Query Responsive Awareness Software: Inventory Control Case Study

Brian J. d'Auriol
Department of Computer Engineering
Kyung Hee University, Korea
dauriol@oslab.khu.ac.kr

Pramod Chikkappaiah
CCIG*

Weiwei Yuan
Department of Computer Engineering
Kyung Hee University, Korea
weiwei@oslab.khu.ac.kr

Sungyoung Lee
Department of Computer Engineering
Kyung Hee University, Korea
sylee@oslab.khu.ac.kr

Young-Koo Lee
Department of Computer Engineering
Kyung Hee University, Korea
yklee@khu.ac.kr

ABSTRACT

Query Responsive Awareness Software (QRAS) is proposed in this paper as a run-time software representable model that incorporates semantic attributes into the software. Specifically QRAS: (a) provides for the representation of properties of the software and its application environment, (b) is compatible for distributed systems and in particular, smart collaborative object systems, and (c) provides human and/or automated query and query-responsiveness. QRAS is based on the Geometric Representation of Programs (GRP) model. A case study based on a simplified inventory control system together with a simulation of various queries concludes a feasibility study of the proposed approach.

1. INTRODUCTION

Software is ubiquitous in modern day high-technology human population centers. These environments typically include technology infrastructures with wired and wireless access available in urban and rural areas where software embedded technologies are economically available. Such software can be augmented with sensor capabilities and more local processing of the sensor-acquired data thereby leading to 'smartness' in the software. Smart collaborative objects exemplify this vision of ubiquitous computing: they link everyday things with information technology by augmenting ordinary objects with small sensor-based computing platforms. There are nowadays many examples of 'smart'-enabled technologies (see for example [2, 7, 9]). The notion that such objects incorporate sensors together with perception algorithms is now established. And approaches for the latter have been developed, for example, mobile agent-based software, context reasoning, abstractions of sensor data, etc.

In addition, the software used in these technologies differs from traditional software. Often, these technologies are mobile, light-weight and have memory, processing capability and power limitations. Software developed for these technologies also needs to support such light-weightiness. Furthermore, the software often partakes in real-time distributed applications, exchanging both information and control like decisions. Traditional software environments tend to be more monolithic and static in structure.

This paper introduces awareness software as an alternative light-weight software for 'smart'-enabled technologies. Awareness software represents 'smartness' by a geometrically structured group of semantic descriptions that describe aspects of the software. *Query Responsive Awareness Software* (QRAS) is software that has two special components: a) awareness software, and b) the capability to respond to queries about its awareness. QRAS software is proposed to assist in the self-management of software distributions over large-scale heterogeneous 'smart'-enabled technologies. The formulation is based on the Geometric Representation of Programs (GRP) model, informally proposed by d'Auriol in [4, 5]. A case study of an inventory control system is presented in the paper. The GRP model is used to develop a 'smart' inventory control system. A simple simulation is presented that provides additional clarity about how such software behaves.

The rest of this paper is organized as follows. The next section, Section 2 develops the QRAS model. A discussion of related works appears in Section 3. Section 4 describes the inventory control case study. Conclusions are presented in Section 5.

2. MODEL DEVELOPMENT

The Geometric Representation of Program (GRP) model consists of three domains: the *computation* domain that

*Computation and Communication Integration Group, A World and International Research Group, http://www.ccig-research.net

represents the execution processes, the *data* domain that represents the data inputs and output of the computations, and the *awareness* domain that represents the semantics of the computations. This paper concentrates on the latter domain. These domains are integrated as illustrated in Fig. 1 and so both the computation and data domains appear in the subsequent case study analyses.

Semantic variables that abstract semantic information about the domain are identified during the software design stages. Let $a \in A$ denote an *awareness axis* that models such a variable. The dimension $|A|$ denotes the number of such axes. All axes are orthogonal but may be either dependent or independent. Specific values of these variables index integer coordinates in the awareness space, A-space. Each computation instance (from the computation domain of the GRP model) is mapped to such a coordinate; thereby, each computation instance is fully indexed by the awareness variables. However, sub-spaces of the A-space are formed by subsets of the awareness axes; these correspond with subsets of the semantic variables. Computation instances existing in the sub-spaces are partially indexed by the awareness variables. Constraints on the semantic variables are modeled as partitions on the A-space and its relevant sub-spaces, that is, conditional expressions partition half-spaces and an intersecting set of such expressions determine a bounding polytope enclosing a relevant set of integer points, and hence, a set of computation instances.

The proposed query model is based on the enclosed space given by all the partition constraints. Let `QRAS`$=(R_A, F(R_A))$ where R_A is some data structure representation of A and F is a set of methods that operate on R_A (i.e., an ADT, class object, etc.) and provide for query formulation, initiation and response. Information that queries may make consist of either or both awareness information and awareness state. The former describes the semantics of the awareness axes and awareness regions of the awareness space and its various sub-spaces. The latter describes particular state values that can be further associated with a single computation (i.e., a point in the awareness space) or with an awareness region. Queries are initiated from QRAS-enabled software and directed to other QRAS-enabled software. However, it is also possible to have a user-driven interface that allows user initiated query requests to be passed to the software.

Specifically, the following operations are defined.

- **Association**: Computation together with data instances are associated with specific values of the semantic variables.
- **Selection**: One or more computation and/or data instances are selected based on given specified values of the semantic variables.
- **Abstraction**: One or more specific values from one or more semantic variables, or, one or more semantic variable concepts (i.e. the variable name itself) are selected.

Figure 2 illustrates the overall operation of QRAS-enabled software. Four modules denoted by `A`, `B`, `C` and `D` are shown in a two-level software design. Standard `USE` relationships are expressed by the think arcs between the modules (for example, `A USE`s `B`). Modules `A` and `B` are QRAS-enabled as illustrated by the inscribed oval and therefore these modules can both initiate queries and respond to queries. The other modules, `C` and `D` can initiate queries to any of the QRAS-enabled modules, but themselves can not respond to any queries. In the figure, `B` is shown as also having a user-driven front-end that allows the user to initiate query and view the results.

Figure 1: Overview of the Geometric Representation of Programs (GRP) model.

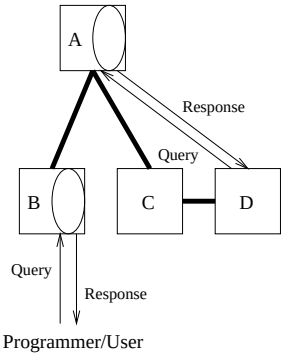


Figure 2: Overview of QRAS concepts: Modules `A` and `B` are QRAS-enabled; module `D` can query `A`, and, the user can query `B`; the three pairs of modules `A` and `B`, `A` and `C`, and, `C` and `D` engage in `USE` relationships.

Scalability of the model is dependent on two factors, first, the number of dimensions and individual states in the A-space, and second, the implementation of the QRAS-enabled software. In the former, although there may be many semantic variables describing a complex software application, only a relatively few are specifically associated with particular specific code fragments. A relatively low dimension A-space is associated with such a code fragment. Thus, the representations are expected to consist of multiple connected low dimensional A-spaces. In the latter, appropriate interval-based or query-based approaches may reduce the software complexity and run-time overhead.

3. RELATED WORKS

In fact, the formulation of computation instances and enclosing polytopes correspond with the well-known iteration space model found in compiler and parallelizing compiler theory (see for example [11,15]). However, an important dif-

ference is that the definition of the axes and the formulation of the representation convey the semantics of the problem domain. In [5], this approach is classified as *non-linguistic-carried* to distinguish it from the linguistic-carried classification of the iteration space model. The inclusion of semantic variables, that is, the inclusion of the awareness domain of the GRP model, augments and extends from the iteration space model.

There are also similarities between the GRP model and the Conceptual Space Model (CSM) [6]. Specifically, the awareness axes are similar to the quality dimensions in the CSM. However a major difference is that the GRP model supports awareness representation in software whereas the CSM supports conceptual representations: hence, operations in the GRP model relate more with the classification, organization and structuring of concepts associated with computations whereas operations in the CSM relate more with conceptual domain representation and manipulation. For example, similarity of objects in the CSM can be defined in terms of a distance function defined over the space given by the quality dimensions, however, there is no direct equivalence of similarity in the GRP model where the closest analogy would be the enclosed region of computations all of which are selected by a common set/range of indexing values in the awareness space.

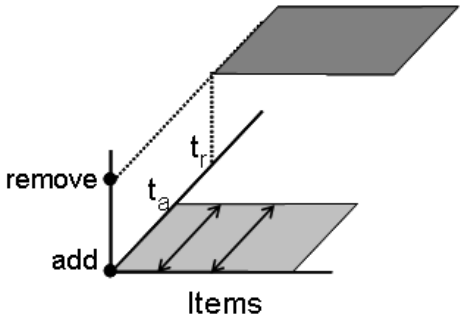
Other related approaches in the literature include concept formalization and software reflection. Concept formalization provides semantic and syntactic descriptions of standardized routines to support generic programming, see for example [17]. Software reflection is defined as: a system that is able to reason about itself [13]; "the ability of a program to manipulate as data something representing the state of the program during its own execution." [12]; and the introspection and intercession performed by an agent about itself [1]. Reflection is included in a number of programming languages, e.g., Java [10,14]. (A Wikipedia page lists a number of languages categorized as 'Reflective languages' [16].) More recently, reflection has been considered in run-time environments [1] and middleware [8].

Our work in this paper is motivated by representing parallel and distributed programming semantics in light-weight software that is suitable for smart-enabled technologies. Common parallel languages are not reflective and Java's reflective capabilities may be too heavy for this purpose. We therefore explore a mechanism in this paper in order to represent the semantics useful for programs in this context. We are not concerned about supporting generic programming or intercession via reflective methods. Our work on awareness software complements the existing literature related with introspection via reflective methods.

4. INVENTORY CONTROL CASE STUDY

The definition of inventory control in [3] is adopted here: "Inventory is the stock of any item or resource used in an organization. An inventory system is the set of policies and controls that monitor levels (threshold value) of inventory and determine what levels should be maintained, when stock should be replenished, and how large orders should be." These items or resources can include: raw materials, finished products, component parts, and supplies.

(a) 3D Awareness space: $A = \{a_o = \text{operations}, a_i = \text{items}, a_q = \text{threshold}\}$.



(b) Illustration of the bounded regions given by the partitions: add: $Q(i) < t_a$ and remove: $Q(i) > t_r$ for all items in the inventory.

Figure 3: Inventory control awareness space (axes displayed rotated such that (i,j,k) corresponds with vertical, horizontal, depth)

Let `add`, and `remove` be operations that can be performed on each of the inventory items: the addition of q quantity to item i provided that the existing inventory quantity $Q(i)$ is less than a threshold t_a, and similarly, the removal of q quantity from item i provided that the existing quantity $Q(i)$ is greater than a threshold t_r.

$$\text{add:} \quad Q(i) \leftarrow Q(i) + q \quad \text{if } Q(i) < t_a \tag{1}$$

$$\text{remove:} \quad Q(i) \leftarrow Q(i) - q \quad \text{if } Q(i) > t_r \tag{2}$$

Here, let a_o denote the concept of add and remove operations, a_i denote the concept of inventory items and a_q denote inventory threshold quantities. Hence: $A = \{a_o, a_i, a_q\}$ and the resulting awareness space of dimension $|A| = 3$ models the necessary concepts in the inventory control problem. The condition $<$, respectively, $>$ specifies a partition on A; more precisely, a partition on the sub-space $A^s = \{a_q\}$. Figure 3(a) illustrates the A-space. Figure 3(b) illustrates the partitions $Q(i) < t_a$ and $Q(i) > t_r$ and the resulting two horizontal planes for the `add` and `remove` operations respectively. The hexahedron formed by these two planes and the four vertical planes connecting the corresponding sides of the horizontal planes represents the enclosed A-space.

Each point in the enclosed A-space region has the associated computation instances mapped to it:

$$\begin{aligned}((\texttt{add}, i, j) \quad &: \quad Q(i) \leftarrow Q(i) + q \quad \text{for} j < t_a \ldots \\ (\texttt{remove}, i, j) \quad &: \quad Q(i) \leftarrow Q(i) - q \quad \text{for} j > t_r)\end{aligned}$$

for all i items in the inventory.

The meaning of a computation instance is given by selections of the awareness axes, e.g., $(\texttt{add}, i, j)$ selects the instance $Q(i) \leftarrow Q(i) + q$. Partial indexing selects multiple instances that are generalized by the associated semantic axis information, e.g. $(\texttt{remove}, ,)$ selects all the computation instances in the `remove` plane (see Fig. 3(b)) and thereby fully represents (2).

Equations (1) and (2) also form the basis for a computer program code fragment as illustrated in Fig. 4 for the addition operation. Here, Line 1 selects on the a_o dimension and Line 2 selects on the a_q dimension; and both specify partitions, `add` and $Q(i) < t_a$, respectively. For a particular given i, this corresponds with a particular arrow shown in the lower light-gray plane in Fig. 3(b). However, for all items in the inventory (i.e., let the code fragment be wrapped within a repetition), this corresponds with the entire light gray lower region in Fig. 3(b).

```
1.   if ( add-operation ) then
2.      if ( Q(i) < ta ) then
3.         add q to Q(i)
4.      endif
5.   endif
```

Figure 4: Code fragment associated with (1)

The association of selection by indexing on the awareness space forms the basis to consider various queries. Table 1 illustrates several queries that can be formulated in this example together with the nature of the response. Figure 5 shows the corresponding dialog from a simplified simulation of the inventory control. The simulation is implemented in Haskell and the code that represents the A-space is shown in Fig. 6. The simulation illustrates that the awareness capabilities as developed in this paper are feasible.

5. CONCLUSION

Query Responsive Awareness Software (QRAS) is proposed in this paper as software that (a) incorporates awareness and (b) can respond to externally generated queries. A simple query model is developed based on some of the ideas of the Geometric Representation of Programs (GRP) model that was informally proposed some years ago. The GRP consists of the three domains, awareness, computation and data. The earlier work on GRP focused on the computation domain, in particular, iteration spaces. This paper extends the earlier work to include the awareness domain. A case study based on a simplified inventory control system together with a simulation of various queries concludes a a feasibility study that shows the reasonableness of this approach. The QRAS model is intended for distributed applications and in particular, for 'smart-enabled' technologies as these have knowledge-processing requirements. The extension of the simulation to apply to general software applications domains and the further formalization of the GRP and QRAS models motivates future work.

```
Main> whatIsPurpose createInventorySpace
"operations items thresholds "

Main> whatAre "items" createInventorySpace
"chocolate candy jelly beans "

Main> whatAre "thresholds" createInventorySpace
"t_a t_r "

Main> whatIsComputation (0,0,0) computationInit
"Q(i) <- Q(i) + q"

Main> whatIsItemData (0,0,0) inventoryInit
1

Main> whatIsItemData (0,0,0) (add 0 10 (extractAxis "items"
createInventorySpace) quantityInit inventoryInit )
11

Main>
Main> howMany "dimensions" createInventorySpace
3

Main> howMany "operations" createInventorySpace
2

Main> howMany "items" createInventorySpace
3
```

Figure 5: Haskell simulation results corresponding with the queries in Table 1

```
-- Semantic Domain
type Semantics = String
type AxisElement = (Int, Semantics)
type Axis      = ( (Int, Semantics), [ AxisElement ]  )
type ASpace    = [Axis]

createInventorySpace :: ASpace
createInventorySpace = [
  ( ( 0, "operations"), [ (0, "add"), (1, "remove") ] ),
  ( ( 1, "items"),      [ (0, "chocolate"), (1, "candy"),
                          (2, "jelly beans")  ] ),
  ( ( 2, "thresholds"), [ (0, "t_a"), (1, "t_r") ] )
```

Figure 6: Haskell representation of the inventory A-space: the three inventory items are chocolate, candy and jelly beans.

Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITFSIP (IT Foreign Specialist Inviting Program) supervised by the IITA (Institute of Information Technology Advancement) and the MIC under the ITRC (Information Technology Research Center) support program supervised by the IITA (IITA-2006-C1090-0602-0002)

6. REFERENCES

[1] M. Ancona and W. Cazzola. Implementing the essence of reflection: a reflective run-time environment. In *Proceedings of the 2004 ACM symposium on Applied computing, (SAC'04:)*, pages 1503–1507, New York, NY, USA, 2004. ACM.

[2] M. Beigl and H. Gellersen. Smart-its: An embedded platform for smart objects. In *Proceedings of the Smart Objects Conference (SOC 2003)*, Grenoble, France, May 2003.

[3] R. B. Chase, N. J. Aquilano, and F. R. Jacobs. *Operations Management for Competitive Advantage*. McGraw-Hill, Inc., Irwin, ninth edition, 2001.

Id.	Query	Discussion
1	What is the program's purpose?	The purpose is described by the semantic axes of the *A*-space, in this example, Operations, inventory threshold of operations of inventory items.
2	What are the inventory items?	The items are the semantic values associated with the items axis.
3	What are the thresholds?	The thresholds are the semantic values associated with the thresholds axis.
4	What is the operation associated with the point (0,0,0)?	The operation is found by selecting the point in the add-items plane, i.e., (`add`,,).
5	What is the inventory associated with the point (0,0,0)?	This refers to the data domain; in this example, the data is associated with the items, i.e., (,0,).
6	How many dimensions in the *A*-space?	Return the number of axis.
7	How many operations in the *A*-space?	Return the number of indeces defined for operations, here, two corresponding with `add` and `remove`.
8	How many items in the inventory (in the *A*-space)?	Return the number of indeces defined for items.

Table 1: Example queries for the Inventory Control Case Study

[4] B. J. d'Auriol. Expressing parallel programs using geometric representation: Case studies. In *Proc. of the IASTED International Conference Parallel and Distributed Computing and Systems (PDCS'99)*, pages 985–990, Cambridge, MA, USA, Nov. 1999.

[5] B. J. d'Auriol. A geometric semantics for program representation in the polytope model. In *Proc. of the Twelfth International Workshop on Languages and Compilers for Parallel Computing LCPC'99, Aug. 4-6, The University of California, San Diego, La Jolla CA USA*, Lecture Notes in Computer Science, Vol. 1863, pages 451–454. Springer-Verlag, 1999.

[6] P. Gärdenfors. *Conceptual Spaces, The Geometry of Thought*. MIT Press, Cambridge, MA, 2000.

[7] H.-W. Gellersen, M. Beigl, and H. Krull. The mediacup: Awareness technology embedded in an everyday object. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC99)*, pages 308–310, Karlsruhe, 1999. Springer. Lecture notes in computer science; Vol 1707.

[8] E. Gjørven, F. Eliassen, K. Lund, V. S. W. Eide, and R. Staehli. Self-adaptive systems: A middleware managed approach. In A. Keller and J.-P. Martin-Flatin, editors, *2nd IEEE International Workshop on Self-Managed Networks, Systems & Services (SelfMan 2006)*, pages 15–27. Springer, 2006. LNCS 3996.

[9] M. Hecker, A. Karol, C. Stanton, and M.-A. Williams. Smart sensor networks: Communication, collaboration and business decision making in distributed complex environments. In *Proceedings of the International Conference on Mobile Business (ICMBŠ05)*, pages 242–248, Sydney, Australia, July 2005. IEEE Computer Society.

[10] G. Kirby, R. Morrison, and D. Stemple. Linguistic reflection in java. *Software - Practice & Experience*, 28(10):1045–1077, 1998.

[11] C. Lengauer. Loop parallelization in the polytope model. In E. Best, editor, *CONCUR'93*, Lecture Notes in Computer Science 715, pages 398–416. Springer-Verlag, 1993.

[12] J. L. W. Richard P. Gabriel and D. G. Bobrow. Clos: integrating object-oriented and functional programming. *Communiations of the ACM*, 34(9):28–35, 1991.

[13] B. C. Smith. Reflection and semantics in lisp. In *Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages (POPL '84)*, pages 23–35, New York, NY, USA, 1984. ACM.

[14] Sun Microsystems, Inc. Trail: The reflection api, 2007.

[15] U. Banerjee. *Loop Transformations for Restructuring Compilers — The Foundations*. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA, USA, 02061, 1993.

[16] Wikipedia. List of programming languages by category. Last modified: 06:52, 9 January 2008.

[17] J. Willcock, J. Järvi, A. Lumsdaine, and D. Musser. A formalization of concepts for generic programming. In *Concepts: a Linguistic Foundation of Generic Programming at Adobe Tech Summit*. Adobe Systems, April 2004.