# Extensive Formalisation of SHERPA Criteria with NoFun

Xavier Burgués Illa, Joan A. Pastor, Xavier Franch

Dept. Llenguatges i Sistemes Informàtics. Univ. Politècnica Catalunya

Edifici C6 Campus Nord. Jordi Girona Salgado 1- 3
08034 Barcelona. Spain

{diafebus|pastor|franch}@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. Now, 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.*

### 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.

We aim at contributing to fill this existing gap in software procurement. In [22] we have shown that it is possible to use a formal language as a support of the SHERPA methodology. We used *NoFun* [4, 5], a language conceived to deal with components and packages selection criteria. We extend here that work to all the SHERPA tables of selection criteria.

## 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 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 four of the sets of criteria proposed by the SHERPA method, namely those concerning functionality, technical, provider and services issues. In all cases, we present a summary that takes the form of a table containing the name of the used criteria and their informal definition.

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 different kinds of modules to capture these and other 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 old = new - 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<br>- custom development<br>- 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 consists in a set of modules related by refinement and decomposition relations. In figure 1 we show the modular architecture; dashed arrows correspond to decomposition while solid ones correspond to refinement.

We begin the description with the definition of three basic modules, which may be useful in many other contexts (see fig. 2). The third module, *MEASUREMENT*, offers abstract mesurement-related attributes which may be customized to many situations by means of domain refinement, as we will do in the module *ORIENTATION* above in the next figure. This module would eventually define more attributes and could reside in a non-functional modules library. As the first module specifically related to our target we find *FUNC_DOMAINS* which encapsulates company areas and product specific supports. They are left abstract for further refinement, since different scenarios may require different values.

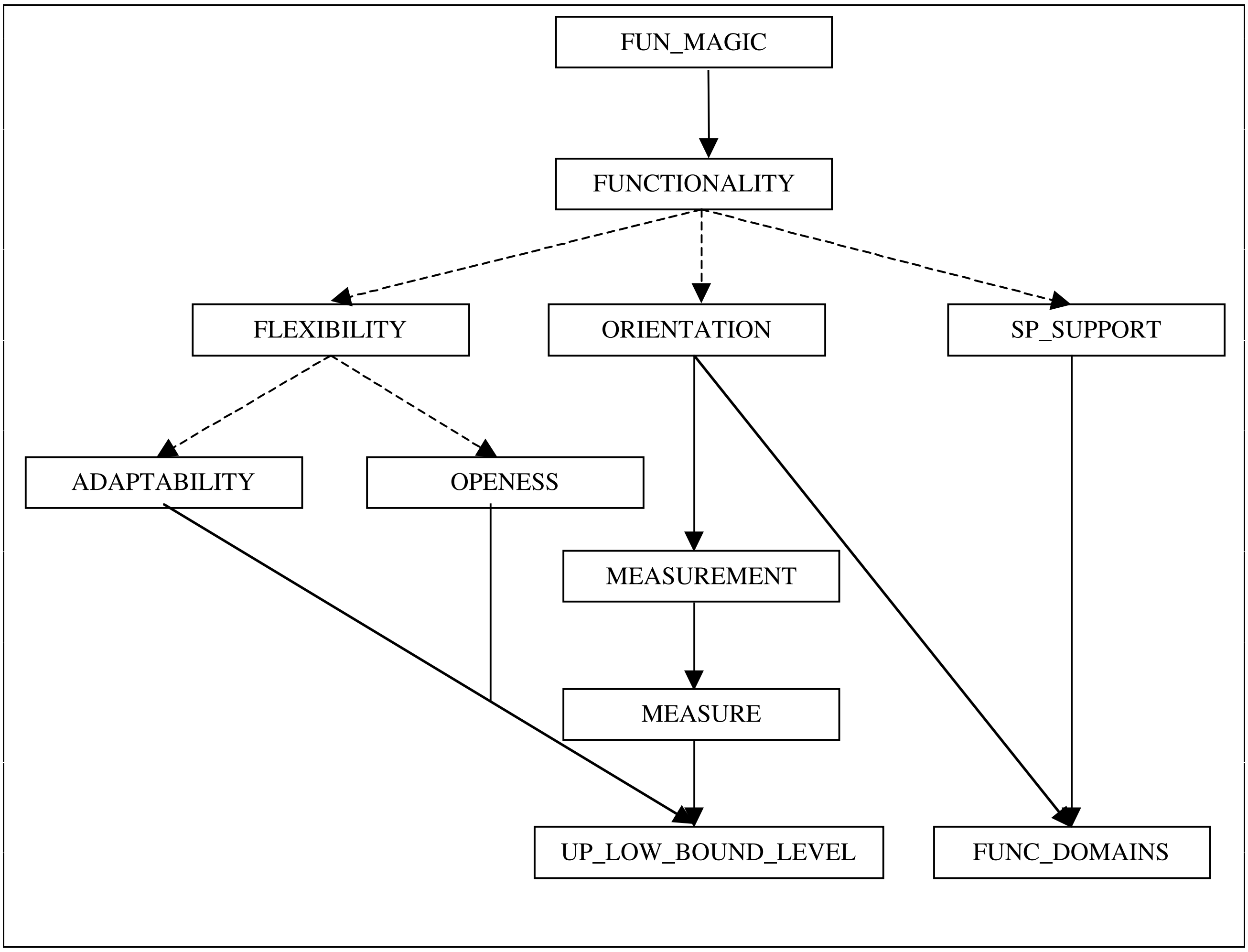


**Fig. 1. Modular architecture for functionality description**

On top of the basic modules we define in fig. 3 the subcharacteristic modules *ORIENTATION*, *FLEXIBILITY* (decomposed into *ADAPTABILITY* and *OPENESS*) and *SP_SUPPORT*. The first one binds the *aspect* domain (which represents the aspects to be measured) to *areas* (the company areas for wich we want to measure the degree of coverage of any candidate ERP). Subsequent modules use the same *level* domain to measure additional features of ERP; note that this decision deliveratealy imposes the same scale to measure different although related things.

Finally, the characteristic module *FUNCTIONALITY* just imports the previous modules (see fig. 4).

The characteristic module *FUN_MAGIC* presented in fig. 5 provides the extensions induced by the real case described in [20] coming from the Magic company.

```
domain module UP_LOW_BOUND_LEVEL
  ordered domain level = bottom None, top Optimal
end UP_LOW_BOUND_LEVEL

subcharacteristic module MEASURE over UP_LOW_BOUND_LEVEL
  domain aspect
  type measureType: function [aspect, level] default None
  measure: measureType
end MEASURE

subcharacteristic module MEASUREMENT over MEASURE
  best: set [aspect] = {x / measure(x) = Optimal}
  worst: set [aspect] = {x / measure(x) = None}
end MEASUREMENT

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

**Fig. 2. Basic domains for functionality criteria**

```
subcharacteristic module ORIENTATION over FUNC_DOMAINS,
                  MEASUREMENT (rename best by mainAreas, measure by coverage)
  explanation Incl. functionality and Main target
  aspect = areas
end ORIENTATION

subcharacteristic module FLEXIBILITY
  subcharacteristics ADAPTABILITY, OPENESS
end FLEXIBILITY

subcharacteristic module ADAPTABILITY over UP_LOW_BOUND_LEVEL
  explanation  Adaptability row
  adaptability: <general, company: level> default <None, None>
end ADPTABILITY

subchracteristic module OPENESS over UP_LOW_BOUND_LEVEL
  explanation  Openess row
  openness: <bespoke, ext: level> default <None, None>
end OPENESS

subcharacteristic module SP_SUPPORT over FUNC_DOMAINS
  explanation Corresponds to Specific supports row
  availSpecSupport: set [specSupports] default { }
end SP_SUPP
```

**Fig. 3. Subcharacteristic modules for functionality criteria**

**characteristic module** FUNCTIONALITY
**explanation** SHERPA's functionality table
**subcharacteristics** ORIENTATION, FLEXIBILITY, SP_SUPP
**end** FUNCTIONALITY

**Fig. 4. Functionality characteristic**

**characteristic module** FUN_MAGIC **over** 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. 5. Specialisation of the functionality criteria in the Magic case study**

### 3.3 Technical criteria

The technical table is reproduced as table 2. 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 (*MEASUREMENT* will now be refined by many modules).

**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<br>- Printed manual<br>- Online help<br>- Tutorials | Type of user documentation for training and helping to use the ERP. |
| Technical documentation<br>- Database schema<br>- Source code<br>- Design | Technical documentation provided about internal structure of ERP master programs and data bases.. |
| External connectivity<br>- Internet/Web<br>- Remote<br>- EDI | Types of external connectivity supported. |

The *NoFun* description modular architecture is shown in figure 6.

There is a module (fig. 7) which introduces domains involving databases. Note the extension relation between them.

Apart from this basic one, the modules in figure 7 introduce several measurement domains and attributes. First, some refinements (*CONNECTIVITY, CONTROL, USER_DOC*) of *MEASUREMENT* appear. This time we rename the *level* attribute so that it may be refined in different ways to measure different aspects. On the other hand, we have decided to use the same scale to measure the degree of usage of tools and data bases. To make explicit this decision, *TOOLS_USE* and *DB_USE* refine *MEASUREMENT* through a copy of it, *USE_MEASURE*.

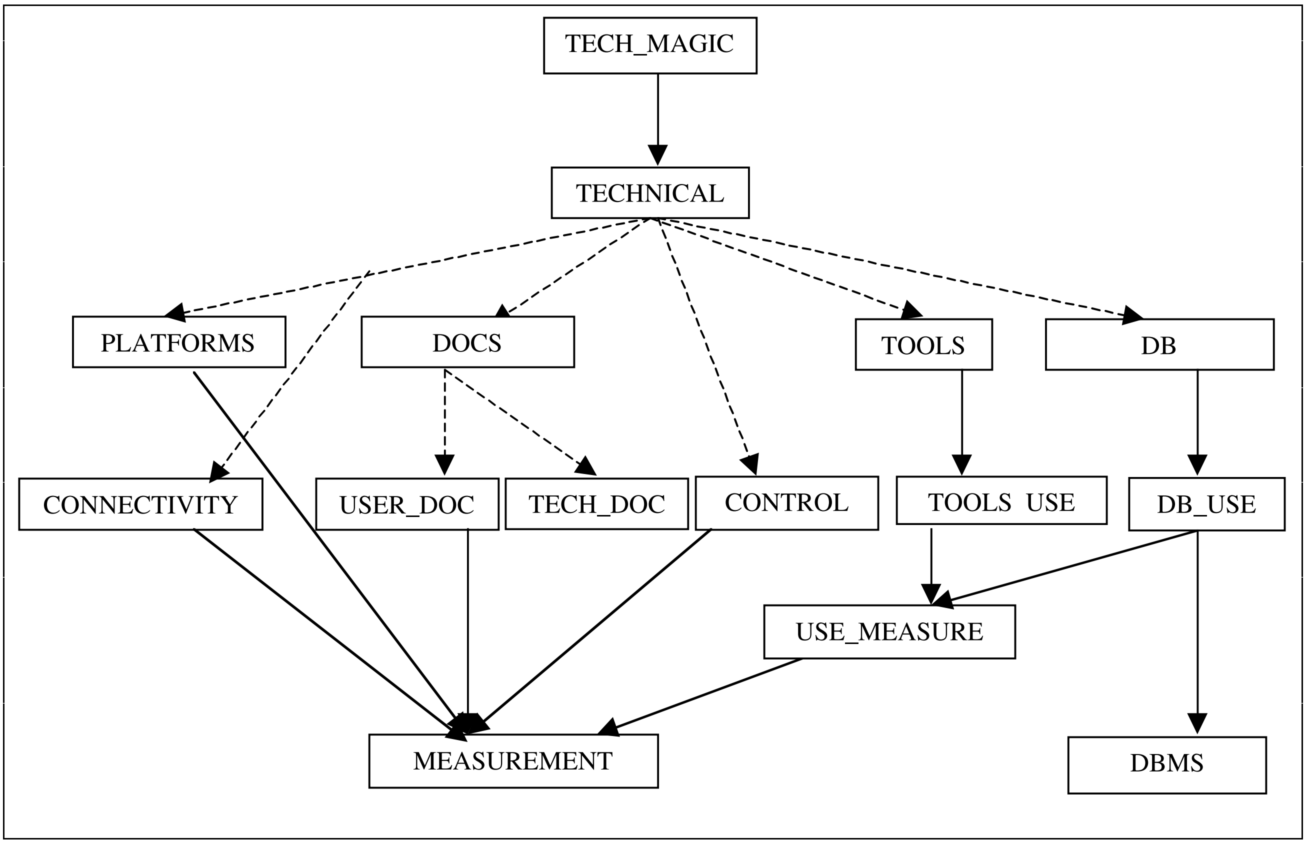


**Fig. 6. Modular architecture for technical description**

Fig. 8 adds attributes and domains 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.

```
domain module DBMS
  explanation databases.
  domain stdSQL, dataBases = stdSQL
end DBMS

subcharacteristic module CONNECTIVITY over
                    MEASUREMENT (rename aspect by connType,
                                 level by connLevel,
                                 measure by connMeasure)
end CONNECTIVITY

subcharacteristic module CONTROL over MEASUREMENT(rename aspect by ctrlAspects,
                                 level by ctrlLevel,
                                 measure by ctrlMeasure)
end CONTROL

subcharacteristic module USER_DOC over MEASUREMENT(rename aspect by userDoc,
                                 level by docLevel,
                                 measure by docMeasure)
   availUserDoc: set [userDoc]
end USER_DOC

subcharacteristic module USE_MEASURE over MEASUREMENT
  explanation Refining modules are expected not to refine the level in order to share the scale
end USE_MEASURE

subcharacteristic module TOOLS_USE over USE_MEASURE(rename aspect by tools,
                                 level by toolsUseLevel,
                                 measure by toolsMeasure)
end TOOLS_USE

subcharacteristic module DB_USE over DBMS,
                    USE_MEASURE (rename level by DBUseLevel,
                                 measure by DBMeasure)
  aspect = dataBases
  usedDB: set [dataBases]
end DB_USE
```

**Fig. 7. Measures for technical criteria**

The *supPlat* attribute is a set containing the platforms for which there is at least one aspect covered. Finally, module *TECHNICAL* imports all the defined attributes.

Fig. 9 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.

```
subcharacteristic module TOOLS over TOOLS_USE
  InternetComp: function [tools, Bool]
end TOOLS

subcharacteristic module DOCS
  subcharacteristics USER_DOC, TECH_DOC
end DOCS

attribute module TECH_DOC
  domain techDoc
  availTechDoc: set [techDoc]
end DOCS

subcharacteristic module DB over DB_USE
  ODBCComp: function [dataBases, Bool]
end DB

subcharacteristic module PLATFORMS over
                                   MEASUREMENT (rename aspect by capabilities,
                                                         level by capLevel)
  domain platforms
  platSupLevel: function [platforms, measureType]
  supPlat: set [platforms] derived as {x/ ∃ y: y ∈ capabilities: platSupLevel(x)(y) ≠ None }
end PLATFORMS

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

**Fig. 8. Adding attributes and domains to measures**

```
characteristic module TECH_MAGIC over TECHNICAL
  explanation Technical table adapted to MAGIC
  platforms = AS400, WinNT
  capabilities = GraphWin, TextInt, Network
  standardSQL = Oracle, SQLServer; DB2/400
  ctrlLevel = bool
  docLevel = None, WinReq, NoWinReq
  docType = HardCopyMan, OnLineHelp, Tutorials
   ....................................
end TECH_MAGIC
```

**Fig. 9. Specialisation for technical criteria**

### 3.4 Provider criteria

The provider table is reproduced below as it appears in [18]. The NoFun translation will differ slightly in order to be able to express requirements which refer to elements not present in the table (for instance, future expectations of the provider).

**Table 3. Provider criteria**

| Criteria | Definition |
|---|---|
| Provider characteristics: history, staff, clients, income, benefits... | Main provider characteristics as a company. |
| Localization | Localization of provider's offices (headquarters, delegations, etc.) |
| Similar implementations | Similar clients that use the ERP. |
| References | One or more clients, similar to the customer company, which use the ERP and could be asked for comments. |
| Experience | Experience of the provider in general and in the industrial sector of the company. |
| Confidence | Confidence of the project team on the provider and its ERP. |

The modular structure is very simple now (*PROVIDER* module is decomposed into *DESCR_DATA*, *LOCALISATION*, *CLIENTS* and *CONFIDENCE*); thus, we simply present the description in figure 10, skipping the modular structure.

We should think about building library module including cities and distances, as these attributes may be useful in the future.

```
attribute module DESCR_DATA
  explanation Descriptive data about the provider of the ERP: past and expected number of
              employees, clients, income and profit
  currentYear: integer
  subdomain past [integer] = {x / x ≤ currentYear}
  subdomain future [integer] = {x / x > currentYear}
  pEmpl, pCli, pIncome, pProfit: function [past, integer] default 0
  fEmpl, fCli, fIncome, fProfit: function [future, integer] default 0
  firstYear: integer derived as MIN(x: x ∈ past:
                  pEmpl(x) ≠ 0 ∨ pCli(x) ≠ 0 ∨ pIncome(x) ≠ 0 ∨ pProfit(x) ≠ 0)
  lastYear: integer derived as MAX(x: x ∈ future:
                  fEmpl(x) ≠ 0 ∨ fCli(x) ≠ 0 ∨ fIncome(x) ≠ 0 ∨ fProfit(x) ≠ 0)
end DESCR_DATA

attribute module LOCALISATION
  explanation kinds of installation (headquarters, ...) and cities where are located
  domain instal
  domain cities
  dist: function [cities, function [cities, real]]
        restricted to ∀x: x ∈ cities: ∀ y: y ∈ cities: dist(x)(y) = dist(y)(x)
        default 0.0
  locs: function [cities, set [instal]] default Ø
end LOCALISATION

attribute module CLIENTS
  explanation a sample of clients, clients similar to our company and similar clients which
              have communicated their degree of satisfaction with the ERP
  ordered domain satLevel
  clients: set [string]
  similar: set [clients]
  references: set [similar]
  Eval: function [references, satLevel] default MIN(satLevel)
end CLIENTS

subcharacteristic module CONFIDENCE
  ordered domain confLevel
  conf: confLevel
end CONFIDENCE

characteristic module PROVIDER
  subcharacteristics DESCR_DATA, LOCALISATION, CLIENTS, CONFIDENCE
end PROVIDER

characteristic module PROV_MAGIC over PROVIDER
  explanation Refinement of year of reference, cities, kinds of installations and levels
end PROV_MAGIC
```

**Fig. 10. Provider NoFun description**

### 3.5 Services criteria

We finish the section devoted to modeling the selection criteria addressing the services-related ones. As before, we present the informal table and the corresponding NoFun description.

**Table 4. Services criteria**

| Criteria | Definition |
|---|---|
| Implementation method and implementation experience. | Existence of an organized, documented and experimented implementation method. |
| Implementation services<br>- Installation, Adaptation,<br>- Training, Support,<br>- Custom development,<br>- Connection to other systems,<br>- Maintenance,<br>- Platform, Others. | Services offered by the provider during and after the implementation and regarding ERP and global IS issues. |
| Type of implementation | Strategy or strategies proposed by the provider for the implementation. |
| Implementation estimated time | Provider estimate for implementation time. |

```
subcharacteristic module IMPL_METHOD
  explanation Degree of sistematisation of the implementation method and number of
                    implementations performed using it
  ordered domain sistLevel
  sistDegree: sistLevel
  #impl: integer
end IMPL_METHOD

attribute module IMPL_SERVICES
  explanation Services offered by the provider and required by us
  domain reqServices, services = reqServices
  offServices: set [services] default Ø
end IMPL_SERVICES

attribute module STRATEGY
  explanation Strategies proposed by the provider
  domain strategies
  proposedStrategies: set [strategies]
end STRATEGY

characteristic module SERVICES
  subcharacteristics IMPL_METHOD, IMPL_SERVICES, STRATEGY
  implTime: integer
end PROVIDER

characteristic module SERV_MAGIC over SERVICES
  explanation Refinement of level and  interesting strategies and services.
  reqServices = Instal, Adapt, Train, Supp, SpecificDev, Connect, Maint, PlatServ
  strategies = AreaBased, Horizontal
  sistLevel = Poor, Acceptable, Good, Optimal
end SERV_MAGIC
```

**Fig. 11. Services NoFun description**

## 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 used during the ERP selection process for the Magic company, expressed in terms of the attributes in section 3.

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.

Figures 12 through 15 show the requirements over the criteria presented in the last section used to perform the selection. 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*.

It has to be said that it is not usual to find a candidate which satisfies all the requirements. That's why it is interesting to stablish some kind of priorities in the requirement in such a way that you may identify the candidates which satisfy the requirements with higher priorities. This is done decomposing a requirement in subrequirements; the order in which the subrequirements appear stablish their priority. See, for instance, the *HighOpeness* requirement in fig. 12. Another mechanism to cope with this question is the relevance assigned to each requirement.

**req module** FUN_REQ_MAGIC **on** FUN_MAGIC
**relevance =** Marginal, Weak, Medium, Strong, Mandatory
FullCoverage: Mandatory
**informal** Should cover all the company areas
**defined as** $\forall$ a: a $\in$ Areas: Coverage(a) > None
RightOrientation: Strong
**informal** Should emphasise commercial and logistic
**defined as** { Commercial, Logistics } $\subset$ MainAreas
HighOpeness: Medium
**informal** Should be as open as possible, both for adding functionality and interconnecting with other software.
**decomposed as**
HighOpeness-1: **defined as**
Open.Bespoke $\geq$ Strong $\wedge$ Open.COTS $\geq$ Strong
HighOpeness-2: **defined as**
max(Open.Bespoke) **and** max(Open.COTS)
FullSpecSupport: Medium
**informal** Should support all specific features
**defined as** SpecSupport = ImplSpecSupport
**end** FUNC_REQ_MAGIC

**Fig. 12. Requirements on functionality criteria**

**req module** PROV_REQ_MAGIC **on** PROV_MAGIC
**relevance =** Marginal, Weak, Medium, Strong, Mandatory
Experience: Strong
**informal**: there are clients similat to us
**definded as** similar $\neq \emptyset$
HistoryAndFuture: Strong
**informal**: the provider has a certain history and expectations for the future
**defined as** firstYear $\leq$ 1990 $\wedge$ lastYear $\geq$ 2010
NearbyLocalisation: Mandatory
**informal**: the provider has to be located in Barcelona
**defined as** $\exists$ c: c $\in$ cities: locs(c) $\neq \emptyset$ $\wedge$ dist(c)(Barcelona) $\leq$ 20
**end** PROV_REQ_MAGIC

**Fig. 13. Requirements on provider criteria**

**req module** SERV_REQ_MAGIC **on** SERV_MAGIC
**relevance =** Marginal, Weak, Medium, Strong, Mandatory
ImplMethod: Strong
**informal**: the implementation method is a sistematic and experimented one
**defined as** sistDegree $\geq$ Acceptable $\wedge$ #impl $\geq$ 5
ImplServices: Mandatory
**informal**: all services are provided
**defined as** OffServices = Services
ImplStrategies: Strong
**informal**: Area based strategy is preferred
**defined as** AreaBased $\in$ proposedStrategies
ImplTime: Medium
**informal**: implementation should last no more than one year
**defined as** implTime $\leq$ 12
**end** SERV_REQ_MAGIC

**Fig. 14. Requirements on services criteria**

**requirements module** TECH_REQ_MAGIC **on** TECH_MAGIC
**relevance =** Marginal, Weak, Medium, Strong, Mandatory
RightPlatform: Strong
**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** $SupPlat \neq \emptyset \wedge$
$AS400 \in SupPlat \Rightarrow PlatSupLevel(AS400)(TextInt) = Yes \wedge$
$PlatSupLevel(AS400)(GraphWin) = Yes \wedge$
$WinNT \in SupPlat \Rightarrow PlatSupLevel(AS400)(Network) = Yes$
RightDataBase: Strong
**informal** Should be std. SQL, Oracle or SQLServer for NT or DB2/400 for AS/400. ODBC compatible.
**defined as** $StandardSQL \cap SupportedDB \neq \emptyset \wedge$
$WinNT \in SupPlat \Rightarrow SupportedDB \cap \{Oracle, SQLServer\} \neq \emptyset \wedge$
$AS/400 \in SupPlat \Rightarrow DB2/400 \in SupportedDB \wedge$
$\exists$ db: db $\in$ SupportedDB: ODBCComp(db)
InternetCompTools: Mediun
**informal** Tools should be internet compatible
**defined as** $\forall$ t: t $\in$ Tools:
ToolsUseLevel(t) > None $\Rightarrow$ InernetComp(t)
RequiredCtrl: Mandatory
**informal** Possibility to define users and access levels; basic authentication
**defined as** CtrlAspLevel(AccLevels) $\wedge$ CtrlAspLevel(Users) $\wedge$ CtrlAspLevel(Authent)
RequiredUserDoc: Weak
**informal** Complete documentation and readable by non-windows users.
**defined as** AvailUserDoc = DocType $\wedge$ $\forall$ d: d $\in$ AvailUserDoc: DocTypeLevel(d) $\geq$ NoWinReq
RequiredTechDoc: Strong
**defined as** AvailTechDoc $\supset$ { DBEsch, Dictionary }
RequiredConnectivity: Mandatory
**informal** Internet and remote connection should be available for commercial agents
**defined as** Connectivity(Internet) > None $\wedge$ Connectivity(Remote) > None
**end** TECH_REQ_MAGIC

**Fig. 15. Requirements on technical criteria**

## 5. ERP description

The last facet of *NoFun* allows to describe the candidate packages with respect to the selection criteria. We show in fig. 16 a particular case, which is a translation of the informal description found in [18] for the ERP finally selected by the Magic company.

```
behaviour module FUN_PROD valuates 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

behaviour module TECH_PROD valuates TECH_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

behaviour module PROV_PROD valuates PROV_MAGIC
   pEmpl(1986) = 100
   pCli(2000) = 624
   locs(Barcelona) = { ...  }
   locs(Madrid) = { ...  }
   clients = { ...  }
   similar = { Braun, Casbega, Hoesch, BioProt, Aguilar, Matri Vallès, RJR  }
   references = { BioProt,  Matri Vallès }
   eval(BioProt) = ...
   conf = High
end PROV_PROD

behaviour module SERV_PROD valuates SERV_MAGIC
   sistDegree = Good
   offServices = { Instal, Adapt, Train, Supp, SpecificDev, Connect, Maint, PlatServ }
   implTime = 7
end SERV_PROD
```

**Fig. 16. Description of a candidate ERP**

## 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.

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.

## 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.

22. X. Burgués, X. Franch, J.A. Pastor. "Formalising ERP Selection Criteria". 10th Int. Workshop on Software Specification and Design (IWSSD 2000). San Diego, California, USA. Nov. 2000.