A Scenario-Based Architecture for Reliability Design of Artificial Intelligent Software

Juguo Wang*‡, Guoqi Li†and Yun Pu*
*Institute of Public Administration, Southwest Jiaotong University, Chengdu China
†Department of Reliability and System Engineering, Beihang University, Beijing China
Email: gqli@buaa.edu.cn
‡Human resource department, Agricultural Bank of China
Email: wang_juguo@sina.com

***Abstract*—Artificial Intelligent is well invested both in industry and academic fields. However, the software reliability design for the applications is a big challenge. In this paper, we present a scenario-based architecture for reliability design of artificial intelligent Software. Scenarios are divided into environmental scenarios and structured scenarios. Both of the two kinds of scenarios are considered in the framework. The quantitative reliability of the software based on the framework could be evaluated and predicted.**

***Keywords*-artificial intelligent; reliability design; scenario-based;**

I. Introduction

Artificial intelligence (AI) is attractive for safety-critical fields. For example, a typical potential application is unmanned aerial vehicle, widely used for military purposes and also widely interested by academic researchers [1]. However, there have been few success cases, for the AI technique is usually lack of determinism and predictability, which is usually regarded as a disqualifier in a safety context. In this paper, we present a scenario-based architecture for reliability design of artificial intelligent Software. Scenarios are divided into environmental scenarios [2] and structured scenarios [3]. Both of the two kinds of scenarios are considered in the framework. With the framework and cooperated with the "Component-Dependency Graph" (CDG) [4], the quantitative reliability of the AI software could be evaluated before invoked.

In the next section, a case study is given for illustration as a typical application. In section 3, the representations of environmental scenarios and structured scenarios are described in detail, as well as the introduction of the framework.

II. Typical Application

Left side of Figure 1 is the Picture of the tiny mobile robot developed by our research group. Right side is its three layer architecture. The object of the system is automatic navigation by pattern recognition through video capture. It can recognize the obstacle in the flow of video. We selected a ARM11 processor (Samsung S3C6410) as main control processor, and select Windows CE as operation system.

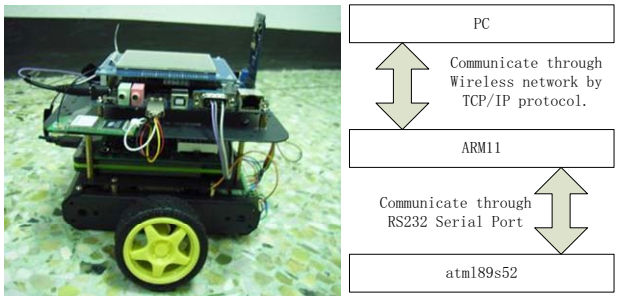

Figure 1. The picture of the tiny mobile robot and its three layer architecture.

Main control processor will be connected to atml89s52(a kind of 8051 single chip microcomputer) through RS232 Serial Port, and then atml89s52 control the motors through a motor drive module. The motion control program is written in the RAM of atml89s52. Ranging sensor provide additional assurance to avoid crash. The ranging sensor connect to atml89s52 directly by I/O port. Operationally, Main control processor capture and send digital image signal to PC upper computer through Wireless network communications by TCP/IP protocol. To implement automatic navigation, PC upper computer need to do digital image processing and pattern recognition through digital image signal and send control command to atml89s52 module through ARM11 module. There is also an interface in the PC to control the motion of the robot manfully.

Figure 2 is the deployment diagram of the distributed software system.

III. A General Framework and Scenarios

A. Environmental scenarios

It is obvious that the recognition accuracy of the mobile robot is affected by the environment. For example, the light intensity, at daytime the recognition is about 96.7% and at twilight, the accuracy is lower and various with different illumination. Visibility is another factor. Fog and dust could reduce the accuracy. So, we add a light intensity sensor

978-0-7695-4297-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CIS.2010.9

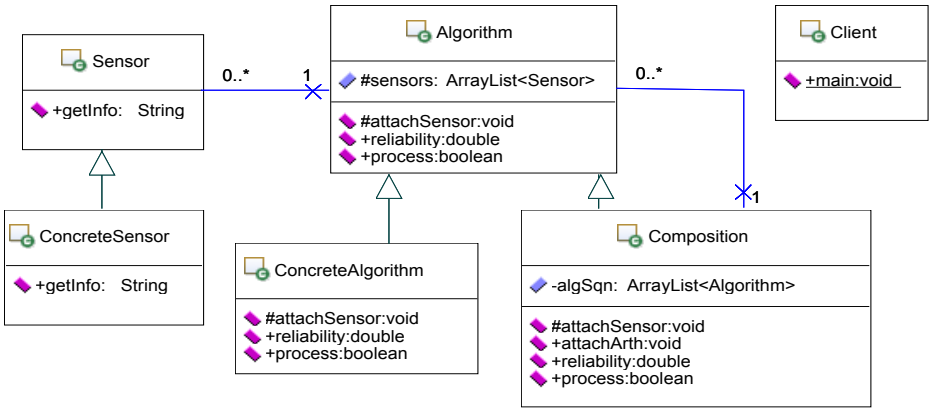

Figure 3. The framework designed for mobile robot with multi-sensors.

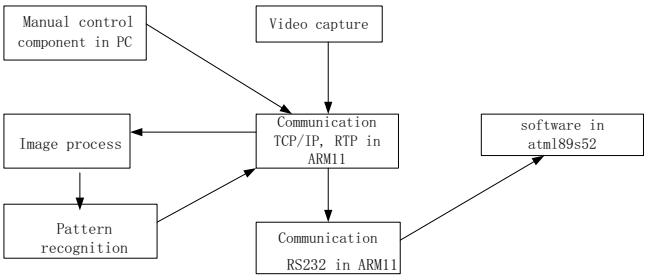

Figure 2. The deployment of the mobile robot software.

and a software component to calculate the visibility of the environment for the tiny robot. On the other hand, number of samples for training is also an important factor for accuracy. Typically, cross-validation is used to evaluate the accuracy of a pattern recognition algorithm.

Generally, we define the reliability R of a pattern recognition algorithm as its accuracy under specified E and s_E:

$$R = A(E, s_E) \tag{1}$$

where $E = \{e_1, e_2, \ldots, e_n\}$, means environmental properties having affection to accuracy. s_E is the number of samples under the specified E. $s_E > s_{E0}$, s_{E0} is the threshold for sufficiently training.

Additionally, environment may also affect the composition of the system, such as configuration or construction of modules. We describe the affection as rules: $E \to C$.

Consequently, we define the rules:

$$\begin{cases} (E, s_E) \to R \\ E \to C \end{cases} \tag{2}$$

as environmental scenarios for pattern recognition algorithms.

B. The Framework

With environmental scenarios, we could evaluate the reliability of pattern recognition algorithms. It's not enough, because it is also a problem that how the pattern recognition algorithms were integrated into systems. A sample architecture should be given for further analysis.

Figure 2 is an architecture designed for pattern recognition systems with predictable reliability. Although there exist other designs, it is a sound solution. In fact, Figure 2 is the class diagram of UML. The classes in the figure can be replaced by software modules in other similar forms.

Practically, algorithms are combined to complete a pattern recognition mission. For example, in the recognition of a signpost for the tiny robot, there are 3 steps:

1) Image enhancement, including intensity histogram modification, noise cleaning, edge crispening, signpost location and etc..
2) Feature extraction for the located signpost.
3) Classifier training and classification.

The parameters of algorithms and composition model should be changed according to the environmental scenarios. For example, when light intensity changes, the noise cleaning algorithm should modify its parameters to adapt the change. At the seam time, to complete image enhancement, the composition model of related algorithms may also need reconstructed. In addition, all the pattern recognition algorithms and the builder of composition model should have ability of checking the information of related sensors when needed.

To satisfy the requirements, in the architecture we apply observer and composite patterns, which are classical design patterns of object-oriented method [5] and do some modification according to requirements.

The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependants, called observers, and notifies them automatically of any state changes. In our architecture, the

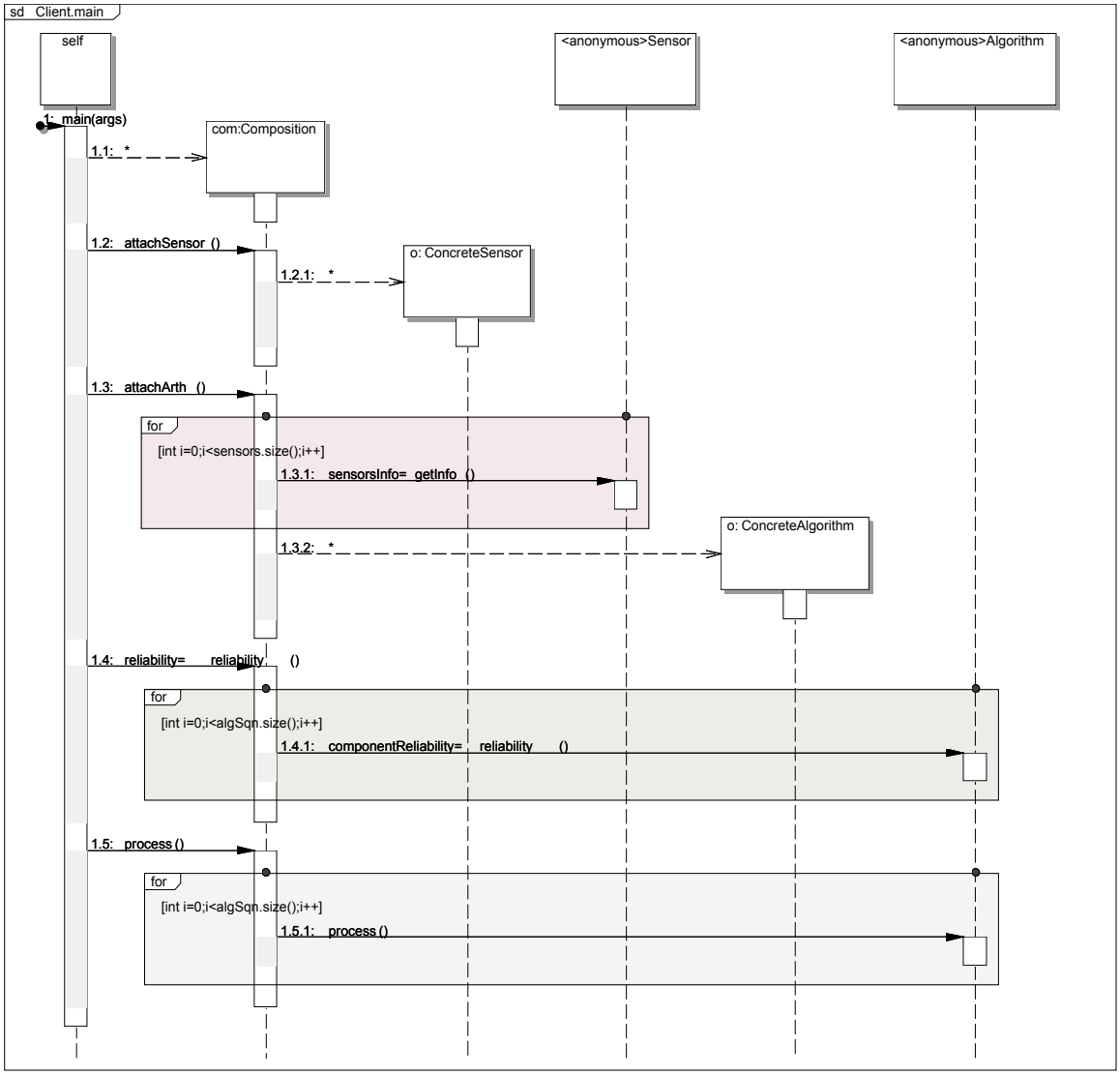

Figure 4. The sequence diagram of main function of a client based on the framework.

four classes, ”Sensor”, ”ConcreteSensor”, ”Algorithm” and ”ConcreteAlgorithm”, are organized like classes in observer pattern. ”Algorithm” maintains a list of its related sensors and can check their states when needed. ”ConcreteSensor” is generalized from ”Sensor” and may be hardware or software sensor.

The intent of composite pattern is to ”compose” objects into tree structures to represent part-whole hierarchies. In our architecture, the three classes, ”Algorithm”, ”ConcreteAlgorithm”, ”Composition”, are organized based on composite pattern. ”Composition” maintains a list of ”Algorithm” and all the listed ”Algorithm”s are combined to complete the required missions. ”Composition” is generalized from ”Algorithm” and also maintained a list of sensors, it can check information of related sensors and modify the composition model according to the information by modify the list of ”Algorithm”s.

C. Structured Scenarios

The term of ”structured scenarios” is derived from application of layered model for ubiquitous computing [3] and means that the scenarios are organized with structured method. Here we call the scenarios as structured scenarios, for they are used to describe the interaction between components of system and they are tight coupling with the architecture of system.

The scenario-based reliability analysis method for component-based software [4] (SBRA) are applied to calculate the quantitative reliability of the whole system. To use the method there need two precondition:

1) The quantitative reliability of components have been determined. We have done that for pattern recognition algorithms in 3.1.
2) The architecture of the system have been built.

SBRA is specific for component-based software whose analysis is strictly based on execution scenarios, which is a kind of structured scenarios. Using these scenarios, construct a probabilistic model named "Component-Dependency Graph" (CDG). CDGs are directed graphs that represent components, component reliabilities, link and interface reliabilities, transitions, and transition probabilities. In CDGs, component interfaces and link reliabilities are treated as first class elements of the model. Based on CDGs, an algorithm is also given in the paper to analyze the reliability of the application as the function of reliabilities of its components and interfaces [4].

Practically, the sequence diagram of software components are treated as structured scenarios, or so called execution scenarios in SBRA. Figure 3 is the sequence diagram of a client of the architecture.

IV. Conclusion

In this paper, we present a scenario-based architecture for reliability design of artificial intelligent Software. With the architecture and scenario-based analysis, the reliability of pattern recognition system could be evaluated and predictable, so that it could be qualified in safety critical applications.

References

[1] P. Abbeel, A. Coates, T. Hunter, and A. Y. Ng, "Autonomous autorotation of an rc helicopter," in *ISER*, 2008, pp. 385–394.

[2] M. Zhizhin, E. Kihn, V. Lyutsarev, S. Berezin, A. Poyda, D. Mishin, D. Medvedev, and D. Voitsekhovsky, "Environmental scenario search and visualization," in *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. New York, NY, USA: ACM, 2007, pp. 1–10.

[3] K. Yanagida, Y. Ueda, K. Go, K. Takahashi, S. Hayakawa, and K. Yamazaki, "Structured scenario-based design method," in *HCD 09: Proceedings of the 1st International Conference on Human Centered Design*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 374–380.

[4] S. Yacoub, B. Cukic, and H. Ammar, "Scenario-based reliability analysis of component-based software," in *Software Reliability Engineering, International Symposium on*. IEEE Computer Society, 1999, pp. 22–31.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.