ontology* represents a knowledge of ancestry. For example activities are comprised of sub-activities. This ontology has received special attention in [12, 13].

The *quality management system* is a separate set of concepts. The *activity-process* mapping relates the terms of activities and processes.

We are defining the V&V ontology as is depicted in Figure 3, where the V&V activities form part of a set of descriptions, which in turn are based in a theory describing software projects. The building blocks are concepts which form the basis for the rest of ontologies.

### 3.2 A Cost Ontology

Related to any activity of the software life cycle, it is important to control its costs. A software project does not differ too much from other collaborative activities that we can find in the business world. So, we can reuse entities in the Enterprise Ontology, including cost concepts taken from the activity-based costing (ABC)

ontology form the TOVE Project [21]. These concepts are expanded including new ones that are essential for software projects and V&V activities.

The ontology related to costs, and used for ABC [21], defines a set of possible states related to the state of the resources that the activity uses or consumes; in our case, the state values must be also useful for the planning and estimation process and the approach must be different. An ACTIVITY STATE can be one of the following:

- Planned: Activity is defined, but not having detailed scheduling information (dates and resources)
- Scheduled: Activity has a defined start and end dates and resources allocated to it
- Committed: PRE-CONDITION(s) for the activity are true and activity can be started
- Started: Activity has started but not yet finished
- Finished: Activity is considered finished
- Closed: Every cost and effort measurement about the activity is held and considered as definitive. An activity can consume resources during its finished state, meaning a resource consumption due to non planed rework.
- Refined: More fine grained Sub-activities have been defined for a task while in the state Planned or Scheduled. Allocation of resources for Refined activities is used only for reference purposes.

### 3.3 Business Process and Process Management Languages

As stated previously, it is necessary to model processes as part of the V&V activities. Currently, there are several works within the Semantic Web arena to do so and some convergence will be required in the future. Examples of Business Processes Frameworks include the ebXML to define inter-processes communications. More interesting from the V&V point of view are other standards to define the internals of processes such as the BMPL (Business Process Modelling Language) [6] which builds upon existing Web Services standards including the Web Services Description Language (WSDL), XPath, XQuery and the Web Service Choreography Interface (WSCI). BMPL can be described as a process interface definition language for business processes. The BMPL can be used for V&V activities in a different number of ways. For example, one of such activities inside V&V activities that can be further explored using BMPL is the possibility of representing scenarios and Use Cases [11] using such languages so that several testing tasks can be automated, e.g. generating documentation, etc.

## 4 Conclusion

We are in the early stages of identifying and meta-modelling the main concepts of V&V, within the context of software management. V&V deserves a separate ontology as the activities carried out intend to provide *compliance* to other activities.

Because of V&V activities are an integral component of any development project in general and software project in particular, the main key issue is to integrate these activities into the software development plan. The goals of the ontology are:

- Embed V&V activities in a the general framework of other project activities
- Enable traceability between V&V activities and components of the project
- Keep the information related to cost and effort, including both scheduled and actual

Some ontologies, already available, may be used as building blocks for the V&V meta-model. The attempt to build an ontology for testing and validation is not directed towards the use of artificial intelligence techniques (although it is an aside effect) but to define data models and measures that the project manager may find useful for the tasks of software V&V management.

Our last goal is to make usable and exchangeable all the information generated in different projects within different organizations. As an intermediate, but essential step, we have to model the software V&V.

## Acknowledgments

This work is supported by CICYT TIC-1143-C03-01, CICYT TIC-1143-C03-03, UPV-EHU EA-8099.

## References

[1] A. Abran and A. Sellami, "Models of the Measurement Concepts in the ISO Vocabulary," presented at 12th International Workshop on Software Measurement IWSM 2002 Workshop Program, Magdeburg, Germany, 2002.

[2] V. R. Basili, G. Caldiera and D. Rombach, "Experience Factory," in *Encyclopedia of Software Engineering*, vol. 1, J. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 469-476.

[3] K. Bassin, S. S. Biyani and P. Santhanam, "Metrics to evaluate vendor-developed software based on test case execution results," *IBM Systems Journal*, vol. 41, pp. 13-30, 2002.

[4] T. Berners-Lee, J. Hendler and O. Lassili, "The Semantic Web," *Scientific American*, vol. 284, pp. 28-37, 2001.

[5] A. Bertolino, "Software Testing," in *Guide to Software Engineering Body of Knowledge*, A. Abran, J. W. Moore, P. Bourque and R. Depuis, Eds. IEEE Press, 2001.

[6] BPML, "BPML 1.0: Business Process Modelling Language," 2002.

[7] V. Devedžic, "Understanding Ontological Engineering," in *Comm. of the ACM*, vol. 45, 2002, pp. 136-144.

[8] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM systems Journal*, vol. 41, pp. 4-12, 2002.

[9] V. Harmelen, "Reference description of the DAML+OIL ontology markup language," 2002.

[10] IEEE, "IEEE Standard for Software Verification and Validation," IEEE 1012-1998, 1998.

[11] I. Jacobson, G. Booch and J. Rumbaugh, *The Unified Software Development Process*. Reading: Addison-Wesley, 1999.

[12] H. M. Kim and M. S. Fox, "Using Enterprise Reference Models for Automated ISO 9000 Compliance Evaluation," presented at HICSS, 2002.

[13] H. M. Kim, M. S. Fox and M. Grüninger, "An Ontology for Quality Management - Enabling quality problem identification and tracing," *BT Technology Journal*, vol. 17, pp. 131-140, 1999.

[14] B. A. Kitchenham, R. T. Hughes and S. G. Linkman, "Modeling Software Measurement Data," *IEEE Transactions on Software Engineering*, vol. 27, pp. 788-804, 2001.

[15] O. Lassila and R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," W3C 1999.

[16] R. A. Paul, T. L. Kunii, Y. Shinagawa and M. F. Khan, "Software Metrics Knowledge and Databases for Project Management," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 255-264, 1999.

[17] S. R. Rakitin, Software Verification and Validation. A Practitioner's Guide: Artech House, 1997.

[18] C. Rankin, "The Software Testing Automation Framework," *IBM Systems Journal*, vol. 41, pp. 126, 2002.

[19] M. Satpathy, R. Harrison, C. Snook and M. Butler, "A Generic Model for Assessing Process Quality," in New Approaches in Software Measurement - 10th International Workshop on Software Measurement (IWSM'00), vol. 2006, Lecture Notes in Computer Science, R. Dumke and A. Abran, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 94-110.

[20] C. Tautz and C. G. von Wangenheim, "REFSENO: A Representation Formalism for Software Engineering Ontologies," Fraunhofer IESE IESE-015.98/E, 20 Oct 1998 1998.

[21] D. Tham, M. S. Fox and M. Gruninger, "A Cost Ontology for Enterprise Modelling," presented at Third Workshop on Enabling Technologies - Infrastructures for Collaborative Enterprises, West Virginia University, 1994.

[22] J. Tuya, P. Fernández, M. A. Prieto, J. Aguilar, I. Ramos, J. Riquelme, F. Ferrer, M. Toro, M. Ruiz-Carreira, D. Rodriguez, M. Satpathy, R. Harrison, A. Ruiz de Infante, J. J. Dolado, R. Matilla and M. A. Álvarez, "Integration of Information in a Training Environment for Software Project Management," presented at Software Quality Management 2001 SQM'01, Loughborough, UK, 2001.

[23] M. Uschold and M. Grüninger, "Ontologies: Principles, Methods, and Applications," *Knowledge Engineering Review*, vol. 11, 1996.

[24] M. Uschold, M. King, S. Moralee and Y. Zorgios, "The Enterprise Ontology," *The Knowledge Engineering Review*, vol. 13 Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate), 1998.

[25] Y. Wand and R. Weber, "An Ontological Model of an Information System," *IEEE Transactions on Software Engineering*, vol. 16, pp. 1281 - 1291, 1990.

[26] F. Xia, "On the Danger of Developing Measures Without Clarifying Concepts," presented at IEEE, 1998.