UML기반의 창의 공학용 로봇 설계

정회원 홍 선 학*

Creative Engineering Robot Design with UML

Hong Seon Hack* *Regular Members*

요 약

본 논문에서는 통일 모델링 언어(UML)기법을 사용하여 창의공학에서 활용할 수 있는 로봇 설계기법을 제시한다. 최근 IT 분야의 교육 목표를 창의적 사고방식을 키우기 위한 시대적인 요구에 맞추어, 본 연구를 통하여 창의공학용 로봇 활용을 GUI 프로그램과 접목시킬 수 있도록 UML기반의 객체지향 프로그래밍(OOP)개념을 적용한다. 창의공학 분야의 프로그램에 사용되는 Java 또는 C#과 같은 객체지향 기법은 시스템 전체를 유연하게 설계하고, 강인하며 유지 관리에서 효율적이며, 여기서 얻어진 결과를 창의 로봇 분야의 시스템 개발도구로 활용할 수 있도록 하였다. 또한 본 논문에서는 GUI 환경에서 UML기반의 창의공학용 로봇 설계기법을 실험을 통하여 구현하였다.

Key Words : Creative Robot, UML, OOP, Refactoring

ABSTRACT

This paper proposes the techniques and theory works of the creative engineering robot with UML(Unified Modeling Language) base. Now, with the IT development of school curriculum has been interested, we apply the OOP(Object-Oriented Programing) concept of UML to the GUI program which used to creative engineering robot fields. The object-oriented analysis and design skills are essential for the creation of well-designed, robust, and maintainable software such as Java or C# with creative engineering robot and are especially applicable to the system development tools of creative engineering robot. This paper experiments the design method of creative robot with UML under GUI programming environments.

Ⅰ. Introduction

The Unified Modeling Language(UML) has become the universally-accepted language for software design blueprints. UML is the visual language used to convey software design patterns that are what allow us to describe design fragments, and reuse design ideas. Design patterns give a name and form to abstract heuristics, rules, and best practices of object-oriented techniques. In the beginning of 1990th, James Rumbaugh developed the OMT(Object Modeling Technic), Grady Booch handed out the Booch method, and Ivar Jacobson introduced the OOSE(Object Oriented Software Engineering), and then they worked together at IBM and collaborated on a work of developing the UML.[1].

The UML started as an effort by Booch and Rumbaugh in 1994 not only to create a common notation, but to combine their two methods - the Booch and OMT methods. Thus the first public

※ 본 연구는 2007학년도 서일대학 교내학술연구비지원으로 수행되었음.
* 서일대학 컴퓨터전자과(hongsh@seoil.ac.kr)
논문번호 : 08027-0415, 접수일자 : 2008년 4월 15일

draft of what today is the UML was presented as the unified Method. They were soon joined at Rational Corporation by Ivar Jacobson, the creator of the Objectory method, and as a group came to be known as the three amigos. It was at this point that they decided to reduce the scope of their effort, and focus on a common diagramming notation-the UML-rather than a common method.

The UML is a visual language for specifying, constructing and documenting the artifacts of systems. The visual in the above is a key point-the UML is the standard diagramming notation for drawing or presenting pictures related to software-primarily OO software. At this point, diagrams help us see or explore more of the big picture and relationships between analysis and software elements, while allowing us to ignore or hide uninteresting details. That's the simple and essential value of the UML. The UML includes class diagram to illustrate classes, interfaces, and their associations.[2,3]

Ⅱ. Basic Theory

This chapter provides a fundamental overview of the creative robot system. Object oriented programming (OOP) has a significant difference that has powerful implications: usually defined by a theme. For example, imagine you are trying to program a robot to scan an object in a 3-dimension scanning robotics system. The robot needs to be able to rotate, and also to scan an object.

2.1 OOP of Robot programming

One object would contain functionality centered on the theme of rotating, and another object would contain functionality for scanning an object. These units are defined by classes in Java. A block diagram of this OOP robot programming system shown in Fig. 1 illustrates the graphical overview of the objects in a robotics application.[4,5,6].

The ScanningObject class would contain all the methods for choosing which object to scan, and the RobotRotation class would contain all the methods to move object around the 3-D scanning robotics system. Classes are created using methods

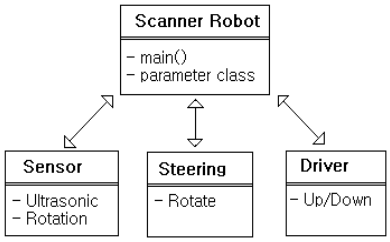

그림 1. 객체의 로봇 응용
Fig. 1. Objects in a robotics application

(functionality) and fields(data). Once the class is defined, it can be instantiated, which means an object is created which can be manipulated in code. One class can be used to create many objects. Another powerful feature of OOP is type extensibility. This is the ability of classes to inherit data and functionality from other classes.

2.2 Definition of robot classes

The purpose of modeling or sketching UML is primarily to understand, not to document. That is, the very act of modeling can and should provide a way to better understand the design of creative robot. In this paper, a domain model is used for creating the classes of creative robot OOP analysis.

A domain model can acts as a source of inspiration for designing the software objects and will be an input to several artifacts. So domain model is a conceptual perspective of objects in a real situation of the robot environment, not a software perspective. There are four classes which are robot class, driver class, rotation class, and sensor class. 3-D scanning robotics system will be operated by the driver, controlled by the rotation, and monitored by the sensor.[7,8]

2.3 Bite into Bluetooth

In this paper, we use the bluetooth communication, a wireless control of robots that gave us the enormous resources and brainpower that can be harnessed through wireless tools. A single Bluetooth dongle effectively acts like many wireless USB ports. Bluetooth eliminates cables from keyboard, mice and game controllers.

The major advantage of Bluetooth versus the infrared (IR) system used by other robotic developing system

is the elimination of in-line-with problem. Radio waves travel through solid objects and in all directions, whereas the IR port on the other system had to point right at the IR tower with no intervening structures.

Bluetooth operates on 2.45GHz and is capable of transmitting data at 460.8 Kbit/s. Multiple Bluetooth dongles can be used in the same area since Bluetooth uses frequency hopping to avoid conflicts. And therefore, We use the pairing technic with robotics to the PC. Pairing occurs when one device tells another device that they can be friends. The reason for pairing is to avoid someone accessing your Bluetooth devices without permission. By pairing, we gave the robotics permission to interact as like plugging a USB cable into a computer. Pairing is done by locating the Bluetooth device, then entering a four digit combination on both devices.

Especially, we used the iCommand that is a package for remote control from a computer. this package seeks to mirror robots as closely as possible. which means the classes and methods tend to look similar. The iCommand controls robots by sending individual commands wirelessly. In some more detail, Speed is often a concern for robot programmers. Because all commands are sent to the robotics through Bluetooth, the iCommand reacts a little slower than on the other devices.

However, most robots are slow and don't require fast processors. It takes a few seconds to move a foot, so a hundred milliseconds here and there isn't a big deal for most robot applications. Since the iCommand is running on your computer and uses the standard Sun Java API, we have access to an incredible number of resources, such as the Internet and other hardware devices on our computer. We used the Eclipse, an IDE to develop the iCommand code.[9,10]

2.4 Setup the leJOS API

The name lejos means 'far' in spanish. In the name leJOS, the letters JOS are capitalized because those letters stand for Java Operating System. Since le means 'the' in several languages, this would mean the Java Operating System. The leJOS JVM (Java Virtual machine) is written in C code in a very platform independent style, which means it is easily ported to other machines. So far it has appeared on the Gameboy Advance and the NXT bricks.

There are also tools on the PC side to compile and upload code to the leJOS JVM. leJOS is multiplatform, and recently that means Windows, Linux, and Macintosh. leJOS software is available for each of these platforms, allowing you to develop the program under your favorite operating system. In this paper we use the Windows that is to download the latest version from www.lejos.org. One of the best open source IDE(Integrated Development Environment) is Eclipse by IBM. It's free, powerful, and easy to use. It makes sense to use a more advanced IDE with the hardware since our code can grow quite large. Eclipse can be downloaded from www.eclipse.org.

We could build the following robot: it wanders around our house avoiding objects with the distance sensor. If the sensor misses an object, the robot can still tell if the wheels are stuck by monitoring decreases in rotation speed. If the robot tips over, it uses a tilt sensor to identify the problem. This is possible with leJOS software. And we should know where to find the methods in the API. There are many classes in the API such as the battery class, the button class, the sensor classes and the motor class. Especially the sensor classes include the lightsensor, the soundsensor, the touchsensor and the ultrasonicsensor.

The battery class allows to determine the voltage produced by the batteries. Rechargeable batteries provide approximately 7.4 volts, while alkaline batteries produce 9 volts. The button class contains static instances of the four buttons. These four instances are ENTER, ESCAPE, LEFT, and RIGHT. This class implements the ButtonListener interface, which contains two methods definitions: buttonPressed() and buttonReleased(). These interface methods must be defined in the class implementing the interface. The motor class contains three instances of motor: Motor A, Motor B, and Motor C. Speed is in degree per second.

The actual maximum speed of the motor depends on battery voltage and load. The lightsensor class contains the readvalue() method which returns a number between 0 and 100. The soundsensor class allows access to the sound sensor, and is operated in two modes, dB and dBA. The ultrasonicsensor class is used to obtain distance readings from the ultrasonic sensor.[11,12].

Ⅲ. 3D Robotic Scanning

In this paper, 3D robotic scanning program is somewhat large because, besides needing to control the actual scanning of real-images, it also needs to enable to adjust the image threshold and select a low or high resolution scan.

3.1 Programming the robotic scanner

The important part about programming this device is that it needs to know x, y, and z coordinates for every point that it scans. However, the distance sensor only returns one value: the distance to the object. So we'll need to use trigonometry to produce coordinates from this value.

The robotic scanner take a measurement at equal increment all around the object as described by Fig. 2.

In this paper, 3D robotic scanner produces exactly the same results as moving a sensor around the object. At the second scan, the effect is the same as if the sensor were moving around the object. It's easier to understand the calculations by imagining the platform standing still and the sensor moving around the object. We'll designate the center of the platform as (0,0,0) on our coordinate system. The position of the sensor when it begins scanning will be designated at 0 degree, and rotation will occur counterclockwise.

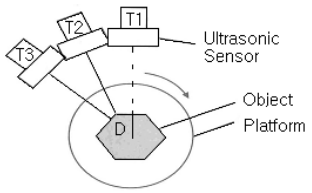

그림 2. 물체 스캐닝
Fig. 2. Scanning an object

3.2 Calculating the scanning algorithm

The scanner rotates around the object taking various distance measurements. At T_1, the y coordinate is zero and the x coordinate is the distance from the center of the platform to the position at the edge of the object. The distance(d_1) from the center of the platform to the sensor is constant, and we can easily find it by measuring. However, we want the distance from the center to the edge of the object.

$$d = d_1 - d_2 \tag{1}$$

At T_2, the x and y coordinates are positive which can be calculated from d and the angle formed at point A. The equations to calculate all points around the object are as follows:

$$x = d \times \sin(A) \tag{2}$$
$$y = d \times \cos(A)$$

The z coordinate is the easiest to calculate, since it is merely the current height of the ulrtrasonic sensor. Each time the sensor rises, we increase the z coordinate value.

In order for our software to correctly and accurately control and interpret the platform rotation and sensor movement, it needs to know some measurements from the scanner unit: how many degrees the motor must rotate for a single platform rotation, distance from the ultrasonic scanner to the center of the platform, and the distance it raises the scanner after every complete scan. Geometrical configurations of 3D robotic scanner is listed in the Table 1.

Rather than directly measuring the distance from the ultrasonic sensor to the center of the platform. We used the sensor to measure the distance to an

표 1. 3D 로봇 스캐너의 특성값
Table 1. Characteristic values of 3D robotic scanner

Measurement	Value	Unit
Distance from sensor to center	20	cm
Full rotation counts	12600	
Vertical movement per scan	2.5	cm

object with its edge on the center. The ultrasonic sensor returned the specified value. It is better to go with the ultrasonic sensor rather than the actual measurement, since all our subsequent values will come form the sensor. With these measurements we can calculate everything else we need to know in order for the scanner to give relatively accurate readings.[13,14,15].

In this paper, sometimes we know the x and y coordinates and must calculate. In this case we use inverse functions. If we know the length of each side of a triangle then we can calculate the angles by rearrange the equations slightly:

angle - asin(Opposite/Hypotenuse) (3)
angle = acos(Adjacent/Hypotenuse)
angle = atan(Opposition/Adjacent)

Asin, acos, and atan are merely words for inverse sine, inverse cosine, and inverse tangent. There is one problem with atan - it can't calculate angles greater than 90 degrees. However using Math.atan2(x,y) we can calculate this accurately as atan2() can produce any angle between 0 and 360 degrees.

During scanning, the 3D robotic scanner don't have a right angled triangle. In this case, as long as we know the lengths of all three sides of a triangle, we could figure out the angles. This means we can calculate the inner angle as illustrated by Fig. 3.

The law of cosine applies to any triangle in which the lengths of all three sides are known

$$c^2 = a^2 + b^2 - 2ab\cos A \quad (4)$$

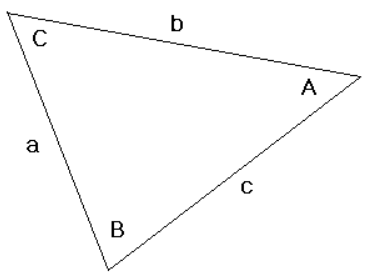

그림 3. 코사인 법칙
Fig. 3. Law of Cosines

In this form, the equation isn't useful if we want to calculate angle A, so we rearrange as follows:

$$A = acos(\frac{a^2+b^2-c^2}{2ab}) \quad (5)$$

3.3 Refactoring for 3D Robotic Scanning

We apply the refactoring method to 3D robotic scanner for the purpose of improving the design of existing program. Refactoring is a structured, disciplined method to rewrite or restructure existing code without changing its external behaviour, applying small transformation steps combined with re-executing tests each step. Continuously refactoring code is another extreme programming(XP) practice and applicable to all iterative methods. The essence of refactoring is applying small behavior preserving transformations at a time. After each transformation, the unit tests are re-executed to prove that the refactoring did not cause a regression. Therefore, there's a relationship between refactoring and TDD (Test Driven Development) - all those unit tests support the refactoring process. Each refactoring is small, but a series of transformations - each followed by executing the unit tests again- can, of course, produce a major refactoring of the code and design, all the while ensuring the behavior remains the same.

Code that's been well-refactored is short, tight, clear, and without duplication - it looks like the work of a master programmer. Code that doesn't have these qualities smells bad or has code smells. In other words, there is a poor design. Code smells is a metaphor in refactoring - they are hints that something may be wrong in the code. it might turn out to be alright and not need improvement. That's in contrast to code stench - truly putrid code crying out for clean up. The remedy to smelly code are the refactorings.[16,17].

Like patterns, refactorings have names, such as Extract method, There are about 100 named refactorings. The description of refactoring for the 3D robotic scanner is summarized in the Table 2. Figure 4 illustrates the flow chart of 3D robotic scanner.

표 2. 리팩토링에 대한 상세
Table 2. The description of refactoring

Refactoring	Description
Extract Method	Transform a long method into a shorter one
Extract Constant	Replace a literal constant with a constant variable
Introduce Explaining Variable	Put the result of the expression, or parts of the expression.
Replace Constructor Call with Factory method	In Java, replace using the new operator and constructor call with invoking a helper method

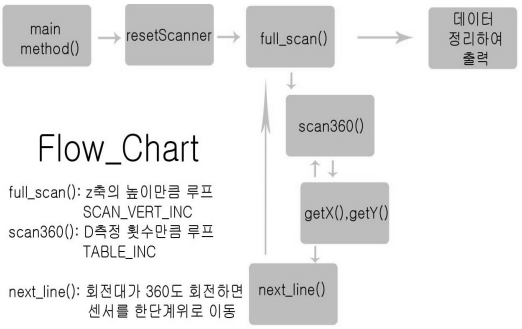

그림 4. Flow chart of 3D robotic scanner
Fig. 4. 3D 로봇 스캐너 플로우 챠트

Ⅳ. 3D robotic Scanner Experiments

The purpose of UML method is primarily to understand the 3D robotic scanner. From this viewpoint, the purpose of doing UML is not to create many detailed UML diagrams that are handed off to a programmer, but rather to quickly expolre alternative and the path to a good OO design.[18,19].

4.1 Agile Modeling for 3D Robotic Scanner

JVM supports the parameterized types which are most commonly used for the element type of collection classes, such as the elements of lists and maps. In this paper, we used the parameter class which includes the Driver_Speed, Steering_Speed, and Sensor_Threshold. And therefore everything of the Robot, Driver, Steering, and Sensor classes references the parameter class. The instance of parameter class is transferred to the start method of robot class for the purpose of initializing the class.

Driver class includes the setParameter method which specify the revolution speed of motor and

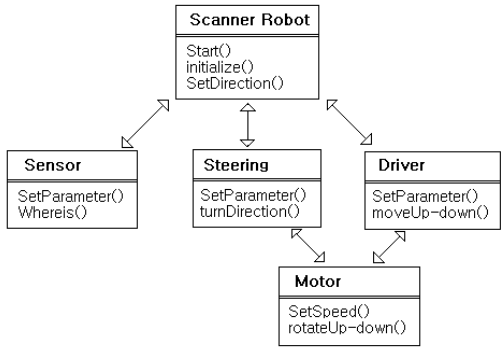

그림 5. 클래스 다이어그램 관계도
Fig. 5. Class diagram relationship

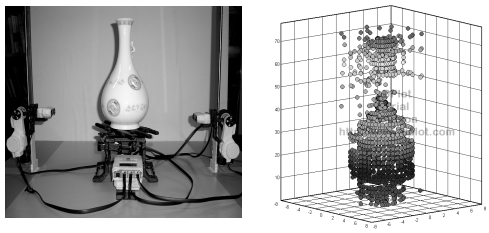

그림 6. 물체의 DPlot 스캐닝
Fig. 6. DPlot Scanning of the object

Forward method which move robot to the forward direction. We summarize the class diagram of relationship among Robot, Driver, Motor, steering and Sensor class as illustrated by Fig. 5.[20,21]

4.2 Results of the 3D scanning

The ultrasonic sensor produces the measurements in one centimeter increments, which prevents it from picking up fine detail, and the field view of the ultrasonic sensor is wide. If it had a narrower field of view, small details in objects would be detectable. And therefore, we can actually change the range of the sensor from 255 centimeters to something smaller and produce more accurate results. The scanner program outputs a series of 3D coordinates to the console screen. The results can be displayed in a variety of ways. For example, Excel spreadsheet is almost ubiquitous to all platform though, we use the DPlot software for the purpose of highly-grade displaying the images.[22,23].

Figure 6. shows the image of 3D robotic scanner of DPlot displaying the scan of an object. Dplot trial version is www.dplot.com and Graphics download

is www.kylebank.com.

V. Conclusions

In this paper, the 3D robotic scanning technique and theory works on the basis of UML are experimented. UML has become the universally accepted language for software design blueprints. UML is the visual language used to convey design ideas throughout this paper, which emphasizes how we really apply frequently used UML elements, rather than obscure feature of the language. 3D robotic scanner needs to integrate multiple sensor to execute tasks such as pattern recognition, and optimal image scanning.

We use the Java software that allows to program the 3D robotic scanner, and we would improve the performance of the 3D scanner with helping the Java Integrated Development Environment(J-IDE) which have the class, threads, arrays, floating point numbers, recursion, and total control methods.

참 고 문 헌

〔1〕 CRAIG LARMAN, APPLYING UML AND PATTERNS 3rd, IE PRENTICE HALL, 2005.

〔2〕 Brian Bagnall, Building Robots with Java Brains, VARIANT PRESS. 2007.

〔3〕 Watanabe, Software Design Technic. Cyber Pub. 2004.

〔4〕 Ambler, S. The Unified Process-Elaboration Phase, Lawrence,KA,:R&D Books, 2000

〔5〕 Beck, K.. Pattern and Software Development. Dr. Dobbs Journal. 2. 1994.

〔6〕 Jason Gu, Max Meng, Al Cook, Peter X, Liu. Sensor Fusion in Mobile Robot, Proceedings of the 4th World Congress on Intelligent Control and Automation, June 10-14, 2002.

〔7〕 R. Gartshore, A. Aguado, C. Galambos. Incremental Map Building using an Occupancy Grid for an Autonomous Monocular robot, 7th nternational Conference on Control, Automation, Robotics & Vision. Dec, 2002. Robot, MIIT Press, 1991.

〔8〕 W. Burgard, D. Fox & T. Schmidt, Estimating the absolute position of a mobile robot using position probability grids, in Proc of National Conference on Artificial Intelligence, 1996.

〔9〕 Coad, P. Objected-oriented Patterns. Communications of the ACM. 9.1992

〔10〕 Fowler, M. Draft patterns on object-relational persistence services. 2001.

〔11〕 CHRIS CANT, Writing Windows WDM Device Drivers, 2001. R&D Books

〔12〕 J. Michael Jacob, Industrial Control Electronics, Prentice-Hall, 1989.

〔13〕 Martin, R. Designing Object-oriented C++ Application Using the Booch Method, Englewood Cliffs, NJ: Prentice-Hall.

〔14〕 Frank L. Lewis, Optimal Estimation, John Wiley & Sons, 1986.

〔15〕 Gordon McComb, Robot Builder's Bonanza, TAB Books, 1987.

〔16〕 Dave Prochnov, Mindstorms Hacker's Guide. McGraw Hill, 2007.

〔17〕 Benjamin Erwin, Creative Projects with Mindstorms. Addison Wesley, 2001.

〔18〕 Jim Leden, Embedded Control Systems in C/C++, CMP Books, 2004.

〔19〕 Fred G. Martin, Robotic Explorations, A Hand-On Introduction to Engineering. 2001.

〔20〕 An Introduction to ROBOT TECHNOLOGY, Philippe Coiffet, 1982.

〔21〕 Thomas, M. IT Projects Sinks or Swim. British Computer Society Review. 2001.

〔22〕 Rumbaugh,.J., et al. Object-Oriented Modelling and Design. Englewood Cliffs. NJ: Prentice-Hall. 1991.

〔23〕 Wirfs-Brock,.R., Wilkerson. Designing Object-Oriented Software, Englewood Cliffs, NJ.: Prentice-Hall. 2002.

홍 선 학 (Hong Seon Hack) 정회원

1985년 광운대학교 전기공학과 학사졸업

1988년 광운대학교 전기공학과 석사졸업

1994년 광운대학교 대학원 박사 졸업

1992년~현재 서일대학 정보기술계열 부교수

<관심분야> 제어, 컴퓨터응용, 로봇분야 등>