CARE: A Computer-Aided Requirements Engineering Tool for Problem-Oriented Software Development

Guoyuan Liu
College of Computer Science and Information Technology
Guangxi Normal University
No. 15 Yu Cai Road, Guilin,
Guangxi 541004, China
153123439@qq.com

Zhi Li*
College of Computer Science and Information Technology
Guangxi Normal University
No. 15 Yu Cai Road, Guilin,
Guangxi 541004, China
zhili@gxnu.edu.cn

Zhaofeng Ouyang
College of Computer Science and Information Technology
Guangxi Normal University
No. 15 Yu Cai Road, Guilin,
Guangxi 541004, China
751194151@qq.com

Abstract—**This paper presents a tool to help software design in the development process. This software prototype will promote further development of Problem Frames framework (PF) and drive it to maturity, i.e., from theoretical research to practical applications.**

Keywords-Problem Frames (PF);Problem diagram;Computer-Aided Requirements Engineering (CARE)

I. INTRODUCTION

Software requirements engineering plays an important role in software development projects. So how to conduct the practice of requirements elicitation, modeling, analysis and transform the results into correct software specifications is a key factor contributing to the successes of software development projects. We designed and implemented a prototype based on the theoretical foundations and principles of a problem-oriented requirements modeling framework–Jackson's Problem Frames approach [1,2] (PF for short). PF has been regarded as one of the major requirements engineering approaches for assisting system analysts in structuring software development problems. They deploy problem diagrams for capturing and describing important contextual information for the software solutions to be built.

Over the years, there have been many extensions and advancements in Problem Frames research. For example, Hall *et al* have proposed Problem-Oriented Software Engineering (POSE) as a theoretic framework for software development [3,4]. Other researchers have made many theoretical extensions to PF and applied them to requirements analysis and reasoning for safety-critical systems [5,6], and identifying reliability concerns [7]. However, how to embed the PF framework in a software development practice remains an open problem.

In this paper, we present a computer-aided requirements engineering (CARE) tool for system analysts to use in the requirements analysis phase of software development. This work is motivated by the challenges and difficulties faced by many software development practitioners when communicating, modeling, analyzing and elaborating requirements in the early phase of a software development project. The tool not only can animate a visual transformation of requirements models, but also provide a vehicle for an automated textual transformation of requirements statement, accordingly.

II. PROBLEM MODELLING AND ITS TRANSFORMATION RULES

The PF is further development of Jackson's work on JSP (Jackson Structured Programming) and JSD (Jackson System Development) [9]. Its basic tenet is that in requirements analysis phase, we should first understand the contextual environment in which the problem occurs, before giving any software solution. The rationale behind it is that most modern software systems inevitably interact with their surrounding environment to serve their ultimate purposes – satisfying the problem owner's needs. PF deploys problem diagrams – a visual modeling notation as a way of concretizing the problem owner's needs or wishes into observable or measurable phenomena [2]. The following is a typical problem diagram describing an insulin injection control system for diabetes.

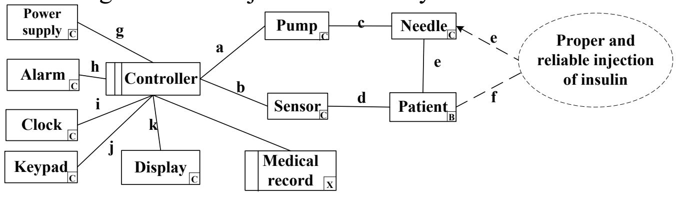

Fig. 1 Insulin injection control system for diabetes

Figure 1 shows that problem diagrams are extended context diagrams, with the following extensions:

- Rectangles with double stripes represent the computerized machine domain on which the software runs, e.g. the Controller domain;
- Application domains are represented by rectangles, which represent physical equipment (e.g., the Pump

*Zhi Li is the corresponding author. The research was supported in part by the Natural Science Foundation of China under Grant No.61262004, and the authors' joint research is sponsored by the National Science Foundation of Guangxi Province under Grant No.2012GXNSFCA053010, and the Guangxi Scientific Research and Technological Development Project under Contract No.(Gui-Ke-He)1347004-22. Zhaofeng Ouyang is sponsored by University Innovation and Startup Project (No. 201410602104).

domain and the Sensor domain - the sub-labels with the symbol "C" represent "causal", which means their properties or behaviors are predictable); or living beings (usually people, e.g., the Patient domain – the sub-label with the symbol "B" represents "biddable", which means the domain has his own freewill but can follow orders or pre-determined rules after being trained or notified);

- Rectangles with a single stripe represent domains which can store information, e.g., the Medical record domain - the sub-label with the symbol "X" represents "lexical", for instance, USB disks or other data storing devices;
- The solid lines labeled "a", "b", "c", "d" , "e" , "f" , "g", "h", "i", "j", "k", "l", represent observable or measurable phenomena shared between domains;
- The dotted oval labeled "Proper and reliable injection of insulin" in Figure 1 represents the requirements. The text is a statement of needs or wishes of the problem owner (the diabetic patient in this case). The application domains that the statement concerns are connected with the oval by dotted lines – the Needle domain and the Patient domain; the label "e" and "f" represent the observable or measurable phenomena (either internal to the domains or external phenomena of the Needle and Sensor domain shared with other domains);
- The dotted rectangle which is connected with the Patient domain represents the Patient domain's properties, i,e, "f" represents the phenomena "the patient's blood sugar reaches abnormal level", "d" represents the phenomena "the sensor detects the patient's blood sugar reaches critical threshold level", then f->d represents a cause-and-effect relationship that is the property of the Patient domain

From Figure 1, we can observe that PF represents a broad perspective on software development problems, in which the hardware, software and relevant application domains should all be treated as first-class citizens in the modeling process. Solving this kind of problems is a process of reasoning and moving from the dotted oval and the dotted lines towards the controller machine domain. In PF modeling, this process is known as problem transformation. In order to implement this transformation, we have defined three classes of transformation rules, namely the "cause-and-effect substitution rules", "switching [domain's] perspective rules", and "removing [unconnected] domain rules", see [8] for more details.

The contribution of this paper is that we have developed a computer-aided tool to implement the three classes of rules for problem transformation.

- The "cause-and-effect substitution" rule: since the application domain's properties in PF modeling mainly describe causal-and-effect relationships in a form like "a->b", we can substitute "cause" events with "effect" events or vice versa. The algorithm of this rule can be described by the following pseudo-code (variables are in italics):

```
foreach (i in D.event) //search all events of Domain D
  if (i == a) //find the event a
    foreach (j in R.event) //search all events of requirement R
      if ((a -> b)is in D.propety) //if a->b belongs to D's property
      R.event [j]=a; //substitute a for b
```

- The "switching perspectives" rule: since adjacent domains share exactly the same set of phenomena, the requirement statement involving the shared phenomena can be switched from the viewpoint of the receiving end to the sending end, and vice versa. The algorithm of this rule can be described by the following pseudo-code:

```
foreach (i in D.event) //search all events from domain D
   foreach (j in R.event) //search all events of requirements R
         if (i == j) //if a equals b
            {  D'.lines++; //add a new line to domain D' of R
               D.lines--;   //delete the old line connected to domain D
            }
```

- The "removing domain" rule: after the requirements are connected with the computing machine domain, all references and constraints of the requirements are on the computing machine, therefore, a requirements engineering problem becomes a pure programming problem, thus all other diagrammatic elements can be deleted. The algorithm of this rule can be described by the following pseudo-code:

```
if (R.ID is in Controller S.connectedID) //if requirement R is connected to the
                                          //Controller
   for (int i=0; i <Domain.count; i++) //search all the domains
   delete (Domain[i]); //delete domain
   for (int i = 0; i> Model.count; i++) //search all the models
   delete (Model[i]); //delete model
```

III. TOOL OVERVIEW AND IMPLEMENTATION

A. Tool Overview

Figure 2 shows the overview of CARE, which consists of three main modules: the Application Domain module, the Transformation module and the Requirement module.

By combining the three modules we can draw a full problem diagram, in order to system analysts in requirements elicitation and analysis. More importantly, the transformation module can enable diagrammatic transformation, which is the core innovative part of the tool. So a computer-aided requirements engineering (CARE) prototype system has been designed. Its aim is to try to get prospective user involved in the process of requirements elicitation, modeling, analysis and transformation as early as possible and as visually engaging as possible. Another motivation for developing the prototype is to empirically evaluate the feasibility and practicality of the PF modeling and transformation framework.

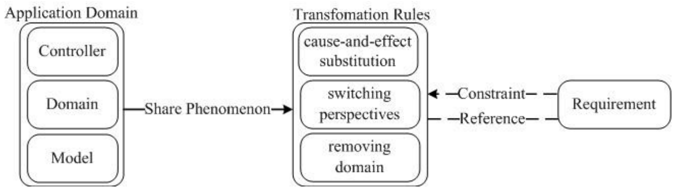

Fig.2 Design overview of CARE

B. Implementation

Since symbols and diagrams can be recognized at a glance and quite often, good representation can assist intuitive understanding of the meanings of visual and diagrammatic modeling. In addition, they can highlight some complex and important relationships among different entities without verbose or ambiguous texts.

Our prototype tool is designed to facilitate system analysts in drawing and editing problem diagrams, inputting a semi-structured textual statement of requirements, as a way of modeling requirements and relevant contexts. In addition, the tool also allows for visual transformation of the model and a textual transformation of requirements, which we believe can simulate or animate system engineers' process of reasoning while trying to solve engineering problems. From a requirements engineering perspective, this tool supports retaining requirements traceability, which is essential for requirements engineering [10]. The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.fig.3 shows the interface of our CARE.

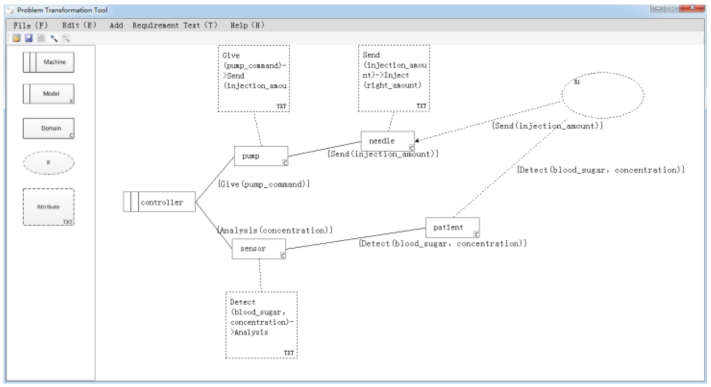
Problem Transformation Tool
File (F)
Edit (E)
Add
Requirement Text (T)
Help (H)
Machine
Model
Domain
R
Attribute
TXT
Give (pump_command)->Send (injection_amou
Send (injection_amount)->Inject (right_amount)
R1
needle
pump
controller
patient
sensor
{Send(injection_amount)}
{Give(pump_command)}
{Detect(blood_sugar, concentration)}
{Analysis(concentration)}
Detect (blood_sugar, concentration)->Analysis

Fig.3 CARE: the Computer-Aided Requirement Engineering tool

IV. EVALUATION SETUP

We evaluated the usability of the CARE tool under both MS Windows and Android with many practical examples. We also report on the results of an initial empirical evaluation of the approach based on the prototype problem transformation tool. A total of 47 students took part in the evaluation, and the results are shown in table I.

TABLE I. PARTICIPANTS' ANSWERS TO THE QUESTIONNAIRE

How helpful is the tool to you in understanding Problem Frames?			
Extremely helpful	very helpful	somewhat helpful	not helpful
15	20	7	5

V. CONCLUSION

In this paper, we present a computer-aided requirements engineering tool. This is part of the second authors' long-term research towards moving PF closer to practice [8,12,13]. Currently, an early version of the prototype can be downloaded from the website http://www.se.gxnu.edu.cn/tooldemo. There are also several versions of the tool on mobile platforms, for example, an Android version and a Windows Phone version of the tool are also available on the website. When all tools are matured enough, we plan to empirically evaluate them by embedding PF theory and the CARE tools in realistic software development projects.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for the valuable comments, which helps improve this paper.

REFERENCES

[1] Jackson M. Software requirements and specifications: a lexicon of principles, practices and prejudices［M］. Boston: Addison-Wesley, 1995.

[2] Jackson M. Problem frames: analyzing and structuring software development problems［M］. Boston: Addison-Wesley, 2001.

[3] Hall G H, Rapanotti L, Jackson M. Problem-oriented software engineering: a design-theoretic framework for software engineering, 2007[C]//Proceedings of the 5th IEEE International Conference on Software Engineering and Formal Methods.Los Alamitos : IEEE CS Press, 2007 : 15-24.

[4] Hall G H, Rapanotti L, Jackson M. Problem-oriented software engineering: solving the package router control problem[J].IEEE Transactions on Software Engineering,2008,34(2):226-241.

[5] Strunk E A, Knight J C. The essential synthesis of problem frames and assurance cases[J]. Expert Systems, 2008, 25(1): 9-27.

[6] Mannering D, Hall J G, Rapanotti L. Towards normal design for safety-critical systems[C]//Fundamental Approaches to Software Engineering. Springer Berlin Heidelberg, 2007: 398-411.

[7] Yin B, Jin Z, Li Z. Reliability concerns in the Problem Frames Approach and system reliability enhancement patterns[J]. Jisuanji Xuebao(Chinese Journal of Computers), 2013, 36(1): 74-87.

[8] Li Z, Hall J G, Rapanotti L. On the systematic transformation of requirements to specifications[J]. Requirements Engineering, 2013,(doi: 10.1007/s00766-013-0173-8),online first article.

[9] Berry M D. Software requirements and design: the work of Michael Jackson[J].ACM SIGSOFT Software Engineering Notes,2011,36(2):39-40.

[10] Jane Cleland-Huang, Orlena Gotel, Jane Huffman Hayes, Patrick Mäder, Andrea Zisman:Software traceability: trends and future directions. FOSE 2014: 55-69

[11] Sommerville I. Software Engineering 9th Edition[M]. Boston:Addison-Wesley, 2011.

[12] Rapanotti L,Hall G J, Li Z. Deriving specifications from requirements through problem reduction[J].Journal of IEE Proceedings-Software,2006,153(5):183-198.

[13] Li Z,Hall G J,Rapanotti L, On the construction of specifications from requirements[C]//Procs of the 14th Workshop on Requirements Engineering. Rio de Janeiro, Brazil: BDBComp, 2011: 431-442