Revista Educación en Ingeniería
ISSN 1900-8260
Julio a Diciembre de 2013, Vol. 8, N°. 16, pp. 136-147 • © 2013 ACOFI • http://www.educacioneningenieria.org
Recibido: 24/07/2013 • Aprobado: 24/11/2013

GUÍA PARA EL DESARROLLO ÁGIL DE APLICACIONES EN ROBÓTICA

METHODS AND TOOLS USED IN FAST DEVELOPMENT OF ROBOTICS PROJECTS

Alo Peets
Universidad Simón Bolivar, Colombia

Resumen

La robótica es un área multidisciplinaria que hace uso de los recursos de vanguardia de las ciencias afines, lo que la convierte en uno de los campos de mayor investigación en la actualidad. Este artículo consiste en la investigación de recursos, tecnologías y metodologías para aumentar el desarrollo y los resultados de los proyectos de la robótica. Se revisaron el diseño y las partes de un robot para mejorar la flexibilidad, eficiencia, uso de herramientas y recursos disponibles para ejecutar los proyectos de automatización y robótica con **éxito**. Además, se explica el proceso de selección del microcontrolador como parte principal de un robot, que permitirá dimensionar las posibilidades reales de creación de sistemas robóticos y la capacidad de los procesadores presentes en el mercado. Se muestra que es posible aumentar la producción de proyectos de robótica aplicando metodologías de desarrollo rápido, lo cual permite mejorar la eficiencia, la programación y el diseño de proyectos de esta área de conocimiento.

Palabras claves: robótica, desarrollo ágil, gestión de proyectos, productividad del trabajo, microprocesador.

Abstract

Robotics is multidisciplinary area where all the main sciences and technologies come together, which makes it one of largest growing investigation fields. The paper focuses on main topics of robotics like project management, innovation, software, electronics and mechanics. The paper will cover more precisely resources, tools and examples to efficiently construct electronics, mechanics and software of robotics projects. The main recommendation the article gives is not to focus on optimizing hardware cost nor man-labor separately, but as a mixture using prefabricated electronics modules or mechanic parts with fast software development and efficient project management. The document is focused on engineers starting with robotics and thus is written in general manner, but should give new ideas to professionals also.

Key words: robotics, fast development, project management, labor productivity, microprocessor.

Introduction

Robotics is the branch of technology that deals with the design, construction, operation, and application of robots. It is a diverse science that mixes mechanics, computers, mathematics, electronics, programming, and number of other sciences to one seamlessly working thing. There are various types of robots for various purposes (Industrial robots, military robots, service robots, home robots, etc…). Industrial robots started to rise in the 50s in America and Japan and have been the main developer of robotics industry for past 60 years. According to International Federation of Robotics the sales of industrial robotics have 5% of annual growth rates between 2013 and 2015 (IFR, International Federation of Robotics, 2012a). Though steady increase, in past 10 years educational, service and entertainment robots have taken over the robotics industry. In 2011, it was estimated that 1,7 million domestic robots (automated vacuum cleaners) and 0,85 million entertainment robots (mainly automated toys) were sold, which in numbers are many times more than industrial robots (166 028 in 2011) (IFR, 2012b). According to Japan Robotics Association, it is expected that service and personal robotics market will grow for the year 2025 three times to 51.7 Billion US dollars.

Sadly, things in Latin-America are not good: According to the The Robot Report there are only 3 manufacturers in South and Central America who sells more than 1000 robots a year. In figure 1, every marker points out one robotics manufacturer who sells more than 1000 units a year or startups (green) who have potential to sell 1000 units a year (The Robot Report, 2013). This map is not complete as this information is very hard to find but gives quite good overview of division of robotics industry and knowhow. Most companies are located in developed countries (United States, Western-Europe, Japan and Taiwan) where the use of technology is very high to overcome high cost of human resources. Nevertheless as with computer industry 20 years ago (Greenstein, 2010), soon robots and automated machines will be installed and used in other parts of the world also and therefore it is best time to learn and move into robotics business. Good way to do it is to complete successfully some small and medium size robotics projects and/or participate in research. In this paper we give overview of technologies and tools commonly used in fast development of robotics projects.

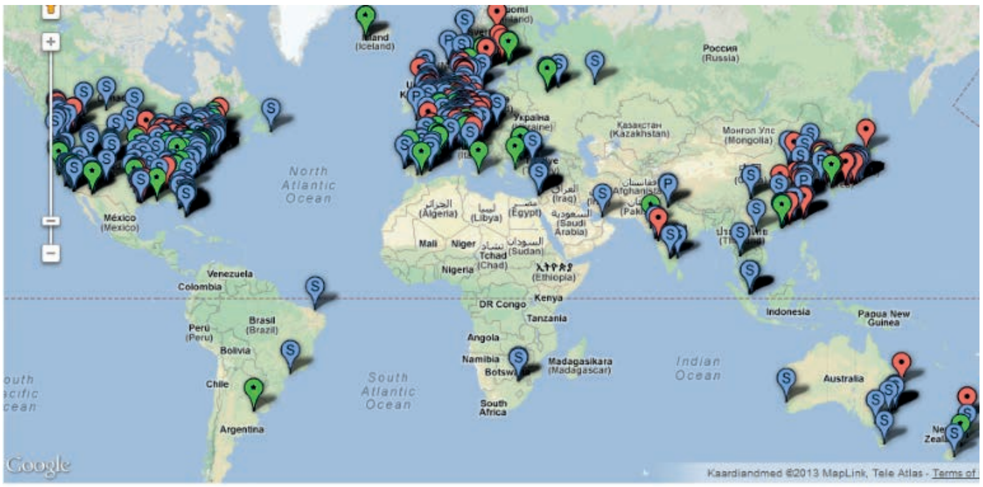

Figure 1. Distribution of robot manufacturing companies around the world (The Robot Report, 2013).

Fast Development of Projects

Project Management Overview

In 21st century time is always limited, therefore doing things efficiently and thus completing tasks fast with minimum effort in order to optimize labor cost is the goal. It can be achieved by various ways like motivating workers with bonus for performance or automating some tasks so company can eliminate repeating tasks. Opposite to people, robots are very good in doing same activities many times and bad in creativity. Robots do not get tired nor have as many errors as humans tend to do. Therefore, the industry has been using robots and machines for more than 50 years to be more productive and lower the cost of manufacturing.

This also means there are numerous projects in companies and universities that try to automate some tasks or make a machine that can make humans life easier or better. As robotics is so diverse it is close to impossible to be an expert in robotics thus many of the engineers does not have the experience nor full knowledge needed to complete or create the project plans in the starting phase of the projects. This also means, often not completing the requirements, going over the budget or deadline and unsuccessful completion of the projects. Therefore right methods of project management must be used and viability of proposed project must be evaluated as early as possible (Bloch, Blumberg, & Laartz, 2012).

Designing something using a limited timeframe and methods of fast development is nothing new, but it is only in last few years when large companies are seeing it as an alternative to classical project management cycle. Nowadays there are various methods and techniques accepted and widely used in company management (ITIL, ISO). Fast development and rapid development are gaining popularity all over the IT sector, but for the last 10 years software industry has been the main leader of the implementation of such methods. The idea behind agile software development is that not all tasks in project have same impact on the results and therefore some of them can be optimized or eliminated. In this section of the paper we will review few of the most famous principles and examples in fast development.

Pareto Principle

The Pareto principle (also known as the 80–20 rule) states that, for many events, roughly 80% of the effects come from 20% of the causes. Vilfredo Pareto originally applied the concept of 80/20 to distribution of wealth (20 percent of the people owned 80 percent of the wealth) and Joseph Juran to quality (20 percent of the defects cause 80 percent of the problems), over the years the 80/20 rule has been expressed in a number of different ways. Some of these are:

- 80 percent of the results are achieved by 20 percent of the group.
- 20 percent of your effort will generate 80 percent of your results.
- In any process, few elements (20 percent) are vital and many elements (80 percent) are trivial.
- If you have to do ten things, two of those are usually worth as much as the other eight put together.
- 20 percent of the tasks account for 80 percent of the value.

The 80/ 20 rule seems to be almost an universal truth and can be applied to practically all aspects of management and even to our personal lives. When used correctly, Pareto Analysis is a powerful and an effective tool for making continuous improvement and in problem solving. Continued application of this rule will greatly improve productivity, quality and profitability (Koch, 2004; Narula, 2005).

Google 70/20/10 Model

The 70/20/10 Model is a business resource management model pioneered by Eric E. Schmidt and articulated about Google in 2005 (Lombardo & Eichinger, 2000). This model dictates that, to cultivate innovation, employees should utilize their time in the following ratio:

- 70% of time should be dedicated to core business tasks.
- 20% of time should be dedicated to projects related to the core business.
- 10% of time should be dedicated to projects unrelated to the core business.

As a motivation technique, Google uses a policy often called Innovation Time Off, where Google engineers are encouraged to spend 20% of their work time on projects that interest them. Some of Google's newer services, such as Gmail, Google News, Orkut, and AdSense originated from these independent endeavors. In a talk at Stanford University, Marissa Mayer, Google's Vice President of until July 2012, showed that half of all new product launches at the time had originated from the Innovation Time Off (Mayer, 2006).

On Google Friday team members are encouraged to set aside their normal day-to-day work and instead “play” with technologies that interest them and may have a use in the Library. It’s a very interesting approach to development and leads to high motivation within the team to explore and innovate. Google employees explained that everyone looks forward to Google Friday as they enjoy the opportunity to be creative, test new ideas, mix with colleagues in the team that they may not normally work with and try things that might fail but doing so in a comfortable, unpressured environment. Some of the developments that come from Google Friday are implemented, others are after discussion are set aside but in the knowledge that something was learned from the experience and work (Batelle, 2005).

Agile Project Management

Agile management or agile project management is an iterative method of determining requirements for engineering and information technology development projects in a highly flexible and interactive manner, for example agile software development. Agile techniques may also be called extreme project management. It is a variant of iterative life cycle where deliverables are submitted in stages. One difference between agile and iterative development is that the delivery time in agile is in weeks rather than months.

Agile techniques are best used in small-scale projects or on elements of a larger project, or on projects that are too complex for the customer to understand and specify before testing prototypes. *The Agile Project Leadership Network* provides a community of practice for those using agile methods, with international conferences and online forums (Agile Resources, 2013).

Garage48

In 2010 six young entrepreneurs from Estonia started a movement called Garage48. Garage48 events usually start at 5pm on a Friday evening. All participants gather together in a big room and pitch about 30 to 40 ideas on stage. Each idea is put on the wall and everyone can choose their favorite idea and team. Usually about 12-15 ideas will be selected and teams start working. Organizers provide facilities, mentors, food and drinks over the weekend, while teams are working on their projects. Sunday night 5pm is the deadline to step on the stage again and live-demo your project or prototype. There are the jury and audience to vote for their favorites and choose the winners (Garage48, 2013).

Government and private companies are promoting entrepreneurship and innovation by sponsoring the event. The event have been huge success every time and results in participants being hired by large companies or developed into spin-offs by participants to shape their ideas into business.

When to use?

To sum up this part we can say that all the methods described before are few examples of great success using methods of fast development and motivation to be efficient. Fast development or the 80-20 model obviously cannot be used in a time consuming and be-there jobs. Nevertheless automation and modern industrial robots are minimizing classical man-labor rich and manual tasks therefore in 21st century more and more people are working with brains, not hands. We can see that one thing that is common in those examples are that the products created are innovative and development is urged by self-success, motivation and pride. Author’s previous experience in robotics projects of the University of Tartu have proved many times that in small and medium size robotics projects, where quantity is low and uncertainty high, methods described before have led to successful completion of projects and innovative products.

Many people believe that modern software development and robotics is like making art, you have to be really creative and you must do something that nobody have never done before. Nevertheless you must do it not for money, but because you like to do it and you believe in it. It has been observed that many of the modern successful IT companies and innovation are created by young students who have not yet been ruined by classical 6-sigma enterprise management, where failing and being wrong is hardly an option (Ressi, 2011). Therefore it is in some cases recommend to use less planning and more just-do-it approach in projects where exact solutions are unclear and difficult to decide in a planning phase. It gives more freedom and better adaptability if instead of a 3 month project phase you have two week phases and a regular feedback from client. At the end, use the planning and management scheme that suits best for you and your team.

How To Build A Robot

Robotics is a wonderful diverse science. It mixes mechanics with computers, mathematics, electrical design, programming and a number of other sciences. Building your first robot can be real fun, but also a hard and time-consuming work in a process of mastering variety of technologies and tools. Luckily you don't have to be an expert in every aspect of robotics but you still have to have good understanding of how it all comes together to form one seamless machine (McComb, 2008). The process of building a robot can be divided into subtasks:

- **Management**: coming up with a great idea, forming into a project and successful completion of requirements,
- **Mechanics**: selecting right materials, fabrication tools, designing a body and finally assembly,
- **Electronics**: selecting all the sensors, microprocessors, motors, power supply and putting it all together without errors,
- **Software**: writing, programming and testing a program that gives brains to the machine.

How Complicated It Is?

In 21st century some children learn how to use computers and technology before they learn how to read and write. For example in Estonia Government funded Tiger Leap Foundation started connecting schools to internet in 1997. Current educational project called ProgreTiiger emphasis on teaching programming to children from age 7 and upwards. They are convinced that understanding the functioning and use of modern technology is vital to our children, who will be the main work force of our society in a few years (Tiigrihüppe SA, 2013b).

Even more, they say that learning a programming language is like learning a foreign language, it gives you understanding and access to different world and culture where computers are integral part of our lives. Another project "Kooliroboti projekt" (Robot in Schools) is also in execution and focuses on giving students first experience and practice with LEGO NXT robot (Tiigrihüppe SA, 2013a). Both projects are very popular amongst the students and have received various educational awards. If even pupils can be successful robot creators, most of the current engineers have capability of being a robotics engineer.

Electronics

After numerous projects in University of Tartu, Estonia we concluded that choosing the right micro-controller (MCU) and communication interface with sensors to your robot is key element to efficient and successful completion of your project. Writing software to your MCU and designing an electronic circuit needs a lot of experience and effort. The rule of thumb are to choose the MCU that everybody is using as you have thousands of code and circuit examples available online and communities are very welcoming with beginners. Another recommendation is that as every MCU family is different, therefore software and connections are not compatible. It is good practice to stick with one MCU for few years until it is outdated, which makes choosing a right MCU for the first project even more important.

As you can see from table 1, current MCU market leaders are Renesass and Freescale. Nevertheless, they are mainly focusing on large scale manufacturing and are not highly used by hobbyists, as they tend to be harder to begin with. Google trend is a great tool to investigate popularity of key search terms in the internet. In Figure 2 there are visualization of main MCU companies and Arduino (most popular hobby board in electronics and robotics).

Table1. 2011 worldwide microcontroller revenue share by supplier.
Source: Databeans Market Reports (Databeans, 2012).

Company	2011 Rank	2011($M) Revenue	2011 Share	Change 2011/2010
Renesas Electronics	1	2,624	17,3%	-1%
Freescale Semiconductor	2	1,538	10,1%	4%
Atmel	3	1,114	7,4%	25%
Microchip technology	4	1,010	6,7%	6%
Infineon Technology	5	1,007	6,6%	18%
Texas Instruments	6	838	5,5%	4%
Fujitsu	7	834	5,5%	-17%
Samsung	10	510	3,4%	-1%
Others		5680	37,6%	-1%

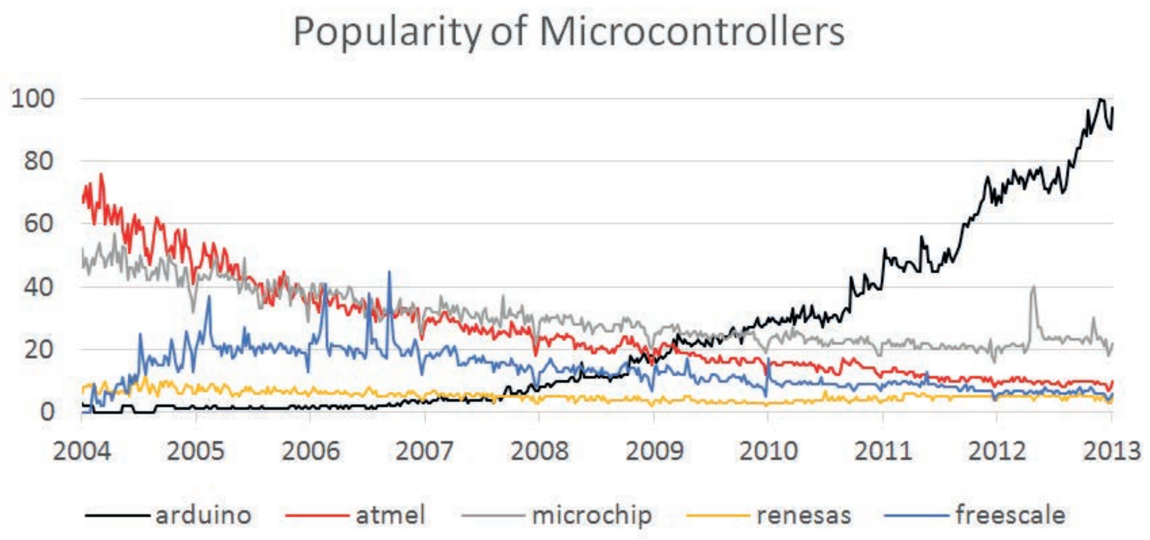

Figure 2. Google trend results for most common MCU-s Manufacturers (Google Trends, 2013).

With latest changes in electronics to cheap and popular open-source circuits and boards, building a small demo robot is surprisingly easy. Arduino ©, MSP430 ™, Lego Mindstorm NTX ®, IntelliBrain Bot ™, Lunxmotion Hexapods are just few of the examples of modern platforms that need very low knowledge to start but have almost unlimited options customize and from into cool robot. As a result of investigation and personal experience, table 2 has a revision of commonly used development boards by researchers and hobbyists. Cost and size of electronics demo boards have dropped in recent years to just few dollars, i.e. MSP430 from Texas instruments have price tag of only 9,99 USD and expedited shipping is included (Texas Instruments, 2013b). It takes only few hours to install all the needed software, write your first blink the LED code, connect the board to computer and program the MCU.

Table 2. Comparison of popular ready to use demo-boards
(PJRC, 2013; Texas Instruments, 2013b; Arduino, 2013; Raspberry Pi, 2013).

Name of the board	Teensy 2.0	MSP430 Launch-pad	Arduino Leonardo	Rasperry Pi
Size W x H x D	3,0 x 1,8 x 0,5	6,0 x 5,0 x 1	6,9 x 5,1 x 1	8,6 x 5,4 x 1,5
Processor	ATMEGA32U4	MSP430G2553	Atmega32u4	ARM1176JZF-S
Working frequency	16Mhz	16MHz	16Mhz	700 MHz
Flash Memory	32 kB	16 kB	32 KB	4GB (SD)
RAM Memory	2.5 KB	512 B	2.5 KB	512 MB
EEPROM	1 KB	0	1 KB	0
I/O pins digital	25	14	20	8
Analogic input lines	12	8	12	8
PWM Channels	7	8	7	
UART, I2C, SPI	1,1,1	1,1,1,	1,1,1	1,1,1,
Working voltage	5V	3,3 V	5V	5 V
Supply voltage (recommended)	5V (USB)	5V (USB)	7-12	5V (USB)
Programming environment	Teensy Loader, WinAVR	Code Composer Studio 5	Arduino Software	LXDE, IDLE3
Price	16$	9,99$	21 $	35$

If you cannot use prefabricated boards, because of the size, performance or some other limitation, you have to design your own electronics and choose every component. It is not recommended for small and medium size projects as it can take many weeks of debugging and testing to get your own board to work. Additionally, many larger robots nowadays have smart-phone or laptop onboard, which is connected to low level electronics via UART or USB. Having a computer onboard gives high processing power for complex tasks like image processing and path planning. But also makes decision making and communication between processors more complicated and erroneous.

Software

Writing a code to a robot and giving them intelligence is relatively easy, you just have to install IDE (Integrated Development Environment) that corresponds to your choice of MCU manufacturer and choose the exact model of MCU. After that you can open or download demo programs for all mayor functionality and modify it to your needs and configuration. This means that fair amount of debugging and testing are still needed to get things working but usually it is not so frustrating and help from well written documentation or Google would answer your problems eventually.

In robotics Software can be divided into two large groups (MacDonald, Biggs, & Collett, 2007):

1. Microcontroller based software (8-bit, 32-bit, ARM and FPGA architectures)
2. Computer run software (x86 or AMD64 architecture)

The main difference is versatility and computing power. Microcontrollers usually work at low speeds like 20 MHz and are not suited to do heavy calculations like image processing or path planning. Nevertheless most robots today are controlled by MCU-s and all intelligent robots have at least one processing unit on them. Modern general computer processors have high performance but are not able to communicate directly with real world, which means they do not have A/D converters nor are they suited to communicate with low speed interfaces. Designing and writing software to be run in a computer is quite similar to regular software development process. Even a few years ago most of the microcontrollers were programmed in low level languages like Assembler. Currently, the evolution of compilers, open source libraries and higher performance have led to extensive use of medium or high level languages like C, C++, python and java. One thing that hasn't changed is that you still must know to which pin each of your sensors are connected to and basic knowledge about functioning of the MCU will be tested quickly also as in real world not all things start working magically.

This means that programming a robot is easier than ever and an experienced professional using advanced programming languages and libraries can create a working code with very limited timeframe and effort. Even though Integrated Development Environment (IDE), like Atmel ® AVR Studio or Code Composer Studio, have automated many hardware and low lever functions from programmers, the software still needs to be written specifically to every family of processors and have to work closely with hardware. This means, reusing your code is limited and software in robotics is hardware dependent. Using IDE with high level programming language and libraries can speed-up your code writing many times and is key element to fast development of software. Unfortunately some of them can cost up to 1000 USD for commercial company, but luckily most of them also have free limited edition or cheap educational license which can easily be used in universities and smaller projects (Atmel Corporation, 2013; Texas Instruments, 2013a).

Almost all agile software development methods can be used in robotics also, especially in computer-run software. The Software Development Life Cycle (SDLC) is an extremely efficient method through which a software product is developed from scratch, or is rid of any problem. It consists of phases, procedures and steps that guide a developer from the beginning to end of the Development process. Some of the popular process models currently available include Spiral model, Rapid Application Development (RAD), Iterative model, V-model, and the Waterfall model. The Waterfall model is the most applied model in the field because of its versatility and clarity. It can easily and clearly be applied to projects of varying sizes. It also involves a lot of testing and documentation, thereby making it easy even for outsiders to follow and identify any problems with the product.

The waterfall process consists of 6 main phases that each consist of tasks that should be completed within that phase. The 6 main phases are: investigation, Analysis, design, coding, testing, maintenance. Even though it is relatively modern and successful development method, RAD and agile development can be more efficient in robotics as code size and development teams are usually smaller.

According to, M. Iqbal and M. Rizwan. Application of 80/20 Rule in Software Engineering Rapid Application Development (RAD) model, from 115 activities of the Rapid application development (RAD) model 70 were marked in questionnaire as important (giving 80% of the result). Also two years before they published an paper *Application of 80/20 rule in software engineering Waterfall Model,* where they were able to show similar results for Waterfall model. With that information they were able to show clear improvement of efficiency over non-edited development model. Complete list of the activities to be ignored can be found in the original papers (Rizwan & Iqbal, 2011; Iqbal & Rizwan, 2009). Example of activities that could be ignored:

- Formulating alternatives
- Collecting Non-functional Requirements
- Doing analysis of Non-functional Requirements
- Gather and establish Non-functional Requirements
- Designing and documenting the algorithms of the modules
- Design the Project Definition and Management Procedure
- Design the Software Programming Procedure
- Documenting all these tasks
- Test the code using Reviews
- Reviewing the code tests
- Approving the change by the user/ client
- Create test data like Live Data and Sampling
- Provide audit or review reports
- Develop system test requirements (Standards)
- Retire system.

Mechanics

Mechanics gives articulation and movement to robots. It is very hard to give any clear recommendation about hardware and the skeleton of a robot, as every robot is different. Nevertheless, the key rule is to keep it simple, but rigid. Just a very simple task as teaching a robot how to drive a straight line can be incredibly hard for a robot with bad mechanics. That said, designing, electric tools used, choosing the material, motors and functionality as a whole must be well considered. Having a good body of a robot can save many hours of programming a complicated control algorithms and testing.

Best recommendation is to learn from others mistakes. Take a day or two to Google and investigate similar solutions to your need and search for a prefabricated body or schematics that could be modified. If you already have some experience and can plan ahead using a 3D-printer or CNC machine can give you precision of micro-meters and speed up your construction many times. The modern prototyping technology are nowadays relatively affordable and best way to test your design and functionality in small quantities. Choosing a right motors or batteries are very important also as large motors can give you a lot of power but weigh a lot and need large batteries. DC and stepper motors are still very popular but newer technologies like brushless motors are good alternative if high power or size limits are desired. Currently almost all robots use Li-Po or similar technology batteries which are affordable, light and support high energy needs without problems. Though Li-Po might seems more expensive and problematic than NiMH or Lead-acid batteries they are worth the money as they are more reliable and versatile. If Li-Po batteries still seem expensive in your local electronic store, ebay and other online vendors currently have really good offers and fast shipments. For small Li-Po batteries price of 1 USD for 1 Watt hour is generally accepted, for larger batteries like 3-cell 11,1V the price is around 20 USD which result in 0,6 USD for Wh (McComb, 2008).

Cost of the Product in Project management

In the free global market economy there are many providers and customers are searching for highest quality with a lowest price available. Therefore, companies have to lower the price for the clients and increase the quality. In reality, this is very hard to achieve, but there are still ways a company can cut the budget of a project. In robotics a product can be: a working robot (hardware), a software that controls the hardware and automates tasks; or a service that makes sure everything works and prevents or fixes errors. In order to provide best quality and satisfaction to users, many modern companies have decided that technology is a complex machine and thus they have to control the full cycle from prototype to servicing finished product (Grossman, 2011). In this paper we consider a product being mixture of hardware, software and service for the end user.

In information technology, software product cost can also be considered the cost of the labor required to deliver a full service to a customer. Minimal product cost is the cost of direct labor, direct materials, and manufacturing overhead that are consumed to create the product. This can be true if the number of products fabricated are large, but for smaller projects planning, designing, testing and managing comes with an additional cost also. The formula 1 describes the cost of a project.

$$(1)\quad \begin{matrix} Project \\ cost \end{matrix} = \begin{matrix} Manufacturing \\ cost \end{matrix} + \begin{matrix} Development \\ cost \end{matrix} + \begin{matrix} General \\ cost \end{matrix}$$

Correct project cost or budget planning is vital to long-term and sustainable business. It can be defined as a sum of manufacturing cost (fabrication), Development (design and testing) and general cost (management). Project cost can be divided into a number of products created multiplied by the cost of one unit. It is also quite clear to divide manufacturing cost to the cost of material needed make one product plus labor cost for manufacturing times number of products fabricated. Development cost and general cost are complete indirect costs and thus have no direct correlation with how many units are fabricated. They depend on complexity and structure of the company, for better overview we can add them together and call overhead. The result can be seen at the formula 2, where *N* refers to number of products made.

$$(2)\quad N \times \begin{matrix} Product \\ unit\ cost \end{matrix} = N \times \left(\begin{matrix} Material \\ unit\ cost \end{matrix} + \begin{matrix} Labor \\ unit\ cost \end{matrix} \right) + \begin{matrix} Overhead \\ total\ cost \end{matrix}$$

From formula 2 we can calculate the *Product unit cost* which can be seen at formula 3.

$$(3)\quad \text{Product unit cost} = \frac{\text{Total direct materials} + \text{Total direct labor} + \text{Total overhead}}{\text{Total number of units}}$$

In formula 3, *total direct materials* are the cost of material in the manufacturing and testing phase; *total direct labor* is the time and effort needed to manufacture and test the number of products; and *allocated overhead* is the development cost, general expenditure to keep company running, cost of management and profit. In another words, the cost of a product on a unit basis is typically derived by compiling the costs associated with a batch of units that were produced as a group, and dividing by the number of units manufactured.

Now we have identified different parameters for evaluating the efficiency of a project and cost of the product. It is clear that lowering the cost of material result in a cheaper product and is general recommendation for any projects. In robotics, choosing cheaper hardware results in many cases in higher labor cost as hardware has no functionality without software or results in lower quality and thus in higher failure rate or service need. Almost every decision in robotics projects have impact in another phase of the project and has to be well calculated. For example choosing smaller microcontroller and saving a few dollars means that the programmer has code-size and performance limits of the microcontroller.

The main question in robotics projects is the quantity of products manufactured. If the quantity is low and the functionality is limited, the product can be called prototype, prototypes can be developed and designed into demo models which are basically a limited edition or trial product of full scale manufacturing. End product is usually the electronic or robot you can buy off the shelf or order online and is generally available to public. Overview of products and their respective unit price can be found in table 3.

Table 3. Characteristics of prototype, demo model, and finished product in robotics

Product type	Quantity Manufactured	Functionality rate	Cost of development per unit	Manufacturing cost per unit	Price per unit	Total cost of project
Prototype	Few (1-3)	Two sigma (>69%)	Medium (30% time)	Medium	Medium	Low
Demo model	Some (1-10)	Tree sigma (>93,3)	High (70% time)	High	High	Medium
Finished Product	Many (>10)	Four sigma (>99%)	Low (100% time)	Low	Low	High

The cost of the management and expected profit is generally in correlation to the total budget of the project, not to quantity of products. It depends on the company, but generally 10-20% of cost of the robotic project is calculated as general expenditure and is added to the expected total cost of the project. The cost of the development is in correlation with the quality and functionality, which means that in order to make a better product you need more designing, development and testing. Thus making more products results in a lower unit price as development cost for one or 100 units is very similar in robotics.

For many years, robotics and robot projects in universities were mainly for education and innovation, doing things as easy as possible with as much functionality and high working rate as possible (building prototypes). Industrial robots are very different from another type of robots as their functionality is quite limited and the main design goals are low error rate and cheap manufacturing cost. Still, many of the projects executed in the universities include 80% of research and innovation and 20% of the time-consuming tasks as quantities are small and companies are doing feasibility studies with the help from Universities. Technical development of processors and electronics and boost in consumer robotics have gave us limitless opportunities to create robots that have the demand on the market all over the globe.

Conclusion

In this paper we reviewed different aspects of robotics construction and project management. Robotics are very interdisciplinary area and if you save money by buying cheaper components for fabrication you probably increase complexity of the solution and thus increase labor cost in design and development. By choosing cheapest available parts for electronics you find out soon that you have power supply problems or unpredictable brownouts. Thus for the authors of the document the solution to fast development of robotics projects and lower projects cost is by optimizing not cost of manufacturing but by optimizing cost of development and more precisely minimizing engineering and testing hours. This will result in more time for testing and better quality of product.

The key point to efficient and profitable project are:

- improving motivation and efficiency of an engineer by choosing solutions and tools that will result in lower development cost,
- lowering the amount of management and general running cost of enterprise,
- using prefabricated electronics and mechanics to save time from re-designing and re-fabrication,
- using high level programming languages like C, C++ and java with libraries and publicly available code examples,
- save testing and redesigning time by choosing high quality components and 30% more power output for motors and batteries than mathematically needed.

Obviously every project is different, thus methods, tools, parts and solution must vary. Author of the paper, have participated in a project where we had large budget for electronic components and the next project we had almost zero budget to buy hardware as client told it is best to solve most tasks in software, because it makes final unit price very low. To sum up, be adaptive and responsive to change in the market and find a ways to be more efficient in all areas and motivate your engineers to me more productive and innovative in their solutions.

References

Agile Resources. (2013). Retrieved from Agile Leadership Network. Retrieved from: http://www.agileleadershipnetwork.org/resources/

Arduino. (2013). *ArduinoBoardLeonardo*. Retrieved from http://arduino.cc/en/Main/arduinoBoardLeonardo. July 15 2013.

Atmel Corporation. (2013). *Atmel® Studio 6*. Retrieved from Supporting Two Architectures: AVR and ARM, with One Integrated Studio. Retrieved from: http://www.atmel.com/microsite/atmel_studio6/.

Batelle, J. (2005, December 1). Thr 70 percent solution. Retrieved from: http://money.cnn.com/magazines/business2/business2_archive/2005/12/01/8364616/index.htm.

Bloch, M., Blumberg, S., & Laartz, J. (2012, August 21). Delivering large-scale IT projects on time, on budget, and on value. *Financial Times*. Retrieved from: http://www.ft.com/intl/cms/s/0/d34acf86-eba8-11e1-9356-00144feab49a.html#axzz2Hsj9i2uy.

Databeans. (2012). 2012 Microcontrollers - Market Research Report. Databeans Inc.

Garage48. (2013). Retrieved from From idea to prototype in 48 hours: http://www.garage48.org/about.

Google Trends. (2013). *Web Search interest: arduino, atmel, microchip, renesas, freescale - Worldwide, 2004 - present.* Retrieved from: http://www.google.com/trends/explore#q=arduino,%20atmel,%20microchip,%20renesas,%20freescale.

Greenstein, S. (2010). Chapter 11 - Innovative Conduct in Computing and Internet Markets. In E. Bronwyn H. Hall and Nathan Rosenberg, *Handbook of the Economics of Innovation* (Vol. 1, pp. 477-537). North-Holland. doi:10.1016/S0169-7218(10)01011-7.

Grossman, L. (2011). A Man in Full. *Time Magazine, Steve Jobs The Genius Who Changed Our World*, 78-89.

IFR. (2012a). *International Federation of Robotics*. Retrieved from Industrial Robot Statistics 2012. Retrieved from: http://www.ifr.org/industrial-robots/statistics/.

IFR. (2012b). *International Federation of Robotics*. Retrieved from Service Robot Statistics 2012. Retrieved from: http://www.ifr.org/service-robots/statistics/.

Iqbal, M., & Rizwan, M. (2009, August 15). Application of 80/20 rule in software engineering Waterfall Model. *Information and Communication Technologies, 2009*, 223-228. doi:10.1109/ICICT.2009.5267186.

Koch, R. (2004). *Living the 80/20 Way: Work Less, Worry Less, Succeed More, Enjoy More.* London, UK: Nicholas Brealey Publishing.

Lombardo, M. M., & Eichinger, R. W. (2000). *The Career Architect Development Planner* (3rd ed.). Lominger Limited.

MacDonald, B., Biggs, G., & Collett, T. (2007). Software environments for Robot Programming. In D. Brugali, *Software Engineering for experimental Robotics* (pp. 107-124). Berlin, Germany: Springer.

Mayer, M. (2006). Innovation at Google. Stanford University. Retrieved from http://www.youtube.com/watch?v=soYKFWqVVzg.

McComb, G. (2008, December). So You Want to build a Robot. *Servo Magazine*, pp. 62-65.

Narula, A. (2005). *80/20 Rule of Communicating Your Ideas Effectively.* UBS Publishers Distributors Limited.

PJRC. (2013). *Teensy USB Development Board*. Retrieved from: http://www.pjrc.com/teensy/index.html. July 15 2013.

Raspberry Pi. (2013). *An ARM GNU/Linux box for $25. Take a byte!* Retrieved from: http://www.raspberrypi.org/. July 15, 2013.

Ressi, A. (2011, May 28th). *Is There A Peak Age for Entrepreneurship?* Retrieved from: TechCrunch: http://techcrunch.com/2011/05/28/peak-age-entrepreneurship/.

Rizwan, M., & Iqbal, M. (2011). Application of 80/20 Rule in Software Engineering Rapid Application Development (RAD) model. *ICSECS 2011*, 518-532. doi:10.1007/978-3-642-22203-0_45.

Texas Instruments. (2013a). *Code Composer Studio.* Retrieved from Integrated Development Environment (IDE) v5. Retrieved from: http://www.ti.com/tool/ccstudio.

Texas Instruments. (2013b). *MSP430 LaunchPad Value Line Development kit*. Retrieved from MSP-EXP430G2 - TI Tool Folder: http://www.ti.com/tool/msp-exp430g2. July 15 2013.

The Robot Report. (2013). Retrieved from The Robot Report's Global 1,000 Robot Makers. Retrieved from: http://www.therobotreport.com/index.php/site/TRR-Global-Map/.

Tiigrihüppe SA. (2013a). *Kooliroboti projekt*. Retrieved from: WikiRobo: http://robootika.ee/index.php/Kooliroboti_projekt. July 15 2013.

Tiigrihüppe SA. (2013b). *ProgeTiiger*. Retrieved from http://www.tiigrihype.ee/et/progetiiger. July 15 2013.

Author

Alo Peets

Is robotics and computer engineer originally from Estonia, University of Tartu, where he finished his University studies in 2010 with master thesis "The Solar System Model for Tartu Old Observatory". In five years (2006-2011) working as robotics engineer in Institute of Technology of University of Tartu he participated and successfully finished more than 10 small and medium size robotics projects. From September 2011 till July 2013 he worked in the Universidad Simón Bolivar (Colombia) as a professor and investigator.

apeets@unisimonbolivar.edu.co