Formalising ERP Selection Criteria

Xavier Burgués Illa
Univ. Politècnica Catalunya
Edifici C6 Campus Nord
Jordi Girona Salgado 1- 3
08034 Barcelona. Spain
+34 934 017 006
diafebus@lsi.upc.es

Xavier Franch
Univ. Politècnica Catalunya
Edifici C6 Campus Nord
Jordi Girona Salgado 1- 3
08034 Barcelona. Spain
+34 934 016 965
franch@lsi.upc.es

Joan Antoni Pastor
Univ. Politècnica Catalunya
Edifici C6 Campus Nord
Jordi Girona Salgado 1- 3
08034 Barcelona. Spain
+34 934 017 021
pastor@lsi.upc.es

Abstract

We present a proposal for selecting ERP products from a formal description of their relevant characteristics. The work is based on a previous and successful collaboration with a midsize company in the field of software package selection. An ERP was selected following a systematic methodology called SHERPA. In this past experience, SHERPA relied on natural language descriptions of the application domain, user needs and candidate ERP solutions. In this paper, we show that a formal language may be used for modeling this application domain, translating user needs into requirements over the ERP products, and for reflecting how concrete ERP products adjust to them. Having selection criteria used during ERP acquisition formally modeled, as well as user needs and ERP product descriptions, we expect to obtain more reliable and understandable results in this process.

Keywords
Software selection, requirements specification, ERP.

1. Introduction

Current practices in the software industry give an increasing importance to software procurement [3]. A significant instance of software procurement relevance growth is the clear trend in both private and public companies with regard to their current options for software-based management information systems: the fast and wide proliferation of large packaged ready-made Enterprise Resource Planning (ERP) systems, surely among the most extreme examples of current Customisable Off-The-Shelf (COTS) software packages [2].

We have experienced the domain of ERP selection by means of a collaboration with a midsize company, that we nickname Magic for privacy reasons.

The collaboration ended with a successful selection of an ERP for this company and was carried out using a new methodology called SHERPA [19, 20]. The methodology states systematic selection phases and involves documents and information which were expressed in natural language in this reported experience.

Despite the satisfactory results of the experience, we believe that it is possible to improve a particular aspect of this work. We think that procurement processes should be as well-defined and systematic as possible [6]. For this reason, as other authors agree [1, 7, 14], we advocate the use of formal notations for structuring the criteria used during software selection, for describing software components with respect to these criteria and in general to help the whole selection process, including tool support.

In this paper we aim at contributing to fill this existing gap in software procurement. We will show that it is possible to use a formal language as a support of the SHERPA methodology. We use *NoFun* [4, 5], a language conceived to deal with components and packages selection criteria. The formal resulting descriptions of selection criteria, which we consider satisfactory enough, are detailed in sections 3 to 5, after a short explanation of the SHERPA methodology appearing in section 2.

2. Overview of Sherpa

For the purposes of this overview, we define *ERP software acquisition* as the following decision process: clearly define the need that should be fulfilled with the help of an ERP product and/or related service; find suitable products and services in the market that may help in the fulfillment of such a need; establish appropriate criteria for the evaluation of ERPs; evaluate products and services in the light of these criteria; select the best available product and service, or the best possible

0-7695-0884-7/00 $10.00 © 2000 IEEE

combination of products and services; and negotiate the final contract with the product vendor or service provider.

Both the lack of methods for ERP acquisition and the lack of ERP procurement case studies motivated us to develop, apply and propose SHERPA (*Systematic Help for ERP Acquisitions*), a rigorous methodology tailored for small and medium companies for which it is difficult to apply other existing, more generic and sophisticated methods. It covers all the ERP acquisition process, from the search for candidate ERPs to the signature of the contract with the provider of the selected ERP and related services. In the other end, it does not cover the implementation of the selected ERP, and neither its usage nor maintenance, extension, evolution or retirement. In this section we briefly overview the different phases of SHERPA, while mentioning the criteria tables used in the evaluation process. We refer the reader to [19, 20] for a detailed account of the whole method.

- *Phase 0: Study strategy and business processes and decide to acquire an ERP.*

This phase is divided in two very different stages. In the first stage, the project team studies the business (mission, strategy, etc.), its departments and business processes. This is something that we consider fundamental if the team is going to evaluate how well each ERP adapts to the organization. In the second stage, a committee has to decide if the company has to acquire an ERP. This decision consists on a deep study of each alternative (internal or external custom development, integrating best-of-a-breed vertical packages, maintaining existing systems, etc), in order to adopt one (or a mixture of some) of them.

- *Phase 1: Search for candidates and first filter.*

Based on the knowledge about the company obtained on Phase 0 and on some minimum requirements about candidate ERPs (maximum cost affordable, platform, etc.), the project team conducts a market research initiative looking for ERPs suitable for the organization. The project team has to obtain enough minimum information on each ERP so, applying the minimum requirements, the number of candidates can be reduced to a small number (between 5 and 8 at the most in the Spanish ERP market for midsize companies).

- *Phase 2: Dig into the candidates and second filter.*

Here the project team needs much more information about the ERPs obtained in Phase 1. This information should be obtained in one or more interviews with the providers, getting as many fact sheets, catalogs, articles, etc., as possible. Applying a long list of more detailed selection criteria —which has to be refined and adapted to the organization—, the project team should select 2 or 3 ERP candidate solutions. This phase and the following are the ones in which the use of a formal notation may be more helpful.

- *Phase 3: Analysis and demonstration of candidates and visits to the providers.*

At this point the ERP providers have to demonstrate their products to the project team, the company top management, the mid-level management and a selected group of future final users. The purpose here is to obtain a much deeper knowledge on each solution, specifically on its functionality and adaptability to the organization. The project team gathers all the opinions; reviews and refines the application of the list of criteria to each candidate ERP; and prepares a selection proposal, which has to be approved first by IT management and, finally, by top management.

- *Phase 4: Final decision, negotiation and planning.*

The project team negotiates the contract with the selected ERP provider, including the estimation of the cost and the overall implementation plan, two very important aspects that should be estimated by the ERP provider, and a contingency plan. Finally, IT management and top management give their final approval and the signature of the contract with the ERP provider may proceed.

A key point of applying SHERPA methodology is that several selection criteria tables are built along the various phases, by incrementally refining and enriching the specific evaluations for the set of ERP solutions being considered in each phase. In particular, for the Magic case we derived a list of around 30 first-level evaluation criteria, which become many more when refined, and that we have grouped in 6 categories: strategy, functionality, technical, provider, services and economic [19]. Overall, they cover all typical aspects that can affect a decision of buying an ERP. Formalisation of these criteria is the current focus of our work, as explained below.

3. ERP selection application domain modeling

We will show in this section the use of the *NoFun* language [4, 5] for describing two particular sets of criteria proposed by the SHERPA method, namely those concerning functionality and technical issues. In both cases, we present a summary that takes the form of a table containing the name of the used criteria and their informal definition. It must be noted that, as [20] points out for the case of European midsize companies, organizations consider "best-fit" functional requirements as the most important ones when evaluating ERP solutions.

As a previous interesting remark, it must be said that, although a primary goal has been to follow exactly the

criteria as they were used in [19], their formalisation has pointed out some inconsistencies between informal documents or parts of a same document. For instance, the informal requirements often refer to selection criteria that were not introduced during the establishment of relevant factors. Therefore, *NoFun* description of the tables contains some additional elements that are not present in the original case study. On the other hand, some criteria appearing in earlier phases of SHERPA do not appear on later phases (requirements and/or product description). We believe that these sorts of side effects are main benefits of criteria formalisation, since more accurate descriptions and requirements improve ERP selection reliability.

3.1 The *NoFun* language.

We give here a very introductory explanation of *NoFun* in order to avoid difficulties in understanding the formal descriptions of SHERPA's criteria.

Currently, *NoFun* is designed to adhere to the ISO/IEC guides on software quality measurement [8, 10], which decomposes main software quality characteristics into subcharacteristics, and these ones into measurable software attributes. The language includes three different kinds of modules for these concepts, each kind with its own specific features. Modules may introduce domains and attributes. Attributes can be defined in terms of other attributes (i.e., they are derived) or not; in the last case, they are given a default value. On the other hand, attributes are of a certain type, which may be a domain, a function (declared as **function**[range, range]), a tuple (<type, type, ..., type>) or a set (**set**[type]). These type constructors may be freely combined.

Modules may perform refinements of previous modules (domains are inherited with the previous values and may be extended with new values in the new module or collapsed -declared as **refines to**- to a compatible domain), allowing incremental and modular construction of descriptions. Additionally, this structure allows to set general scenarios which will be further specialised depending on the kind of software product required to deal with, and even on the concrete characteristics of a company. Some particular features of the language used in the formal descriptions are not defined here for the sake of brevity; we think that their meaning is self-explained.

3.2 Functionality

The functionality table addresses three main aspects: which functional areas are covered by the product; how flexible the product is with respect to adaptability and openness; and some ERP specific features.

Table 1. Functionality criteria

Criteria	Definition
Included functionality	Areas or functions of the company that the ERP has to serve. It is described how the ERP covers each function.
Main target	Functional area or areas for which the ERP is specially oriented or strong.
Adaptability	Possible level of customisation in general and for the specific company.
Openness for - custom development - working with other systems	Level of openness to additional bespoke development (internal or external) and to other existing applications (for example, vertical applications, API, CRM, SCM, etc.).
Specifics supports	For example, Y2K, euro, ISO-9000, etc.

The *NoFun* description begins with the definition of two basic modules, which may be useful in many other contexts (see fig. 1). The first one, *FUNC_DOMAINS*, encapsulates company areas and product specific supports. They are left abstract for further refinement, since different scenarios may require different values. The second one, *FUNC_MEASURES*, declares also an abstract domain (used in most of the other functionality criteria) for classifying attributes into levels; just least and greatest values are fixed. The domain is declared as ordered for comparison purposes. The domain will be used in next modules as the range of a function that evaluates functional areas, and also for the flexibility criteria domain with pairs of values in both cases.

```
attribute module FUNC_DOMAINS
  explanation ERP areas and specific supports referred
              to in the SHERPA table
  domain Areas, SpecSupports
end FUNC_DOMAINS

attribute module LEVEL
  ordered domain Level = bottom None, top Optimal
end LEVEL
```

Fig. 1. Basic domains for functionality criteria

On top of the basic modules we define in fig. 2 the subcharacteristic modules *ORIENTATION*, *FLEXIBILITY* and *SP_SUPP*.

```
subcharacteristic module ORIENTATION
  explanation Incl. functionality and Main target
  imports FUNC_DOMAINS, LEVEL
  Coverage: function [Areas, Level] default None
  MainAreas: set [Areas] derived as
    {x / Coverage(x) = Optimal }
end ORIENTATION

subcharacteristic module FLEXIBILITY
  explanation Adaptability and Openness rows
  imports LEVEL
  Adaptability: <General, Company: Level> default
                                    <None, None>
  Openness: <Bespoke, Ext: Level> default
                                    <None, None>
end FLEXIBILITY

subcharacteristic module SP_SUPP
  explanation Corresponds to Specific supports row
  imports FUNC_DOMAINS
  AvailSpecSupport: set [SpecSupports] default {}
end SP_SUPP
```

Fig. 2. Subchar. for functionality criteria

Finally, the characteristic module *FUNCTIONALITY* just imports the previous modules (see fig. 3).

```
characteristic module FUNCTIONALITY
  explanation SHERPA's functionality table
  imports ORIENTATION, FLEXIBILITY, SP_SUPP
end FUNCTIONALITY
```

Fig. 3. Functionality characteristic

The characteristic module *FUN_MAGIC* presented in fig. 4 provides the extensions induced by the real case described in [20] coming from the Magic company.

```
char module FUN_MAGIC refines FUNCTIONALITY
  explanation Functionality table adapted to MAGIC
  Areas = Commercial, Logistics, Production, Finance,
          Staff, Quality, Technics, Management
  Level = None, Marginal, Weak, Medium, Strong,
          Optimal
  SpecSupports = Year2000, ISO9000, Euro,
                 Multicurrency
end FUN_MAGIC
```

Fig. 4. Specialisation of the functionality criteria in the Magic case study

3.3 Technical criteria

The technical table is reproduced below. We remark that all these criteria involve some kind of measuring which yields to the corresponding domains and attributes; this will impact in the *NoFun* specification.

Table 2. Technical criteria

Criteria	Definition
Platforms	Information technology platforms supported
Database management systems	DBMS or DBMSs used as base for the ERP.
Languages and development tools	Languages and development tools used to customise the ERP.
User management tools	Management capabilities: users, user groups, access levels, roles, authorisations, etc.
User documentation - Printed manual - Online help - Tutorials	Type of user documentation for training and helping to use the ERP.
Technical documentation - Database schema - Source code - Design	Technical documentation provided about internal structure of ERP master programs and data bases..
External connectivity - Internet/Web - Remote - EDI	Types of external connectivity supported.

The *NoFun* description uses three basic modules (fig. 5). Note the extension relationship in domains involving databases. The domain *measure* will be further refined with the different aspects to measure; therefore the level scale may be differently refined for each aspect.

```
attribute module SQL_DB
  explanation Standard SQL DBMS
  domain StdSQL
end SQL_DB

attribute module DATABASES refines SQL_DB
  explanation Known DBMS (includes all Std. SQL)
  rename StdSQL to DataBases
end DATABASES

attribute module MEASURE
  explanation A common pattern to measure several aspects
  ordered domain Level
  domain Aspect
  measure: function [Aspect, Level]
end MEASURE
```

Fig. 5. Basic domains for technical criteria

```
subcharacteristic module CONNECTIVITY refines
MEASURE
  rename Aspect to ConnType, Level to ConnLevel,
         measure to ConnMeasure
end CONNECTIVITY

subcharacteristic module CONTROL refines MEASURE
  rename Aspect to CtrlAspects, Level to CtrlLevel,
         measure to CtrlMeasure
end CONTROL

subcharacteristic module USER_DOC refines MEASURE
  rename Aspect to UserDoc, Level to DocLevel,
         measure to DocMeasure
  AvailUserDoc: set [UserDoc]
end USER_DOC

subcharacteristic module USE_MEASURE refines
MEASURE
  explanation A common scale to measure degree of usage
  import LEVEL
  rename MEASURE.Level to UseLevel
  UseLevel refines to LEVEL.Level
end USE_MEASURE

subcharacteristic module TOOLS_USE refines
USE_MEASURE
  rename Aspect to Tools, UseLevel to ToolsUseLevel,
         measure to ToolsMeasure
end TOOLS_USE

subcharacteristic module DB_USE refines
USE_MEASURE
  import DATABASES
  Aspect refines to DataBases
  rename UseLevel to DBUseLevel, measure to
DBMeasure
  UsedDB: set [DataBases]
end DB_USE
```

Fig. 6. Measures for technical criteria

```
subcharacteristic module TOOLS
  import TOOLS_USE
  InternetComp: function [Tools, Bool]
end TOOLS

subcharacteristic module DOCS
  import USER_DOC
  domain TechDoc
  AvailTechDoc: set [TechDoc]
end DOCS

subcharacteristic module DB
  import DB_USE
  ODBCComp: function [DataBases, Bool]
end DB

attribute module MEASURE_TYPE
  explanation A common type pattern to measure
  ordered domain Level
  domain Aspect
  type measure: function [Aspect, Level]
end MEASURE_TYPE

subcharacteristic module PLATFORMS refines
MEASURE_TYPE
  rename Aspect to Capabilities, Level to CapLevel
  CapLevel = bottom Null
  domain Platforms
  PlatSupLevel: function [Platforms, measure]
  SupPlat: set [Platforms] derived as
      {x/ ∃ y: y ∈ Capabilities: PlatSupLevel(x)(y) ≠ Null }
end PLATFORMS

characteristic module TECHNICAL
  explanation Corresponds to SHERPA's technical table
  import CONNECTIVITY, CONTROL, TOOLS, DOCS,
         DB, PLATFORMS
end TECHNICAL
```

Fig. 7. Adding attributes over measures

Fig. 6 shows the definition of such refinements (*CONNECTIVITY, CONTROL, USER_DOC*). We then introduce the module *USE_MEASURE*; it defines a measure pattern the refinement of which will share the same measure level. Such refinements (*TOOLS_USE, DB_USE*) appear at the end of fig. 6.

Fig. 7 adds attributes to complete the formalisation. It is worth to highlight that formalisation here requires some decisions to be made, solving some open questions that were somehow hidden in the former application of SHERPA. More specifically, the platform technical criteria has been refined as an evaluation of the support to several aspects (like graphical interface; *Capabilities* attribute) for each platform (for instance, Windows, UNIX, etc.; *Platforms* attribute). We define the *PlatSupLevel* attribute as a (higher-order) function from platforms to functions from capabilities to the scale.

The *SupPlat* attribute is a set containing the platforms for which there is at least one aspect covered. It is useful to define a measure type to allow easy declaration of higher-order functions in further modules. Finally, module *TECHNICAL* imports all the defined attributes.

Fig. 8 shows the corresponding specialised module to extend domains for the Magic case. We remark that the measure *CtrlLevel* is refined indeed as a boolean, just indicating presence/absence of a feature.

4. Requirements specification

In this section we aim at showing one of the more explicit advantage of using a well-structured formal language during the software procurement process: the specification of user needs by means of quality requirements. To do so, we use the requirements (over

functionality and technical issues) used during the ERP selection process for the Magic company, expressed in terms of the attributes in section 3.

```
characteristic module TECH_MAGIC refines
TECHNICAL
  explanation Technical table adapted to MAGIC
  Platforms = AS400, WinNT
  Capabilities = GraphWin, TextInt, Network
  StandardSQL = Oracle, SQLServer; DB2/400
  UseLevel = bottom None, top Full
  CtrlLevel refines to bool
  DocLevel = None, WinReq, NoWinReq
  DocType = HardCopyMan, OnLineHelp, Tutorials
  .....
end TECH_MAGIC
```

Fig. 8. Characteristics for technical criteria

Requirements are expressions built up over attributes and are related to their informal statement, together with a reference to the document where it can be found (not shown in the examples below for the sake of brevity). They are encapsulated in another kind of module.

Fig. 9 and 10 show two examples encapsulating requirements over functionality and technical issues. The "informal" statements should preferably be read before the formalisation. We remark the use of some mathematical stuff: quantifiers, set operators, first-order formulae and so on. However, the resulting expressions tend to be clear enough to serve the understandability purpose of using *NoFun*. Note the possibility of decomposing requirements into simpler ones, as done in the *HighOpeness* one.

5. ERP description

The last facet of *NoFun* allows to describe the candidate packages with respect to the selection criteria. We show in fig. 11 a particular case, which is a translation of the informal description found in [18] for the ERP finally selected by the Magic company.

6. Conclusions and related work

We have formalised part of the knowledge related to a real experience of ERP procurement using the SHERPA methodology. The selection criteria concerning functionality and technical aspects have been formalised, as well as the user needs on these criteria and the description of a particular ERP product. The chosen formalism has proven to be rich enough to catch all the specificities required by the SHERPA methodology, and the result has been a complete, formal description of all these elements. We think that the notation may be used to describe quality aspects of general software components. We are in fact using *NoFun* for two different applications in which we are also obtaining satisfactory results about the usefulness of the notation: compilation of information about class libraries (for now, LEDA, Booch and STL) and documentation and classification of automatically generated user interfaces.

```
req mod. TECH_REQ_MAGIC on TECH_MAGIC
 RightPlatform:
  informal Should run on NT, AS/400 or both, support
    text. interf. to AS/400, graph. int. to AS/400 and NT net
  defined as SupPlat ≠ ∅ ∧
              AS400 ∈ SupPlat ⇒
                PlatSupLevel(AS400)(TextInt) = Yes ∧
                PlatSupLevel(AS400)(GraphWin) = Yes ∧
              WinNT ∈ SupPlat ⇒
                  PlatSupLevel(AS400)(Network) = Yes
 RightDataBase:
  informal Should be std. SQL, Oracle or SQLServer for
        NT or DB2/400 for AS/400. ODBC compatible.
  defined as StandardSQL ∩ SupportedDB ≠ ∅ ∧
        WinNT ∈ SupPlat ⇒
          SupportedDB ∩ {Oracle, SQLServer } ≠ ∅ ∧
        AS/400 ∈ SupPlat ⇒ DB2/400 ∈ SupportedDB ∧
        ∃ db: db ∈ SupportedDB: ODBCComp(db)
 InternetCompTools:
  informal Tools should be internet compatible
  defined as ∀ t: t ∈ Tools:
            ToolsUseLevel(t) > None ⇒ InernetComp(t)
 RequiredCtrl:
  informal Possibility to define users and access levels;
            basic authentication
  defined as CtrlAspLevel(AccLevels) ∧
          CtrlAspLevel(Users) ∧ CtrlAspLevel(Authent)
 RequiredUserDoc:
  informal Complete documentation and readable by
            non-windows users.
  defined as AvailUserDoc = DocType ∧
            ∀ d: d ∈ AvailUserDoc:
                  DocTypeLevel(d) ≥ NoWinReq
 RequiredTechDoc:
  defined as AvailTechDoc ⊃ { DBEsch, Dictionary }
 RequiredConnectivity:
  informal Internet and remote connection should be
            available for commercial agents
    defined as Connectivity(Internet) > None ∧
            Connectivity(Remote) > None
end TECH_REQ_MAGIC
```

Fig. 9. Requirements on technical criteria

```
req module FUN_REQ_MAGIC on FUN_MAGIC
  FullCoverage:
    informal Should cover all the company areas
    defined as ∀ a: a ∈ Areas: Coverage(a) > None
  RightOrientation:
    informal Should emphasise commercial and logistic
    defined as { Commercial, Logistics } ⊂ MainAreas
  HighOpeness:
    informal Should be as open as possible, both for adding
      functionality and interconnecting with other software.
    decomposed as
        HighOpeness-1: defined as
          Open.Bespoke ≥ Strong ∧ Open.COTS ≥ Strong
        HighOpeness-2: defined as
          max(Open.Bespoke) and max(Open.COTS)
  FullSpecSupport:
    informal Should support all specific features
    defined as SpecSupport = ImplSpecSupport
end FUNC_REQ_MAGIC
```

Fig. 10. Requirements on functionality criteria

```
description module FUN_PROD[FUN_MAGIC]
  Coverage(Commercial, Finance, Staff,
           Quality, Technics) = Strong
  Coverage(Logistics) = Medium
  Coverage(Production) = Optimal
  Coverage(Management) = Marginal
  Adapt = <Optimal, Strong>; Open = <Strong, Strong>
  AvailSpecSupport = {Year2000, Euro }
end FUN_PROD

description module TECH_PROD[FUN_MAGIC]
  PlatSupLevel(WinNT)(Network) = Yes
  PlatSupLevel(WinNT)(GraphWin) = Yes
  AvailTechDoc: SelTechDoc = {DBEsch}
  AvailUserDoc: SelDocType =
       {HardCopyMan, OnLineHelp, Tutorials}
  SupportedDB: SelDB = {SQLServer}
  DBUseLevel: (SQLServer) = Full
  DocTypeLevel = NoWinReq
  CtrlAspLevel: CtrlAspMeas = bottom(CtrlLevel)
  Connectivity(Internet, EDI, Remote) = Full
end TECH_PROD[FUN_MAGIC]
```

Fig. 11. Description of a candidate ERP

We think that our proposal may be the starting point of further work and research with the goal of making procurement as systematic as possible by means of formal languages and tools. In our opinion, some improvements may be expected from using this approach, including:

- A detailed and formal description of the domain is obtained, which provides a comfortable and precise framework for reasoning about the involved ERPs.
- It can help to determine organisational, business and user requirements that will facilitate more mature evaluations of alternatives, as well as clarify how the solution eventually selected fits these requirements.
- Not only the product but also the formalisation process is interesting by itself. During formalisation many questions may arise concerning the criteria, which otherwise would remain hidden. Also inconsistencies and ambiguities are discovered earlier, as some examples have shown in this paper.
- If we choose a modular language, we will be able to reuse part of the decision process. It will also be easier to deal with the dynamic nature of requirements, which may change before the end of ERP procurement and subsequent implementation.
- From our point of view, comparison of ERPs with respect to formally expressed criteria is a step towards ensuring that the decisions taken during procurement are adequate. This is specially true when dealing with more than a few candidate ERPs.
- Eventually, the existence of a widely accepted formal language would provide a lingua franca to which: 1) ERP vendors could adhere for elaborating uniform descriptions of their products, and 2) ERP procurement specialists could improve their ability for searching and examining ERP solutions.
- A structured and formal notation can be used as a basis for building ERP procurement toolkits.

Other authors promote the use of structured notations [1, 7, 11, 14], being [1] the closest to our view. It proposes a template with many parts (functional and non-functional information, related components, compliance, etc). They ask vendors for filling these templates as part of their business. Furthermore, they advocate the use of these descriptions as the basis for contracts between vendors and clients, mainly for upgrading purposes.

Also, PORE (Procurement Oriented Requirements Engineering Method) [14] is a method that proposes a 5 processes iterative method that, after gathering requirements from users, serves to evaluate candidate products and to select one or more, and to negotiate the contract. PORE is a quite elaborated and generic method, which provides templates for the processes, and documents and guidelines for the project team. We believe that PORE demands more effort than SHERPA while being more generic, i.e. it does not provide tables of ERP-focused criteria. Additionally, the criteria used in PORE have so far not been formalised to the level presented in this paper. Anyway, PORE represents a very good balance between effort and completeness. PORE is being adapted to address the evaluation of large COTS packages [13], a category that presumably includes ERP systems.

Another approach is OTSO (Off-The-Shelf Option) [11]: a method oriented to the search, evaluation and

selection of reusable software in a six-phase process. The four initial phases roughly correspond to Sherpa's ones. OTSO proposes a hierarchical definition of criteria similar to the kind of descriptions encouraged by *NoFun*. Non-basic criteria are evaluated consolidating lower level values using the AHP (Analytic Hierarchy Process), a methodology to support multiple criteria decision-making. We believe that *NoFun* modules could be used to support the OTSO method encapsulating, criteria descriptions (replacing and/or complementing the template tables adopted in OTSO) and implementing AHP. An interesting idea to evaluate costs and values of alternatives is the baseline concept (cost is the effort needed to attain the baseline; value is the set of features which exceed the baseline).

There are other approaches proposing formal description languages for information systems, expert systems and so on [4, 12, 16]; regardless of their different domain, most of the ideas have a common background.

Another procurement related proposal is the work in [15], based on the eight phases of the procurement process of standards ISO/IEC-12207 and ISO/IEC 9126 [7, 8], focusing in large-scale software, risk management and quality assurance. Other relevant work is Software Acquisition Capability Maturity Model [17]. This model helps an organisation to evaluate its general software procurement process, obtaining a maturity level, and to improve it progressively. See [20] for a more detailed description of other related work.

Although we have presented a case study formalising a SHERPA-driven selection by means of the *NoFun* language, we think that the central idea of formalisation may be a source of improvements in the general field of software procurement. In this sense we will explore the possibility of using some widely spread language to aid in COTS activities. We think that the formal notation should be powerful, flexible and usable for non-technical users. It would be interesting to find out some graphical notation as an alternative or complement to the textual one.

Acknowledgements

This work has been partially supported by the spanish granted project TIC97-1157. We thank F. Sistach for his previous work and comments as well as the anonymous referees for their valuable comments.

References

1. J. Dong, P. Alencar, D.D. Cowan. "A Component Specification Template for COTS-based Software Development". In [2].
2. First Workshop on Ensuring Successful COTS Development. Workshop at 21st ICSE, 1999.
3. A. Finkelstein, G. Spanoudakis, M. Ryan "Software Package Requirements and Procurement". 8th Int. Workshop on Software Specification and Design, Berlin (Germany), 1996.
4. X. Franch. "Systematic Formulation of Non-Functional Characteristics of Software". Int. Conf. on Requirements Engineering (ICRE), Colorado Springs (USA), 1998.
5. X. Franch, X. Burgués. "A language for stating component quality". Proceedings of 14th Brasilian Symposium on Software Engineering (SBES), Joao Pessoa (Brasil), 2000.
6. X. Franch, J.A. Pastor. "On the Formalisation of ERP Systems Procurement". Workshop on Ensuring Successful COTS Development (22nd Int. Conf. on Software Engineering).
7. S. Guerra, A. Finkelstein. "Specification of COTS-based Systems". In [2].
8. ISO/IEC 9126:1991. *Information technology – Software product evaluation – Quality characteristics and guidelines for their use*, ISO, www.iso.org, 1991.
9. ISO/IEC 12207:1995. *Information technology – Software life cycle processes*, ISO, www.iso.org, 1995.
10. ISO/IEC 14598. *Information technology - Software Product Evaluation*. ISO, www.iso.org, 1999.
11. J. Kontio. "OTSO: a systematic process for reusable software component selection". Univ. of Maryland research report CS-TR-3478 UMIACS-TR-95-63, 1995.
12. D. Landes, R. Studer. "The Treatment of Non-Functional Requirements in MIKE". In Proceedings of Fifth European Software Engineering Conference (ESEC), Barcelona (Catalunya, Spain), LNCS 989, Springer-Verlag, 1995.
13. N. Maiden, L. James, C. Ncube. "Evaluating Large COTS Software Packages: Why Requirements and Use Cases are Important". In [2].
14. N. Maiden, C. Ncube. "Acquiring COTS Software Selection Requirements". IEEE Software, Vol. 15, No. 2, March/April, 1998.
15. J. Mayrand, F. Coallier. "System Acquisition Based on Software Product Assessment", Proc. of ICSE-18, 1996.
16. J. Mylopoulos, L. Chung, B.A. Nixon. "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach". IEEE Trans. on Software Engineering, 18(6)
17. SEI. *Software Acquisition Capability Maturity Model* Version 1.01, Technical Report CMU/SEI-96-TR-020, http://www.sei.cmu.edu, December 1996.
18. F. Sistach, L.F. Fernández, J.A. Pastor. "Towards the methodological acquisition of ERP Solutions". Research Report LSI-98-63-R, http://www.lsi.upc.es, December 1998.
19. F. Sistach, J.A. Pastor. "Methodological acquisition of ERP solutions with SHERPA", in First World Class IT Service Management Guide (Ed. J. van Bon), tenHagenStam, 2000.
20. F. Sistach, J.A. Pastor, L.F. Fernández. "Towards the methodological acquisition of ERP solutions for SMEs", keynote paper in Proc. of the First Int. Workshop on Enterprise Management and Resource Planning: Methods, Tools and Architectures (EMRPS'99), Venice (Italy) , 1999.
21. Y. van Everdingen, J. van Hillegersberg, E. Waarts. "ERP Adoption by European Midsize Companies". Communications of the ACM, Vol. 43, N. 4, April 2000.