

# Capturing Software Requirements of Business Processes in Multinational Firms: A Conceptual Framework for Practitioners

Mingtao Shi

FOM Fachhochschule für Oekonomie & Management

University of Applied Science

Bismarckstr. 107, 10625 Berlin, Germany

Tel: 49-171-2881-169 E-mail: Consulting\_Shi@yahoo.de

Received: December 16, 2010

Accepted: December 28, 2010

doi:10.5539/cis.v4n3p95

## Abstract

While capturing software requirements of business process, multinational firms are confronted with the fact that their business operations are scattered over a number of national markets. This paper suggests the approach of central-local-central loop to tackle the contextual complexity. First, Business process information is elicited, analysed and elaborated at the central headquarters. Second, requirements engineers must validate the centrally documented requirements at the local national markets where the application will ultimately be deployed. Empirical experience shows that observation, apprenticing and online-interview are effective field elicitation methodologies if they are applied in a combined and balanced manner. Third, the adapted, changed and added information shall be satisfactorily verified centrally and agreed upon by local and central product champions and decision makers.

**Keywords:** Central-local-central loop, Observation, Apprenticing, Online-interview, Activity-diagram

## 1. Background: Software Requirements

Broadly speaking, software for industrial applications is by nature mainly made in two ways. On the one hand, larger firms with abundant IT resources continue to “manufacture” their software applications internally. On the other hand, the last two decades have witnessed breathtaking growth of technology houses specialised in software production. Software vendors, such as SAP, have induced the emergence and development of industrial applications based upon core systems that can be individualised by parameterisation and customisation. Both types of software applications are commonplace in the industrial context. Small and medium-sized Enterprises (SME) have rather adopted the latter strategy and been “importing” from specialised vendors, because their internal IT department is too small to conduct software development from scratch for comprehensive business operations. Thus, SMEs tailor the parameterisation-capable and customisable software applications to their firm-specific business requirements.

No matter through which way the software comes into being, capturing the requirements of a firm’s operational activities is always indispensable. Although agile methodologies of software engineering exist, the essence of achieving software excellence continue to lie in several key phases, including *requirements analysis*; *architectural design*, *code construction*, *software testing*; *application configuration* and *delivery*. Requirements engineering as an important software activity is ubiquitous, no matter which application and which industry is looked at. From the vantage point of the author, an effective investigation of the software requirements encompasses sufficient analysis and documentation of a firm’s business *processes*, *products* and their *parameters* in a carefully defined *project* management environment. This paper focuses predominantly on the software requirement analysis of a firm’s business processes in a *multinational operational environment*.

Business processes are descriptions of how a firm’s value chain is built and how its operations are conducted and accomplished by different classes of users and end-consumers. These descriptions naturally shed light on system features, behaviour and interactions with users. The more seamlessly the system is integrated into the business processes defined by the operational and product managers, the more forcefully the application software is supporting the overarching business strategy of the firm.

## 2. General Patterns

While analysing and documenting business processes, multinational firms are faced with a set of similar contextual conditions generally. Business operations are normally carried out in different *local national markets*

and supervised or controlled by *central headquarters*. The knowledge of business details reside in both central headquarters and local subsidiaries. Intelligibly, central instances strive for global standardisation of business procedures, in order to achieve associated scale and scope economies. However, because the *central planers* are not located in the immediate vicinity of business reality, their definition or sometime even “imagination” of business processes is fraught with uncertainties. Their counterparties, the *local product line or product managers*, can provide considerable value-added business process information. Thus, the harmonic collaboration among these product champions is vital. Central planer’s definition of business product is often subject to *local adaptation*. Global standardisation is unavoidably to be supplemented by local responsiveness. Not only the business definition but also the *decision-making* take place both centrally and locally. *Corporate top managers* fairly frequently have to succumb to their local counterparties, because the “*national “kings”*” are the ones who are ultimately responsible for the profitability in the local marketplace and therefore have the ultimate say of how the local businesses should look like.

The extraction of software requirements from business information, in contrast, is most probably carried out by *central requirements engineers with local assistance*. Such personnel is located at the corporate centre and in contact with either the local product champions directly or the local IT employees who are drawing upon the business knowledge of the local product managers. The structure of this configuration is caused by the fact that on the one hand, setting up both central and local teams engaged and specialised in software requirements is too costly, and on the other hand, qualified software requirements engineers are relatively rare. Particularly in less developed national markets, such human resource infrastructure is not always readily available. Likewise, software development (code construction) also tends to be concentrated at the corporate centre. Scale advantage, central control over the user access rights, source or application codes are some of the additional reasons for the *central concentration of the software construction*.

Also a similar *modus operandi* exists for multinational firms. When it comes to map the software requirements of business processes, central requirements engineers usually organise interviews and workshops with central planers and decision makers, aiming at unveiling salient business information. These information-absorbing sessions often happen among knowledge workers with *cultural similarities*. Employees working and living at the location of corporate centre usually possess high cultural proximity. Communication modes and mentality are fairly similar. Requirements engineers however are often forced to conduct *field investigations*, in order to assimilate information for local adaptation. Software specialists can sometimes experience challenging *cultural discontinuity* in the field work, the art of which can decisively influence the quality of the application.

### 3. A Conceptual Framework

This paper suggests a conceptual framework of capturing the business process requirements from the perspective of a practitioner. Empirical reality shows that *locational iteration* is always indispensable. The software requirements in a multinational environment must be captured centrally and locally. This important software-related activity takes place in form of a *central-local-central loop*. First, Business process information is elicited, analysed and elaborated at the central headquarters. Second, requirements engineers must validate the centrally documented requirements at the national operational sites where the application will ultimately be deployed. Finally, the adapted, changed and added information shall be satisfactorily verified centrally and agreed upon by local and central product planers and decision makers. Undoubtedly, both hard and soft skills of the software engineers are required in this technically complex and organisationally sensitive process. The requirements engineer can certainly apply the concept of central-local-central loop repeatedly, thereby refining the understanding and description of the business process requirements. However, business practice has taught the practitioners repeatedly that time and budget is always in shortage. It is the reasonable applicability but not the overall perfection that dominates the thinking of the top managers in such requirements projects. That is why the requirements engineer should always try to capture the maximal information and underlying logics in each meeting and talk, particularly when the field research has commenced. To hope that there will a next round frequently turns out to be futile.

The first work products of central workshops can typically be the *activity diagrams* and/or *use cases* of the firm’s business processes. The author recommends the use of activity diagram at the beginning, because it is more time-saving and easier to use and understand. Many software engineers use to think in and work with Unified Modelling Language (UML) as a standard platform to map the business processes for requirements analysis purposes. However, in order to gain the necessary skill, potential analysts have to take part in formal trainings. Furthermore, commercial UML tools equipped with regular upgrades and updates are likely to be expensive. Experience in the practice indicates that Microsoft Excel, on the contrary, is available to most SMEs and can also

adequately serve the purpose of mapping the business processes. Whichever tool is used, a number of essential aspects must be taken into account to sufficiently decipher important information in business processes.

A *role* is a class of system users performing the same functions in the business processes. The description in the activity diagram needs to unambiguously define the role conducting a certain *activity* (process step). If necessary, detailed explanations can be given to each activity, in order to map the essential content of the activity. It is highly recommended that the *system-related activities* are highlighted. By doing so, the business specialists and the software engineers can subsequently defined the data fields of *system inputs and outputs* at a particular system process step. It is beneficial that these inputs and outputs are streamlined, e.g. in form of a *data dictionary*, at a later stage to make the data structure of future application more meaningful. *System printouts* need to be indicated, analysed and carefully defined at relevant process steps. If an activity involves both system-related and non-system sub-activities, the requirements engineer should differentiate these sub-activities by using different coding in the activity diagram (e.g. different colour or different font). System designer and developer will benefit from the precision and richness of the requirements description in a later stage extensively. Furthermore, if the use of an *external hardware* (e.g. a POS terminal is not necessarily a party of a restaurant billing system) exists at a certain process step, its involvement, its handling by the role and its possible printouts are ought to be depicted clearly.

As stated above, based upon local data, centrally gathered information must be supplemented and changed to reflect the business reality more accurately in the respective national marketplace. Sometimes, it may be expedient to see the central information as the way the central planners perceive how the business *should* be done and the local data as the way the local product managers think how the business *is* done. Informational imperfection and constraints caused by legacy software systems have partially contributed to these different viewpoints.

*Observation* may be the least resource-consuming methodology for requirements elicitation. As a typical field methodology for requirement elicitation it is recommended widely in the literature (e.g. Wiegers, 2003: 114). The field experience of the author shows that while the business processes are observed, the requirements engineer must avoid turning up as a controller or auditor, and thereby effectuating possible process distortion. The operational staff should be informed about the purpose of the observation transparently and work on the business process as usual and in a collaborative attitude. The requirements engineer should familiarise himself with the available information of the business process in advance, in order to ensure efficiency of the observation. If necessary, the requirements engineer should also collect and study the system printouts (e.g. reports, vouchers, slips) of the real-life business in advance. With the knowledge of centrally collected information, the requirements engineer is able to judge, in detail, the sufficiency and correctness, and notify the inadequacy of the existent data quickly. Nevertheless, observation only tracks the normal business procedure and does not reveal exceptional business scenarios in most cases. Although these scenarios only occur occasionally, they can have important implications for systems. Software professional are aware of the fact that a significant portion of time expenditure in the software development process deals with exceptional business cases.

A special form of observation is the so-call *apprenticing*. On the one hand, the operational employees does not feel observed because the trainer-trainee relationship make them more confident and less reluctant to completely show thorough business practice, on the other hand, the requirements engineer can gain a more deeper glance at how the business is done in reality, and thereby understanding more precisely what the underlying logic at each process step is. However the author concurs with Rupp (2007: 122) that apprenticeship unveils as little information of exceptional business cases as observation does and it is not appropriate for critical business processes where trainee's errors during the business operation may cause financial or other detrimental consequences (e.g. an error input in a logistic system may cause costly corrections afterwards).

This paper also recommends *online-interview* as the third useful methodology that should be combined with the above mentioned two methodologies. Requirement engineers should harness time gaps effectively to ask context-free questions as they observe or are apprenticed to process trainers. Questions are asked in the real-time mode and should be formulated at a high level without narrowing down the options of possible answers. Purposeful questions can exploit business process requirements in form of problem statements and solution suggestions put forward by the operational staff or product champions. Field practice also shows that process requirements that are more difficult to be captured, including exceptional business cases, operational risks and fraud potentials, system pitfalls and defects, and operational constraints grounded on insufficient functionality provision of the legacy system, can be more adequately coped with if online-interviews are applied additionally in the local requirements elicitation process. Notably, at the local operational sites, the requirements engineer is also typically dealing with other languages, other cultural behaviours and symbols. *Intercultural competencies*,

such as ability to communicate, capability to understand and act correspondingly in a multicultural context (empathy), are additional reasons why the demand of hard and soft skills on requirements engineers have become reasonably high.

Information elicited at the local market must be notified and written down preferentially immediately after observation, apprenticing and online-interview taking place. After the requirements engineer has returned to the central headquarters, the data collected should be reviewed and compared with the previously centrally mapped information and differences must be explained. During this reflection phase, the requirements engineer should always try to negotiate, intermediate among and in communication with central and local stakeholders. The result of this endeavour is to build a third activity diagram packet which is based upon the previously centrally mapped information and locally obtained field data. This packet should again describe how the business processes *should* be conducted, but this time it is much less theoretical and rich reality data has been added to the knowledge base.

The essential outcomes of the requirements elicitation and analysis therefore include three “business process maps”, resulting from the activities conducted at the corporate centre, in the local marketplace and again at the central headquarters. If the central-local-central loop is performed properly, the maps are able to capture business information systematically. The quality of the activity diagram packets may be effectively assessed by the extent to which the requirements information unveils the fundamental software design components, including objects (relationship between the objects), their services, constraints and even partial mini-specifications (Pressman, 2007: 128-131).

#### 4. Conclusion

A number of system-related processes, including product definition, decision making, requirements elicitation, elaboration and documentation, take place both centrally and locally in a multinational context. First, Business process information is elicited, analysed and elaborated at the central headquarters. The first work products of central workshops can typically be the activity diagrams. Second, requirements engineers must validate the centrally documented requirements at the national operational sites where the application will ultimately be deployed. The requirements engineer must observe, learn, ask and capable of carrying out intercultural tasks in the field work. Observation and apprenticing should form the basis for generating effective questions for online-interviews. Although other field methodologies exist, a balanced combination of the methodologies mentioned in this paper can become satisfactorily effective and, in terms of time and financial resources, more efficient for requirements elicitation. Field elicitation should also cover data related to exceptional business cases, operational risk and fraud potentials. Finally, the adapted, changed and added information shall be verified centrally and agreed upon by local and central product planners and decision makers. Requirements analysis and documentation is probably the most time-consuming and costly learning activity in software development cycle for decision makers, product champions, requirements engineers, software designers, coders and testers, but it is the key of a high-quality software package. It is therefore worthy paying the tuition fee.

#### References

- Pressman, R. S. (2009). *Software engineering: A practitioner's approach*. (7th ed.). New York, NY: McGraw Hill.
- Rupp, C. (2007). *Requirements-Engineering und Management: Professionelle, iterative Anforderungsanalyse für die Praxis*. (4th ed.). München/Wien: Carl Hanser Verlag.
- Wieggers, K. E. (2003). *Software requirements*. (2nd ed.). Redmond, Washington: Microsoft Press.