The Case for Cooperative Requirement Writing

Vincenzo Ambriola and Vincenzo Gervasi

Dipartimento di Informatica, Pisa, Italy
{ambriola,gervasi}@di.unipi.it

Abstract. Ensuring high quality in requirements is difficult and costly, and often requires extensive study on the part of the requirement engineer (RE) of the applicative domain. An interesting alternative to the traditional approach is the adoption of a cooperative requirement writing (CRW) process. In CRW processes, several stakeholders, both technical and non-technical, are directly involved in drafting the requirements for a system. Each participant can then contribute his or her own competences without the need for costly knowledge transfer activities.
Due to the different backgrounds of the various stakeholders, CRW processes need good support from appropriate automatic tools to be effective. In this paper, we sketch the needs of CRW processes and describe CIRCE, a Web-based CRW environment targeted mainly at supporting cooperation on natural-language requirements.

1 Cooperation in requirement processes

Exploiting cooperation among the stakeholders in a software project is probably the best means to capitalize on available knowledge and competence without long, costly, and error-prone knowledge transfer processes. In order to allow smooth cooperation, however, a number of problems have to be overcome. One of the most difficult lies in what we call *communication impedance* among the stakeholders, stemming from their different backgrounds, habits, languages, and goals. Other relevant problems are the handling of conflicts and inconsistencies in specifications, and the integration between cooperative requirement analysis and subsequent phases of the development process.

These problems can be solved, at least partially, by using a CRW supporting environment providing some basic capabilities, namely the capability of expressing and examining requirements through different representations (and different models), the capability of performing inter- and intra-model validation checks, and the capability of tracing metrics and other data relevant to the overall management of the development process.

In a traditional setting, cooperation in requirement processes is limited to the participation of the customer as a passive information source from which a RE extracts relevant knowledge. This investigation, often referred to as "elicitation", is usually quite difficult. It is highly probable that some important fact is missed, or that errors are injected; as a consequence, customer validation is needed, and thus requirements must be readable by such non-technical participants.

The effect of direct user participation in requirement analysis has been already considered in a number of works, e.g., [2–6], with a consensus on the fact that projects with greater user involvement have been more successful than those with weaker participation. It is surprising, then, that in the vast majority of cases there is no tool assistance available *to the user*. Even in the most advanced settings, the RE has to act as interface between the customer and some supporting environment. This solution is quite unsatisfactory, in that the RE has to translate back and forth between the customer's and the system's views of the problem, while making his own mind in the process.

Although this inconvenience may be acceptable in simple cases, this style of interaction becomes unbearable as the number of non-technical participants in the process grows. In fact, we advocate that the positive effects of direct involvement should be extended to as many stakeholders as possible, not limiting oneself to consider only a single customer and a single RE. In this kind of process, each stakeholder is supported by automatic tools, that complement (not substitute) direct interaction among stakeholders.

Requirements go through two basic operations: they are first *expressed* by some stakeholder, and then they may be *understood* by some other stakeholder. As soon as requirements are expressed, they assume a syntactic and lexical form (in some arbitrary language and notation) that encodes their intended semantics. It is worthwhile to notice that this encoding can well be imprecise: actually, the most common language for expressing requirements, natural language (NL), is typically quite fuzzy when it comes to precise semantics; the same holds for other informal or semi-formal notations that are in widespread use.

Ideally, the semantic content of a requirement should be independent from a particular language, and in most cases can be communicated by different means (e.g., an entity-relationship diagram can be "read aloud"). On the other hand, the lexical content (i.e., the choice of names for things) depends on the real-world entities found in the domain, and can usually be moved among different languages and notations (e.g., "user" can be used both as a noun in a NL requirement, or as a label in a data flow diagram). Correct and complete communication among the stakeholders occurs when the semantic content of a requirement, tied to the problem domain via its lexical content, can be transferred without degradation from "the head" of a stakeholder to "the head" of a different one.

This observation takes us to the main requirement that a CRW supporting system should satisfy: it must be able to successfully translate requirements forth and back among different models, languages and notations. Naturally, a full supporting environment has an opportunity to perform several other useful functions, e.g., the formal validation of models, the extraction of product and process metric data, the handling of conflicts. However, in this paper we concentrate on the issue of providing multiple views on the requirements.

2 Circe: supporting CRW processes

In order to experiment with an effective CRW setting, we developed a prototype environment called Circe. At the architectural level, Circe can be described in terms of four classes of components, communicating through a shared blackboard:

- a centralized *semantic repository* stores the (possibly inconsistent) knowledge extracted from the requirements expressed by the users of the system (i.e., by the stakeholders);
- a number of *modelers* use the information stored in the repository to build *models* from the requirements; higher-level modelers can also exploit models built by lower-level modelers;
- *translators* provide an effective representation for the models, turning them from abstract entities into text, tables or graphs;
- last, these representations are output in an appropriate *interaction environment* that takes care of showing them to the user and, in some case, allows the user to edit them. In such cases, the whole series of transformation can be taken in the opposite direction so that user editing is reflected by an updated semantic repository.

Our current prototype includes a translator called Cico that handles NL input and output [1], and several modelers for data flow diagrams, entity-relationships diagrams, aggregation and derivation diagrams, communication diagrams and a limited dynamic behaviour representation. A user of the system can freely move among all these different views on the requirements he or she (or even a different user altogether) has expressed.

Circe also includes a number of modelers devoted to the extraction of metric data and to the validation of the requirements, both at the representation level (e.g., linguistic correctness of the NL requirements, ambiguities, redundancies) and at the model level (e.g., well-formedness of data flow diagrams).

As interaction environment we opted for a standard Web browser complemented by some custom Java and JavaScript code. This choice has several advantages; among them, the immediate availability of the interaction environment on practically every computing platform, and the opportunity to make good use of the layout control offered by HTML. Another advantage that has proven to be of paramount importance in practical use is the seamless integration of the various representations with widespread personal and group communication methods: stakeholders can exchange their views via email or discussion groups, and can easily publish snapshots or even "live" diagrams on the Web. This integration greatly simplifies direct interaction among the stakeholders, and this in turn favours conflict resolution and composition.

3 Conclusions

Cooperative requirement engineering is a promising methodology, but it cannot deliver its promises without a flexible and practical supporting environment. Previous work in this area has concentrated in providing support to the *requirement engineer*, but not to other classes of stakeholders. In view of the substantial number of stakeholders that should participate in the requirement definition in a true CRW process, these approaches turns out to be too weak.

We believe that with environments providing easy integration of different views, like our prototype described in Section 2, and by effective use of existing inter-personal communication facilities, cooperative requirement writing can prove not only more complete, but also simpler and less expensive than the traditional, labour-intensive mediated approach.

References

1. V. Ambriola and V. Gervasi. Processing natural language requirements. In *Proceedings of ASE 1997*, pages 36–45. IEEE Press, 1997.
2. P. G. Brown. QFD: Echoing the voice of the customer. *AT&T Technical Journal*, 70(3):18–32, Mar.-Apr. 1991.
3. A. Cucchiarelli et al. Supporting user-analyst interaction in functional requirements elicitation. In *Proceedings of the First Asia-Pacific Software Engineering Conference*, pages 114–123, Los Alamitos, California, Dec. 1994. IEEE CS Press.
4. J. M. Drake, W. W. Xie, W. T. Tsai, and I. A. Zaulkernan. Approach and case study of requirement analysis where end users take an active role. In E. Straub, editor, *Proceedings of the 15th International Conference on Software Engineering*, pages 177–186, Baltimore, MY, May 1993. IEEE Computer Society Press.
5. K. El Emam, S. Quintin, and N. H. Madhavji. User participation in the requirements engineering process: an empirical study. *Requirements Engineering Journal*, 1(1), 1996.
6. M. Keil and E. Carmel. Customer-developer links in software development. *Communications of the ACM*, 38(5):33–44, May 1995.