Embedded Systems for
Computational Garment Design

Megan Lee Galbraith

B.S. Mathematics with Computer Science,
minor in Women's Studies
Massachusetts Institute of Technology, 2001

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology
June 2003

Megan Galbraith
Program in Media Arts and Sciences

John Maeda
Associate Professor of Design and Computation
Thesis Advisor

Andrew Lippman
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Embedded Systems for
Computational Garment Design

Megan Lee Galbraith

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology
June 2003

Abstract

In an age where identity is increasingly fluid and multifaceted, the static clothing and unresponsive materials we wear are often an insufficient means of expression. Clothing designers want to create systems of clothing that react, collect information, and enrich our interactions with spaces and people; however, technical barriers inhibit designers interested in building computational garments. Designers need a tool that is attainable and usable in order to successfully work in the field of computational garment design.

This thesis introduces a powerful, intuitive tool named Zuf which provides a new approach to control embedded devices using fuzzy logic. Zuf is a prototyping and simulation environment for programming and testing embedded devices. Users write code by establishing simple, natural language rules. The rules are translated into fuzzy algorithms which run on the devices. Zuf enables fashion designers to think abstractly about computation as a medium.

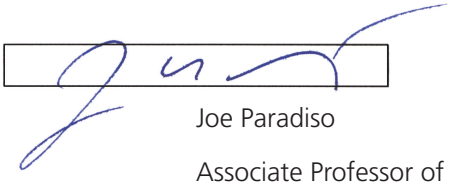
Thesis Supervisor: John Maeda
Associate Professor of Design and Computation

Embedded Systems for
Computational Garment Design

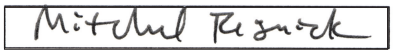
Megan Lee Galbraith

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology
June 2003

Thesis Readers



Joe Paradiso
Associate Professor of Media Arts and Sciences
Responsive Environments Research Group
MIT Media Laboratory



Mitchel Resnick
Associate Professor of Media Arts and Sciences
Lifelong Kindergarten Research Group
MIT Media Laboratory

Table of Contents

Computers, like finger paint and beads, should be used as a "material" for making things...

Mitchel Resnick, Closing the Fluency Gap, 2001

Chapter One
Introduction

Clothing has been used to transform the body and alter our understandings of self and the human form for years. Much like corsets and bustles work as underlying systems to mold, shape, and support our body and clothes, we can use new technologies, materials, fabrication processes, and computational systems to adapt to, respond to, monitor, and alter our bodies and clothing. Through these systems we can create rich new experiences and modes of expression.

Technology as a medium for design enables new instantiations of style and personal identity. Applications for technology that relate to the physical human form are rapidly developing as lighter and more robust components get introduced to the market. Despite the growing interest in technology as a means for expression - particularly in the area of fashion design - no discussion or vocabulary exists about how technology changes our definition of fashion or transformations of the body.

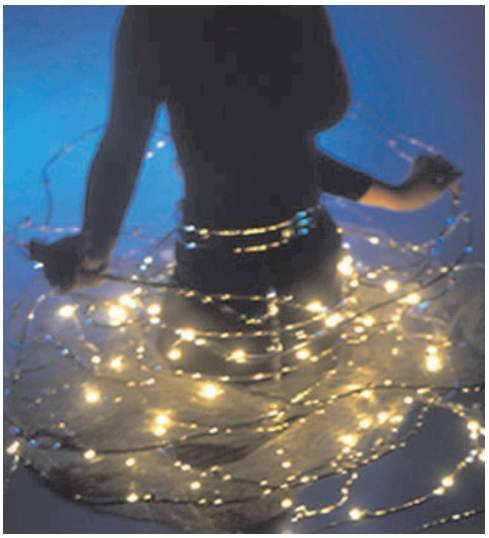
Work developed in the area of wearable computing focuses on the technical capabilities of garments instead of the aesthetics. It is not simply a matter of sewing MP3 players and cellphones into our jackets. Nor is it simply about being able to wear personal computers rather than carrying them in cases.

Introducing concepts of aesthetics, kinetic sculpture, and visual design into the field of wearable computing changes the nature and the goals of the work. The focus shifts from the technology to the garment, from functionality to context. This shift marks the foundation of a new area of research called computational garment design. Computational garments are garments that contain embedded technologies designed to enhance clothing and make it reactive or dynamic. Computational elements can breathe a sense of life into previously inanimate objects, redefining how we relate to, wear, and think about our clothes.

The field of computational garment design is starting to take root amongst technical fields of research, but fashion designers are unable to use computation in their designs because technical barriers are generally too high, too impermeable, or too intimidating. Developing new approaches to control and use embedded devices enables fashion designers to think abstractly about using computation as a technical medium for their designs without getting thrown off by the burden of implementation. In order to encourage fashion designers to think about and use embedded devices in their work, this thesis focuses on the development of a new environment designed using a web-based, fuzzy logic programming model, shifting away from traditional device control practices.

The chapter in this thesis entitled "Background" is designed to give an overview of the different areas of research pertinent to this work. These areas include Wearable Computing, Computational Garment Design, Embedded Systems, Fuzzy Logic, and Computational Literacy. The chapter contains a discussion of each area and provides samples of work and references to related papers or writings. It is intended as a resource for others interested in this vein of research.

The next chapter, entitled "Preliminary Work," contains additional motivation for this research which stems from a collection of computational garments that I constructed during the period between September 2001 and March 2002. Each of the garments brought to light many of the challenges, needs, and frustrations that designers might encounter during the development process of a computational garment. Elroy, Iris, Peppermint and the others were indispensable projects for testing and exploration. They carved out the functionality and needs of a development tool aimed for designers. Each garment brought to light at least one important element implemented in the Zuf software. Their contributions are highlighted in the third chapter.



An illuminating skirt created by designer Erina Kashihara.

The fourth chapter, entitled "Zuf: A Fuzzy Control System," gives a detailed description of the fuzzy programming tool including the interface design, software and hardware design, and interaction design. The fifth chapter, "Analysis & Applications," provides an in-depth analysis of the system, including people's experiences using the system, its ability to empower designers, and its weaknesses. The thesis concludes in the final chapter, "Conclusion," which reiterates this area of research and its approach, then discusses where the field is going in the future.

Empowering Designers

For adults, computer programs exist that provide control capabilities for hardware and development spaces for software, but these programs are typically not for novice users. An inexperienced adult interested in building a project with a microcontroller has a variety of options to chose from, but the truth is that most programs are ineffective tools for learning the underlying concepts. The programs are difficult to set up, difficult to understand, and extremely intimidating. Some provide high level languages to use to control the devices, but the languages still require a basic fluency in writing procedural programs. Others require a deep understanding of the circuity in order to write programs. Users must rely on dense maps and complicated circuit diagrams that show the connections between pins, buses, and other elements in order to write working code.

Meanwhile, educational tools developed for the computer might help designers use computation in their work; however, they are primarily designed for children. Bright colors, cartoon-like characters, animals, associations with toys, silly sounds, and graphical elements construct the interface for a myriad of programs designed as exploratory spaces. Through these programs, children can learn about and understand the complex world of computation. These programs are powerful because they provide a stepping stone for children to explore tasks in relation to computation, mathematics, science, identity, and, in some cases, social responsibility. But these same programs fall short for adults who want to use computation for their professional work.

Adults with weak mathematics or science backgrounds think very differently about computers and computation than adults who come from stronger technical backgrounds. This is not to say that "non-technical" adults

will not (or do not) find computers useful in their daily life, work, or hobbies. In general, people are becoming more adept at using computer programs or familiarizing themselves with operating systems and simulation environments. Many adults use computers to accomplish tasks such as word processing, email, or commerce. The catch is that these adults operate in abstraction levels far removed from the internal structure, design, and operation of the computer as a machine.

Hobbyists and computer enthusiasts have a conceptual model of the computer as a machine and therefore a desire to mess around with its functionality. My father is one of these enthusiasts, and in his spare time he likes to add internal devices, reinstall operating systems, or fuss with his scanner and digital camera. His approach is methodical and calculated. Unlike engineers or experienced hackers, however, he is fearful of damaging the computer, so he reads the manual and instruction booklets with fervor. His progress is slow, precise, and safe. Adults like my father have the confidence to open up machines and toy with their controlling mechanisms, but they often have no space in which to do so with complete freedom or lack of fear. We need an educational tool that helps my father, fashion designers, and other adults delve into complex ideas about computation and programming. Yet no appropriate tool exists.

Development tools designed for embedded devices assume a complex understanding of computation and are no place for beginning users to explore and build computational projects. Some commercially-available, high-level packages include the BasicStamp, MiniJava, I-Cube, and others. The BasicStamp, for instance, is a development platform geared towards beginners which couples a software development tool with microcontroller. Its reliance on a form of the BASIC language, however, creates a confusing and difficult barrier for adults who have no programming experience. Since most development platforms are not geared toward beginners, they are dramatically complex and difficult to use.

Educational tools, on the other hand, are generally available to people of any age, but adults are less likely to use the programs because of age-related taste differences or irrelevance to their occupation. No clear connection exists between the output of these programs and the tasks adults need to accomplish in their career. There is little motivation for adults to conceptualize useful projects to build with computational devices because the appropriate tools do not exist for use by anyone without an engineering degree. Computationally educational programs are associated with children's toys. "Real work" can't be accomplished with a child's video game, play calculator, or educational computer program. A designer, for instance, would not feel comfortable showing a prototype garment controlled by LEGO parts to their client when they are expected to build and design professional work.

How, then, are adults supposed to learn about and use embedded devices if there is no useful resource available to them? It's infeasible to expect them to learn everything in a college classroom. Without a colleague or peer familiar with the work to guide them, the task is daunting enough to be avoided altogether.

Building Blocks

For this thesis, I developed a system called Zuf for fashion designers and non-technical adults interested in building computational garments or programming embeddable devices but do not have the technical skill, background, or intuition to delve into such projects alone. After building computational garments for over a year, I realized there should be a better method for developing these garments. Each time I set out to build a new garment, I had to reimplement the same or similar programs and circuits that either I or other researchers had built many times before. I felt like I was reinventing the wheel.

I spoke to many students and designers interested in building these types of garments. Across the board, they were all highly capable of developing strong concepts for computational fashions, and were highly capable of designing non-functioning prototypes of their ideas. However, they were rarely able to bring these ideas into reality and make actual, functioning garments.

At the time, my colleagues and I were in the process of developing Nylon, a computational system for programming and simulating microcontrollers. Through this work, I realized the benefit of having a highly integrated, aesthetically oriented system for novices and designers to use for experimentation and development. It seemed obvious that such a system was needed for fashion designers and other adults interested in building computational garments, especially for those who do not have the benefit of being able to study technology in school or work around highly skilled, technical people. It needed to be a system that was powerful and elegant to use, as well as easy to pick up. The system needed to let people teach themselves how to use and build computational projects without the headaches that are so common during traditional hardware development.

Two things seemed particularly important as I ventured into this work. First, the system needed to have a viable amount of abstraction away from the hardware specifics. Second, there needed to be a strategic step away from traditional programming methodologies. The people who I imagined would be interested in this system wouldn't care at first about the underlying architecture of the devices, sensors or hardware they want to use. For the purposes of their work, there is no need for them to. Therefore, if the field of computational garment design is to really open up and blossom as a new approach to designing, building, and thinking about clothing, then such walls must be torn down.

It is not entirely pertinent for a fashion designer to be an expert programmer or hardware engineer in order to create beautiful or evocative designs for their computational garments. It is pertinent, however, for designers to use computational materials comfortably and with some amount of control in order for the designs to move off the sketch pads and into the physical world. The Zuf system developed for this research does not require a perfect understanding of electrical engineering or programming in order to be useful and effective, nor does it take away complete control of the device from the user. The system is instead designed as a gateway between two worlds, allowing room for experimentation and exploration before requiring a hard leap into the technological unknown.

A Fresh Outlook

Generally the approach for programming small devices begins with mastering specific applications which require complex initialization procedures to access the devices. Situating the Zuf environment within a website makes the process more familiar for designers. The explosion of Internet use over the last decade has resulted in general familiarity with website visitation, email checking, online shopping, etc. The web is a medium used for many purposes by people from varieties of backgrounds and education levels. A successful website developer designs for the user-experience of a site because it is critical for their client's business to ensure that users can utilize the software to its full capabilities and intentions. Thus the web becomes a powerful tool for reaching people who are wary of technology but have found comfortable spaces to occupy online. It is precisely for this reason that Zuf utilizes the web as a space for hardware development, and demands the same attention to user interface design as a high traffic website.

There is an intellectual opportunity in this domain of work. Computational garments will emerge as key actors in our lives once they are accessible, inexpensive, and well-designed. They offer the ability to provide new dimensions for interaction, performance, and social cues, not to mention they will enhance the aesthetics of clothing. Currently, the outfit or uniform we choose to wear each morning generally remains on our bodies throughout the day, and yet each evening the garment knows nothing more about our lifestyle, friends, or work environment than it did when it was donned. Computational garments can provide elegant and unobtrusive ways to gather, store, and access information. Medical and rescue industries might employ particular concepts and technologies to improve missions and increase performance, while athletics and sporting industries might use them to monitor and enhance the performance of athletes and teams. Computational garments can provide new ways to interact with people and spaces, and provide cues about identity, belief systems, sexuality, or economics. Such results should bring forth notable social behaviors or expressions of self, or evoke new types of relationships.

Initial technological developments required to develop computational garments are already in place, but there is a large amount of work yet to be done. Appropriate technologies, education, vocabulary, and a community must be established. Collaboration between many industries is critical to the success of new fashion design. Wearable computing researchers have made considerable headway in the development of electronic components that are small, lightweight, robust, or flexible. The fashion industry, however, is quickly falling behind in terms of adapting to and adopting these new technologies. The tools for designers need to be developed in order for the field to avoid becoming a niche, strictly accessible to computer scientists and electrical engineers. The research for this thesis attempts to lay the foundation for some of these needs.

"Visually, things have to make sense, in an almost mathematical way. I'm not anal or anything.. but when it comes to what I wear, I'm very precise. I know what's going to work, and I know what's going to suck."

Cameron Diaz, in Vogue, May 2003.

Chapter Two

Background

This chapter gives an overview of related work in the fields of Wearable Computing, Computational Garment Design, Embedded Devices, Fuzzy Logic, and Computational Literacy. Each of the five sections is distinct in goals and work; however, they are equal contributors to the foundation upon which this research is built. The ideas, technical innovations, and images presented in this chapter have served as inspiration and motivation throughout the construction of the Zuf system and its preceding projects.

Wearable Computing and Related Work

Steve Mann, a pioneer in the field of wearable computing, defines a wearable computer as a computer that is small, worn on the body, and taken into the personal space of the user. (Mann, 1998). Wearable computers are always on and always accessible, "more than just a wristwatch or regular eyeglasses: [they have] the full functionality of a computer system." (Mann, 1998). Generally, these devices consist of eyewear, helmets, belts, vests, and a variety of bulky gear. Entire systems of garments and accessories contain the workings of a fully functional personal computer.

Wearable computers began development in the late sixties, when Morton Heilig patented a stereophonic television Head-Mounted Display. Heilig is known more for his "Sensorama Simulator," a virtual reality simulator developed in 1962 with handlebars, binocular display, vibrating seat, stereophonic speakers, cold air blower, and a device close to the nose that generates odors which fit the action in a corresponding film (Rhodes, 1997).

Wearable computers from the 1970's (top) and 1980's (bottom). Images taken from Jay Levine's photoshoot prior to the "Origins of Cyberfashion" show during the 2000 TED Conference. University of Toronto, 2003.

The 1970's brought the development of the first wearable devices, as well as one-handed keyboards, eye-mounted displays, and Hewlett Packard's algebraic calculator wrist watch (Rhodes, 1997). Early wearable computers initially challenged the idea of computers as immutable constructs. They specifically addressed the ability to run computers on batteries and, for the first time, brought the computer and human into mutually beneficial relationships (Wearable Computing, 2003). Thanks to work of wearable computing researchers throughout the seventies, eighties, and nineties, traditional interaction between human and computer is no longer limited to the comfort of a desk chair.

Wearable computer of the 1990's. Image from Jay Levine's photoshoot prior to the "Origins of Cyberfashion" show at the 2000 TED Conference. University of Toronto, 2003.

Mann makes the distinction that wearable computers are reconfigurable and programmable by the wearer (Mann, 1998). Computational garments, in contrast, are not required to be reconfigurable, although they can be. A common misconception is that computational garments are synonymous with wearable computers. The fields overlap and depend upon each other in many ways, however they come from different ends of the continuum and it is important to understand their distinctions.

Computational garment design concerns itself with the aesthetics of garments enhanced by technology or innovative materials with reactive properties. The technology is not the focal point of the garments, rather it is the aesthetic beauty, the unique interactions, and the interesting behaviors that emerge as a result of embedding technology in clothing. Technology in computational garments should be as invisible as the thread that holds the fabric of the garment together.

Scrolling text T-shirt from Cyberdog, 2003.

Wearable computing, on the other hand, is concerned with the functionality, robustness, and usability of the technology as it inhabits spaces on the body. To the general public, the phrase "wearable computing" elicits images of cyborg humans donning devices that transform them into something untouchable and unapproachable. Technical features offered by wearable computers take precedence in design over the aesthetic and physical properties of the garments that house them. Wearable computers tend to be considered a fashion faux pas. Bruce Knaach, a manager at IBM working on the development of wearable computers on the industry side explains,

> *Wearing a computer isn't a socially acceptable thing. You look like a soldier wearing half a helmet.. kind of Borg-ish. That forces it [the devices] into environments where people either don't care what they look like or have it as a condition of employment (PCWorld, 1999).*

In defense of wearable computers, when headphones and portable music devices were introduced in the late seventies, no one expected them to become a trendy, techno-accessory. Yet now they have a market all their own, with the most sophisticated, funkiest, sleekest, and ergometric designs landing on the shelves to date. People looking for a way to block aural stimuli from their surrounding environment and place them in a netherworld of pleasing sounds turn to their headphones. They base purchasing decisions on form, color, and technical quality. As Sony touts on their website, headphones have become "Not just an audio accessory, a fashion accessory..." (Sony, 2003).

It can be argued that headphones are in fact the first product in the field of wearable computing which has successfully entered the consumer market and become accepted by mass culture. We might wonder why wearable computers haven't followed suit in popularity or trendiness, since technically they offer leaps and bounds more functionality than your basic portable music player and headphones can provide. Perhaps it is because the physical devices are bulky and awkward, conjuring images of cyborgs or unpleasant science fiction characters, or perhaps they need only withstand the test of time to slowly creep into public consciousness and acceptance.

Wearable computers are becoming a steadfast fixture in Internet lore. Their strong cult-following among scientists, electrical engineers, and Internet junkies might soon be their redemption, for these are the people doing the research that will eventually make the devices smaller, more efficient, and potentially more sleek and socially acceptable. Academic communities are firmly in place for wearable computing research. Over the last six years, the Institute of Electrical and Electronics Engineers (IEEE) has held an International Symposium on Wearable Computers (ISWC).

A sketch of the *Mushroom Cap: The Year 2018*, a garment designed by Nanae Hashimoto, Ai Mizuno, Seonhyu Na, and Jennifer Healey for the MIT Media Laboratory's Wearables fashion show. 1997.

At the MIT Media Laboratory, Prof. Sandy Pentland, Rich DeVaul, Thad Starner, the Wearable Computing group, MIThrill, and other faculty and research students have worked in this domain throughout the last twenty years. In 1997, the Media Laboratory hosted the first wearable computing fashion show, called "Beauty and the Bits." The show was a collaborative effort between the Media Laboratory, Bunka Fashion College in Tokyo, Creapole in Paris, Domus in Milan, and Parson's School of Design in New York (MIT Wearables, 2003).

Other institutions have joined MIT in wearable computing research. Stanford University's Computer Science Department contains a Wearable Computing Laboratory which works on the "design of highly wearable general purpose PCs and improved technologies for the human interface to wearable computers" (Stanford University, 2003). Their research includes work in the areas of speech input and output and digital sign languages (Stanford University, 2003). The University of Oregon's Wearable Computing Laboratory in the school of Computing and Information Science "investigates the use of cutting-edge mobile and wearable computing technology to assist people during social encounters in the real world" (University of Oregon, 2003).

At Carnegie Mellon University, the Wearable Group is an interdisciplinary team of researchers working on software, hardware, and interaction design as it relates to wearable and pervasive computing (CMU Wearable Group, 2003). Wearable computing research tends to overlap with research in ubiquitous computing, pervasive computing, context-aware computing, augmented realities, dynamic interfaces, shared information systems, personal computing, and safety control systems.

In addition to academic developments, the United States Military is funding many of the projects and research in wearable computing. The military is interested in improving the uniforms of soldiers in combat, such as providing them with clothing that would make them invisible, help protect them against injury, or heal wounds. The military has already deployed high-tech gear for urban warfare that includes global positioning, heads-up displays and networking capabilities (Kumagai, 2001).

Research in wearable computing must not only focus on technical quality and improvements, but must forge new ground in physical design and ergonomics. Only recently have various companies and academic labs started to conduct research that makes a concerted effort to improve the aesthetic design of wearable computers. Studies are being conducted to find optimal regions where hardware should be placed on the body.

Manufacturers like Xybernaut are producing accessories with specially sized pockets to house portable technology. Still, wearable devices are bulky, awkward, and techno-centric. In order to increase demand and begin acceptance into mass culture, wearable computer designs have a long way to go.

Computational Fashion Design and Related Work

Maggie Orth's Firefly dress (front) and Hussein Chalayan's motorized dress (back), taken during the 2003 Wear Me exhibition in Rotherham, UK.

Computational fashion design focuses on the aesthetic and social properties of clothing, questioning how technology might be embedded into garments to change the behavior of clothing and its physical properties. Technology is used as a material for construction, but does not take focus off the clothes themselves as beautiful elements, indicative of style, personality, and identity. The spectrum of behaviors possible from embedding technology into garments fall into the following categories.

Dynamic / Static
Reactive / Disregarding
Disposable / Permanent
Mutating / Preserving
Communicative / Withdrawn
Informative / Mysterious
Humorous / Solemn

Computational garments can fall along any of the axes and also overlap with multiple axes. Each property is realized through choice of technology, material, color, fit, shape, texture,

and concept. An informative garment, for instance, should utilize an embedded system with good storage capabilities, whereas a communicative garment needs an embedded system with the ability to pass data to other systems via wireless connectivity, RF, or other means. Garments that mutate or change shape must be mechanically and structurally sound.

Computational garment design is just breaking forth into industry after years of academic work and research development. Some key players in the development and growth of computational fashion design as a field are described in this section. The researchers, artists, engineers, and designers described are pioneers. Their work has refocused the aims of many individuals and companies.

In addition to wearable computing research, the MIT Media Laboratory is conducting research in computational fashion design. Maggie Orth, Elise Co, Prof. Joe Paradiso, and others have developed conceptually beautiful and innovative projects in this area. Joe Paradiso is the director of the Responsive Environments Group. In the late nineties, he developed expressive footwear as a performance interface for dancers. The shoes wirelessly transmitted measurements recorded by sensors embedded in the shoes, thus providing dancers with control over the volume, tempo, and other musical parameters (Paradiso, 1999).

Dancers' expressive shoes were developed by Prof. Joe Paradiso and his students for the 1999 American Dance Festival. MIT Media Laboratory.

Maggie Orth was a Ph.D. student who worked with Rehmi Post and others to develop conductive threads that could be sewn into fabric and used to create soft circuitry for jackets, tablecloths, and dresses. The Firefly Dress, the Musical Jacket, and

Top: Elise Co's Perforation, 1999.
Bottom: Co's Garment Chimerical, 1999.

embroidered musical instruments are some of the projects Orth worked on during her studies. The Firefly Dress uses conductive fabric to distribute power throughout a dress so that as the wearer walks, LEDs brush against the power layer and illuminate (Orth, 1998). The Musical Jacket has an embroidered keypad on the top left chest which triggers the jacket to play musical notes when the keypad is touched (Post, 2000). The embroidered musical instruments likewise use conductive threads to create pressure sensors that trigger a computer to play music. The sensors look like intricately embroidered patterns on a soft, squeezable ball (Orth, 1998).

Elise Co presented the idea of computational fashion design in her Masters thesis "Computation and Technology as Expressive Elements of Fashion" (Co, 1998). She created very elegant examples of computational garment design, including Perforation, the Garment Chimerical, and Puddlejumper, among others.

Co's project Perforation plays with the transparency of light through the body. A winding fiber optic belt transmits light between matched arrays of perforations and the effect is a sense of transparency cutting through the core of a human body (Co, 2003). The Garment Chimerical uses a flat-screen LCD panel and computationally generated 3D graphics to project imaginary clothing into the

Sketch of student work in Parson's Fashionable Technology Class, 2003.

physical world. "The display, worn on the back in a custom-built pack, shows a virtual garment in the context of an abstracted 3D representation of a male back. Through sensors embedded in an arm unit, the chimerical garment responds to body movements, breath and temperature" (Co, 2003). Puddlejumper is a luminescent raincoat that responds to droplets of rain (Co, 2003).
Internationally, the Swiss Federal Institute of Technology's Electronics Laboratory has a Wearable Computing Laboratory that "focuses on the hardware and system architecture challenges posed by the wearable computing vision" (ETH, 2003). Throughout Europe, research is starting to conceptualize in the areas of computational fashion design, in particular at institutions such as Italy's Interaction Design Institute Ivrea and London's Royal College of Art.

The intersection between fashion and technology is being explored by students at Parson's School of Design in New York. Parson's Center for New Design offers a course in Fashionable Technology. The course investigates the relationship between wearable technology and fashion (Seymour, 2003).

Commercial products from Levi Strauss & Co., Burton Snowboards, Apple®, Charmed Technologies, and other companies are starting to enter the market and capitalize on technology embedded into garments and accessories. Motorola Inc. offers the i90c limited edition phone with pearlized finish for Bloomingdale's customers so that they can have sophisticated communication devices that are also fashion accessories (Motorola, 2002). Motorola is also developing garments that integrate

Left: Motorola Inc. design on display at the Wear Me exhibition, 2003. Right: The Burton Amp jacket, 2003.

PDA's, phones, and MP3 players in interesting, ubiquitous ways. These garments are not commercially available yet, but were on display at the Wear Me exhibition that took place in the UK during 21-26 April 2003.

In the spring of 2003, Burton Snowboards partnered with Apple® Computer to create a snowboarding jacket that houses a sound and control system for Apple's iPod. A fabric keypad on the arm, developed by the company SOFTswitch™, controls an iPod that is stored on the chest. The "Burton Amp" jacket is made out of 3L GORE-TEX® fabric, and is on sale for a limited time at the non-trivial price of $500, iPod not included.

A relatively new company, ElekSen, is also working on soft switching technology which depends upon contacting layers of conductive fabric or varying resistances across partially conductive fabric. Their ElekTex™ fabric is capable of electronic sensing and can be stretched, scrunched, and washed (ElekSen, 2003). They recently announced a commercially available flexible keypad that will come with Orange SPV Smartphones (ElekSen, 2003).

Royal Philips Electronics and Nike, Inc. have likewise created an alliance to merge athletic and digital technology expertise because "athletes want technology that stimulates and enhances the athletic experience" (Royal Philips Electronics, 2002).

Reima Smart Clothing is a research company that has the goal of improving basic clothing through adjustable insulation and ventilation, although their current work has primarily focused on integrating clothing with mobile devices (Reima, 2003). Cyberdog is a company based in the United Kingdom that sells T-shirts and vests with reflective material or reprogrammable illuminating displays. Their "Light S/S Red Alert" shirt, for instance, has a red 32 character scrolling display on the chest of a black shirt. It sells for approximately 90 dollars (Cyberdog, 2003).

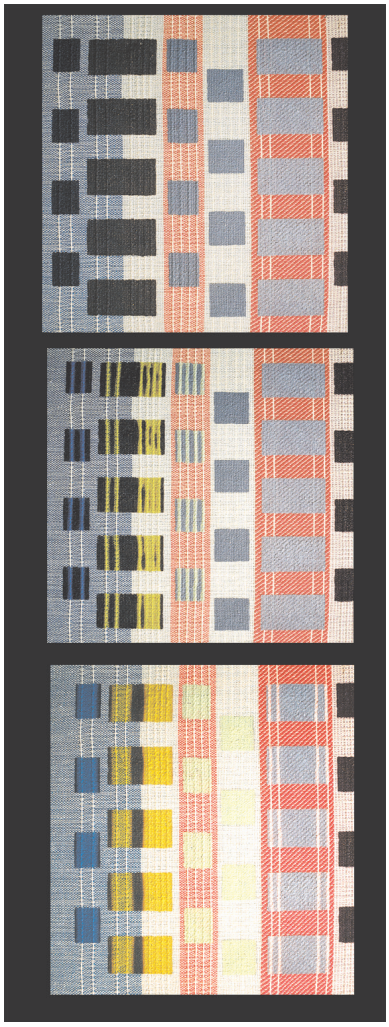
In addition to academic and commercial endeavors, a few fashion designers have explored the area of computational garment design. These designers are the exception, however. In general the fashion industry has been slow to incorporate technology into their designs. Hussein Chalayan's Spring 2000 collection, Before Minus Now, contained the Airplane Dress, a motorized dress with white panels that open and close around the body. Erina Kashihara has created skirts

Hussein Chalayan's motorized dress (bottom) and a close up view of one motor (top), 2003.

with glowing orbs and layers. In addition, many designers have embellished elements of clothing with imagery of printed circuit boards or other technical looking patterns.

Overall, the fields of computational garment design and wearable computing are slowly moving towards each other on the continuum as better technology, increased research, and efficient designs are being developed. But until fashion designers are as free to explore and design with technology as they are with fabric and threads, the industry and all the possibilities for this work will continue to lag behind.

Images of International Fashion Machine's Electric Plaid, 2003. Photo courtesy of Maggie Orth.



Technology Behind Computational Fashions

Outside the scope of wearable computing and computational garment design, interesting scientific research is being done that will eventually impact the design and construction of garments.

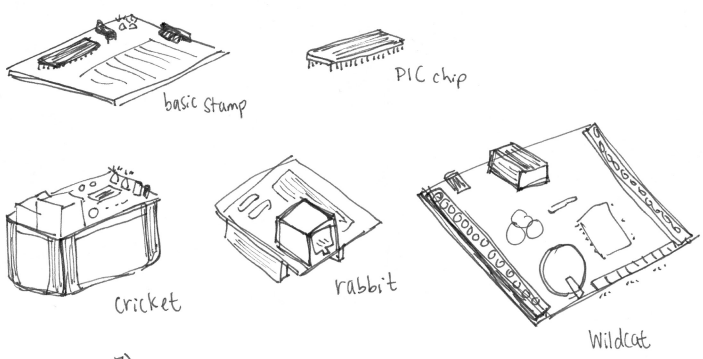
Powering garments is a tricky and burdensome problem because batteries tend to be heavy and bulky. Innovations in the area of power will greatly improve the aesthetics, feel, and drape of computational garments. For instance, at the University of California, Berkeley, chemists are developing solar cells that are cheap, can be produced easily, and can be placed on plastics. They are based on inorganic nanorods and so they are also small (Sanders, 2002). In addition, Material scientists at MIT are working to develop flexible and thin batteries (Sadoway, 2002). At the Media Laboratory, Joe Paradiso, Nathan Shenck, and others worked to develop a pair of shoes that generate power from excess energy expended while walking (Shenck, 2001).

Companies are also working on problems associated with power. Casio, for instance, is developing small, high efficienc fuel cells for potential use in PDA's, laptops, and mobile devices (Casio, 2002). VoltaFlex is a company developing high power, thin-film batteries based on the research from MIT's Materials science department (VoltaFlex, 2003).

Improvements in sensing technology will have an enormous impact on the development of computational garments. Currently, researchers are working on improvements and developments that include capacitance loading, flexible switches, embedded fringe sensing, and more. Danilo DeRossi of the University of Pisa is working with students to develop wearable strain gauge sensing technologies that record posture and movements (DeRossi, 2002).

In the textile industry, textile scientists, polymer chemists, physicists, and bioengineers are starting to brainstorm applications for "intelligent" fibers and fabrics that they hope to develop (Waters, 2002). Maggie Orth's company, International Fashion Machines, is developing hand woven textiles that use a reflective color changing medium to create fabric weaves that have dynamic, programmable patterns and colors. They call the medium Electric Plaid, and it premiered at the Cooper Hewitt National Design Triennial in New York City in April 2003, as part of the project Hydra-House (Orth, 2003). At StarLab NV/SA, a private lab in Belgium, researchers developed "Fiber Computing," which consisted of transistors fabricated into silicon fibers that could then be integrated into textiles (Cakmakci and collaborators, 2001).

Top: A flexible, thin-film battery produced by VoltaFlex, 2003. Bottom: Parasitic Shoes. Courtesy of Joe Paradiso, MIT Media Laboratory. 2003.



Embeddable Devices

Many popular, powerful, and relatively inexpensive microcontrollers are on the market these days. Functionality ranges from basic logic control up to fully functioning computers with wireless Internet, RF communication, BlueTooth, global positioning capabilities, and more. Microchip, for instance, has a vast selection of programmable devices called PIC microcontrollers. Hundreds of families of PICs exist, and users can write code for them using assembly language, a variation of C, even BASIC. In fact, microcontrollers exist that can be programmed by most popular software languages, such as Java, or Logo. Others require a hardware specific language, machine code, or assembly code. Rabbit Semiconductors, Parallax, Atmel, and Texas Instruments are some of the many companies which manufacture microcontrollers and embeddable devices.

With so many options on the market, it can be daunting for a designer to know where to turn when building a computational garment. The languages, functionality, and cost are all factors that must be taken into consideration when choosing the right device. Particular programming, engineering, and mathematics skills are generally required or assumed in order to use the devices. The skills are desirable but may not exist yet for designers setting out to build computational garments.

Fuzzy Logic

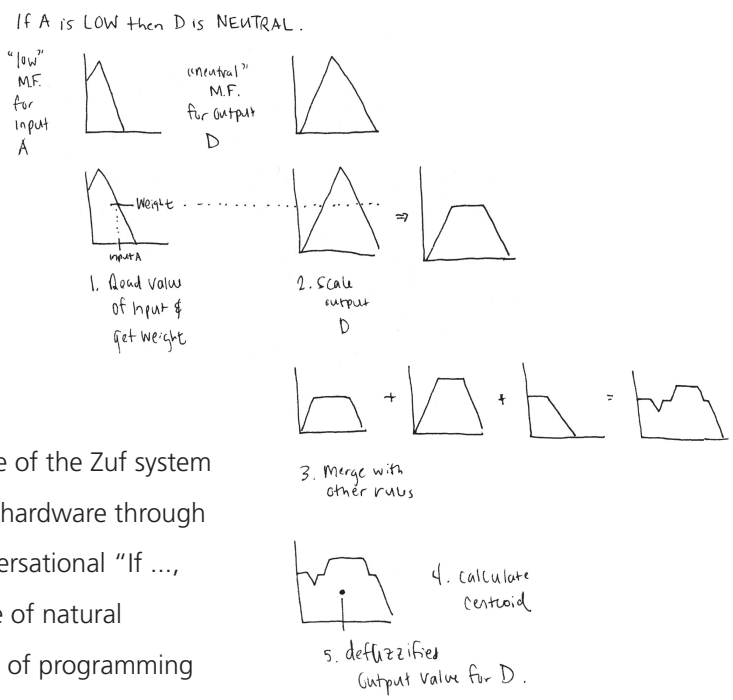
Fuzzy logic was designed to model the uncertainties of natural language, thus becoming an obvious choice as a way to provide designers with a powerful, yet intuitive, programming language. Fuzzy logic is a form of boolean logic that is capable of handling partial truths, or values that are not absolute (Buckley, 2002). Instead of assigning values to be completely true or completely false, elements fall in a subset between two absolute ends. Thus, a mapping from one set to another is not discrete. Subsets are described by membership functions, which describe the degrees to which elements belong to the set. For instance, the membership function for a set S describes to which degree the statement "Element X is in set S" is true. Users of fuzzy devices establish rules for how they expect their device to behave given certain circumstances, or inputs. The collection of rules is mathematically analyzed in order to determine actual behavior.

Fuzzy logic was developed by Lotfi Zadeh, a professor in the Graduate School of the Computer Science Division, Department of EECS, University of California, Berkeley. He is also director of Berkeley's Initiative in Soft Computing (BISC). Dr. Zadeh has been an influential researcher of system theory and decision analysis. More recently, however, his work has focused on the theory of fuzzy sets, fuzzy logic, soft computing, computing with words, and newly developed computational theories of perception and natural language. Applications for his work include artificial intelligence, linguistics, logic, decision analysis, control theory, expert systems and neural networks (Zadeh, 2003).

Dr. Zadeh's recent work on computing with words is particularly interesting and important in this discussion of building a fuzzy logic programming environment. Much of the success of this environment lies in its ability to provide a means for non-technical and non-mathematical users to think about and connect with programming and computing. Taking advantage of the structures of natural language is an obvious way to approach the problem. Humans communicate with language on a daily basis. They are typically more familiar with quantifying and describing relationships and behaviors with language and words than they are with numbers and equations. Finding a logical, rigorous way to do our computation with language and words rather than with discrete values means we have found a key to unlock the door between programmer and designer.

In 2001, Paul P. Wang put together a book entitled "Computing with Words" that explores the ways in which we can harness the expressive power of words and propositions through computational means. One of the reasons why fuzzy logic is so powerful is because of the way it uses language to compute. The input and output devices are defined using nouns, and each variable is assigned to an adjective which modifies the nouns. The creation of simple English language sentences is enough to create robust and elegant computational capabilities.

Fuzzy logic has been implemented in control systems for many years. General Electric created a steam turbine controller that uses fuzzy logic, Nissan developed a fuzzy automatic transmission controller, and Matsushita Electric has built fuzzy washing machines and vacuum cleaners (Chiu, 1998). Researchers have even worked on building fuzzy windshield wipers (Cheok and collaborators, 1996). Fuzzy logic has many applications in feedback control systems because of its ability to enhance the capabilities of devices, reduce operating costs, and mimic human intuition (Chiu, 1998).



The fuzzy logic infrastructure of the Zuf system allows users to control their hardware through linguistic signifiers and conversational "If ..., then ... " sentences. The use of natural language makes the process of programming more intuitive and familiar for users with non-technical backgrounds. Instead of writing discrete steps dictating exactly how the hardware should behave, a user generates a collection of rules. The mathematics behind fuzzy theory allows concrete control to be generated from these rules, even if the rules are uncertain or conflicting. Vague declarations for how a device should behave need not result in uninteresting or inoperable programs, rather they might evoke obscure - perhaps even charming - behavior due to the mathematics of fuzzy theory. Fuzzy logic reasoning allows for organic behavior on the part of the devices.

A fuzzy algorithm works by applying a construct called a membership function to a set of rules. Membership functions are graphical representations defining how much an element belongs to a set. They place a weight on the value of each input. Through a combination of scaling, merging, and calculating the center of mass of all membership functions, the fuzzy algorithm creates a discrete output value. Generally, membership functions are triangular in shape, but can also be trapezoidal or bell shaped. The area covered by a membership function determines how acutely and strongly different elements belong to the set it describes.

The simple fuzzy algorithm implemented by the Zuf system proceeds as follows. First, each input is mapped to a membership function. Second, the input values obtained from the sensors are mapped to a membership value obtained from that input's membership function. This determines the input value's weight. Next, for each rule, the input value's weight scales the membership function of the rule's output. Each rule is processed mathematically and compared against values set by other rules using min-max comparisons. Then, all scaled membership functions for an output are merged together. After each rule has been taken into consideration, a centroid calculation generates the discrete value for the output in question.

The centroid is calculated using the following equation:

$$\frac{\sum_{i=1}^{i=n} m_i x_i}{\sum_{i=1}^{i=n} m_i}$$

Fuzzy logic algorithms are capable of processing complicated rules that relate several elements, using the boolean relationships AND, OR, and NOT. These rules maintain the basic "If ..., then ..." structure, such as "If A and B, then C." The AND operation relates to the intersection of two sets. The rule would be processed by taking the minimum of the weights of A and B. The OR operation relates to taking the union of two sets, and rules containing an OR relation would be processed by taking the maximum of the weights. Finally, the NOT operation relates to the complement of a set, and rules containing a NOT relation would be processed by subtracting the weight from one.

In the context of computational fashion design, fuzzy devices make sense as the glue to bind the behavior of output elements (i.e. motors, lights, buzzers) to the values of input sensors (i.e. microphones, photoresistors, fabric switches). The sensors trigger certain behavior from the outputs, and one garment might contain multiple sensors or outputs. With each additional component, the structure of the driving code increases in complexity. However with a fuzzy device, the designer simply adjusts their set of rules to accommodate for new components, and if some cases are unaccounted for, the device still behaves reasonably.

Computational Literacy

In addition to the technical background, this research was built upon theoretical ideas about computational literacy and a constructionist approach to education. The goals of the Zuf system are to empower designers, excite them about learning and using computational elements, and to enable them to freely design with embedded systems as a medium. The Zuf system must increase their level of computational literacy, therefore it is more than just a development tool, it is an educational tool.

The Zuf system builds on the ideas of Jean Piaget, a child psychologist who believed that intelligence is a form of adaptation, and knowledge is constructed through the processes of assimilation and accommodation. Piaget developed a theory called "Constructivism" that believes our interaction with objects and events helps us conceptualize solutions and ways of thinking.

> *To understand is to discover, or reconstruct by rediscovery, and such conditions must be complied with if in the future individuals are to be formed who are capable of production and creativity and not simply repetition (Piaget, 1974).*

Seymour Papert, a mathematician and one of the early pioneers of Artificial Intelligence (AI), collaborated with Piaget for many years. Papert conducted research on the ways technology can be used for learning and creative thinking. Piaget's theory of constructivism helped Papert develop his own theory, which he called "Constructionism." Constructionism adds an extra layer, asserting that people construct ideas most effectively when they are constructing personally meaningful objects.

Constructionist ideology resonates with this research and my desire for computational designers to have the skills and access to build any component of their work, to learn about their relationship to technology, and to develop depth in their projects as a result of exploration during the development process.

Seymour Papert is interested in the kinds of computational models that lead to better thinking about powerful developmental processes (Papert, 1980). In my work I am trying to construct such a model for hardware development by fashion designers. My research in embedded system design has been influenced and informed by many existing educational programs available on the market and within academic research settings. Its theoretical foundations lie in the research about ways people learn through designing with technology, such as the writings of Seymour Papert, Mitchel Resnick, Sherry Turkle, and Alan Kay, to name a few.

Computer Literacy can be defined as knowledge or competence with a computer, such as having the ability to turn it on, play a CD-ROM, browse the web, or use the mouse. As Andrea diSessa describes in his book, *Changing Minds: Computers, Learning, and Literacy*, computational literacy is different than computer literacy, for it goes beyond a casual familiarity. On a large scale, computational literacy would enable civilization to achieve things previously unimaginable (diSessa, 2000). For this thesis, the term "Computational Literacy" means having the ability to program or take advantage of the computer's ability to compute, as well as having an awareness of the internal hardware and processes of a computer. The term refers to both the operational understanding of a computer and the ability to utilize the computing power of a hardware device or machine. Computational Literacy implies an intuition about how computers work and how they can change or impact the ways we think and view the world.

Concepts about transparency are important to this work as well. Sherry Turkle describes the idea of "transparent" computing in her book, *Life On The Screen: Identity in the Age of the Internet*. According to Turkle, transparent computing once referred to the idea that software programs, operating systems, and user interfaces didn't block a user from being able to "get inside" the machine and take control over what was going on at the machine code and hardware level. Nearly ten years later, however, the definition of transparent computing had shifted. People who spoke of transparency meant that their machine didn't block them from getting work done. Their machines were transparent because they could "easily see how to make it work" without necessarily knowing how it's working underneath (Turkle, 1995).

Both of these definitions of transparency apply to the construction of the Zuf system. On one hand, the fuzzy logic control is a form of the second definition of transparency in that it allows designers to get their work done without necessarily understanding the process behind it. On the other hand, the system supports the initial definition of transparency because it encourages designers to make computation transparent by enabling them to build and construct technology themselves.

The target audience will be interested in building garments and finding the beauty in computational components. The instruments they use should enable them to create projects that reflect these aesthetic standards. As Resnick states in his paper, *Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation*,

> *The merits of the instrument-building tradition go beyond the immediate needs of research. Indeed, one element of that tradition is a design philosophy that emphasizes elegance and beauty in the material objects of scientific work (Resnick, 2000).*

Several systems have been designed by Prof. John Maeda and by Ben Fry and other students in the Aesthetics + Computation Group at the MIT Media Laboratory that focus on the aesthetic merits of scientific work. Design By Numbers (DBN), and Proce55ing are two systems that introduce basic ideas of computer programming within the context of graphic design. In DBN, dots, lines, and fields are drawn using computational concepts like iteration, repetition, variables, and conditional statements. Proce55ing is a learning program and environment for creating systems in JAVA with real time three-dimensional graphics, color, and other features that DBN lacked. The spirit of Proce55ing is to act as an electronic sketchbook where people can learn the fundamentals of computer programming within the context of the electronic arts (Reas, 2003).

> *"It is (Maeda's) belief that the quality of media art and design can only improve through establishing educational infrastructure in arts and technology schools that create strong, cross-disciplinary individuals" (Aesthetics + Computation Group, 2003).*

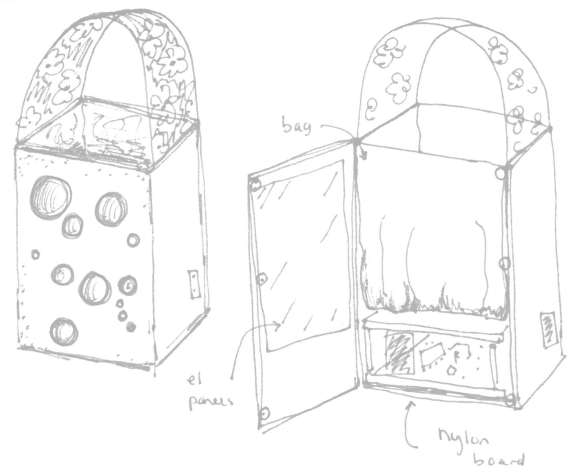
DBN and Proce55ing are powerful educational tools designed for adults, but they only allow users to create programs that manifest themselves on the screen. With Proce55ing, they can create beautiful graphics that are both interactive and dynamic. Other programming systems are also limited to the screen. Squeak, a project developed by Alan Kay and colleagues, is a program designed for children to express ideas about math and science. On the Squeakland website it claimed that its threshold is set low enough for five-year-olds. With Squeak, users can build computational systems.

> *Squeak is an idea processor for children of all ages (Squeakland, 2002).*

Boxer is a project that began at MIT but is now under development at UC Berkeley. Boxer is a computational medium based on a literacy model, and designed for building screen-based tools with ease (diSessa, 2000).

In each of these programs, users are unable to extend the concepts into the physical realm of hardware control. Other groups, however, have explored ideas about hardware control. For example, members of the Lifelong Kindergarten Group (LLK) at the MIT Media Laboratory are working on the development of educational tools for learning concepts about hardware and computation. Over the past decade, LLK worked on the development of Programmable Bricks. Programmable Bricks are tiny computers that control motors, receive information from sensors, and communicate with infrared. The bricks can be used for robotics or other investigations, such as body-monitoring and data collecting.

Programmable Bricks have expanded into two areas. On the industry side, they became the foundation for LEGO Mindstorms, a commercially available toy that lets users build robots and computational projects using LEGO blocks. Academically, the Programmable Bricks evolved into tiny computers called Crickets. Crickets are controlled using a dialect of the Logo programming language, called LogoBlocks. LogoBlocks is a procedural language that includes constructs like *if*, *repeat*, and *loop*, among others. Users program by snapping graphical blocks together, much like snapping LEGO bricks into place. (LLK, 2002).



In addition to Crickets, I worked with my colleagues Justin Manor and Simon Greenwold to create a programming environment and language called Nylon that lets users write the controlling code for programmable devices in addition to developing dynamic graphics. Nylon builds off DBN in that it was designed for artists and visual designers. Nylon and LogoBlocks are each very powerful and are used in a variety of educational settings, however they are based on procedural languages and follow traditional models of computation. For this thesis, I move one step further to develop a system that doesn't rely on procedures and algorithms, but instead upon the structures of natural language and fuzzy logic calculations.

All of the work in this chapter has provided the framework upon which I am building the Zuf system. Without the theoretical foundations of such a varied collection of educational systems, it would be difficult to understand and discuss the intricate relationships and experiences that adults and children have with computational systems. These programs and projects provided my work with a basis to expand upon, both conceptually as well as in execution.

Whether we decide to fight them or join them
by becoming computers ourselves, the days of the
human race are numbered.

Katherine Hayles discussing the computational universe, 1999

Chapter Three
Preliminary Work

An important component of this research has been the process of building garments and thinking about computational garment design. A collection of garments and handbags were constructed that explore the different conceptual spaces of computational garment design. This component of the research is two-fold. On one hand it facilitated the process of defining and clarifying the field of computational garment design by illustrating key characteristics in each garment, while on the other hand, the garments helped pinpoint particular problematic spaces or challenging tasks that designers would encounter during the development process of their own garments. The projects described in this chapter made it possible to isolate the needs and concerns of designers building computational garments, and this knowledge was used to develop the Zuf programming system.

The following collection of work directly correlates design decisions that went into the development of Zuf. The decisions include the ability to program devices over the Internet or to easily redesign relationships between sensors and outputs. The work also illustrates different properties of computational garments, such as being reactive, dynamic, mechanical or mutable. In addition, these garments helps us understand the social, emotional, personal, and aesthetic characteristics of computational garments, pressing our understanding beyond simply technical knowledge. Reviewing this collection of work adds depth to our understanding of computational garment design.

Peppermint

Peppermint is a handbag designed to provide information to its owner over time in a way that is meaningful to the owner but ambiguous to others. Peppermint is a conceptual piece that wasn't implemented, however a prototype of its form was built and 3D CAD models of the bag were generated using Rhinoceros, a NURBS modelling software for Windows. The front plate of the bag was built from laser cut acrylic and implemented using a collection of eight servo motors.

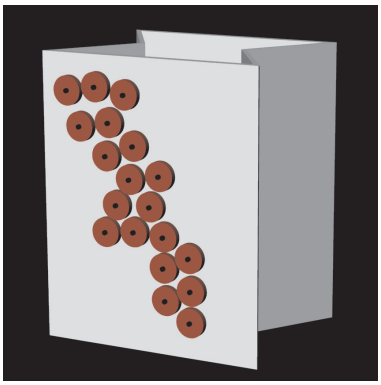
The idea behind Peppermint was to build a bag that subtly informs the owner of specific, important events during the day. The time of the events would be progammed at the start of the day by the owner, through a serial interface and simple software that runs on a personal computer or portable device. Small rotating discs are fixed on the front plate of the bag and move around in slow, elegant, and methodical patterns. As the notable event (an appointment, meeting, test, show, etc.) comes near, the discs alter

Top view of the Peppermint prototype, 2001.

their speed and the pattern in which they move. These subtle changes occur gradually, becoming more intense as the event approaches, thus triggering the memory of the owner. Quicker, more sporadic behavior alerts their attention to the bag, reminding the owner of the event programmed at the start of the day. Much like the way people tie a ribbon around their finger or keep a rubber band around their wrist as reminders for an important event, Peppermint uses non-explicit methods to trigger the memory of its owner and keep their schedule in tact.

Typical alarms beep or sound in loud and often annoying ways, interrupting the public soundscape and drawing unnecessary attention to a person. Peppermint, on the other hand, is a non-imposing alarm that reminds the owner through a change in his or her perception. Other people who see the bag might take note of the fact that its behavior changes over time and may find it interesting because of its aesthetic, however they would be unaware of its ability to trigger memory, and unaware of the meaning behind its kinetic patterns. The beauty behind the aesthetic design of the bag would act as a mask to the world, keeping its secret capabilities and intents hidden to all but the owner in the same way that it hides and protects physical objects carried within the bag.

Left: 3D rendereing of Peppermint, 2001. Right: front view of the Peppermint prototype, 2001.

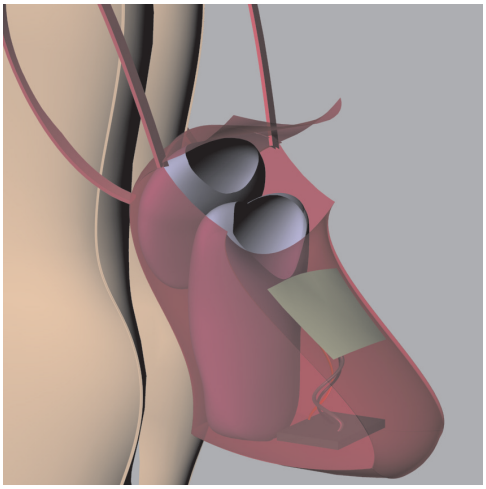


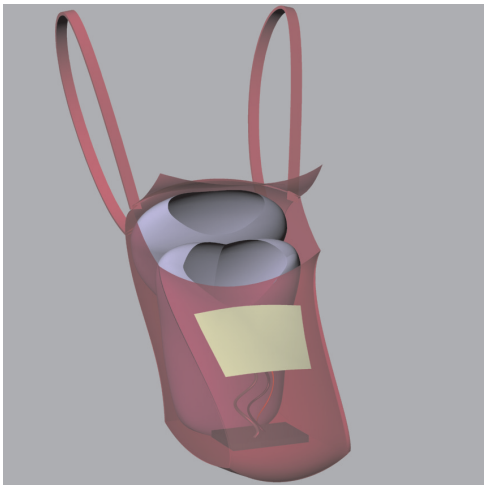
Peppermint is an example of a garment that would require new information to be loaded into it frequently. Users of a garment like Peppermint might not be able to bring their laptop or desktop computer with them everywhere, yet they might want to reprogram Peppermint while travelling or away from their personal computer. Peppermint was a major motivating factor in the decision to create a system that is programmed over the Internet. Being able to communicate with your garments regardless of where you are seems important since clothes are inherantly mobile and non-static.

Belly

Belly was intended to help designers think about and visualize the space of the body and spaces around the body that our clothing and technology inhabit. By designing garments with careful consideration to their form, shape, size, and weight, technology can be embedded into garments that don't weigh down the wearer or cause undue stress or strain upon their body. It also means that technology can be embedded in such a way that it becomes virtually invisible to the wearer. Lighter and more flexible components are being developed in the electronics industry, however there is still a long way to go until electronic components are trivial in size and weight. Therefore it is extremely important for designers building computational garments to think about the spaces technology takes up and how they impact the movement and physical comfort of the wearer.

3D renderings of Belly, 2001.





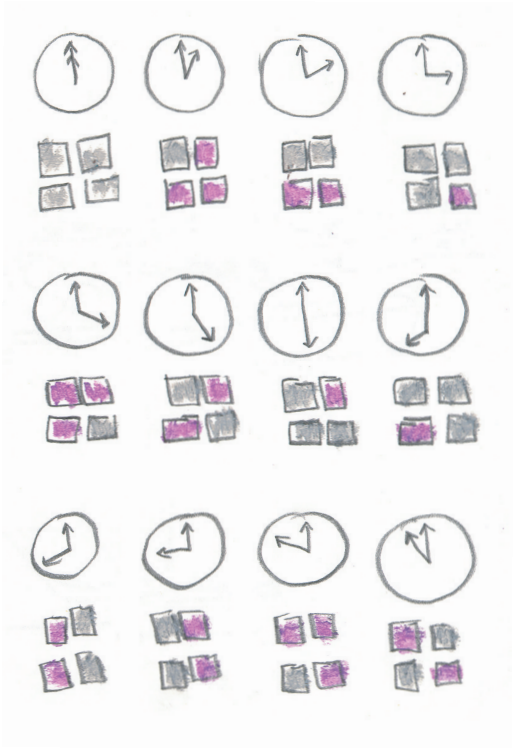
Belly is a series of 3D CAD renderings of a human form wearing a backpack that morphs to the shape of the human's body. The bag is designed to provide enough physical space for the technology which drives the computational element of the bag, and for the storage and carrying capabilities of the bag. The technology is embedded in a space that rests along the small of the back and is out of the way of excessive movement and contact with the body. This protects the technology from damage, and protects the human from discomfort.

Elroy

Elroy was the first fully operational and wearable garment completed for this research. Elroy is a dynamic, illuminating dress that encodes the time of day and displays it on the wearer. The hours of the day are encoded in binary along the right breast, while the minutes are broken into fifteen minute periods and flash down the left leg. Elroy contains several Panasonic electroluminescent (EL) panels cut to size and sewn into the dress. The clock and signalling is controlled by a Rabbit 2000 microprocessor, a product of Rabbit Semiconductors. An inverter turns the five volt DC signal into 120 volts AC current needed to light the EL panels. The inverter is made by JKL Components Corporation, part number NDL-217. Each panel also needs an optoisolator, which allows the logic levels from the microprocessor to control the on/off states of each panel, switching the AC current on and off. The optoisolator used in Elroy is the MOC3043 from Fairchild Optoelectronics Group.

Elroy is sundress made from green, water-resistant polyester fabric. It has wide, one-inch straps that leave the shoulders and meet at a T along the nape of the neck. A zipper runs along the back of the dress, from the neck to the waist. There are seams along the left leg that border the electroluminescent panels, seams around the right breast, surrounding the chest panels, one seam that runs along the hips entirely around the dress, separating it into a low-waisted garment, and a seam that runs directly down the center of the dress vertically, on both the front and back. The seams are visible but stitched using army-green thread so as not to draw the eye's focus away from the illuminating panels.

Photograph of the Elroy hour panels, 2001.



The pattern of the illuminating panels behave in the following way. During the first fifteen minutes of an hour, the top panel on the leg flashes once every five seconds for the first five minutes, twice every five seconds for the second five minutes, and three times every five seconds for the last five minutes. During the second fifteen minutes, the top panel stays lit while the second panel down from the top flashes in the same pattern. During the third fifteen minutes, the top two panels stay lit while the third panel flashes, and so on. The panels reset at the top of each hour and repeat this behavior.

The four panels on the chest, which encode the hours, change patterns at the top of each hour, remaining in one illuminated pattern during the entire hour. Each time one of the minute panels flashes on, the hour panels turn off briefly. This design is for two reasons, the first to save power by reducing the number of panels simultaneously lit. The second reason is to add more dynamic and eye-catching behavior to the dress.

The binary pattern for the hour panels was determined by numbering each panel in a clockwise manner from zero to three, starting at the top left. The zero panel represents the 2^0 place, the one panel represents the 2^1 place, the two panel represents the 2^2 place, and the three panel represents the 2^3 place in binary representation. Every hour from one o'clock to twelve o'clock in a twelve hour time cycle can occur, since only four binary places are needed at maximum to encode the numbers one through twelve.

Elroy plays with the idea of being able to obtain information from our clothing that is meaningful to the wearer but not to the general public, much like the idea behind Peppermint. The time information displayed by Elroy is encoded in such a way that external viewers would not understand that it tells time if they passed the dress on the street. With practice, however, the information can be quickly decoded by the wearer with a quick glimpse. The panels are located in positions easily grazed by the eye, continually providing the wearer with a sense of the passing time. Elroy has meaning to one specific, intended person, but not to the public at large. With so much information that can be collected, stored, and viewed through the use of sensors, infrared, or wireless communications, it is important for designers to think about how the information is made available and processed. Questions must be asked during the design process about whether it is necessary for one person, a small group of people, or the entire public to be able to understand the information displayed by our clothing.

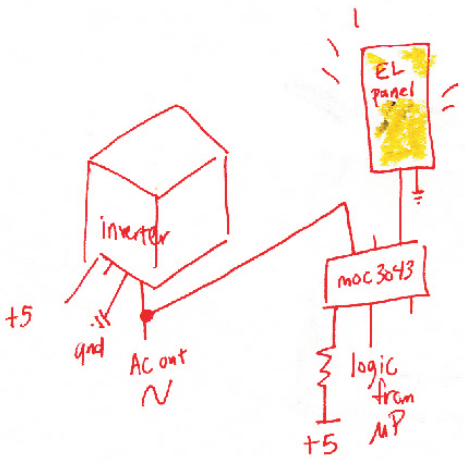
Top: image of Elroy's minute panels. Bottom: Elroy in use by the author. 2001.

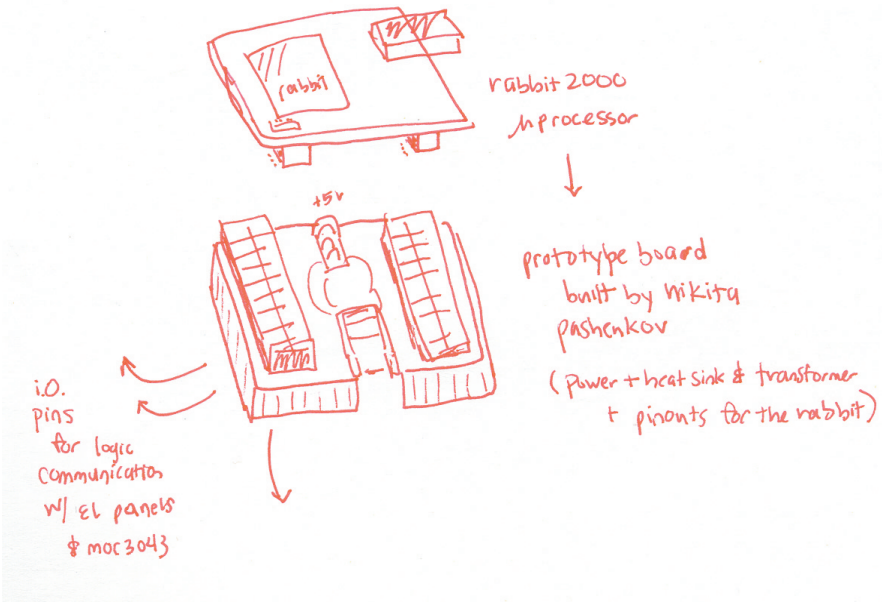
One area that would be interesting to explore is the development of visual languages for the body, designed so that subsets of people can understand data and interact with garments to display information on demand or to input information at whim. The languages, a form of "body slang," might reflect varying cultural or style differences among groups of people. They might relate to the gestural languages currently being researched to improve human-computer interaction. Elements of these languages could also be tailored to exploit the relationship between human, body, and clothing.

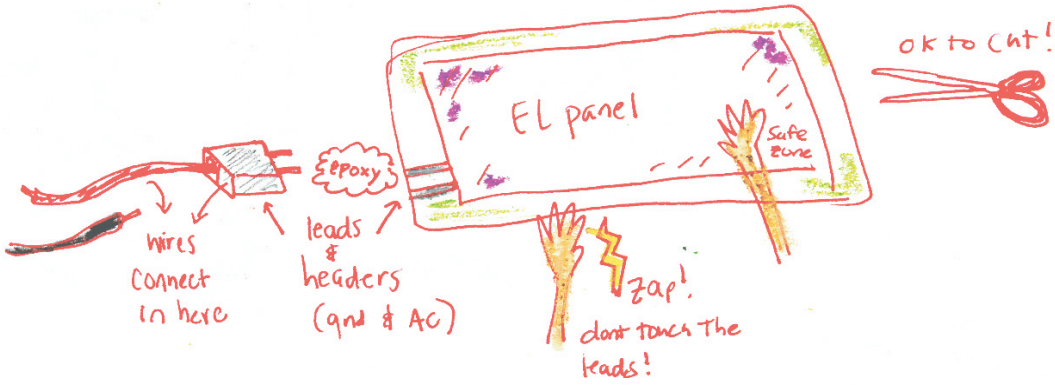
Since Elroy was the first garment I built for this research, and also the first time I programmed for the Rabbit2000 microprocessor, I was amazed and daunted by the slow learning curve required to use the hardware. First, the development environment for the microprocessor was not intuitive to use. It comes equipped with debugging capabilities that I hardly took advantage of because the documentation about how to use the debugger was poor, and because it assumed a very low level understanding of the hardware, an understanding that was uneccesary for my project. Second, the language and methods for addressing the input and output pins took many weeks to master, as they were again established assuming a very low level understanding of the harware. When designing the Zuf system, it became a requirement that both the interface and simulation did not prove to hinder or block novice users from being able to succesfully think about, design, or use an embedded device. As their experience with the devices increases, they might chose to switch to more sophisticated methods which let them work with the microcontrollers on a low level. However, for an entry level tool that should help encourage, entice, and excite designers, such requirements are unnecessary and counter-productive.

Left: Sketches of the Elroy time sequencing.
Below: Elroy in use by a model, 2001.

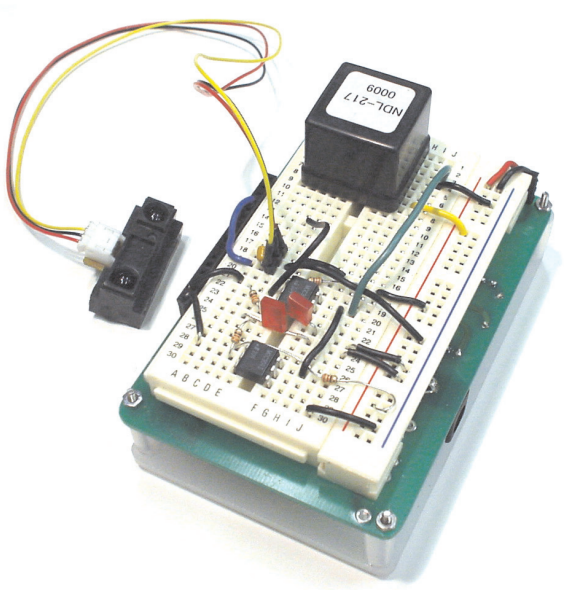
All images on these pages are sketches of Elroy's hardware, drawn by the author, 2001.







The internal ciruitry for Iris, 2002.



Iris

Iris is an example of a reactive garment. It is a small handbag that contains two electroluminescent panels, a piezo buzzer, and an infrared sensor. When the sensor detects motion of the bag moving past the body of the wearer or past objects in its vicinity, the panels and buzzer respond in a quick, alternating rhythm. The result is a playful interaction between wearer and garment that resonates with the bag's movement.

Iris explores ideas about interacting with and altering our garments based on our physical behavior. The way our body moves in our clothes and in relation to our accessories has strong effects on how long a garment lasts, where it wears out or breaks down first, and how it make us physically feel when we wear or carry the item. The way our garments move will also impact the computational components of technologically-enhanced garments, in both destructive and constructive ways. Constructive ways include intentional aesthetic changes, protective strategies to prolong usability, or triggers to change a garment's dynamic behavior. Sensors might detect that the garment is moving or being jostled, versus when it is inactive or still. They might detect when the garment enters a room with hotter or colder temperatures or varying light levels. They might sense when the garment is in contact with the body or other objects. Each of these inputs can trigger the garment to change its state, such

as small changes in color or light, as well as larger changes such as structural shifting or changes in material properties. In essence, the garments become alive. On the contrary, destructive ways include breakage, general wear and tear, power shortages, or possibly erratic and unpredictable behavior.

Images of Iris in use by the author, 2002.

Iris was built using the Nylon system, an integrated software and hardware development platform described later in this chapter. Using the system made the development of Iris very fluid, and helped me realize the concept quickly. The project was especially fun to build because of the ease at which I could move between building the bag, working on the hardware, sewing, and programming. Iris verified the importance of having an integrated system aimed at designers. Its only drawback was the reliance on Nylon's procedural language. The fuzzy logic reasoning implemented in Zuf would have made the construction of Iris even more fun, more elegant, and more robust. Less time needed to be spent calibrating the input sensor for exact values and worrying about covering all cases so the system didn't fail. The fuzzy logic algorithm used by Zuf would have meant I needed to establish only a few rules in order to generate the same behavior.

The rotating shapes on Saturnpants, 2002.

Saturnpants

Saturnpants are kinetic Capri pants that detect the proximity of approaching strangers and respond to the direction of the stranger's movement and distance from the wearer. Soft fabric shapes dance around the legs of the pants. The direction in which the shapes turn and the position in which they stop directly correlates to the movement of an approaching or retreating person.

Two servo motors and one ultrasonic sensor are embedded into the garment. The sensor is made by Devantech and is called the SRF04 UltraSonic Ranger. It measures from three centimeters up to three meters in distance. The device runs on five volts, weighs 0.4 ounces and is just over an inch and a half in length. Positional control of each motor is determined by taking the input from the sensor, processing it with a PIC microcontroller, and outputting the appropriate pulse-width modulated (PWM) signal to the motors.

Saturnpants are made from orange and green wool, and fit a woman of dress size eight. The one and a half inch waistband is secured by Velcro along the front. There are no pockets on these pants. The back of the left leg from the center seam to the left side seam are made of green wool, whereas the rest of the pants are a salmon colored wool. The right leg has three patches sewn to the surface. One green patch is placed halfway down the leg just to the back of the side seam, and contains the sonar sensor, which peeks out through two holes. The second green patch is

placed four inches from the hem of the leg, just to the front of the side seam. It contains the servo motor which controls the circle shape. The third patch is salmon and is just to the back of the side seam, two inches above the hem. It contains the servo motor which controls the square shape. Wires that connect the sensor and two motors to the PIC run along the side seam and convene at the hem of the right leg.

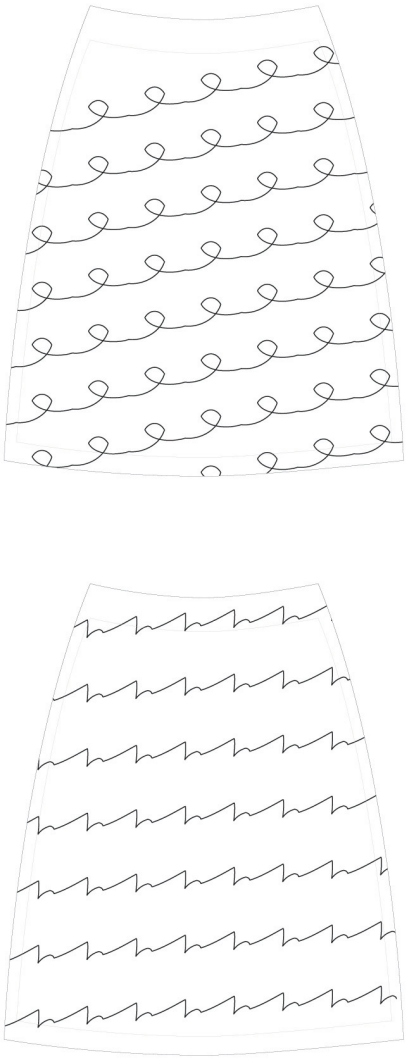
Much like Iris, the Saturnpants project explores ideas about garments which respond to external stimuli and environmental changes. The difference between Saturnpants and Iris is that one exhibits changes through kinetic motion while the other exhibits changes through light and sound. It is important to recognize the different ways in which garments can be altered, as we are not limited to simply one kind of technology or one type of dynamic behavior.

Building Saturnpants required a lot of patience to test different behaviors with the shapes. I played around with the speed and abruptness at which the shapes responded in order to elicit different emotional reactions from people who interacted with the garment. Slow, subtle changes created a different reaction and felt less playful than quick, jumpy behavior. Each time that I decided to try new behavior, I had to wait through a several minute cycle while the code compiled and burned onto the PIC microcontroller. This bottleneck discouraged me from making many changes or experimenting with the garment.

This project helped me realize the importance of being able to quickly change code that runs on a device. It also brought to light the value in a software simulation. I would not have had to wait through the compile/burn cycle just to test a new behavior or debug an interaction if I could have simulated it in software first. I took both of these realizations to heart when building the Zuf system. It is important that a program like Zuf lets designers easily and quickly experiment with their code and with the behavior of the garments they are building.

Scribble

Scribble is a software program that lets a user quickly illustrate and render a pattern for a skirt. The program allows the user to computationally generate the pattern printed on the garment, then it outputs the rendered design to a PostScript file. In a matter of minutes, hundreds of different designs can be generated and queued up for printing. The file contains the outline of a basic skirt pattern so that the printed material is ready to be cut and sewn together directly after printing. The process not only eliminates the step of cutting and pinning fabric and paper patterns for clothing, but it also lets the user regain an element of control over the design of their clothes. A large format inkjet printer, such as the DesignJet 1055CM from Hewlett-Packard, can print the rendered files onto a large roll of cotton fabric or other materials.



Images of skirt patterns generated by Scribble, 2002.

The software for Scribble was written using Proce55ing, a language developed by Ben Fry and Casey Reas, of the MIT Media Laboratory and Interaction Design Institute Ivrea, respectively. To use the software, the designer follows three steps. First, simply draw a line pattern, then click the mouse. Next, adjust the spacing between a repeating pattern of these lines, then click the mouse. Finally, adjust the line width and coloring. Upon clicking the mouse after this final stage, the postscript file is rendered and ready to be printed.

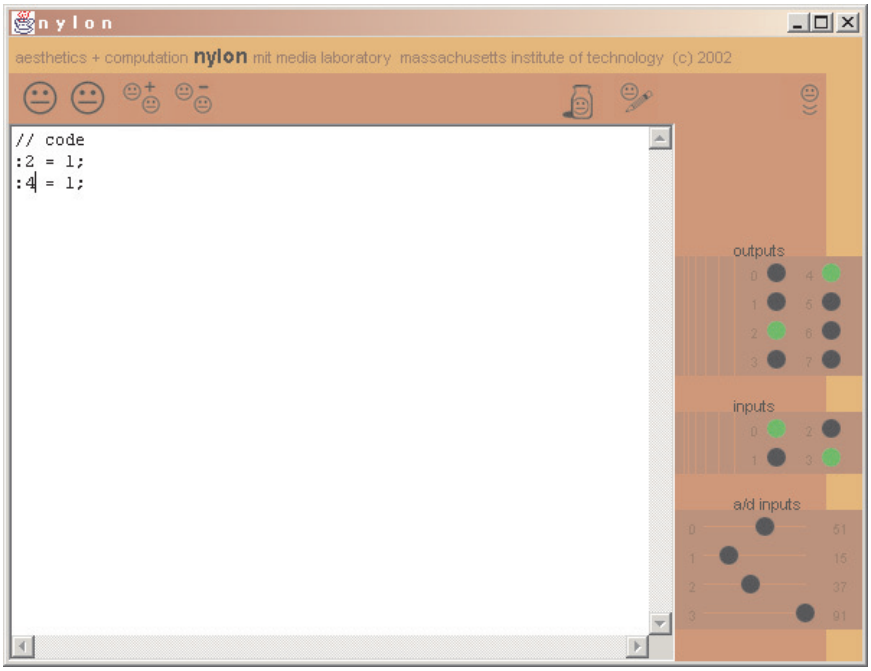
When designing Scribble, I imagined being able to use technology to design and build new clothes each morning before going to school or work. I envy the idea of having my entire wardrobe hand-crafted to fit my body perfectly, but do not want to sacrifice the ease and simplicity of going to a store and purchasing an item directly off the rack. Scribble toys with the idea of disposable clothing that comes and goes with each day.

Women in the early to mid 1900's went to dressmakers for the hottest fashions of the day, a concept alien to most women and girls in today's society. Instead, modern women rely on ready-to-wear clothing for a significant part (if not all) of their wardrobes. This changes the relationship women have with their clothing. Mass production revolutionized the fashion industry forever, but it also altered our mindset about the permanence and fluidity of our wardrobes.

The Scribble technique creates a dual commentary on the fabrication of fashion garments. The speed and simplicity not only contrasts the carefully constructed dressmaker's clothes, but it also contrasts the generic clothing we buy in our local chain store because of the wearer's involvement in generating the design of the fabric. The design process behind the construction of ready-to-wear clothing is completely isolated from the customer.

Scribble is powerful because it lets users see and build garments of their own design in a quick, step-by-step process. The interaction, visualization, and customization of Scribble give the user a sense of ownership over what they create. The process is easy and clear. When thinking about the design of the Zuf system, I wanted the process to be easy and clear as well. The step-by-step approach works well for novice designers, and the minimal interface helps keep the focus and concepts of the project clear. Finally, the visualization of the system in the software simulation provides the user with that sense of ownership, involvement, and understanding that is so useful in the Scribble software.

Screenshot of the Nylon development environment, 2002.



Nylon

The Nylon system is an integrated programming environment for interactive computer graphics and hardware controlled by a microprocessor (Aesthetics + Computation Group, 2003). It was developed in the spring of 2002 by myself and my colleagues, Justin Manor and Simon Greenwold.

The Nylon board was designed for prototyping hardware. It can be programmed directly from a computer's serial port using the Nylon software. The basic syntax of the Nylon language is much like Java or C, however special linguistic constructions were designed to control the hardware input and output pins. The language is versatile because it lets the user elegantly control both the hardware and software components as an integrated and intertwined system.

The Nylon programming environment is the development hub of the Nylon system. It brings together the written code, software simulations, and hardware module. Code can be written and interpreted in the environment, then run in a simulation mode. Once a user is satisfied with the behavior of their code, they can attach their hardware module to a serial port and upload their code with the click of a button. In addition, we developed a 10x14 LED display called Hotpants, that attaches to the Nylon hardware and runs computer graphics generated by the code.

The programming environment contains a text editing space to write code, a simulation space where the input and output pins of the board are simulated, and the ability to add and remove displays so that graphics code can be simulated on the computer screen as it would appear on a Hotpants display. Pausing, resuming, and debugging capabilties exist in the Nylon environment.

Working on Nylon was a great introduction to building an integrated system like Zuf. Many of the design decisions that went into Nylon were tested by a class of undergraduates who used Nylon for their assignments. Feedback from the students in regards to the language, the interface, and the hardware component proved to be invaluable when I set out to build Zuf. I was able to address mistakes made in Nylon when building the Zuf simulation, and when creating the Zuf interface.

Intelligent systems should be able to reason about their own knowledge.

Terry Winograd, Fernando Flores

Chapter Four

Zuf: A Fuzzy Control System

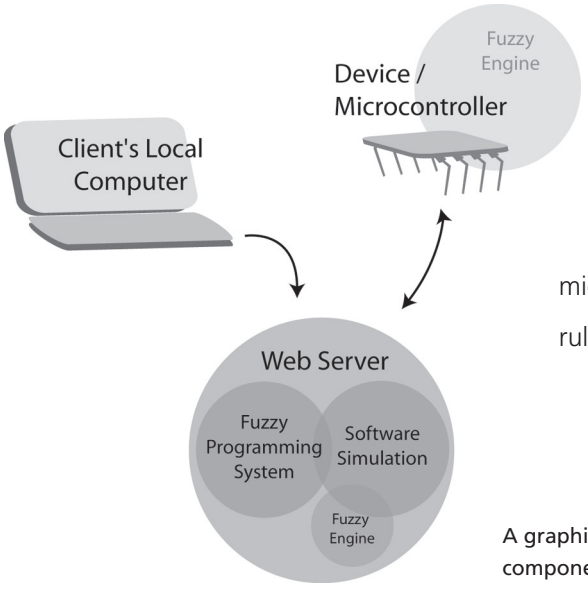
This chapter discusses the Zuf system in detail, first giving an overview of the system and its parts, then discussing the development process when using Zuf with a concurrent discussion of all components in detail. Finally, the chapter wraps up with a discussion of improvements for the system as a whole.

Zuf is a programming system for controlling small embedded devices and microcontrollers using fuzzy logic. It was designed for fashion designers interested in building computational garments, however the underlying concept and process is applicable to many fields of research and development.

System Overview

Zuf is a web-based application which runs on a remote server rather than locally on a designer's machine, like most development environments. Designers access the system by visiting the website from a remote location. To write a fuzzy program, users must first step through a series of web pages, where they are prompted to specify and name input and output modules attached to the device they are controlling. Next, they specify a bank of "If... then..." rules that controls the fuzzy-logic reasoning. Finally, a software simulation of the fuzzy code generated by Zuf illustrates exactly how the device will behave given the rules the user has established.

At any point during the process, users can step back and change parameters, names, or rules. For instance, if it is found during the simulation that the device is behaving differently than anticipated, a user is free to alter the behavior in several ways. The user might delete rules which were already established. The user might also add new rules or change existing rules. Perhaps the user would find that one



A graphical model of the Zuf system, its components, and their interactions.

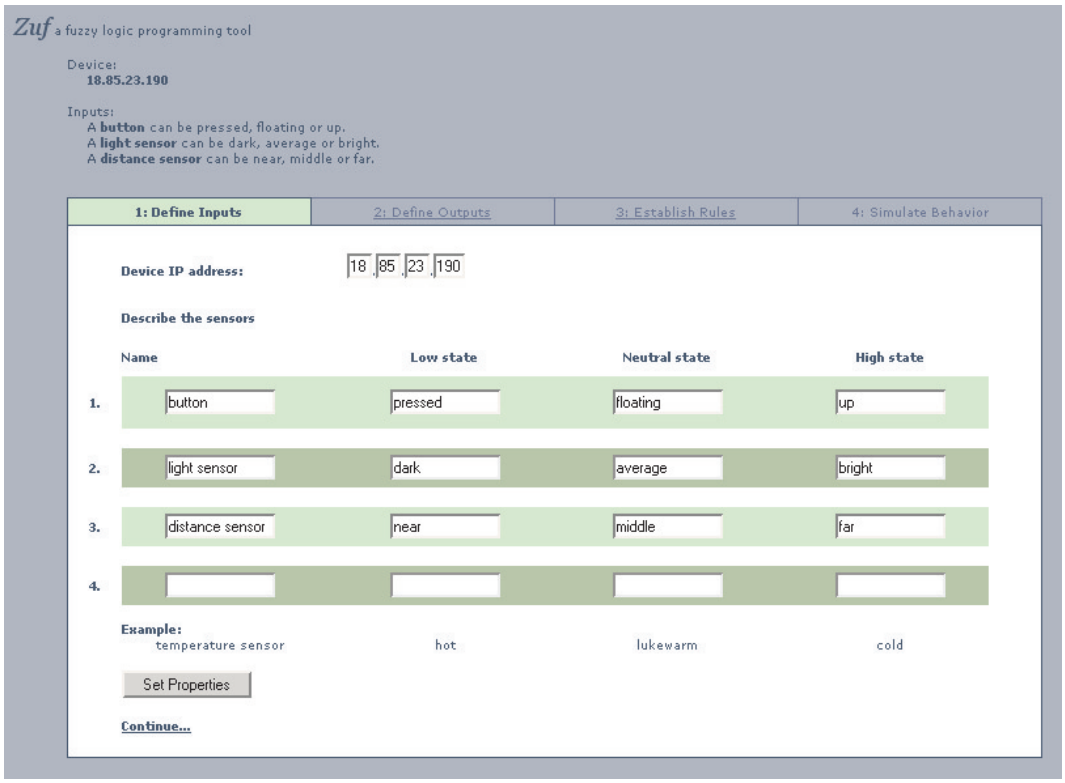
of the inputs is irrelevant, and can therefore be removed from the design. Zuf lets them play with the code in a trial-by-error fashion. The user can see directly how changes affect behavior, and in the end the programmer develops a more intuitive sense about the relationship between hardware, software, and human interaction.

There are three main components to the Zuf system. First is the client, or end-user. The client accesses Zuf from a remote location in order to build the program for the device. The client should have in possession easy accessibility to the second component of the system, the embeddable device. For the first iteration of Zuf, the embeddable device is required to be the BL2000 microprocessor module from Rabbit Semiconductors, otherwise known as the Wildcat. Future iterations will allow for the client to choose between a Wildcat, PIC microprocessor, BasicX module, or other popular system. The main constraint on a client's choice of device is that the device needs the capability to communicate via TCP/IP in order to download information from the Zuf web server.

The third component of the Zuf system is the web server, which houses the development environment and software simulation. The interface for the development environment is powered by Java Server Pages (JSP). It consists of a JSP engine which dynamically generates HTML pages with each request from the client's computer. The interface collects information from the client about device peripherals and behavior, storing this information remotely on the server. The JSP pages modify the interface dynamically to reflect the client's input as they step through the programming process.

Development Process

There are three steps to this process, which include defining the inputs and their states, defining the outputs and their states, and establishing a bank of rules. The inputs and outputs are established by providing a name and linguistic modifier for the low state, middle (neutral) state, and the high state. Although the fuzzy calculations are done on a continuous spectrum between the high and low states, only three modifiers are provided which approximate what range the component is in at any given time.

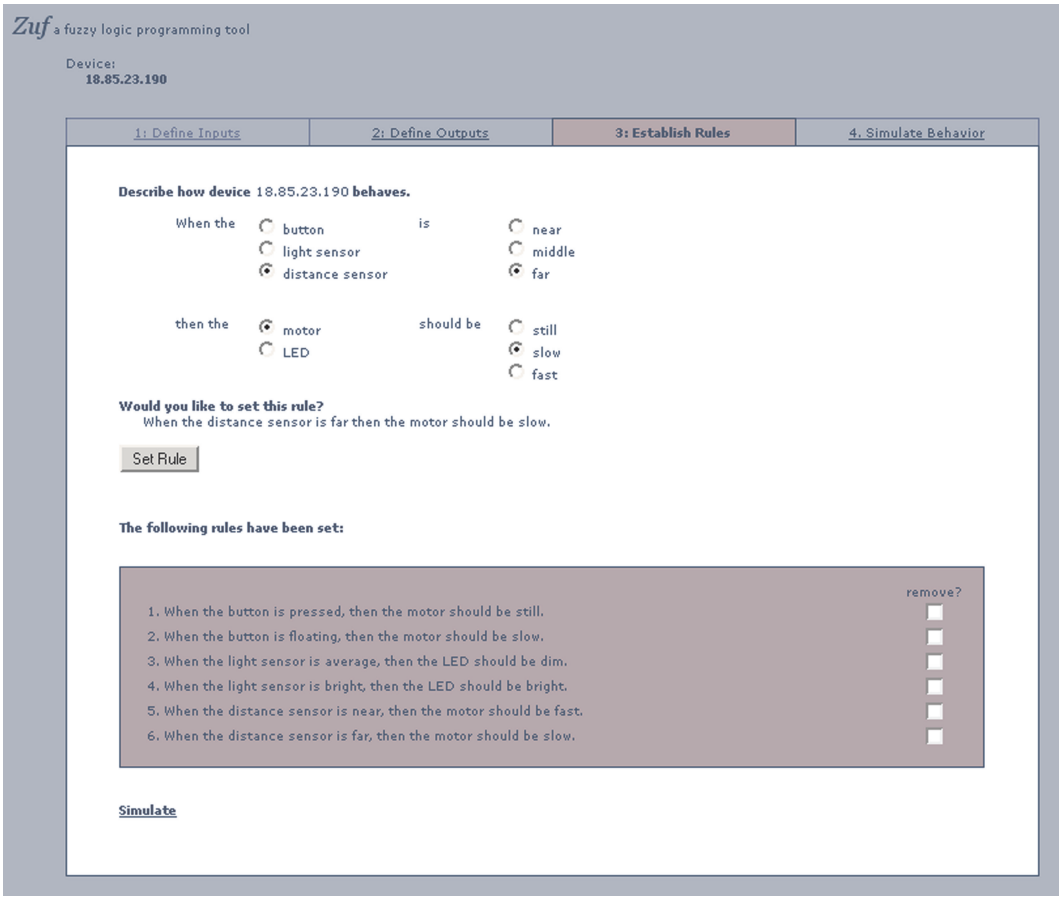


A screenshot of the first step in the Zuf system, entering information about the input devices.

The user enters all of this data into two sequential web forms. Then at the third step, the rules are generated through an interface that dynamically regenerates English sentences describing the rules, depending on which input, output, or states are selected. This is done through radio buttons. A running list of the rule bank is updated at the bottom of the page. There is a check box next to each rule that lets the user remove a rule if so desired. Repeated rules or conflicting rules are allowable in this system, and will not break the functionality of a device. However users might want to delete rules because they affect the behavior in undesirable ways. All computation which happens during these steps is processed remotely on the Zuf server. The software simulation, however, is run locally on the client's computer and consists of a Java applet generated during the last stage of the interface. The JSP engine which drives the development environment ensures that the most recent information gets passed from the Zuf server during simulation.

A screenshot of the second step in the Zuf system, entering information about the output devices.

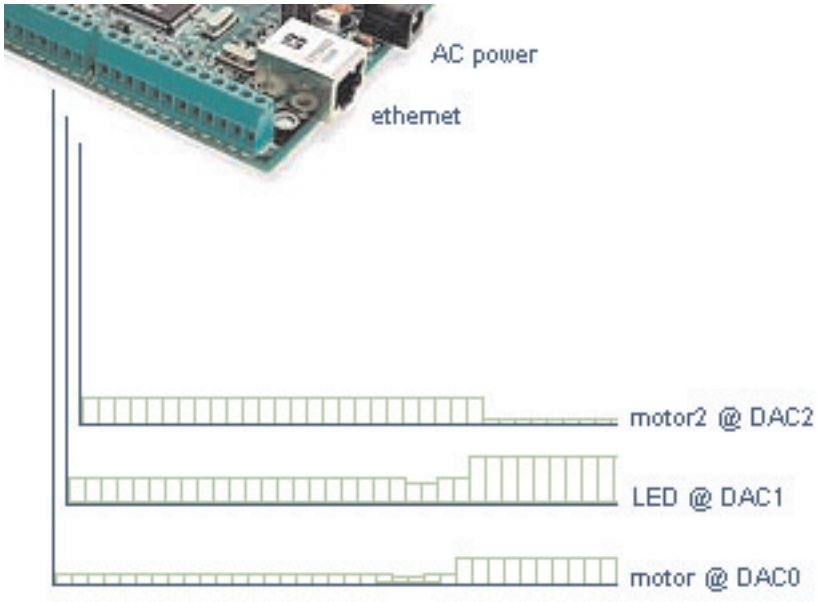
While running, there are two modes in the simulation, a straight-forward visualization of the embedded device and a graphical illustration in the context of computational garment design. In both modes, the client has the ability to manipulate simulated input values and view how the output modules are affected by input changes and the fuzzy computation. Located on the screen during both modes is one slider per input module. The client manipulates the value of an input by dragging the mouse on the corresponding slider. In the first mode, dynamic graphs of the fuzzy membership functions and the changing output values are drawn on the screen in real-time. Toggling to the second mode lets the client watch illustrations of motors, lights, or other output modules as they change state in response to the changing inputs.



A screenshot of the third step in the Zuf system, establishing a bank of rules.

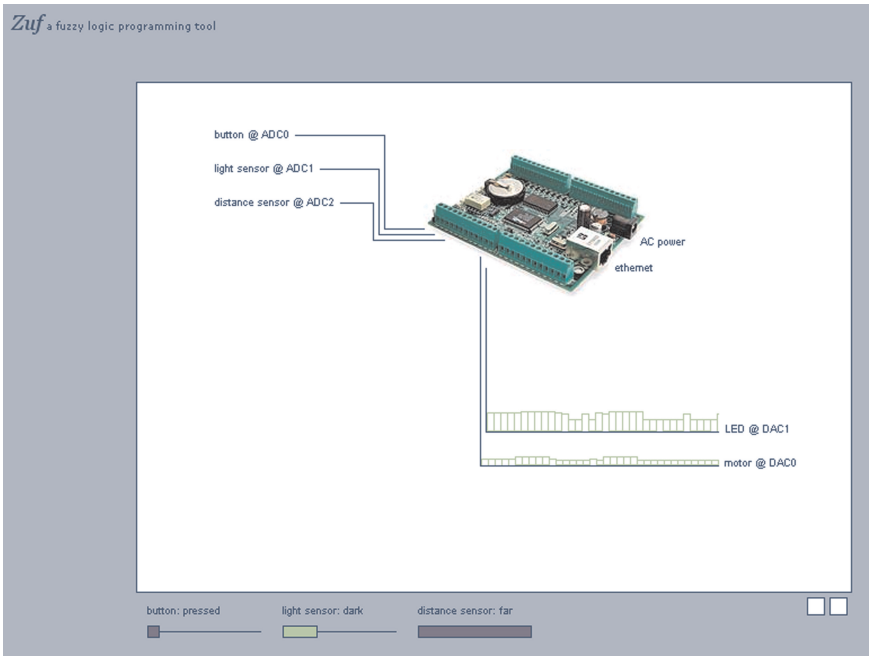
In the first mode of the simulation, the users view a collection of dynamic graphs to monitor if their device behaves reasonably with the rules they generated. First, there are graphs of the voltage level that appears on the output pins. The graphs visually originate from a picture of the device and scroll dynamically across the screen. The graph maps to the output value at any given time during execution. It is directly affected by the input values and are the result of processing through the fuzzy logic algorithm. The client can watch the graph change over time and it quickly becomes clear whether the device is behaving in a reasonable manner.

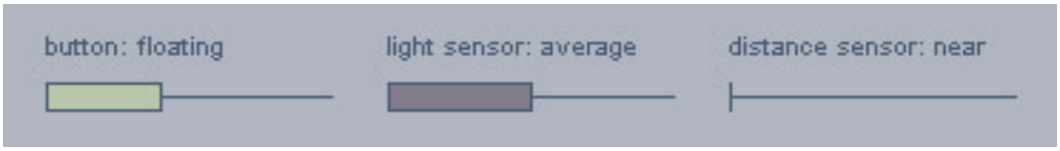
The second type of dynamic graph that appears in the first mode is a mapping of the fuzzy membership functions and their center of mass. Each output has a final membership function which determines to what amount the output should belong in the fuzzy set. The value read on the output pin is generated by taking the center of mass of the final membership function as it changes. These graphs help the user develop an understanding of why they see the values they see and how the fuzzy calculations are affected by the input values.



A close-up screenshot of graphs during the device simulation.

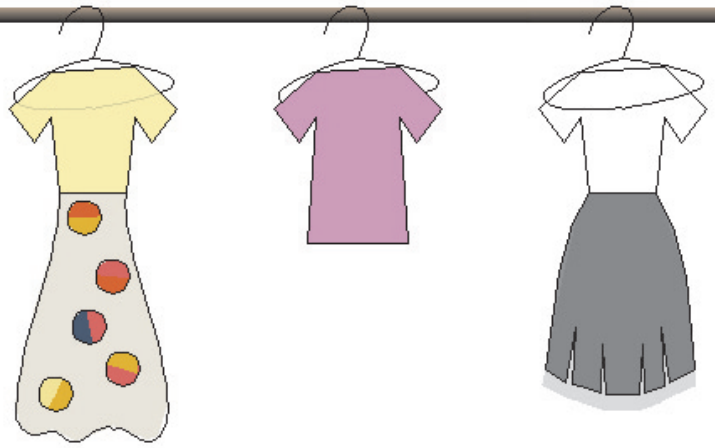
Screenshots of the fourth step in the Zuf system, simulating the device behavior.





In the second visualization mode of the simulation, the client sees the system in an abstract, general form. This mode is designed specifically for the context of fashion design, however one can easily imagine the illustration being tailored to many industries or applications. In this case, each output has an image of itself drawn in relation to the body or as an integrated feature on a garment. Each garment hangs from a hanger on a bar which spans the screen. The idea is to give the feel of stepping into someone's closet, where all the clothes are dynamic and reactive. The technology is so embedded into the scene that it is ubiquitous and thus renders itself both invisible and irrelevant to the idea behind the garment being constructed.

A closer screenshot of the simulation mode in the Zuf system.



In both modes, each input is clearly labeled by name and state. For instance, lets say a client defines an input called “Microphone” which has the low state of “Quiet”, the neutral state of “Hum” and the High State of “Blaring”. As the value of the microphone input slider is adjusted in the simulation, the client is automatically informed which state the microphone is in rather than regurgitating a numerical value. Numerical values of the inputs can be arbitrary and thus meaningless to the user’s understanding of the code. Numerical values are tuned specifically for the embedded device and input module and vary from system to system. By knowing linguistically which state the input is in, the client obtains a more rich understanding of the device behavior and intuitively how it is affected by the rules and sensory inputs. The user’s understanding becomes qualitative rather than quantitative.

A screenshot of another mode during the fourth step in the Zuf system, simulating the device behavior.

Once the designers are happy with the behavior they designed through the Zuf programming system, they are ready to load the code onto their embeddable device. If it is their first time using the device, they must put a bootloader program on the device before programming it. This program allows the device to easily load and run fuzzy programs designed using Zuf. The devices discussed for use with this research are commercially available microcontrollers, and therefore not ready to load Zuf programs off the shelf. The bootloader only needs to be put on the device the first time a designer uses it. From that point out, they can program their controller by powering it up while connected to the Internet over TCP/IP. One input pin is designated on each device to be a momentary "Program" switch. When pressed, the device queries the Zuf server. If code exists for the device then it loads and runs this new program. Otherwise, it continues to execute the code currently saved in memory.

An ideal solution to the bootloader problem would be to design Zuf hardware which has the available features of commercial microcontrollers, comes ready to load and run Zuf code, and is inexpensive. Until such hardware exists, the bootloader is a non-ideal solution to the problem. Loading the bootloader will require that the designer use the development environment which comes with their devices to open and load the bootloader code. This unfortunately means that the designer will have to go through the steps of installing the IDE and dealing with the programming cables and other hassles that come with programming microcontrollers.

The bootloader is written and available for the devices ahead of time so that the designer does not have to program the bootloader on their own. An advantage to using the bootloader program is that experienced users are now given the ability to modify parameters in the bootloader code that fine tunes their device behavior, such as changing the pins each input and output are connected to or adding functionality for more complicated input sensors and output devices. After customizing the bootloader, experienced users only needs to load it once, then they can run Zuf code easily using the TCP/IP connection. This cuts down on compile and load time, and makes the programming process smoother, faster, and less complicated. The reasoning process of the device can be changed easily and quickly without having to reload the driving algorithms for the hardware.

Improvements to the System

The basic structure of the Zuf program was completed for this thesis. However there are many areas where the system can be improved, by adding more functionality and usefulness to the tool. These areas are listed below.

- Develop hardware modules that plug into the devices easily, removing the need to build interface circuitry between hardware and input/output component. *This would be a starting point for hardware development, providing a toolkit of existing circuits for the designers to use. This would include both input sensors and output devices. For example, to attach a DC motor to the Wildcat, a motor driver chip is required to source the correct amount of current without cutting power from the processor.*

- Create a robust back-end for the server which lets users log in, develop a variety of projects, and have access to all variations of their code. *This would require implementing security features in the Zuf web server to control who has access to information saved on the server. It would also require setting up a database for storing and retrieving code, such as the open source database, MySQL.*

- Create a community for Zuf users to share code or expand components and functionality of the system. *This would require implementing a shared space on the website for discussion boards and galleries of work. This could continue to be implemented using Java Server Pages.*

- Improve the software simulation so that designers can build their own visual representations of the outputs, or view both modes simultaneously.

- Handle the identification of devices more elegantly. Move identification away from static IP address identifiers. Implement a system to authenticate identity. *One way to implement this would require users to register their device when they begin development in order for the Zuf server to assign a unique identifier to it. The identifier could then be included in the bootloader code. Another implementation would allow the User to choose an identifier for their device that is approved by the Zuf server to avoid duplicates.*

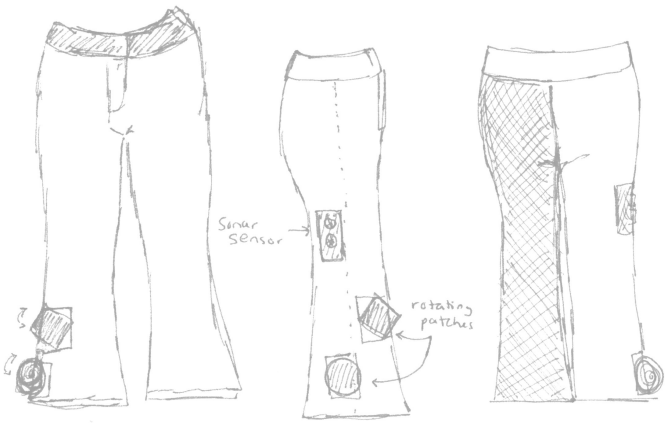
- Implement DHCP capabilities for the devices.

- Implement the system for use with many devices and microprocessors, moving from just the Wildcat to devices such as PIC, BasicX, etc. *This would require writing the necessary driving code for each device that allows them to run the fuzzy logic algorithms. It would also require adding TCP/IP circuitry to those devices that do not come equipped with Internet capabilities.*

- Allow for more complicated rules in the fuzzy algorithm. *This would require updating the software interface to include rules relating multiple devices in "If... and..., then..." statements. It would also require updating the driving code on the devices to handle these rules.*

- Allow for tweaking membership functions and other components in the fuzzy code, such as moving from the min-max centroid calculations to other methods.

Each of these improvements would add more functionality to the tool without changing the basic concept or goals of the Zuf system. Certainly other improvements could be made to the system that are not listed above. When designing a tool, it is important to get feedback and information from people who would use the it. This enables the designer to create a system that is robust and complete and goes beyond the expectation of the User.



The bicycle without a rider balances perfectly well. With a novice rider, it will fall. This is because the novice has the wrong intuitions about balancing and freezes the position of the bicycle so that its own corrective mechanism cannot work freely.

Thus learning to ride does not mean learning to balance, it means learning not to unbalance, learning not to interfere.

Seymour Papert
"MindStorms: Children, Computers, and Powerful Ideas"

Chapter Five
Application & Analysis

In order to evaluate this software as a useful tool for designers, I first had two non-technical adult professionals use the software to evaluate the interface and conceptual grounding of the system. I wanted to evaluate Zuf's ability as an educational tool and to evoke inspiration and computational literacy. Second, I used the software to develop a computational garment of my own in order to test the feasibility of building projects with the Zuf system. Finally, I worked with a class of design students to refine the needs and goals of such a system for fashion designers, and to analyze the usefulness and need for such a tool for their work.

Powerful Ideas and Computational Literacy

Many adults get anxious and nervous when confronted with the idea of programming or getting inside electronic devices to learn about and control their behavior. Through my experience teaching workshops on computational design and also showing my own parents and siblings the work I do as a masters student at the Media Laboratory, I have seen that adults can get excited about programming when someone is with them to hold their hand and explain what is going on. However several people have admitted that they would not attempt such projects on their own. They recognize the power of computation and their dependency on computational objects, but the infrastructure of such devices is otherworldly. The "guts" of their machines occupy a space they dare not tread. This realization became important to me as I set out to design the Zuf system and to refine my ideas about computational garment design. How could these garments aim to inspire them to learn more about computation? How could a system like Zuf and a fuzzy logic approach to programming microcontrollers and embedded devices inspire them to get their hands dirty and build projects of their own?

To evaluate the ability of the Zuf system to stimulate powerful ideas and develop computational literacy, I tested the system with two non-technical adults. The adults, Heather Casey and Laura Davis (whose names have been changed), were able to discuss with me the ability of Zuf to evoke interest, creativity, and ways to conceptualize ideas about computation. During this stage, I was not concerned with the literal application of Zuf to garment design, but rather its ability to reach out to adults, in particular adults that have never programmed before and do not consider computers to be integral to their design or work process. I wanted to know whether Zuf helped users conceptualize the functionality of hardware devices and programming languages. Did it generate new ways of thinking they hadn't experienced before? The fuzzy process does not follow conventional "accepted" methods of control and design, and so I wondered what it meant in regards to its usability, its robustness, and its rigor.

During this stage, I also wondered how adults felt emotionally when approached with the task of writing a computer program. Did Zuf help to relieve this anxiety? Heather and Laura were chosen because they would be impartial to the intent of the system. Neither of these women have programmed a microcontroller before.

According to Sherry Turkle, in her article entitled "Seeing Through Computers: Education in a Culture of Simulation,"

> *Ideas about computer literacy have changed, and it is no longer clear what we need to know or should know in order to become masters of our technology (Turkle, 1996).*

Heather and Laura both use computers frequently, but their perceptions of the computer does not include the complex algorithms, logic gates, transistors, or binary machine code that enables the computers to run. Heather and Laura use the machines without understanding how they work underneath the surface. Concepts used in programming or hardware design, such as iteration or the execution of algorithms, are foreign to them, as they are to many adults.

The first of the women, Heather Casey, is a social worker in Boston. She has a bachelors and a masters degrees in social work, and has worked for the last two years at an early intervention program located in one of the poorer neighborhoods of Boston. Her job consists of working with children who are diagnosed with developmental delays and their families. When Heather has the time, she enjoys engaging in projects like sewing and knitting. Computers are not a prominent component in Heather's daily activities, however she owns a desktop computer which resides at home. She uses the computer several times a week, primarily for checking email, shopping, and visiting websites on the Internet. Heather used to use her computer frequently while still in school, primarily to write papers and create reports for her classes and research. At Heather's agency she shares a computer with several other colleagues. She usually does not have time to use the machine at work, unless it involves obtaining paperwork or conducting research for a client in need of housing, schooling, or other social services.

The second woman, Laura Davis, is a post-doctoral researcher at a medical research facility in downtown Boston. Laura moved to Boston from England just over a year ago. In the spring of 2002, she travelled back to England to defend her dissertation and finish her doctorate. When asked if she used computers and email to help exchange revisions of her work with her committee overseas, she said no. Before she left England, her committee members supplied her with an outline of what they expected of her work, and this outline became her primary guideline while she worked independently.

As a biological researcher, Laura spends a lot of time at her lab growing and working with cells. The lab is equipped with digital cameras designed for use in conjunction with the microscopes and other high-precision instruments. Laura recently bought herself a laptop so that she could complete many of the tasks required of her research, such as writing papers, putting together presentations, and analyzing data. Laura primarily uses her computer for word processing, spreadsheets, checking email, and managing her personal finances. In addition, she uses digital photography to record the developmental stages of her cells, so Laura is familiar with paint and imaging programs, like Adobe Photoshop and the ones included with the cameras.

Heather, Laura, and many adults learn to understand their computer through exploration and interaction. Some adults enjoy reading manuals to learn how to use a computer program. Others, like Laura, operate on a need-only basis. Laura learns about the functionality of her machine only as she stumbles upon new tasks she must accomplish with it. Despite using a computer at work or at home, neither Laura nor Heather think about their machine as a calculated and methodical robot. They do not visualize its behavior in terms of data flow mapped by procedures, algorithms, and variables. They are not exposed to the ordered structure, the synchronized gates, or verbose chunks of code that lie within. It's not that they are incapable of knowing this, they just don't have the need, or else they are scared to discover it for themselves. Heather and Laura each repeatedly told me, "I don't know anything about computers" despite using them frequently and owning personal machines.

Their comments brought to light an important perception that must be considered when designing systems like Zuf. How are such systems and methods capable of expanding a user's bank of knowledge as well as broadening their sense of power and understanding? When does knowledge about the process of computation become necessary and useful to the casual computer user? What would entice an adult to learn more about technology or to want to build projects using programmable devices and computational elements? It is not important for them to know how their computers work in order to use them effectively. When would they find it useful to understand computational concepts?

Each of these women had successful academic careers and slightly higher than average mathematics educations. Heather does not use math or science directly in her social work career, but she took college level Calculus and considered herself a strong math student while getting her undergraduate degree. Laura is a science researcher and uses math both directly and indirectly in her work.

Heather took a computer class in 1988 at her middle school in Florida, when she was in the eighth grade. In the class, they were given the task of executing rudimentary programs in what, we think, was the BASIC language, although this is unclear. She remembers the task as "making colored lines go across the screen" and she had difficulty remembering any other details about what they did. When asked about this experience, Heather was uncertain if the class "counted" as programming experience. She was hesitant to bring it up or describe it to me in detail because it occurred so long ago and existed as a vague memory in her mind. She remembered that it was fun, but didn't feel that it changed they way she thinks as an adult or that the work was useful in her current life.

Aside from Heather's middle school computer class, neither of the women had experience programming either software or hardware, nor did they use computers for tasks outside email, commerce, and the others previously listed. Laura's new laptop is still missing a few programs she'd like to use for her work. We started to discuss what other software programs she wanted to have on the computer, what she would use them for, and how she would install the software on her machine. Laura said she had a friend who could get the programs and install them for her. Through her exasperated facial expressions, it was apparent that installing software was an alien process to Laura, one in which she is very hesitant to get involved in. Laura told me that she thinks installations are probably quite easy once someone knows what they're doing. She admits she could probably install software herself if someone showed her how to do it. But she was also adamant that she didn't want or didn't need to learn. The task required a level of computer literacy that she was not willing to reach. She had friends she trusted to do it for her and do it correctly.

After working with Zuf, Heather and Laura were both surprised that they had actually programmed on their own. Heather felt that the format was more natural than what she understood programming environments to be like, and so it became easier to comprehend what she was doing. She thought that more people would be able to use it because of the ease. She liked that it didn't crash or take a lot of time to use. Once she worked through some initial confusion, she really started to have fun and wanted to add more components and behaviors to the device. Through the simulation and the rules she established, Heather really started to understand how the different components related to the rules she established. She became interested in the types of sensors and inputs available for programmable devices. She asked if there were temperature sensors, or whether it was possible to determine how near other objects were. She thought it would be fun to make stuff she could use with the kids at her job, so they could understand cause and effect.

> *I'd like to make it so that when I talk really loud a car moves around, and the kids would know if I'm angry or not," she said. "I think it would be really helpful to teach them about cause and effect. They like lights and noises and it would be cool to make stuff for them to play with and learn (Casey, 2002).*

Zuf seemed to open a floodgate in Heather's head and new, creative ideas came bubbling forth. Now that she felt empowered and capable of using a microcontroller to build real projects, she had new freedom to plan and dream about what these projects might be. On the floor in Heather's room, I noticed that she had an array of holiday ornaments that she was sewing as gifts for her colleagues at work. I prodded her to see if she'd be interested in building computational stuff for her colleagues, and she laughed at the idea of making squishy candy cane ornaments that blinked or played carols. She said she would definitely be interested in learning how to use motors and lights because she didn't know how to use them.

Laura, on the other hand, seemed very hesitant to call what she had done "programming," even though she'd set up the inputs and outputs and defined the rules herself. She played with the simulation briefly and saw the relationship between input and output values as the system processed the rules. She was not interested in looking at the output graphs generated by the fuzzy logic calculations, and didn't care to ask questions about what they meant or how they related to her car. The entire system didn't fit with her model of programming as a complicated and impossible task, and so she resisted calling it programming. She was sure of her role as someone who "knows very little about programming" and didn't seem to have a desire to explore outside that box, or to admit that she actually had.

Building computational objects seems to have little relation to the work Laura does at her lab or home. Therefore she didn't care to brainstorm about other uses for the system outside the task I laid out for her. It would be fun to let Laura play with programs like MicroWorlds or StarLogo, programs with more relation to her work as a biologist. The physicality of the work that Zuf lets users build does not illicit inspiration or evoke new ideas about computation in Laura. My guess, however, is that programs designed for simulating distributed systems and cellular bodies would be received very differently.

One thing Laura appreciated was that the system was in a format that she could access easily. She finds that her favorite websites are organized clearly and contain links that work and follow through the site.

After discussing with Heather and Laura their experience with Zuf and the things they learned from the program, it is clear that adults need to see constructive, direct uses for computational systems in their careers and daily activities in order for them to get excited about and interested in using educational tools. Alan Kay has previously expressed these same ideas about children,

What really seems to be the case is that children are willing to go to any lengths to learn very difficult things and endure almost an endless succession of "failures" in the process if they have a sense that the activity is an integral part of their culture (Kay, 1995).

The strength of Kay's idea should not be limited to young kids. It resonates throughout my work with Heather, Laura, and I imagine with other adults. My work with Heather and Laura during the early development stages of Zuf reaffirmed that it has the ability to empower designers interested in building computational garments. It is specifically designed for their work, and therefore it docks into their current interests, needs, and initial skill level.

After spending time with Heather and Laura, I was curious about using Zuf with adults who have high levels of programming experience and a very fluid and deep understanding of computers, but who do not use hardware or have an equivalent amount of experience working in hardware as they do in software. Therefore I engaged in an informal discussion over coffee with two graduate students who are working on thesis projects that involve many hours of programming and computer time. Stan Port and Jamie Wood studied computer science and electrical engineering as undergraduates at the Massachusetts Institute of Technology. Stan is currently working on his master's degree at the MIT Media Laboratory, and Jamie is working on her M.Eng. degree with professors at the Whitehead institute.

Neither Stan nor Jamie feel comfortable using hardware, although they have different opinions of hardware as a medium. Stan loves programming in software because of the ease of implementation, the relatively free cost, and the speed at which he can develop complex programs. He has forayed into hardware on occasion, but found it extremely frustrating to pour over catalogs, scratch his head about broken components, and repetitively build the same circuit over and over. He had no idea how to go about starting a hardware project without the help of his colleagues. He explained it to me as an "indescribable phobia of hardware."

Jamie, on the other hand, has a desire to use hardware because she finds it aesthetically beautiful and much more powerful than the screen-based programs she used to create in software. She likes the idea of being able to work in a three-dimensional medium rather than building programs that can so easily get ignored or lost. However Jamie hasn't the faintest idea where to start such a project. She explained that, if given a circuit diagram, she can build it perfectly because she knows all the components and how to put them together, but she wouldn't have any notion of what the circuit did on her own. "I can follow the directions if someone gives them to me, but I want to be able to just put things together and know they'll work. I want big, tangible pieces that I can just play with." Jamie expressed a need to separate hardware from the personal computer altogether.

Speaking to these two helped clarify where the next stages of the Zuf development needed to focus. Heather and Laura helped me recognize that Zuf is fairly successful at making the software and programming component of project development open and accessible to people with no programming background. The fuzzy logic calculations successfully implemented control code that allowed users to program using natural language rather than procedural languages. It created an abstraction that was high enough to be useful to beginners, and transparent enough to encourage questions and curiosity about how things behave. Although Zuf is not the best example of a tool that will excite all users, it is a good tool for people like Heather to start thinking about and creating computational projects. The next step is to create open and accessible modules for designers to build hardware circuitry, to explore electrical engineering concepts, and to feel like they can grasp how their projects behave on many levels beneath the surface.

Using Zuf to Build a Garment

In order to test the Zuf system through an entire design cycle, I used it to develop another garment, named Twirl. The idea behind Twirl was to have a skirt with dynamic components that respond to the posture of the person wearing the skirt. A resistive bend sensor was sewn into the back seam vertically along the rear of the skirt in order to detect if the wearer is standing, sitting, or partially bent over. Two five volt DC motors were embedded into the lower regions of the skirt and control movement of floppy, delicate adornments. The adornments can spin quickly, slowly, or not at all, depending on the input from the sensor and on the fuzzy algorithm currently running on the skirt's microcontroller.

The Twirl project, which was built using the Zuf system, 2003.

A pocket along the right quadricep houses the Wildcat microcontroller and the interface circuitry for the DC motors. The pocket Velcros on three sides and holds the microcontroller firmly in place. It also makes it easy to access the device when new code needs to be uploaded through the TCP/IP connection, because the entire microcontroller does not need to be removed. Instead, a small section approximately one inch in length is a sufficient opening to let the cable through. The input sensor and output motors are connected to the microcontroller through small wires that are stitched through the skirt.

The development cycle for Twirl was very simple because of the Zuf system, and the hardware implementation became secondary to the aesthetic design and garment construction. Initially the skirt was designed in concept through sketches and scenarios, and therefore the appropriate input and output technology could be chosen for the skirt. The actual skirt was built with the devices embedded into it without ever testing the hardware circuitry or needing to write code. Because of the nature of the Zuf system, I was confident that the hardware development would be smooth and painless, and I proved myself correct. Without the Zuf system, development would have required using the Dynamic C programming language and environment, writing a procedural program that reads the input values, calibrates and interprets them, checks for all cases dictating how the output should respond, then sends the appropriate output value to the pin. The process would have required learning the correct syntax, function calls, pin declarations, and other specifics, plus multiple compile cycles before getting to load and test the code on the hardware device.

Once the skirt had completed construction, I downloaded the bootloader onto the Wildcat, then attached the inputs and outputs to it and placed it in the pocket. At this point I was free to toy with many variations of code linking the sensor readings to the output behavior. The Zuf interface made this behavior easy to visualize and quick to modify. Writing new code could be done without taking an excessive amount of time. After several iterations of code, it quickly became clear what set of rules were most effective for my desires for the garment. Tweaking the interaction with the input sensors and establishing the intended behavior from the output devices was accomplished without any headache or long iteration process.

A close view of a Twirl motorized component, 2003.

Describe the sensors

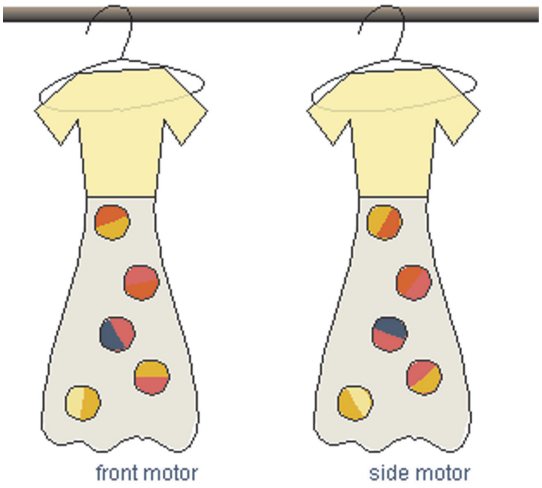
	Name	Low state	Neutral state	High state
1.	bend sensor	straight	slightly bent	very bent

Describe the outputs

	Name	Low state	Neutral state	High state
1.	front motor	off	slow	fast
2.	side motor	off	slow	fast

The following rules have been set:

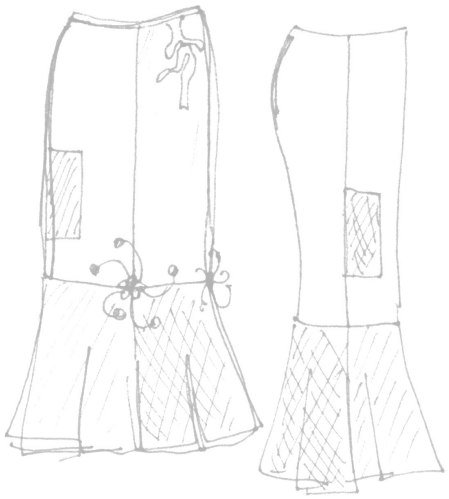
	remove?
1. When the bend sensor is straight, then the front motor should be fast.	☐
2. When the bend sensor is straight, then the side motor should be slow.	☐
3. When the bend sensor is very bent, then the side motor should be off.	☐
4. When the bend sensor is very bent, then the front motor should be off.	☐
5. When the bend sensor is slightly bent, then the front motor should be fast.	☐
6. When the bend sensor is slightly bent, then the side motor should be slow.	☐



bend sensor: slightly bent

Four steps for programming Twirl using the Zuf system:

1. Declaring the inputs (top)
2. Declaring the outputs (second from top)
3. Establishing the rules (third from top)
4. Simulating the behavior (left)

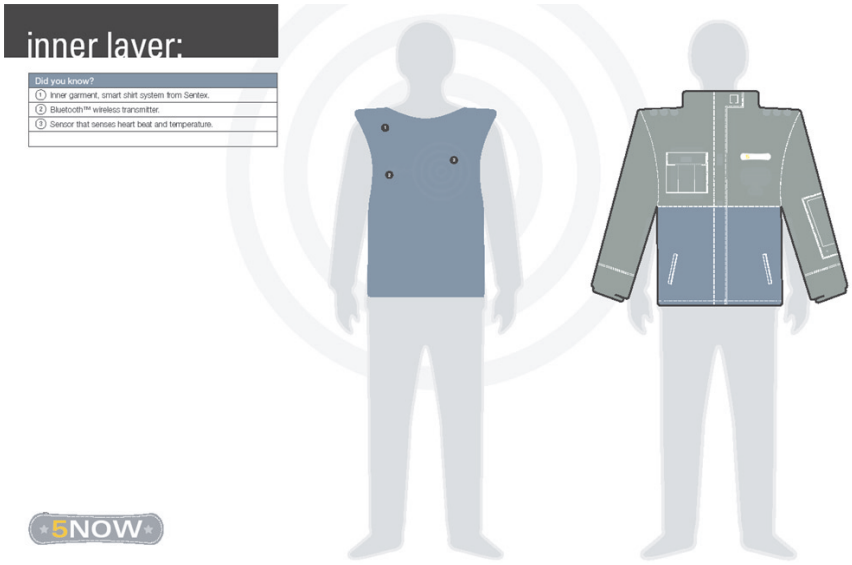


In addition, the Zuf system became a good platform for working on the project with other people because everyone could see clearly what changes were made and how they were made, and could intelligently discuss all the different design paths for the garment. The interface really helped the ideas of the project come through, and made the code intuitive for everyone to understand and think about.

There were a few weaknesses that I discovered during this development process. First, the interfacing between hardware and the microcontroller was a serious breakdown in the system. Motors cannot be directly connected to a microcontroller because they draw too much current and cause the microcontroller to fail. Therefore I had to take extra time to develop interfacing hardware with a motor driver chip. The system needs either "black box" modules that connect without extra circuitry or some method for constructing hardware circuits.

In addition, having programmed entirely with procedural methods, making the shift to fuzzy logic control proved to be difficult. The fuzzy process Zuf uses is a much higher-level approach to programming, and with that comes a sense of a loss of control and of the discrete understanding of the code. It became a matter of how much I trusted my program to work and how much impact I had on its performance.

Design of the inner layer of the fashiontech jacket. Document courtesy of Sabine Seymour, 2003.



Working with Design Students

The final analysis of the Zuf system came when I visited and worked with a class of fashion design students at Parson's School of Design in New York City. The class, entitled "Fashionable Technology," spans two semesters and was taught by Sabine Seymore. The course website describes the class as investigating "the relationship between wearable technology, fashion, and design. An interdisciplinary design process is applied to guide the research, concept development, and prototyping."

"Fashiontech," as the class is referred to, explores both the theory and practicality of developing computational garments, integrating technology into fashion design. Each year the class focuses on a specific topic or context area. Snowboarding was the theme of the class I came to work with. The students were developing jackets for snowboarders to wear during their sport. The goal for the final jacket was to integrate wireless communication, global positioning (GPS), and biosensing to provide emergency location, slope information, and peer-to-peer communication to the wearer of the jacket.

The average age for the class was 28 years old, and all of them had undergraduate degrees from accredited institutions. Half the class had received bachelors of fine arts, and the other half had already received master's degrees in topics ranging from business to arts.

During the fall semester, I visited the Fashiontech class to give a lecture about embedded computing in the context of garment design. This visit was the first stage of my evaluation and research, and it served as a means for me to talk with the students and get a feel for the skills they did and did not have. Through classroom discussion and observation following my lecture, I gathered the following information about a typical fashion design student.

1. Everyone knew how to use a computer for checking email, surfing the web, and using word processing programs.

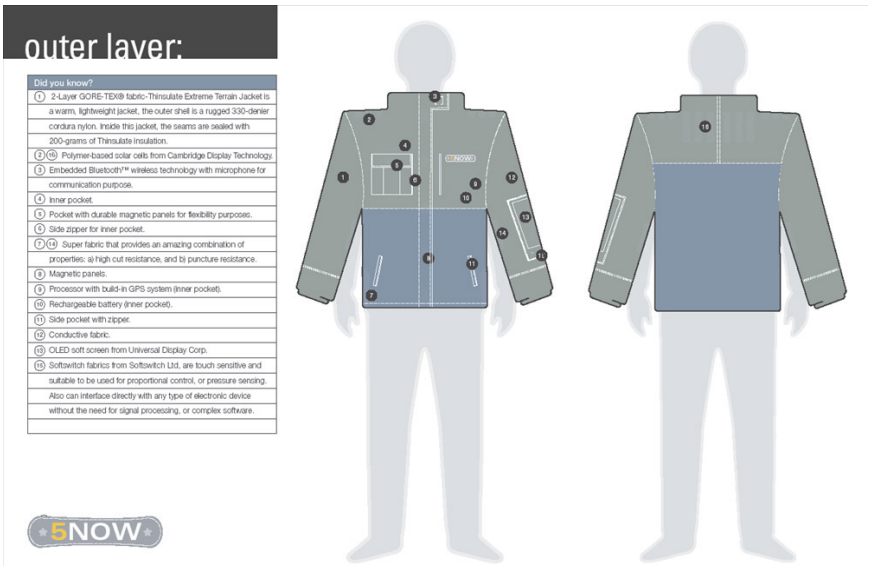
2. Some of the students had created visual design projects on the computer using applications like Adobe Photoshop or Macromedia's Flash.

3. Many of the students had never done any sort of computer programming or computational design projects. Of those who had, most had written short action scripts or HTML code, but not worked with hardware.

4. All students were extremely conceptual and visual in nature.

The students who had worked with Flash or written HTML code and Action scripts seemed to have a higher level of comfort and confidence when confronted with the idea of using microcontrollers and building circuitry. It was good to meet with and talk to all the students during this visit because I was able to gauge their experience level and interest in computational projects. I was able to use this understanding as I set out to design the Zuf system and interface.

Design of the outer layer of the fashiontech jacket. Document courtesy of Sabine Seymour, 2003.



During the spring semester, I revisited the Fashionable Technology class in order to assist them with the construction of their jacket, and to provide a hands-on workshop using my software to evaluate its usability and intent. The class had shrunk in size because conflicting schedules forced some of the students to drop the course, however the students that remained were very enthusiastic and excited about the class, and about my visit.

The collection of jackets they'd conceptualized the semester before had been whittled down to one concrete design, and the class was currently in the process of building presentations, specifications, and detailed documentation of the design and the technology they hoped to incorporate into the jacket. It was interesting to see the process by which they came upon their design. They have a fresh outlook on technology, its implementations, and its applications. They attack their projects with a very top-down approach, as opposed to myself and most of the students I work with at MIT, who attack their projects from the bottom up.

Many engineering students often have very technically strong projects but the concepts are unclear or the final form of the work is poorly designed and constructed. The students at Parson's, in contrast, have very strong conceptual work. The ideas and intentions are extremely clear and illustrated well, the design of the forms are immaculate, but the implementation is a nearly impossible task, too daunting to even know where to start.

In contrast, my usual process when building a computational garment is to find interesting technology and build the garment around it, whereas the students at Parson's would design a garment and concept, then force technology to fit into the design. For these designers, their approach makes sense because they come from a point of view in which they are unaware of the constraints technology can impose on a design. They operate under the assumption that whatever they choose to use will magically be able to work somehow.

Unfortunately, many of the ideas they generated for the class were one of two extremes, either so fanciful that they weren't grounded in reality in regards to existing technology, or they exactly mimicked currently existing garments that incorporate PDA's, cellphones, and other portable devices, instead of challenging the ways in which the aesthetic and physical properties of garments can actually be altered by technology.

The class was very good about accounting for such problems as placement on the body, protection against weather and elements, and other physical issues. Their technical weakness was apparent, however, because they did not design any of the interfacing circuitry that would "glue" their components together and allow for the jacket to function as one system. After they presented their designs to me, one student looked up with inquiring eyes and very innocently asked, "Can this all work? Is this jacket possible?"

The jacket they were designing was unfortunately not far enough along for us to do the hands-on workshop using my software, so I modified the evaluation to consist of a presentation of the work, an optional questionnaire, and a group discussion. During the presentation, I gave a working demo of the Zuf system, explained where I wanted to take it in the future. The questionnaire helped them provide concrete feedback and insight on the work. We also engaged in a meaningful discussion of the Zuf system and their challenge as designers in general.

The strongest observation I perceived from the class was that (understandably) their designs were directly influenced by their current skill set and the depth of their knowledge of technology. The did not know where to research, browse, or search for new technologies or new devices and components they'd never heard of before. Therefore the designs had a tendency to repeat traits from the designs of industry and currently marketed products. This seemed to be largely different from other work that I observed around the building. The student projects and design pieces that were displayed in the hallways and exhibition spaces at Parson's were extremely cutting-edge and creative, pushing the envelope in design, illustration, and fashion. Was it the amount of theory they'd been taught that helped them break through in other domains? Was it the larger collection of existing work from which to draw inspiration that gave them more room to be unique and innovative, as opposed to the extremely small set of existing work in computational garment design? I became extremely curious about where and why this deviation occurred in the Fashionable Technology class.

A map of the interface design and interaction with the fashiontech jacket. Document courtesy of Sabine Seymour, 2003.

Because of the strong contrast between their vantage point on project development versus myself and other MIT students, I was able to get very refreshing and supportive feedback on the Zuf program. The students were genuinely interested in seeing the program and understanding how it works and should be used. My feeling is that this was because it was the first time someone turned to them and said "I understand your challenge and frustrations, I want to help you, and I want to make this work for you." The students have little to no experience working with electronics on such a low level, and they do not have the perspective or support of other students to help them get their footing.

The students explained that it would be helpful for a list of all existing input and output devices to be displayed on the interface or appear in a drop down menus on the pages where the parameters are named. My assumption was that users would want the freedom to write in or add their own naming conventions and devices, and so I left the text fields open and ambiguous. My mistake was that I assumed users would either know what technology exists that they can use, or else they would know where to research and find such items. The reality however is that neither of those assumptions are true and the users are most likely clueless about such things or else not confident enough in their knowledge.

They really resonated with the second mode of the simulation. Being able to visualize the behavior was very meaningful and especially because it was so directly correlated with the fashion projects. They suggested that it would be nice for the users to be able to design and illustrate their own visualizations for each output so that the connections could be made that much stronger and more literal. This suggestion seemed especially nice to me because it got the designers more engaged in the work through a visual medium they understand. It got them thinking about each device, how it would behave, how to isolate its functionality in a design and how it should be placed on the body, then how all these things relate back to the code they're building and behavior they're specifying.

Some of the students had worked on projects for either work or school where they held the position of designer and worked with a team of engineers or technologists. They described these projects as often being frustrating or challenging because it was difficult to communicate their ideas to the engineers, and vice versa. The hardest part was making it very clear to the engineers what type of behavior and functionality they envisioned.

After seeing the Zuf interface, they were very excited because it had an obvious application to their work that I had not envisioned. A program like Zuf would be an ideal tool for them to use to communicate between designer and engineer. Zuf became, in their eyes, an interface between designer and technology - one that both sides could use and understand. The idea of using Zuf to clarify or express an idea, to present work, or to actually build a project was equally important to their work and their needs.

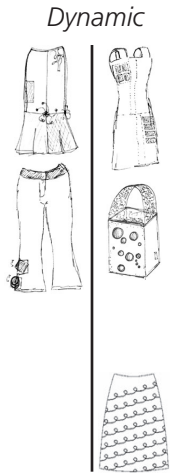
At first I was surprised to hear this because it had not crossed my mind as a useful application for the Zuf software. This use for Zuf clearly addresses a very basic communication problem that exists between designers and engineers. The software could act as a conduit between designer and engineer and still accomplish the task of being a tool that empowers designers, develops powerful ideas, and aids with their computational literacy. Even if a designer wasn't going to build the technology for their work, playing with a program like Zuf still enabled them to work computationally and think about the kind of behavior and functionality their work would possess.

In this sense, Zuf helps designers ground their work in reality and think strategically about the technology they want to use, how it relates to the garment, how it operates, and what it entails to use it. Their designs thus become stronger and more elegant, simpler and more clear. Instead of designing super-garments that can do anything and everything in concept but are technically impossible, they can now design unique, functional, and realistic garments that make it through the development process intact, retaining the true nature and goals of the work. Perhaps it would eventually excite them enough to want to dive in a little deeper and see if they can build the garments themselves.

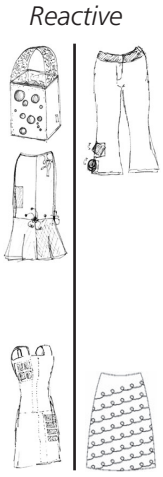
Chapter Six
Conclusion

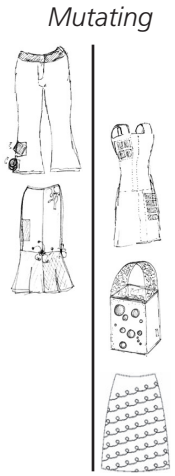
Technical barriers inhibit designers interested in building computational garments. In order for fashion designers to construct systems of clothing that react, collect information, or enrich our interactions with places and people, they will need a tool that helps them realize powerful computational concepts. The tool must lower the threshold and engage designers in meaningful ways during the design process.

This research focused on the development of a powerful tool named Zuf, which was constructed for fashion designers interested in building computational garments. The Zuf system uses fuzzy logic reasoning to control embeddable devices. It contains a programming and simulation environment for designing and testing the devices, and utilizes the familiarity of websites as the programming interface. Its goal is to uproot the process of programming embeddable devices and turn it into a procedure that designers can use confidently and creatively.

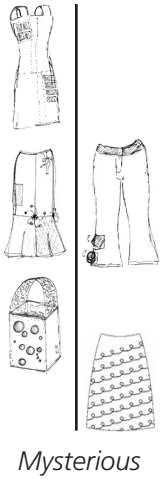


Prior to building Zuf, I created many projects in computational design that helped outline some of the major features and design choices implemented in the Zuf system. These projects included several garments, a handbag, and a hardware programming system. The garments are situated within the axes of computational garment design; *Dynamic / Static; Reactive / Disregarding; Disposable / Permanent; Mutating / Preserving; Communicative / Withdrawn; Informative / Mysterious; Humorous / Solemn.*





Each garment was built to illustrate the different axes but also turned out to contribute a great deal towards the development of the Zuf system. Building the garments illustrated how Zuf would be used to create computational garments; what electrical components and control algorithms were needed to create the different behaviors, and what materials and embedded systems worked most effectively to make each axis clear. As a result, Zuf became a powerful tool that enables designers to elegantly develop projects which fall along any of the design axes by providing a solid foundation for embedded system control.



When designers use the Zuf system, they write code by establishing simple, natural language rules instead of relying on procedural languages or complex algorithms. The rules are translated into fuzzy algorithms which run on the devices. The website and programming methodology lets designers hand-craft the hardware components of their garment designs. Zuf is a potential stepping stone for those interested in exploring more advanced projects.

The Zuf system was tested on two adults who have no programming experience. Then it was used to develop a computational garment. Finally it was shown to a class of fashion design students at a school in New York, where an in depth discussion was held regarding the Zuf system and the needs of designers interested in a new and unfamiliar field.

There are still many features in the Zuf system that should be refined and improved upon, however it is successful as a conceptual model and starting point. In addition to its application as a tool for building computational garments, Zuf has powerful uses as a tool for other domains. It provides a unique approach to controlling embedded devices through fuzzy logic reasoning and its web-based platform. Any application that requires hardware control, particularly those where an inexperienced adult will be working on the design, might find Zuf empowering and inspiring. No matter what type of project provides the context for using a tool like Zuf, the system can act as a stepping stone for the designer and evoke a deeper understanding and interest in hardware control.

Research in the field of wearable computing has moved technology and industry towards building lighter, more flexible, and more powerful computational devices that can be carried or worn on the body. As a result, fashion designers have started to think about embedding technology such as MP3 players and cellphones into clothing. Little work, however, has been done to change the nature of garment design by utilizing technology as an expressive medium.

Computational garments have the potential to become key actors in our lives. On one hand, the garments have infinite possibilities as expressive fashion elements. The kinetic, dynamic properties of computational garments means they are capable of exhibiting components of our identity in new and interesting ways. Clothing no longer has to be static or unresponsive. Rather, it should have the ability to transform itself, repair itself, mutate, adapt, and react. Instead of resting on our bodies like an external layer of dead skin, clothes can come alive.

Computational garments have a huge potential in the fields of medicine, military, rescue operations, business, education, performance, athletics and more. Computational garments have the computing power to collect, store, and share information. The human body acts as a mobile, dynamic canvas upon which to display and process this information. The visibility and interactivity of the human form makes it a unique interface - perfect for many industries, jobs, and applications.

Research needs to move out of the academic labs and into industry in order for computational garments to fall into the hands of the general public. Therefore a lot of work must be done to establish the framework and standards for building the garments in order to empower designers. Zuf is a first step in this direction. By building Zuf, I hope to encourage people on both sides of the story to think seriously about how to enrich the field of computational garment design so it can blossom and mature.

Bibliography

Aesthetics + Computation Group. Design By Numbers website. MIT Media Laboratory. Cambridge, MA. 2003. http://dbn.media.mit.edu/.

Aesthetics + Computation Group. Nylon website. MIT Media Laboratory. Cambridge, MA. 2003. http://nylon.media.mit.edu/.

Bernier, Beatrice. "Fashion, City, People." Master's thesis. Massachusetts Institute of Technology, 1985.

Buckley, J., Eslami, E. *An Introduction to Fuzzy Logic and Fuzzy Sets*. Heidelberg. Physica-Verlag. New York. 2002.

Cakmakci, O., Koyuncu, M., Eber-Koyuncu, M., Duriau, E., Matthewson, A., Donnely, J., O'Neill, B., Healy, T., Clemens, F. "Fiber Computing: Towards More Wearable Computing." 2001.

Carnegie Mellon University Wearable Group. Website. 2003. http://www.wearablegroup.org/.

Casio Computer Company, Ltd. "High Performance Fuel Cells for Programmable Devices." 14 March 2002. http://www.casio.com/.

Cheok, K. C., Kobayashi, K., Scaccia, S., Scaccia, G. "A Fuzzy Logic-Based Smart Automatic Windshield Wiper." IEEE Control Systems Magazine. Volume 16. Issue 6. December 1996.

Chiu, Stephen. "Using Fuzzy Logic in Control Applications: Beyond Fuzzy PID Control." IEEE Control Systems Magazine. Volume 18. Issue 5. October, 1998.

Co, Elise. "Computation and Technology as Expressive Elements in Fashion." Masters Thesis. Massachusetts Institute of Technology. Media Arts and Sciences. 2000.

Co, Elise. Aesthetics + Computation website. MIT Media Laboratory. 2003. http://acg.media.mit.edu/people/elise/

Cyberdog. Company website. 2003. http://www.cyberdog.net/

diSessa, Andrea A. *Changing Minds: Computers, Learning, and Literacy*. MIT Press. 2000.

ElekSen. Company website. 11 March 2003. http://www.elektex.com.

Green, Adam. "Shooting Star." *Vogue* Magazine. May 2003.

Kay, Alan. "Powerful Ideas Need Love Too!" Written remarks to U.S. Congressional Committees. October 1995.

Maeda, John. Design By Numbers. MIT Press, Cambridge, MA. 1999.

Maeda, J., Sakai, Y., Sawada, Y., Komuro, R., "Fuzzy Rules As a Programming Medium for Children." Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks. pp. 709 - 715.

Mann, Steve. "Definition of a Wearable Computer." Wearable Computing website. University of Toronto. May 12, 1998. http://about.eyetap.org/.

MIT Wearables. Research group website. Massachusetts Institute of Technology. 2003. http://www.media.mit.edu/wearables/index.html

Motorola inc. "Motorola and Bloomingdale's Offer a Rare Gem of a Phone." Motorola company website. 2001. http://www.motorola.com/mediacenter/news/detail/0,1958,1409_1074_23,00.html

Orth, Maggie, Berzowska, Joey. International Fashion Machines. Company website. 2003. http://www.ifmachines.com/

Orth, M., Post, R., and Cooper, E. Fabric Computing Interfaces (short paper). Proceedings of Conference on Human Factors in Computing Systems. Los Angeles, CA. ACM Press. 1998.

Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books. 1980.

Paradiso, Joseph, Hu, Eric, Hsiao, Kai-yuh. "The CyberShoe: A Wireless Multisensor Interface for a Dancer's Feet." Proc. of International Dance and Technology 99. Tempe AZ. Feb. 26-28, 1999.

PCWorld. "Wearable PCs Offer Function, Not Fashion." IDG.net. December 23, 1999. http://www.idg.net/go.cgi?id=309556.

Piaget, Jean. *To Understand is to Invent: The Future of Education*. Viking Press. October, 1974.

Post, E. R., Orth, M., Russo, R. R., Gershenfeld, N. "E-broidery: Design and fabrication of textile-based computing." IBM Systems Journal. Volume 39, Nos. 3 and 4. 2000.
Post, E. R., Orth, M. "Smart Fabric or Washable Computing." IEEE International Symposium on Wearable Computers. 1997.

Reas, Casey, Fry, Benjamin. Proce55ing website. MIT Media Laboratory and IVREA. 2003. http://www.proce55ing.net/.

Reima Smart Clothing. Company website. 2003.
http://www.reimasmart.com/.

Resnick, M. "Closing the Fluency Gap." Communications of the ACM, vol. 44, no. 3. March 2001.

Resnick, M., Berg, R., and Eisenberg, M. "Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation." Journal of the Learning Sciences, vol. 9, no. 1, pp. 7-30. 2000.

Rosenschein, Stanley. "Artificial Agent Architecture." MITECS: The MIT Encyclopedia of the Cognitive Sciences.
http://cognet.mit.edu/MITECS/Entry/rosenschein.

Royal Philips Electronics. "Philips and Nike join forces to bring technology to sport and create a new market." Press Information. 25 March 2002.
http://www.philips.com/InformationCenter/Global/FPressRelease.asp?lArticleId=2002&lNodeId=13

Rhodes, Bradley. "A Brief History of Wearable Computing." MIT Wearable Computing Project. 1997.
http://www.media.mit.edu/wearables/lizzy/timeline.html.

Sadoway, D. R., Trapa, P. E., Huang, B., Ryu, S., and Mayes, A. M. "Block Copolymer Electrolytes for High-performance, Solid-state Lithium Batteries." Presented at the 11th International Meeting on Lithium Batteries, Monterey, June, 2002.

Sanders, Bob. "Cheap, plastic solar cells may be on the horizon, thanks to new technology developed by UC Berkeley, LBNL chemists." UC Berkely, Media Relations. 28 March 2002.

Seymour, Sabine. Parson's School of Design, Center for New Design, Fashionable Technology website. http://a.parsons.edu/~fashiontech/.

Shenck, N., Paradiso, J., "Energy Scavenging with Shoe-Mounted Piezoelectrics," IEEE Micro. Volume 21. No. 3. May-June 2001.

Sony Corporation. Company website. 2003. http://www.sony.com/.

Stanford University website. 2003. http://wearables.stanford.edu/.

Starner, T., Mann, S., Rhodes, B. Levine, J., Healy, J., Kirsch, D., Picard, R. W., Pentland, A. "Augmented Reality Through Wearable Computing." Presence vol. 6(4). 1997.

ETH, Swiss Federal Institute of Technology. School website. Wearable Computing Laboratory. Zurich, 2003. http://www.wearable.ethz.ch/.

Turkle, Sherry. *Life on the screen: Identity in the Age of the Internet*. Simon and Schuster Press. New York. 1995.

Turkle, Sherry. "Seeing Through Computers: Education in a Culture of Simulation." The American Prospect. 1996.

University of Oregon website. Computer and Information Science, Wearable Computing Laboratory. 2003.
http://www.cs.uoregon.edu/research/wearables/.

VoltaFlex. Company website. 2003. http://voltaflex.com/.

Waters, Peter. "Clothes that know when you've been sleeping…" Textile News Online. 1 March 2002.
http://www.tft.csiro.au/textile_news/2002_1q/knowing_cloths.html.

Wearable Computing. Group website. "Introduction to Wearable Computing." University of Toronto, 2003.
http://about.eyetap.org/library/weekly/aa061500a.shtml.

Zadeh, Lotfi. Personal webpage. University of California, Berkely. 2003.
http://www.cs.berkeley.edu/~zadeh/.

Acknowledgements

I'd like to thank the following people for the encouragement, inspiration, and assistance they so kindly gave me during my time as a graduate student at MIT.

My Adivsor

Prof. John Maeda, whose wisdom and insight propelled me like a rocket through this research, leaving me no time to doubt - just time to design, build, and absolutely love to work.

My Readers

Joe Paradiso and Mitch Resnick, two of the greatest minds at MIT.

The Aesthetics + Computation Group

Simon and Justin, my never-ending source of competition, comraderie, and compassion. Ben and Tom and their indispensible, awe-inspiring knowledge and guidance. James and Nik, for keeping watch at night. Elise, my resource for great research, food, and shoes. Casey and Cait, so thoughful, helpful, and beautiful. Jared and Golan, for nurturing me as a wee undergrad. Axel, Omar, and Afsheen, each so crazy and yet so amazing. Max and JRoth, because I don't know who has a bigger smile. And Allen, Mimi, Patrick, Max, Saggy, and the other UROPS, who gave me something to adore (their work) and fear (Cher).

Other Media Labbers

All the students I couldn't have done it without - Parul, Michael, Michelle, Ryan, Jeanna, Erik, and the rest. Plus Elizabeth, Missy, and Connie, Csik, Prof. Smith, and John DiFrancesco. And the TTT, I:O, and DL consortia whose funding made this possible.

My Family

My parents, a bottomless well of inspiration and love. My brother, Colin, for staying right on my heels, keeping me one step ahead of the game. My sister, Allison, an awesome chef, the perfect friend, and who always grounds me when I get lost in the ivory tower. My sister, Laurel, the best nurse in the world, and to whom I am forever indebted for taking care of our mom. And all the others in Naples and Colorado because I have the best family in the entire world.

My Girls

Ali Wood, for the long emails and shared mouse clicks. Nicky Stafford, for the ice cream, the phone calls, and plenty of laughs. Carla "The coolest math teacher in New York" Pellicano for letting me visit without any notice. Cat Fooooo, an awesome roommate who loved to share her mom's great food. Ali Snyder, Karen Davis, plus Shawdee Eshgi and the rest of my bike gang - biker chicks and skirts forever.

My Boys

Ian Ingram, who blamed me for his masters thesis, likes to accuse me of stealing his ideas, and somehow manages to be one of my best friends in the world. Captain Lieutenant Admiral Ensign Andrew Sause Money, who washed in and out with the tides but reminded me why life is so damn fun. Dan Chak, for being The Dan and nothing less. Jeff Ma, for the yummy lunches, odd projects, and great talks. Jim Anderson, Stefan Bewley, and the Pirates. Ben Fry, who will always hold a place in my heart. And of course Dr. Brian Bingham, for being the best friend I didn't know I needed, and for giving me love I tried hard not to need.

My Athletes

The MIT Women's Crew Team, in particular Susan, Andrea, Hillary, Rich, Lean&Mean, and all the awesome ladies I got to coach this year. The MIT Cycling Club; Stan, Caitlin, Chip, Jason, Janine, Ariel, the balcony, and all the other great riders. The Korbly's, and the Triathletes, who taught me that when you really want something, there is nothing better than swimming, biking, and running as hard as you possibly can after it.