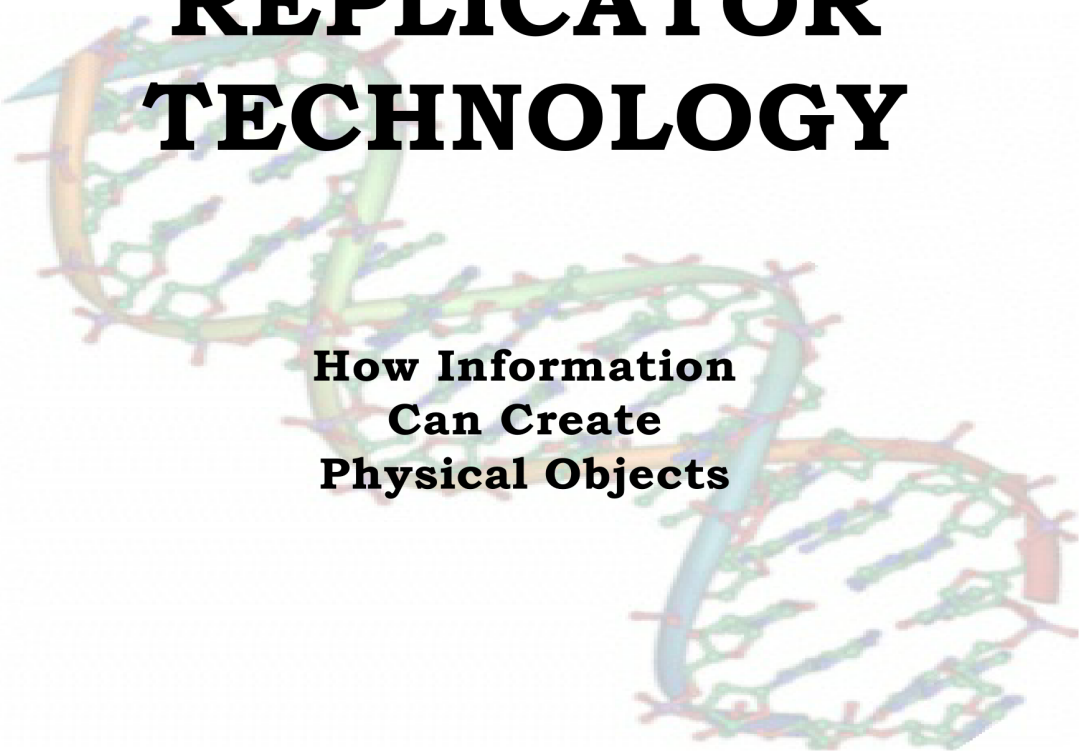
REPLICATOR TECHNOLOGY

How Information Can Create Physical Objects

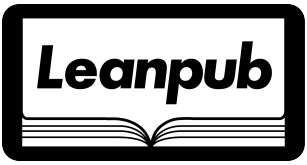


Brian McMillin

Replicator Technology

Using Information To Create Physical Objects

This version was published on 2012-06-03



This is a Leanpub book, for sale at:

http://leanpub.com/ReplicatorTechnology

Leanpub helps authors to self-publish in-progress ebooks. We call this idea Lean Publishing. To learn more about Lean Publishing, go to: http://leanpub.com/manifesto

To learn more about Leanpub, go to: http://leanpub.com

Tweet This Book!

Please help Brian McMillin by spreading the word about this book on Twitter!

The suggested hashtag for this book is #ReplicatorTechnology.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#ReplicatorTechnology

Contents

Preface i
Introduction ii

I Replicator Technology **1**

1 What is a Replicator? **6**

2 What Do We Want To Replicate? **10**

3 What DON'T We Want to Replicate? **13**

II The Challenges of Replication Technology **18**

4 The Challenge of Raw Materials **20**

5 The Challenge of New Materials **21**

6 The Challenge of Processing **22**

7 The Challenge of the Skilled Operator **23**

8 The Challenge of Inorganic Compounds **26**

9 The Challenge of Organic Compounds **27**

10 The Challenge of Large Objects **28**

11 The Challenge of Small Objects **29**

12 The Challenge of Inventory and Storage **30**

13 The Challenge of Molecular-Scale Structures **31**

14 The Challenge of Design Standards **32**

15 The Challenge of Conventional Wisdom **34**

Spacesuits and Glove Boxes 35

Auditory Systems 38

Vision Systems 42

16 The Challenge of Sub-Assemblies **52**

17 The Challenge of Perceived Cost **53**

18 The Challenge of Economics **57**

19 The Challenge of Permanence **61**

20 The Challenge of Quality **65**

21 The Challenge of Environment **68**

22 The Challenge of Yield **69**

23 The Challenge of Waste Management **70**

24 The Challenge of Transportation **71**

25 The Challenge of Assembly **73**

26 The Challenge of Recycling **74**

27 The Challenge of Not Dying **75**

28 The Challenge of Pattern Design **79**

Chocolate Chip Cookie Recipe 80

29 The Challenge of Reliability **85**

30 The Challenge of Obsolescence **86**

31 The Challenge of Hidden Feedback Systems **89**

32 The Challenge of Intellectual Property **93**

33 The Challenge of Semantic Blindness **97**

34 Roadmap to a Replication Future **99**

III Essays **104**

35 I Have A Cat **105**

36 On the Failure of Capitalism **109**

Obfuscation Wins 110

Outsourced Middlemen 111

Mob Rule 112

Cascade Failures 113

Feedback Limits 116

Adapt or Die 117

Buyer Rules 118

Contracts and Lawyers 120

37 On Inventory Management **122**

Background 123

Requirements 124

Visual Object Recognition 126

Modern Examples 128

Other Recognition Techniques 130

Implementation 132

Applications 133

Notes 135

38 What is a Source File? 140

Introduction 141

What Are Source Files? 143

How Are Source Files Used? 146

What Does the Future Hold? 150

What Could Replace Source Files? 151

I Am Requesting Some Feedback 152

39 Malevolent Social Engineering in Open Source Software 153

Background 154

Examples 156

Single-Bit Date-Time Bug 156

Intel FDIV Bug 157

NASA End-Of-Year Protocol 160

McAfee Automatic Update 160

Adobe Flash Player 161

Zune 30GB Music Player Leap Year Bug 162

Sony Root Kit 164

A Tirade Against Digital Rights Management Software 165

Physical Damage to Memory 166

Analysis 169

Vulnerabilities, Exploits and Triggers 169
The Stack As An Unnecessary Vulnerability . . 172
Recommendations 176

Afterword **180**

References **181**

Preface

I have based this work on a lifetime of experiences, observations and interactions too numerous and diffuse to catalog.

I have attempted to verify the accuracy of the anecdotes and statements of historical fact. Any errors that remain are purely my responsibility, and I would appreciate thoughtful comments from my readers.

I hope that any inaccuracies will not detract from the general goal of trying to outline path to a safe, sustainable, local technological future.

Brian McMillin

Watauga, TX

March, 2012

Introduction

This book is the work of a geek, toiling alone in his basement. Figuratively speaking. As I write this, I am taking a concept that I envision and converting it to data in a computer. In a moment, I will send that information to my "publisher". After a few days this physical book will be delivered to my door.

This is an example of Replicator Technology in operation today, in the real world. I have sent a (more or less) complete description of an object into a great, mysterious machine that I know almost nothing about, and the object of my desire has been delivered to me.

Now comes the best part. The information that I created is now stored in "the great computer in the sky". *You* can access that information, utter the appropriate incantations, and cause a replica of this physical book to be created and to appear at your door. This is different from just buying a book off the shelf or ordering out of some inventory in a distant land. "Publishing on demand" is actually creating the physical object only when it is needed.

The term replicator was used in the television series *Star Trek: The Next Generation* to describe a mechanism on the starship Enterprise which could be verbally commanded to create almost any non-living object. This plot device allowed the story to move along without the need to explain how a (comparatively) small ship could have almost any imaginable equipment readily

available. These replicators were a more advanced version of the ubiquitous "food slots" used in the original series.

How can we move forward from today's rudimentary beginnings to a future in which Captain Picard can casually request "Tea, Earl Gray, hot."? What can we achieve in the near term, without the need for breakthrough discoveries? What are the advantages of using Replicator concepts? What are the disadvantages and dangers that we might encounter?

It is my intention to shed some light on a possible technological future by outlining some of these ideals, goals, pitfalls, and areas in need of study. This is a book of questions. I am looking forward to seeing some of the answers.

I Replicator Technology

Our global economy has reached the point that our everyday lives are fundamentally dependent on goods and services provided from great distances, sometimes ten thousand miles or more. This dependence on non-local support began when our hunter-gatherer ancestors first grouped into villages and towns. Local transportation and storage of supplies and materials expanded with trade to distant areas. This allowed mankind to exploit regional specialties, originally based on the varying abundance of natural resources.

Technological humans are those that use tools and fire, build villages and have generally stable populations. The *scope* of a population is the range over which they gather resources. Even though most individuals in a population may not travel very far they may rely on transportation of goods from great distances. Some of these exotic items may be considered luxuries, but if efficient transportation makes items abundant they may come to be viewed as necessities.

The ready availability of inexpensive goods mass produced in specialized centers located at great distances is what makes our

modern technological society possible. Exotic goods, invented mere years ago are now efficiently produced and transported from locations that are completely beyond the control of the consumer. This dependence on distant materials, manufacturing and transportation is a source of great concern. Any disruption due to economic miscalculation, terrorist acts or natural disaster could cascade into virtual collapse of society on a wide scale.

There are communities that have stable populations which use resources from a limited *scope*. Primitive tribes in isolated areas tend, for the most part, to be able to function without outside contact. Amish communities, for example, have an early twentieth-century technology and sustain their populations with minimal *scope*.

The *scope* of the average individual in the United States has grown to encompass a large part of the earth. Cheap foreign mass production and efficient, inexpensive global transportation have made local production of virtually all goods unlikely.

There are no manufacturing facilities for the most basic electronic components (resistors and capacitors) located in America. The specifications for those devices are well standardized and the production has been optimized over a long enough time that virtually all of these components come from a small number of plants in Asia. This transfer of technological competence means that the United States relies completely on foreign sources for the most basic devices. Furthermore, it is likely that there are no longer any individuals in the United States that have the knowledge of how to set up the equipment to produce these components. If it were necessary to begin domestic production it would probably take months or years to reacquire the knowledge.

Interestingly, although I believe the United States is in a dangerous situation with respect to demand for foreign resources I think China is much more vulnerable. They are depleting their resources and shifting their population in a completely non-sustainable fashion. In the quest for global trade, they are duplicating the political, regulatory and environmental mistakes that plagued the United States at the beginning of the twentieth century, with the exception that their production is not headed for domestic consumption but rather is being sold for a fraction of its worth overseas. Any major disruption of demand or transportation will lead to the collapse of their highly specialized manufacturing. This will leave huge populations without the ability to participate in commerce and may impact the availability of life-sustaining necessities to these people. The sudden loss of certain manufacturing specialties will ripple throughout the world and cause unpredictable effects on most of the population.

> The ability to produce goods as needed from simple raw materials on a local basis is the immediate goal of Replicator Technology.
>
> Producing all necessary goods on a sustainable basis without waste or depletion of local resources is the ultimate goal.

Determining the *scope* of a particular population may be trickier than it initially appears. An isolated group may be dependent on rain for agriculture which comes from seasonal weather systems

that cover thousands of miles. *Scope* determination becomes easier in a high-tech environment where virtually all resources flow through a known transportation system. Submarines at sea, the International Space Station and research bases in the Antarctic allow a careful examination of resource utilization, but none have truly closed environments or sustainable populations.

Eventually mankind will try to establish self-sustaining colonies in space or on the Moon or Mars. It will be necessary to use local manufacturing to produce many of the goods that will be needed in such colonies. There will not be adequate transportation or storage area to bring along every commodity that might be required. The population will (initially) be too small to allow skilled artisans to manufacture rarely needed goods.

All of the assumptions that make mass production and transportation on a planetary scale so efficient will not be applicable in tiny, truly isolated colonies. Determining the minimum sustainable population on Mars (for example) is a completely different matter than on Earth. Specialists will be required for maintaining the habitat, obtaining energy and raw materials from the environment, managing health care and reproduction and training young replacements for aging individuals. Each of these tasks will need sufficient redundancy so that accidents or time will not leave the colony short of critical skills.

For the foreseeable future, certain strategic goods will need to be shipped from Earth. But a just-in-time manufacturing facility will be required in any case. With a population of only 250 adults, even carefully screened, one would expect more than 25 cases of diabetes to develop. It would be unreasonable to rely on timely shipments of simple pharmaceuticals from Earth. Deriving insulin from pig pancreases is simple, early twentieth-

century technology. Of course, this presumes that the colony has plenty of pigs.

An important aspect of Replicator technology is that it allows one to focus on defining goals clearly and devising ways of achieving them. Cultural biases and current technologies tend to confine our thinking about both the problems and the solutions.

1 What is a Replicator?

Let's begin by defining what I mean by a Replicator. There is the ideal concept and then there are compromises and simplifications that we need to make to fit into the real world. In the ideal case:

This is a very idealized concept which implies the creation of matter from pure energy. This is certainly possible and is being done today in the real world. A visit to your local particle accelerator might allow you to see for yourself. The amount of matter that can be created is, however, very, very, very tiny. You would probably have to take somebody's word that matter had actually been created at all. And the amount of energy that it takes is prodigious. That bit about waste heat is not really a joke, either. Making the particle accelerator be able to operate without melting is a major consideration.

So, maybe we need to compromise a little and allow some type of raw materials to be used. What we mean by raw materials, where we get them, and how they are delivered to the Replicator are all interesting questions which will require further exploration. Now we have:

This has radically simplified things and moved more into the realm of what we might be able to do practically.

Allowing our Replicator to accept raw materials leaves open the possibility that we could process previously replicated objects into new, completely different objects. This recycling ability forms the heart of a sustainable technology.

Now, what do I mean by information? In this case I am talking about a description of the object in sufficient detail to allow an acceptable copy to be created. It is expected that the Replicator itself will contain a library of detailed information that can be accessed by using nice, user-friendly names. We do not intend to require an atom-by-atom description when all I really need is "Make me a chair".

The Replicator may contain patterns for 10,000 different chairs, but we expect a suitable user interface to allow the most popular or most appropriate version to be selected. Think of it as Googling the Replicator's pattern database. You get a kind of catalog to choose from.

Even though the Replicator contains 10,000 chair patterns, it goes without saying that the one you *really* want is not there. So, the user interface should allow easy customization of objects and some level of error-checking and visualization of the object prior to creation. Thus, "Make me a chair like this, only paisley" would allow you to see if the pattern really matches the decor. And "Make me a chair like this out of mercury" might elicit a cautionary response that the object would instantly melt at room temperature. You could then correct your request to "Make me a chair like this out of titanium."

Where might this Pattern information come from? For simple objects it could be built from manually created Computer Aided Design (CAD) drawings. For complex objects, an entire library of these detailed descriptions would be needed. The information applicable to each different manufacturing process uses its own

set of standards and has certain hidden assumptions that need to be included as part of the Pattern.

For many objects, both simple and wildly complex, it would be very nice if the Replicator could simply duplicate an existing object. This would require a (presumably non-destructive) scanning process, coupled with an ability to identify materials, plan processes and actually derive a Pattern. This Pattern would then be used to duplicate the object.

I argue later that a proper Replicator Pattern must include disassembly and recycling instructions, and that this is at least as important as the object-creation part itself.

We have gone to great lengths to have the information input to the Replicator accurately describe the desired physical object to be produced. The Replicator is expected to produce exactly that object and nothing else. We do not want more than one of the object. We do not want a bunch of left-over, scrap object-pieces, sawdust, used tools, solvents, or radioactive waste. Just one chair.

Right now we are simply trying to define the problem. Actually *achieving* all this is what the future is for. Even getting close in the near term is going to be a challenge for a lot of people to work on. But I think it is a worthy goal to start toward.

Brian's Dictum

Never teach a child to swim.

Teach him to dive. Once he has jumped in he will have to figure out how to get back to the side of the pool. Maybe with a little help the first few times. But if it isn't a big deal to you, it will not be a big deal to him.

Set the goal high. The little things will tend to take care of themselves. And if one of the spinoffs happens to be what you wanted in the first place, so much the better.

---

2 What Do We Want To Replicate?

At the least we would probably wish to replicate solid, three-dimensional objects composed of a single material. I refer to such a device as a Simple Replicator.

> **A Simple Replicator is a Replicator which creates solid, three-dimensional objects from a single material.**

Much of this book will concern replicating complex objects, creating exotic materials and assembling structures with replicated objects. But we must not overlook the advantages of being able to simply and reliably obtain objects made of a single material. I refer to these as solid, three-dimensional objects. For now, I want to defer dealing with liquids and gasses, I want the raw materials to be easily handled, I do not want to have to address the internal environment of the Replicator, and I want to deal with essentially one processing step.

We can get many useful objects by choosing the right raw material and shaping it into a three-dimensional object. Shaping raw materials into objects involves one of three processes: *adding* material, *removing* material, or *deforming* the material. Modern manufacturing processes are generally grouped into (1) casting, (2) molding, (3) forming, (4) machining, (5) joining, or (6) finishing.

Michelangelo was a master of using simple tools to create useful objects out of solid marble. His tools were capable only of

removing material. It seems reasonable to want to design a Simple Replicator capable of delivering, for example, the Statute of David on command.

Purists will complain that marble is not really a single material, but rather a mosaic of crystal polymorphs of calcium carbonate, often with inclusions of other minerals. I will deal with the question of the purity of raw materials later. Marble is a metamorphic rock formed from limestone, dolostone or older marble under heat and pressure. Interestingly, this presents a route for recycling manufacturing waste and used statutes. Given an initial supply of marble, we can presumably reuse it continuously.

Once we have a Simple Replicator for marble objects we can begin to explore the things that can be done with just this single building material. I leave this as an exercise for the reader to list the everyday objects that could be made from marble, presuming the material was essentially free and disposable, and to contemplate the change in philosophy this would bring to our society.

Why would I want to Replicate food instead of just growing it?

1. I don't know how to grow food. I am not a farmer.
2. I don't have the space for a farm or garden.
3. I don't have the materials: soil, water, seeds, etc.
4. I don't have the time. Dinner is at 5:00
5. I don't need much. The recipe calls for a pinch of saffron.
6. I need a lot. The locusts ate the crops.

Why would I want to Replicate fuel instead of just storing it?

1. The Replicator might be the best device for converting an energy source to physical fuel to be stored for later use.
2. The Replicator might be fast enough to convert an energy source to physical fuel as needed.
3. Stockpiling physical fuel might be too dangerous.
4. The desired physical fuel might need exotic structure, such as a solid fuel rocket core.

Maybe we just replicate materials that are destined to be used by a factory outside the Replicator.

If we replicate a liquid or gas we need a container.

Do we want to Replicate a state (hot or cold)?

How do we Replicate an object with significant potential energy such as a cylinder of pressurized gas or a compressed spring?

How big an object? Maybe big things come out in chunks for external assembly.

Maybe really small objects like cells, bacteria, viruses, molecules (buckyballs or insulin).

Maybe long, skinny objects come out on a spool (fiber-optic cable).

3 What DON'T We Want to Replicate?

Presumably it would be reasonable to put some limits on the objects that we want a Replicator to produce. Our definition of a replicator means that it will not produce hazardous waste or pollution other than heat, and that it is capable of recycling any object that it does produce.

The Norse fairy tale *Why the Sea is Salt*, collected by Peter Christen Asbjørnsen and Jørgen Moe, is a cautionary tale that would be applicable to any Replicator technology. The premise that an untrained operator might accidentally command the creation of a dangerous amount of an otherwise innocuous substance is only one of the possibilities that should be guarded against.

The moral aspects of Replication. Creating life.

Maybe cloning an army is bad.

What about yeast? It is needed as a processing step for bread and beer. And we already kill it in the end, anyway.

What about viruses? Is it better to store bio-hazards or to create them on demand for research?

What about explosives? Rocket fuel?

> In 1942, Dr. Isaac Asimov published the short story *Runaround* in which he first posited the Three Laws of Robotics:

> 1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
> 2. A robot must obey orders given to it by human beings, except where such orders would conflict with the First Law.
> 3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Should we have a paraphrased version of Isaac Asimov's Three Laws of Robotics?

> (3) A Replicator shall preserve its own existence unless doing so conflicts with the first or second law.

This is probably a very bad idea. The Replicator should probably have severe restrictions on doing anything that would damage itself, to the point of elevating this above the level of the First Law. Even then, children, terrorists, and the misguided would probably attempt to cause spectacular consequences that the Replicator's safety systems would be unable to anticipate.

Robust systems are designed to anticipate random failures and to function safely even in the presence of single-point failures. A

traffic control system that relies on everyone obeying a law that says "Do not run a red light" is not robust. A single person failing to obey the law can lead to immediate death and destruction. The system needs to be redesigned with inherent safety features that allow safe operation.

An example of a twenty-first century solution would be to eliminate traffic signals altogether. Assigning time slices to intersecting roadways would allow vehicles to cross by simply modulating their speed. No one would ever stop near an intersection. This would be vastly more efficient, especially for cargo vehicles, than the current start-and-stop approach.

This would lead to what is known as *platooning*, where you have groups of cars bunched together and moving in lock-step. Any failed vehicle would simply leave the platoon, and platoons would "see and avoid" obstacles (defective vehicles, construction, etc.) by changing lanes and speed modulation. Think of it like a computer-assisted figure-eight race.

A good first step forward would be to simply standardize road intersections. This was attempted when the interstate highway system was envisioned in the mid-twentieth century, but there were far too many horse-and-buggy concessions left in the design.

Understand that I am opposed to manual operation of automobiles. The height of luxury for the wealthy has always been chauffeur-driven travel. Henry Ford made inexpensive vehicles available to the masses and set in motion an automotive industry

geared toward convincing the public that they actually *wanted* to drive. And had to have a new car every two years.

I do not like to drive. It is a waste of my time. I do not want to be reduced to the level of a collision-avoidance robot. I cannot work on anything else while I am driving. I cannot even enjoy the scenery. Glancing at the drivers of the vehicles around me leads me to believe that they do not want to be driving, either. They are talking on the phone, eating, playing with their children, and enjoying the effects of liquid refreshment and burning agricultural products.

I think that the "car of the future" should have no windows. There will be a large flat-panel display and controls in the traditional driver's position. You don't have to sit in the driver's seat if you don't want - the car is going to chauffeur you from home to work, all by itself. Really want to drive? OK. You have controls and a screen. You can be Mario Andretti racing through the Swiss Alps. Work up a sweat. Enjoy yourself. The physical car is still stuck in traffic on I-35.

Eliminating the windows would allow many new engineering opportunities. The decisions about the frame, safety structures, air conditioning, and seating could all be revisited. Psychological aspects such as claustrophobia could be handled creatively. Early elevators had metal grates or windows in the doors. Modern elevators do not seem to need such features, and most people find them acceptable.

There is also the question of whether I would actually own the vehicle or not. Maybe I just summon one on demand, like an efficient taxi or limousine service. I do not want to have to take care of maintenance. I do not want to pay for an extra vehicle and insurance that I use infrequently. I might not even need a

vehicle sized for shopping, because, in this future, I would order the goods I need and they would be delivered to my doorstep in specialized vehicles.

II The Challenges of Replication Technology

This section is intended to highlight some of the challenges associated with Replicating objects and developing the mechanisms necessary to make the technology practical. My intention is to draw attention to areas that may radically change our views of the world and the objects in it. Each challenge is actually a separate opportunity for rethinking our approach to getting things done. Independently rethinking many of these aspects might be a good idea for our society, even without associating them with Replicator technology.

If you have purchased a physical copy of this book, I commend you and appreciate the financial contribution. The book itself is illustrative of many of the concepts that I will discuss here. I believe that going through the steps of obtaining the book and examining it in light of the comments here will, at the least, be thought-provoking.

If you are reading the electronic version, consider what it would take for you to begin with the text before you, edit it, customize it, and then have it published in physical form.

4 The Challenge of Raw Materials

- Raw Materials limit the scope of what can be Replicated.
- A bias toward “abundant” raw materials leaves the system vulnerable if these materials become scarce.
- Separating the wheat from the chaff. Obtaining the desired raw materials may require an infrastructure all its own with unacceptable by-products.

5 The Challenge of New Materials

- If raw materials are converted to new, exotic materials, the new materials must be able to be extracted and recycled.
- Creating alloys, identifying them, and recycling them.
- Heat treating. Changing crystal structure. Ceramics.
- Coatings and finishes.
- Adhesives and composites.
- Proteins and Prions.

6 The Challenge of Processing

- Multi-step process sequences.
- Batch processing
- Continuous flow processing.
- Specialized equipment. Replicating and recycling the tools.

  Precision and Randomness.

As a machinist, I collect and maintain tools. I enjoy having a variety of tools and being able to use them, on a whim, for different little projects. When I buy a tool, I look for quality and anticipate that I will be able to use that tool for many years.

Replicator technology, on the other hand, assumes that a tool is literally that and nothing more: a tool to be used for a single purpose. The cost of Replicating an object includes the cost of creating and recycling the necessary tools. Under normal circumstances, I would never consider buying a tool for a project and discarding it when done.

I do not have a good idea of what the costs of my whimsical projects are, or what the intangible overhead of owning and maintaining by shop full of tools is. I also do not know how many projects take extra time, or are not even begun, because I do not happen to have the precisely correct tool already available.

Rethinking aspects such as this in light of evolving technologies is important to determine the potential of these concepts.

7 The Challenge of the Skilled Operator

- How to control the flow of material into and through processing steps.
- How to ensure proper alignment, calibration or operation of the process.
- How to handle defects in materials or equipment.

> In October, 2006, after about fifty reported fires in five years, Sony recalled more than seven million lithium-ion batteries because a quality control step failed to detect metal shards left in the cells during manufacturing.

The operators may not really know what they are doing. They may not catch defects in a pro-active manner. They may inadvertently corrupt production by trying micro-optimizations that eventually affect the entire system.

In March, 2007, after a number of cats and dogs died of kidney failure, the FDA reported that wheat gluten imported from China and being used to manufacture pet food was contaminated with melamine.

> **In order to make diluted gluten appear to be of higher quality, the melamine was deliberately added, thus defeating the simple nitrogen assay being used to determine the protein content.**

In April, 2007, rice protein concentrate imported from China was also found to be contaminated with melamine.

In September, 2008, after at least 400 babies developed kidney stones, the Chinese dairy company Sanlu recalled 8,200 tons of milk powder used in infant formula in China because it was contaminated with melamine.

Anyone can recreate the sequence in their mind.

1. They pay me per liter of milk.
2. I can get more money if I add water to make more liters.
3. Woops, I added too much water and the Nitrogen-O-Meter rejects my shipment.
4. Hmmm. If I just grind up this plastic and add it in, the meter likes me again.

5. All is good. Very profitable.

Dead babies are a long way down the road.

What we have here is a perfectly good, working feedback system on a small scale that fails miserably when viewed in the larger context.

8 The Challenge of Inorganic Compounds

- Material properties
- Alloys
- Quality Assurance / Purity / Tolerances
- Crystal Structure
- Identification
- Recycling

9 The Challenge of Organic Compounds

- Processes
- Solvents / Reagents
- Quality Assurance
- Bio-hazards / Medical Uses

10 The Challenge of Large Objects

- Build pieces for external assembly
- How do you define where to divide the object
- Fasteners
- Structural strength and other properties.
- Submarine hull in pieces??
- Build the tools to use externally. Transporters, cranes, etc.
- Disassembly
- Recycling

11 The Challenge of Small Objects

- Holding and manipulating
- Identification
- Quality Assurance
- Recycling

12 The Challenge of Inventory and Storage

- Different size and shape parts must be stored as intermediate steps or for future use when short lead-times are required
- Identification of parts of all shapes and sizes
- Specialized storage requirements: temperature, humidity, etc. Aging in storage. Slow chemical or biological reactions. Rotting.
- Retrieval. Specifying what you want. Knowing that it is in inventory.

How it works on the Space Station. Describing things verbally. Serial Numbers. Bar Codes. Pictures and Videos

How can we automate these processes?

13 The Challenge of Molecular-Scale Structures

- Bucky-balls
- Viruses / capsids
- Nano-machines
- proteins / folding / prions

14 The Challenge of Design Standards

- Units of Measure.
- Material thickness.
- Operating voltages in electronics.
- Power dissipation.
- Overkill for the sake of safety margins.
- Tested objects that work but are not exactly what you need.
- Tested objects that work, but not the way you are trying to use them.
- Hidden design criteria and assumptions.

My wife drives a Mini Cooper, made by BMW. The Mini has power windows. As with most modern cars, the cabin seals very well, sometimes preventing the doors from closing all the way against the air pressure without slamming them.

The Mini engineers came to a clever solution. They arranged for the electronic controls to lower the window about one centimeter

whenever the door opened and to raise the windows all the way after the door is closed and the latch engaged.

Power windows have been around a long time. The mechanism is well understood and reliable. Unless you raise the window every time you use the door. And stall the motor against the mechanical stop each time.

The window motors that were used were DC motors with brushes and a mechanical commutator. Each time the motor stalled under power the brushes burned a little bit. Eventually, the motor would not be able to get power while it was in the window-up position.

I have had three sets of window motors installed (under warranty) in the last five years. I have explained my analysis of the problem to the local service people. There is virtually no way that this information will ever make it back to the engineers that need to know it. There is also no way that I myself could improve my Mini by installing a set of proper, brushless DC motors to prevent the problem in the future.

15 The Challenge of Conventional Wisdom

- Replicating an object without redesigning it.
- Opportunity to redesign from the ground up.

Spacesuits and Glove Boxes

I believe that the space suits currently being used by NASA (and the Russians, for that matter) are in need of complete redesign. Spacesuits are intended to allow a person in a comfortable environment to manipulate objects in a nearby hazardous environment. This is exactly what a "glove box" used for sand blasting, electronic production or chemical handling does here on earth. Why not just invert the view and put the astronaut inside the glove box? This means we get rid of the "suit" concept. Why does an astronaut in zero G need his legs? And separate boots? Why can't he take his hands out of the gloves? The biggest complaint that spacewalking astronauts have is that their hands get cold and tired.

My suggestion is that the entire torso and leg section of the suit be replaced with a rigid canister that the astronaut could rest inside. Perhaps when he "stands up" his head is positioned properly inside the helmet area so he can see, and his arms reach properly into the gloves. When he "sits down", perhaps cross-legged, he has an area where he could eat, drink, relax, and so on. All he would really need is some sort of rigid grapple mounted on the outside of the chest area so he could clamp the suit in position to the structure he is working on. This would keep his motions inside the suit from starting a spin or other undesirable attitude.

A small hole in a glove is very dangerous. Currently, you would have to try to patch it against the escaping air. If you could get your hand out of the glove and apply a patch to the inside it would tend to be self-sealing.

This design would not be significantly different in volume than

the current suits, so the power and air handling systems would be about the same. It would also fit through standard hatches and could be positioned by the same station or shuttle robot arms.

Why hasn't anyone done a design like this? Because it cannot be tested, except in orbit. Astronauts are specifically trained for each EVA, simulating the situations they might encounter while working in space. Current suit designs are used extensively in EVA training sessions underwater. This simulates weightlessness in the general sense, but the astronaut is not weightless *inside* the suit. No astronaut could get valid experience using a glove-box type design until he was actually weightless. Therefore, any realistic design, experimentation, construction, testing, and revision would have to be done in space. There simply are not the facilities, resources, time, or personnel to do this work safely in orbit today. So we are left with the rather ludicrous legacy suits. They are designed to work in two completely different, incompatible environments and do a poor job in both.

A rapid prototyping facility in orbit would allow people in space to build the tools that they envision. And their vision will be completely different from the vision of engineers on earth. Using some of these Replicator concepts should help to make such a facility safe and sustainable.

Many of the systems in use today are the result of legacy discoveries or observations. In many cases entire industries have been founded on the basis of one particular observation or another. As these industries grow and become more ingrained in society, it becomes more and more difficult to rethink the

basic premises. I am concerned that many of the foundations of modern society are based on such fundamental flaws that much of what is being produced today is at risk of catastrophic obsolescence.

I will give two areas of concern, broadly termed Auditory Systems and Vision Systems. I believe we must look to a future when all of our current multimedia industries will seem as though they were turning out daguerreotypes and recordings on wax cylinders.

Auditory Systems

Stereo audio systems are based on the simple observation that people have two ears. The assumption is that spatial discrimination is based on differences in timing and amplitude of the sounds heard by each ear. And you can create some real "Wow" effects by artificially increasing the separation of the two channels.

Unfortunately, this is not all that goes on in audio perception. In the real world, individuals interact with their environment. They turn their head. They use a multitude of cues to identify the location and nature of a sound. This is why surround-sound systems are increasingly popular. They are slightly more realistic.

Human beings are really very good at processing auditory information. Most people just do not realize it. You can tell the difference between standing in an empty room and one with furniture in it, just by the ambient sound. You can tell how close you are to a wall by the sound if you practice for a little while.

Your home theater system will have an audio "sweet spot" where the sounds are most like what the producers intended. If you move away from that spot the audio illusion does not travel with you any more than the visual one does. You get a distorted presentation and the visual cues and audio cues will be mismatched. Some people can take this in stride, like enjoying a roller coaster ride. It makes other people nauseous.

For our purposes here, the problem is that this audio information is insufficient. Each listener needs to be able to interact with the environment. Their own perceptual systems are unique.

The spectral responses of their ears and the changes they expect as they move, breathe, swallow, etc. are all unique and affect the believability of the illusion in subtle ways. Just putting a pounding base line on a sub-woofer does not make a believable motorcycle engine.

Another area of glaring deficiency is the synthesis of voices. One would think that creating believable voices would be much simpler than visual animation. After all, you can recognize the person speaking over a telephone. Single channel, low bandwidth, digitized audio. This should be nothing compared to the data and bandwidth requirements of video.

Even after years of research and development there are only a handful of speech synthesizers that come close to reality. And they are extremely limited, carefully tailored algorithms. I expect a proper speech synthesizer to be able to accurately emulate any human being. If I want it to do Katherine Hepburn, it should sound exactly like I would expect. If I want Sean Connery, that is what I should get.

I should be able to simulate age, gender and accents. I should be able to convey emotion: fear, rage, lust. I should be able to yell or whisper. And foreign languages would be no problem.

Current speech synthesis cannot even get the prosody (rhythm, stress and intonation) of simple sentences right. I expect a proper speech synthesizer to be able to sing. I see nothing unreasonable in requesting my Replicator to produce a rendition of *Pirates of Penzance* as performed by Sean Connery.

In short, I may have a crew of hundreds of animators to synthesize *Shrek*, but I still need Mike Meyers, Eddie Murphy and Cameron Diaz. And the voice parts represent a tiny, tiny amount

of data.

The flip side of this is speech recognition systems. After years of research, they are also a pitiful shadow of what they need to be. Again, we have a tiny amount of data to deal with. No more than eight kilobytes or so per second for digitized audio, like over a telephone. To be rendered into text at no more than about four bytes per second: call it forty words per minute, like a stenographer can do.

Developers discuss such concepts as "speaker independent" voice recognition systems. I contend that there is no such thing. The key to recognizing speech is to have a huge library of speech to compare a given sample to. Our illusion of speaker independence is created by our experience with thousands of different people: we are rapidly choosing among thousands of speaker-dependent patterns. If I were to stand on a street corner and have one hundred random people pass by and speak one hundred random, single words to me I would probably be very slow and inaccurate in my understanding. If, however, each of those hundred random people said a sentence, my brain would be able to pull out age, gender, ethnicity, accent and other factors which would be used to tailor my expectations and make my recognition system much more accurate.

My three-year-old grandson is in the process of building such a universal library for speech recognition. Everyone he hears speak, either directly to him, or in the background, or on television adds to his repertoire of context and recognizable words. He may not know meaning, spelling or anything else, but he will

certainly be able to tell when his mother, father or Spongebob are discussing "crab fishing". As far as he is concerned: same sounds, different speakers, no problem.

Knowledge of the speaker and the expectations that your brain derives from that knowledge is what allows us to pull a single voice out of cocktail party chatter or simple background noise. Tailoring expectations is also what makes it so much easier to understand someone when you can see their lips. The broader your experience and the larger your exposure to different speakers, the more likely it is that your brain will be able to choose a good template to match against the sounds it hears.

I believe that, in the long run, speech recognition and synthesis systems will be parts of a single whole. The speech recognition portion would have examples of Katherine Hepburn to tailor its expectations when analyzing her speech. The speech synthesis would be adaptable and would iteratively feed samples into the recognition system to see how well it approximated the expectations. Just the way a voice actor listens and experiments to learn an accent.

Adaptive systems such as this would make the man-machine interaction much more reliable by allowing the machine to automatically switch to the language pattern most easily understood by the user - for both speaking and listening. This would minimize the misunderstandings in both directions.

Vision Systems

Human vision is very good at spotting important details. We are descended from millions of generations of individuals who were *not* eaten by the saber-toothed cat. We can spot tiny clues to larger patterns hidden in the bushes. Sometimes, we see things that aren't really there, but this is the safe option. To fail to see something that really *was* there could be fatal. As long as we are not too jumpy to find a meal and eat it, we will do OK.

This vision system is very good, but we can have some fun with it. Play some cute tricks. Every grade school child has seen a cartoon flip-book. Make your own little animated character. One still frame after another, your brain interprets it as a moving image. Over the past century, we have taken this trick and built entire industries around it. Movies and Television and Video Games.

The problem is that this flip-book trick bears essentially no relation to what is actually going on in our visual perception. Yes, it usually works. No it does not really allow us to perceive all we could.

Our retina is designed to detect *changes* in light level. We see things when edges pass across the photo-receptors in the eye. If there was no movement we would lose the ability to see any patterns at all within a few seconds as our neurons reached a stable state. Therefore, our eyes are designed to always introduce motion. Tiny tremors known as micro-saccades. There are always edges or changes in brightness moving across our field of vision. The patterns and timing of those changes, coupled with the direction of the saccade are what allow us to recognize objects.

Unlike all current still or digital photography methods, our eyes have no shutter and the spacing of the light-sensitive elements is not uniform. The motion-sensing characteristic is what allows our eyes to function without a shutter. And the distribution of cells in the retina allows us to see fine detail as well as a wide-angle view simultaneously, without resorting to the zoom-lens concept. Even when we are concentrating on fine detail in an object, our peripheral vision is protecting us from lurking cats.

These fundamental differences lead me to believe that the one-still-image-after-the-other movie approach will be replaced with a more appropriately designed technology.

One thing to remember is that the motion detection within the eye is *really* fast. On the order of one hundred times faster than the frame rate in a movie. And your eye position is an interactive part of the perceptual process. If I see an edge move across the movie screen in my peripheral vision I will register one thing. But if I am looking straight at it the edge will skip over so many receptors that I usually just take it for granted that it moved smoothly. In other words, I can tell it is an illusion. Even on very fast frame rates, like an IMAX movie. A big part of the problem is that film tricks like motion blur (as the shutter speed reaches the maximum for the frame rate) just introduce blurred blobs to the retina. There is no sense of direction, just there and not there.

The retina is designed to help figure out what direction an object is moving, and it does so in conjunction with the pattern of micro-saccades and the movement of your body. This is what allows a batter to hit a major-league fastball. He can actually see the seams and accurately judge the motion of a spinning, 2.86 inch diameter sphere coming toward him at over 90 miles

per hour. The total time between the pitch and the time the bat must make contact is less than half a second. There is little chance that anyone could hit a fastball if they were allowed to see only a video or movie of the pitch. It is easy to call it after the fact. But actually seeing it and getting the swing down in time is one of the greatest challenges in all of sport.

The digitization of video images leaves a lot to be desired. Black and white (panchromatic) movie film had excellent sensitivity, resolution and dynamic range. When this was scanned to create old analog broadcast television signals using an old-style "film chain", much of this dynamic range was preserved. In particular, blacks were black and showed a smooth transition to white.

The same cannot be said of modern digital signals and displays, even "high definition" ones. Invariably the sales and marketing hype will emphasize the brightness of the image, or the sharpness of selected scenes. Much of this "Wow" factor comes from unnatural adjustments of color saturation to make the customer think they have been missing something with the older technology.

One key to spotting the limitations that I am talking about is to observe scenes with highlights and deep shadows. Invariably, the shadow will exhibit a "posterization" effect: you can see the contours of digitization steps where the intensity changes by a single integer step. Furthermore, you may be able to spot a "blockiness" in the shadows instead of smooth contours. This is an artifact of the MPEG compression algorithm. Dark shadow areas are also subject to a "crawling" effect caused by slight variations in the way the MPEG algorithm renders the region

from one frame to the next. I contend that the presence of these types of artifacts indicates that this is a technology where the compromises required to "get it to market" have limited the range of material that can be produced for the medium. New productions won't suffer because the directors and cameramen know that shadows don't work. And the old films have now become incompatible with the new media.

This incompatibility is far deeper and more fundamental than the much more obvious and annoying things such as different frame rates and different aspect ratios. All craftsmen strive to achieve quality work within the limitations of their tools. The extraordinary effects that are achieved in one medium may be lost in another. Film makers who, for example, use only the center third of their frame simply because it might eventually be shown on TV are doing a disservice to both their vision and their audience.

There are no synthetic vision systems that take advantage of either the shutterless concept for motion sensing or the fovea-based idea to give simultaneous zoom and wide-angle performance. All because the conventional wisdom says that perception needs a static image of the whole scene. And the flip-book idea is good enough for movies.

Grade school children are also taught all about primary colors and color wheels. Red, green and blue: primary colors of light.

Magenta, cyan and yellow: primary colors of pigment. Simple concepts. It is how color TV screens and computer monitors work. It is how digital cameras work. It is how four color (with the addition of black ink) offset printing works. It is how color film and movies work.

The only problem is that it is *not* how our vision system really works. We are told that there are red-, green- and blue-sensitive cones in our retinas. These are differentiated by three different photopigments within the cells. Upon closer examination, that is only true in the broadest sense. Ten percent of men have only two different working pigments, thus exhibiting red/green color blindness. Up to fifty percent of women have a genetic mutation that produces four different pigments, thus yielding better ability to distinguish subtly different colors.

Many animals such as birds not only have four different photopigments, their cones include specialized droplets of colored oil that narrow the spectral sensitivity of their cones and add to the ability to resolve subtle variations in color. One reason pets do not respond to photographs or television as we might expect is that they perceive colors differently. An image that appears photo-realistic to us will have a cartoon quality to the animals

No matter how much I fiddle with the white balance of my camera or the gamma correction of my monitor I will never be able to come up with a setting that allows my wife and I to agree on a color match. The trichromatic color technology is fundamentally flawed and needs to be revisited in a thoughtful way. We need to transition to full-spectral imaging without an industry-wide upheaval.

Our poor, abused grade school children are also taught all about stereo vision and depth perception. It seems obvious. You have two eyes. The angle between the two as you focus on an object gives you its distance. Ties right in with your geometry class. The only problem is that the effect is far too small to be of much use beyond ten feet or so.

A much more important effect is the parallax of near-field objects against the distant background. You get two slightly different views and perceive it as depth. You can even see the effect yourself with a stereoscope or View-Master.

These stereo effects are all valid, but they do not tell the whole story. You can get depth perception with only one eye. All you need is near-field objects and some motion. When you drive a car your head moves around. The hood ornament or fender or some dirt on the windshield is all that is needed. Using only one eye you can judge distances, park properly, etc. Your brain is fully capable of figuring it out with little training.

I have observed the way a cat's eyes work. In particular, they tend to have eyebrows and whiskers that droop off to the side of their eyes. A little thought on the matter yields the realization that whiskers as near-field objects and micro-saccades give a cat depth perception in their peripheral vision using only one eye. In other words, a cat can be intently stalking a meal, looking straight ahead, and still be aware of exactly how far it is to the nearby branch it is passing. The combination of motion-sensitive peripheral-vision (non-foveal) photoreceptors, micro-saccades, whiskers, the target object and the background gives a tremendous amount of information. Processed by a an astoundingly capable visual cortex, this information allows a level of perception only hinted at by the grade school explanation.

There are many other things at work here. Unlike the modern photographic approach, nature has not attempted to keep the field of vision flat. Distortions arise in the single-element lens and in the spherical curve of the retina. This is not a bad thing - rather it is used to advantage to gain additional information about a scene. The eye rotates about an axis between the lens and retina. As the eye moves, these distortions will help to accentuate and outline nearer objects against the background.

Again, the problem with this misunderstanding of visual perception is that modern technology is only taking advantage of a tiny part of the capabilities inherent in all of us.

These observations have wide-ranging implications. What will an advanced generation of display device look like? How can we make objects with full-spectrum controlled color? Kind of like some sort of super-paint. How will this affect art? What about this interactive, non-static motion-sensing business? How can I design my art so that I control your perceptions? Draw attention to certain parts on a consistent basis.

What can this tell us about pattern recognition? Things oriented at odd angles. Floating in zero gravity.

What about facial recognition? I can easily spot my wife in a crowd. It is harder to do in a picture of her in a crowd, since I don't get any motion cues.. It is really hard in a video of her in a crowd because the camera's point of view is fixed and the resolution is very low. Unlike the real world, focusing on a particular point on the screen doesn't make that area any clearer.

- Implications for data storage and transmission. Alternative to MPEG.

Distribution of cones

Loss and gain of photopigments - not changes in neurology

Learning which neurons are associated with each color

Enhanced edge detection / seven-color spectrum

Unequal angular areas / dim light averted vision

Axis of rotation

Field flatness

Distance accommodation

Kodachrome

- Implications for symbology. Writing, fonts, markings.

Normal writing has a tremendous amount of redundancy. Words are written in a much more complex way than they need to be to convey the minimum information. Most English words, for example can be distinguished from one another by knowing only the letters they contain. The ordering of the letters just gives more, redundant information.

```
Some words are easy to read.
emos dorsw aer aesy ot ader.
```

This is the principle behind the operation of the court reporter's stenograph machine. Just get the important letters together in a group. The order does not really matter.

How can we combine this observation with what we know about vision systems? How can we make fonts or markings more easily recognizable or less ambiguous? If we contemplate weightlessness, how can we make markings easy to read no matter what their orientation?

- Learning systems for pattern recognition.
    - Airport security. Learning prevents false positives, but anything that learns eventually gets bored.
    - Boredom and false negatives. Where to draw the line.

An early Artificial Intelligence system, based on a neural network, was designed and trained to recognize different aircraft as either NATO or Soviet. In the lab, it appeared to perform well but in the field it failed miserably.

The training was done using photographs from *Jane's Aircraft Recognition Guide*. Further research showed that in the training photos, NATO aircraft predominately flew right and the Soviet planes left. The neural net, having no concept of the cold war, was simply figuring out which side of the picture had the pointy-end.

16 The Challenge of Sub-Assemblies

- Maintaining parts in inventory until needed.
- Interfaces. How do parts connect together?
- Fasteners. How do you hold assemblies together?
- Does this introduce unnecessary or undesired features?

17 The Challenge of Perceived Cost

The Fallacy of "Economies of Scale". Yes, you can drive down the cost of an individual object by making a lot of them, but what if you do not really want a lot? You have locked in the design, added the cost of setup for mass production, getting the raw materials in bulk, storing and transporting the finished goods.

Furthermore, you have created the need to convince customers that this specific product is what they need. The entire concept of marketing, market research and the fundamental basis of a Free Market Economy is based on producing more of an item than you needed.

Creating a particular object requires Design, Setup, Operation and Cleanup phases.

If the Operation phase is tiny compared to the total, one would be tempted to repeat it just to create "free" objects. It requires discipline to "Just Say No" to unnecessary production.

If we have the mind set that says we are only going to make one object at a time we are much more likely to engage in trial and error experimentation and make a better product in the end.

When I step into the machine shop today, I rarely have the pleasure of just sitting down and making something. I spend all my time designing. Figuring out how I can tell the machinist how to set up the machines for production. What stock to use. How the CNC program is going to behave for multiple parts. What hand finishing will be needed. I cannot just mill, whittle and gnaw until my object works. I have to remember every detail

of what I do so that I can do it again. And that information has to be in a form that I can communicate to someone else.

The true cost of an object is a function of time and energy.

Long production cycles may also lead to overproduction due to the necessity of starting production in anticipation of demand. If the demand does not materialize you have to deal with the over-supply. And you might have created shortages of other goods preempted by the production of the material you did create.

In biological systems, cellular metabolism is regularly confronted with exactly this type of challenge. The solution that seems to have evolved in nature is that many proteins are regularly created and then digested within the cell without ever being used. The balance between production and recycling determines the amount of the material that is available within the cell. In other words, recycling should be the norm, every bit as important as production. If you need material on an emergency basis, you already have the production in place to handle the demand. You simply cut back on the recycling for a while.

> No one can figure out how much a car costs, least of all the consumer. The modern automotive industry obfuscates costs to such an extent that no individual, inside or outside the industry can tell where the real value lies.
>
> The salesman may attempt to maximize his commission, but this may be at odds with what is best for the dealership. The

dealership manager may attempt to maximize revenue, but this may conflict with the dealership owner's interests which are based on periodic bonuses received for unit sales. The manufacturer may incentivize sales for a particular model using revenue from a different model. Research and development, advertising and overhead costs may be spread across multiple models and years.

In reality, the cost of credit is often the overriding concern when purchasing a vehicle. In the current system there is no way for the consumer to know how much he should pay. Supposed credit scores are based on proprietary formula which may or may not be applicable to the particular transaction. Since there is no reference loan price for a consumer to compare, there is essentially no ability for free-market shopping to operate. There is every incentive for commissioned agents to misrepresent the available options to maximize payments from the customer. Financial institutions should recognize that in a non-transparent system such as this, the abuse comes from their own employees and agents, not the consumer.

The cost of a car to a small colony on Mars would be a completely different matter. It would take a certain amount of time and material to produce the vehicle, and the

```
priority of that production would have to
be weighed against the other needs of the
colony.
```

The Scheduling Conundrum. The Replicator will internally create and destroy the tools and environments for each required stage of processing used to create a requested object. It should be able to schedule its internal use of resources to optimize the fulfillment of multiple, simultaneous Replication requests.

18 The Challenge of Economics

There will always be the need for money, possibly in the form of Replicator Rations. This term was mentioned in *Star Trek: Voyager,* when the crew faced a situation where they were energy-limited.

I believe that the Replicator concepts will allow society to come to a more realistic standard measure of the true cost of goods.

As I have said, the true cost of an object is a function of time and energy. More practically, *inherent cost* is a function of the costs of materials, fabrication, transportation and assembly. This formula can be applied to any object needed at any location.

A house identical to mine might have different inherent costs if it was built in different parts of the country - but only because of different transportation and assembly costs, not because the raw materials were any different or they were fabricated differently. Lumping all these into an inherent cost allows us to examine the contrast with speculative cost. I refer to *speculative cost* as the price that I would pay the builder or current owner of the house if I wanted to buy it.

It seems that the most egregious abuses of the free market concept stem from the deliberate obfuscation of the value of an object. When the consumer has no objective way to determine the value of the item he is buying, the seller has unlimited ability to manipulate prices. The concept of a home appraisal based on local "comparable" sales is a case in point. This can yield only an estimate of the speculative value of a home and can easily be

manipulated on a local basis.

The current mortgage crisis in the United States is due almost exclusively to this kind of abuse. Unlike the stock market with strict disclosure and reporting requirements that allow investors to compare the inherent and speculative values of a company, the mortgage-backed securities market is a speculative market based on speculative values. It is not possible for an investor to know anything about the underlying properties due to the bundled layers obscuring the ostensible inherent value.

Using these Replicator concepts, I propose that sellers of certain classes of objects should be required to disclose the inherent cost of the item. Home appraisals should include a nationally standardized inherent valuation, along with the current speculative appraisal. This would allow ordinary consumers to compare properties in much the same way that fuel efficiency numbers allow a consumer to estimate the operating costs of a car.

The standardized inherent valuation would become a part of the mortgage paperwork and would form an inherent valuation for mortgage-backed securities. This would allow investors to readily determine the comparative risk associated with different offerings by simply comparing the stable inherent valuation with the volatile speculative share price.

There will still be free-market price fluctuations and trade will continue, but better informed investors will exhibit much less risk of unconstrained expansion leading to inevitable collapse. You can still have a housing boom in an area without the artificial and temporary price inflation caused by unbridled speculation.

As part of this change to the appraisal of real property, I would also envision better stability in the taxation and insurance aspects.

Local governments would be better able to budget for their residents if the *property* taxes were a function of the inherent value and thus did not fluctuate with the speculative markets. Local *sales* tax revenue would be based on the speculative price and thus would provide extra revenue during growth and also help to dampen uncontrolled extravagance. This would be a more fair prospect for both long-term residents and recent purchasers.

Insurance companies are currently at a disadvantage in setting fair rates because they have only the arbitrary, speculative appraisal of the property. In general, I would anticipate that the greater the spread between inherent and speculative valuation, the higher the insurance rates would be. You could still insure your home for the speculative value, but the pressure to keep insurance rates reasonable would help to temper any wild increases in that value.

Wise investors will shy away from an investment where the speculative price is too much greater than the inherent value. All that being said, human nature and the lure of easy money will still override common sense. It may be true that the occasional company such as Google will be a good speculative investment at a hundred times their inherent value. But that doesn't mean that every startup in the Dot Com era was worth that valuation. And even the best disclosure laws cannot protect people from the consequences their own greed.

Now, or in the future, I am not anticipating any communist utopia. There will still be plenty of opportunities for abuse. The Martian colony's commissar will still be able to divert resources by scheduling materials for his new dacha in the foothills of Olympus Mons ahead of the weekly bread production.

19 The Challenge of Permanence

Our culture will completely change if we view literally nothing as permanent.

Buildings, rooms, furniture, clothes will all be temporary and disposable. Each would exist for its own time span - buildings might change seasonally, rooms weekly, furniture daily and clothes hourly. The amount of space *allocated* to each individual would shrink as that space performed multiple functions, but the amount of space *available* to each person would increase as the world became less crowded with unused objects.

Architecture

Roads and Bridges Dynamic redesign, construction and maintenance on a continuous basis. Maintenance lanes as a design feature. Continuously adding and removing capacity as demand changes. Budget for continuous smaller projects instead of enormous, long-duration construction / disruption followed by half-hearted maintenance. How would this affect ancillary requirements such as drainage?

Buried and Forgotten Infrastructure How should items such as water mains, aqueducts, sewers, oil and gas pipelines, power and telecommunications cables, steam pipes, etc. be handled? Everything should be installed with an eye toward the ability to continuously inspect, maintain, remove and replace.

Property insurance is based on the assumption that things are more-or-less permanent and that there is a significant cost associated with replacing them. In a Replicator future the concept of

property insurance would probably not exist.

Carried to the extreme, Replicator technology would make the ultimate disposable culture. The design of everyday objects would change since they would need to be created rapidly, used once, then immediately recycled. The containers that products arrive in would also be radically revised.

Much of the trash discarded by modern society consists of containers and packing materials designed to protect a product from the time it is manufactured until it reaches the consumer. This storage interval has led to a tendency to use more and more indestructible materials, even for products with a limited shelf life. Shipping milk with a shelf life of two weeks in a container that will not degrade in ten years is somewhat incongruous.

> The (perhaps apocryphal) story of Henry Ford in the days of the Model T springs to mind. It is said that he sent accountants to inventory junkyards containing Model T's and measure the condition of each part. When the statistics were compiled, he ordered parts that were never found to be worn-out to be built to a lower standard.
>
> If the car was discarded before the part wore out, you paid too much for the part.

I contend that no deregulated wireless telephone company has ever been profitable. The technology and infrastructure is

changing too fast for it to be both installed and paid off before it has to be replaced. Thus, the current revenue of the phone companies is going to pay off debt and write-downs incurred for previous generations of hardware, even while they are borrowing for the future.

I am in favor of dynamic replacement of infrastructure and recognize that the more advanced we get the more maintenance is required to keep the systems functioning.

These costs should be accepted on an ongoing basis, not as part of a wave of speculation based on projections of revenue from future generations. The companies would be making wiser investments if they chose technologies that could actually be profitable within an established area and time-frame, instead of playing a shell game with different technologies while misleading both their investors and their customers.

Doing costly roll-outs of stop-gap incremental technologies just to achieve a small incremental increase in subscriber revenue is such an example. An industry that touts their numbers of new subscribers while obscuring the actual nature or costs of the service they are providing and doing nothing to satisfy or retain their current customers gives the appearance of a pyramid scheme. The real danger is that if the demand falters even for a moment the shift in cash flow in a single large company could cause the collapse of portions of critical infrastructure across the entire country.

Replicator concepts inherently allow for multiple, incompatible technologies. Overlapping functionality can be provided in many different ways. Making these options available to the consumer is important for meeting their differing needs and

requirements. But these choices should be made on an informed basis, not with misleading descriptions and inaccurate costs.

20 The Challenge of Quality

Quality will tend to improve incrementally as patterns are revised.

It will be unlikely to have a million defective objects in the hands of consumers if everything is created and disposed of as needed.

> In the South Seas there is a cargo cult of people. During the war they saw airplanes land with lots of good materials, and they want the same thing to happen now. So they've arranged to imitate things like runways, to put fires along the sides of the runways, to make a wooden hut for a man to sit in, with two wooden pieces on his head like headphones and bars of bamboo sticking out like antennas--he's the controller--and they wait for the airplanes to land. They're doing everything right. The form is perfect. It looks exactly the way it looked before. But it doesn't work. No airplanes land. So I call these things cargo cult science, because they follow all the apparent precepts and forms of scientific investigation, but they're missing something essential, because the planes don't land.
>
> – *Richard Feynman*

Caltech Commencement Address, 1974

Cargo Cults

ISO 9001

Glove boxes and fungi

Anti-static straps

Temperature-critical epoxy and commercial fishing

Adrian Monk and I are probably the only two people in the world who are obsessive about ice cube trays. I like to use ice cube trays as an example when I discuss quality in today's society.

I believe Rubbermaid makes the best ice cube trays. They are shaped correctly so that the ice cubes do not stick and simply fall out when the tray is inverted. They stack correctly so that, even if over-filled, they do not stick together. They last a long time since they are not being forced to bend while cold. I used the same set of Rubbermaid ice cube trays for fifteen years.

There is a lot of engineering that went into these trays. There is no business case for a quality ice cube tray. Nobody else makes a

quality ice cube tray, and the customers all expect a no-good product. You can't tell good from bad until you use it. Nobody advertises ice cube trays.

How can Rubbermaid afford to design and develop a quality product and sell me one set in fifteen years when the competition is selling a set every six months and spending nothing on engineering?

The answer that I usually use in such discussions is that Rubbermaid is protecting their brand. No individual product can justify the cost, but, on the whole, Rubbermaid is more successful because it sells a variety of quality products.

21 The Challenge of Environment

This refers to the environment in which a particular Replicated object must be created or used.

Depending on the size of the object and the tools required to Replicate it, there must be sufficient space available within the Replicator.

There may be exotic requirements for processing within the Replicator such as gravity, air and temperature. This becomes especially important if we contemplate a Replicator in space or on another planet. We should not have hidden assumptions about things like sea-level on earth.

We also must be aware of possible contaminants from the outside as well as from previous or concurrent processes within the Replicator.

We must have mechanisms in place to ensure the required cleanliness and safety within the process flow.

22 The Challenge of Yield

- Cleaning and Inspection
- Wear and Tolerances
- Production Testing
- Rework

23 The Challenge of Waste Management

- Processing by-products
- Scrap materials
- Left-overs from when something doesn't work or the process breaks down in the middle.

24 The Challenge of Transportation

- Getting Raw Materials to the Replicator as needed.

Getting the finished objects to where they are needed. Presumably the Replicator is too big to be on-site for every application.

- Packaging the finished goods.
- Transportation of finished goods to where they are needed.
- Unpacking the objects.
- Recycling anything used in the transportation process.
- Recycling objects after they are no longer needed.

> Consider the resupply situation for the International Space Station. Every few months or so, supplies are sent to the space station aboard the space shuttle or Russian Progress cargo ships at a cost of approximately $1000 per pound.
>
> Used equipment, packing materials, and the perfectly good, working cargo ship are then discarded to burn up in the atmosphere. All because there is no safe way to store or

recycle the materials, some of which could become extremely hazardous in the confined volume of the station. Also, the lack of maintenance and limited lifetime of certain components (especially seals, pyrotechnics and the fuel systems) mean that the cargo ships would be unsafe.

Careful rethinking and redesign with the goal of eliminating all possible "down-mass" operations would make inhabiting space much more cost effective. Designing equipment to be disassembled into reusable material or components inside the station and designing the vehicles themselves for permanent attachment to the growing station would provide much needed additional space as well as shielding and safety from orbital impacts.

If cleverly done, this could provide the basic research into techniques for building really large structures in orbit. Questions about orbital boost and decay, dynamic stability, structural strength, modes of oscillation, etc., could all be studied. But not if the design requires that we continue to regularly discard tons of mass that we already paid to put into orbit.

25 The Challenge of Assembly

Replicating large objects might require assembly outside the Replicator. How is this accomplished and what equipment is needed?

Assembly needs to be automated. How would we ensure that it is done right?

How is this similar to assembling Sub-Assemblies within the Replicator?

26 The Challenge of Recycling

- Patterns for safe disassembly.
- Recognizing objects to be disassembled, even if broken or damaged.
- Getting useful raw materials from recycled objects.

A schoolchild's terrarium is a closed system, except for energy input. Plants grow, transpire, die and decay in a system where decayed material is decomposed by soil microbes. Inside the terrarium, almost nothing is created that cannot be recycled within the environment.

A terrarium is not truly sustainable, but will last a long time. It would have to be much larger to sustain even small mammals.

27 The Challenge of Not Dying

I have made a list of classes of things that can kill a person. In our technological future it would be nice if these things were kept to a minimum.

- Temperature Extremes
- Pressure Extremes
- Acceleration Extremes
- Radiation Exposure
- Impacts
  - Penetrating
  - Crushing
  - Shearing
- Electrocution
- Chemical Exposure
  - Corrosives / Acids / Alkalis
  - Solvents
  - Chelating Agents

- Respiratory Failure
  - Hypoxia / Drowning
  - Poisons
- Starvation
  - Improper Food
  - Vitamins
  - Micro Nutrients
  - Poisons
- Allergens
- Carcinogens
- Pathogens
  - Parisites
  - Bacteria
  - Viruses

Replicator Technology should allow us to create objects needed to guard against these dangers, and should not accidentally introduce any of these things into our environment.

We can, perhaps, agree that it might be a bad idea to replicate sharp, pointy things for a small child. Or medicines. Or cans

of spray paint. But where do we draw the line? The Replicator itself needs internal safety systems but anything further becomes more and more subjective.

The whims of various local jurisdictions become factors. Just as the control of Intellectual Property issues may dominate discussions of Replicator Patterns, actually invoking those patterns may be subject to arbitrary local control.

One of the primary advantages of considering objects as essentially equivalent to their Patterns is that the Pattern can be transferred intact to distant lands and used without the interference of (possibly) well-meaning Big Brothers.

Perhaps unlimited access to LSD would be a bad idea. Timothy Leary would argue that most early problems with LSD were related to the low quality, contaminated product produced by amateurs.

More widespread drugs (such as THC, the chemical that makes marijuana so popular)are tending to swing from outright bans to a more reasoned admission of legitimate medical uses. But the criminalization, stigma, and outright lack of quality or safety in the essentially uncontrolled production and distribution makes any serious discussion practically impossible.

In this day and age, access to acetaminophen (Tylenol) is being curtailed because it is combined into so many different over-the-counter medications that even conscientious adults are overdosing and damaging their kidneys without realizing it.

Posession of, or access to, certain materials is subject to seemingly arbitrary rules. See the essay I Have A Cat for a description of *pet food* available by prescription only. In this case, the prescription-only requirement is simply a very profitable

method of restricting access, controlled by a select clique of intelligensia and their political cronies.

It turns out that the unlicensed posession of an Erlenmeyer flask is a federal crime. And, furthermore, if I had an Erlemneyer flask and gave it to you, we would both have committed a separate crime. How can this be? It seems that Erlenmeyer flasks are on the list of 'Controlled Precursors' in the War on Drugs. Which is apparently just a way for charges to get piled on in case the operator of the local drug lab happens to get crosswise with local law enforcement.

28 The Challenge of Pattern Design

Patterns must describe the desired object in a form that can be implemented by the Replicator.

For a start, let us think of a Replicator Pattern as a kind of recipe. For reference I have included a particularly good recipe. Any cook knows that you (or your mother) must have read *The Joy of Cooking* before you are actually allowed in the kitchen. Thus, you have the background knowledge to be able to measure flour (don't pack it down), and what it means to "cream the butter".

I have modified the presentation from a normal cookbook format to include the required equipment. If you try making the cookies yourself, see how many little things I have left out. Experiment. Try baking cookies on a clear, cold day and again on a hot, rainy one. Do all the cookies come out nicely brown? What do you "get to know" about your oven. Do you really leave them in the oven for 10 minutes, zero seconds? Or do you peek and take them out when they look and smell nice.

How do you get the cookies off the cookie sheet? When? Are we going to need a cookie jar?

Chocolate Chip Cookie Recipe

Equipment:

```
blender          measuring spoons
grater           measuring cup
mixing bowl      cookie sheet
spatula          oven
```

Ingredients:

```
2 cups butter                1 tsp. salt
4 cups flour                 1 Hershey bar, 8 oz.
2 tsp. baking soda           4 eggs
2 cups granulated sugar      2 tsp. baking powder
2 cups brown sugar           3 cups chopped nuts (your choice)
5 cups oatmeal               2 tsp. vanilla
24 oz. chocolate chips
```

Blend oatmeal in blender to a fine powder
Cream the butter and both sugars.
Add eggs and vanilla.
Mix together with flour, oatmeal, salt, baking powder, and soda.
Grate Hershey bar and add to mix.
Add chocolate chips and nuts.
Roll into balls and place two inches apart on a cookie sheet.

Bake for 10 minutes at 375 degrees.

Makes 112 cookies.

Now we have a batch of cookies. And a terrible mess. How do we know what to do with the leftover ingredients and the dirty dishes? Not even *The Joy of Cooking* is any help here. We have common knowledge and some vague instructions from the dishwasher's Operator Guide.

I believe that a Replicator Pattern must restore the system to exactly the initial state. And it should have provisions for cleaning up and recycling the inevitable accidents - like the egg that got dropped on the floor.

1. How many patterns would be required to recreate the Replicator itself?
2. How many patterns would be required to support a sustainable human population?
3. How many patterns would be required to support a modern technological society?
4. How many patterns would be required to support a colony in space or on Mars?

I expect the Replicator to be able to internally derive processing steps and self-optimize operations without explicit direction.

Patterns must include tolerances and test parameters to be used during production to detect flaws.

Patterns must describe deconstruction steps to be used during recycling. This must work with incomplete or damaged objects.

Deconstruction patterns are also used to handle production waste by-products such as materials and used tools.

Software tools must be available to allow ordinary human beings to create and edit patterns safely and successfully.

Patterns must be standardized in such a way that they can be transmitted from one Replicator to another.

On April 14, 1912, a clear, moonless night in the north Atlantic ocean with unusually calm seas, impact with an iceberg fractured hull plates and rivets made brittle by the frigid water leading to the loss of the *RMS Titanic* and the deaths of 1,517 people.

On May 24, 1993, Joe Bill Dryden, one of the most knowledgeable and experienced F-16 test pilots in the world took a brand new, properly performing, aircraft on a clear day over level terrain and performed a Split-S maneuver into the ground. He ejected safely, but the parachute descended through the resulting fireball and he was killed by the impact.

On January 28, 1986, hot gasses leaking past a redundant pair of frozen o-rings on the right solid rocket booster weakened the supporting structure and ruptured the external tank of the space shuttle *Challenger*seventy-three seconds after liftoff. The $1.2 billion spacecraft and its payload were destroyed and all seven astronauts were killed.

On February 23, 2008, at Andersen Air Force base in Guam, water intrusion into three of twenty four skin-flush air-data sensors caused faulty air speed and attitude information to be sent to the flight control computers of the B2 stealth bomber *Spirit of Kansas*, leading to a crash immediately after takeoff. The two pilots ejected safely, but the $1.4 billion aircraft was destroyed.

> On March 27, 1977, at Los Rodeos Airport on Tenerife in the Canary Islands, the KLM Boeing 747 *Rijn* piloted by Captain Jacob Veldhuyzen van Zanten, head of KLM's flight training department, began takeoff without proper clearance and collided with the Pan Am Boeing 747 *Clipper Victor* taxiing in the opposite direction on the runway. Both aircraft were destroyed and 583 people were killed.

The co-pilot knew that the Captain was making a mistake. The control tower knew that there was another plane on the runway. With the top-down command structure that was in place in 1977 no person had the authority to override the pilot in command. It also did not help that, as head of the training department, the Captain reviewed the co-pilot's performance, not the other way around. Since this accident, most airlines have implemented "Cockpit Resource Management" programs in which the flight crew are cooperating team members who cross-check and confirm each other's actions.

29 The Challenge of Reliability

- What if the Replicator breaks?
- How can you be sure that you have parts to repair it?
- How can you be sure you know how to diagnose and repair the replicator?

Consider a space ship. You cannot have skilled specialists, knowledgeable in every aspect of the replicator, just sitting around waiting for it to need repair. If you are looking at the long term, how do you train the next generation of specialists? Especially if the Replicator is fairly reliable and specific pieces only break every 100 years or so. The required skills may have been lost generations ago.

Solution: Only allow your space ship to use a Replicator that was built entirely by a Replicator. This way you ensure that all necessary skills are codified within the machine. And always ship at least two Replicators so that one can repair the other.

- What about Really Bad Things? Bad Raw Materials that might damage all the Replicators? Pathogens: physical or computer viruses that could damage or otherwise render the Replicator inoperable. Maybe damage *all* Replicators in ways that unskilled users could not diagnose.

30 The Challenge of Obsolescence

Replicator concepts allow us to preserve the knowledge and techniques required to fabricate physical objects. In many cases this will be quite valuable, but it must be realized that many objects are useless without their context. As society moves forward and people use newer techniques to accomplish a goal the infrastructure of the past will fall by the wayside.

> Properly operating a steam locomotive is a frightfully complicated task. Unlike most modern engines that essentially have only a starter and throttle, steam engines require careful attention to temperatures and pressures in the firebox, boiler and steam chests. Water levels, fuel flow, lubrication, replenishment of fuel, water and sand also require the attention of the (usually two-man) crew.
>
> A steam locomotive operates differently under different weather conditions: temperature, humidity, rain or ice all significantly affect its operation. The crew must anticipate changes in load caused by hills or curves. The mechanism is very slow to respond to changes in settings and it can be extremely difficult to restore normal

> operation if any single parameter slips out of tolerance. Operating a locomotive at 30 mph is a completely different thing than operating at 60 mph. It took years of on-the-job training for an engineer to learn to safely control a steam locomotive.
>
> A Pennsylvania Railroad steam locomotive (#460, the "Lindbergh Engine") pulling a tender, baggage car and coach car, is said to have reached a speed of 115 mph on June 11, 1927. I am amazed by the skill of the crew, and the level of trust that they would place in the metallurgy, assembly and maintenance of the locomotive, cars, wheels, tracks, crossings, etc.

There is no reason to believe that a Replicated steam locomotive would not be fully functional. But it was designed to be operated by a trained crew and run on properly installed and maintained tracks. If I were to need to actually operate my new locomotive I would need to be able to replicate an entire infrastructure of physical objects as well as draw forth the knowledge and skills needed to use this technology.

Many technologies have fallen into disuse - sometimes through obsolescence, sometimes through simple neglect. The ability to use the technology that will eventually be embodied in the Replicator's "Universal Library Of Everything That Has Ever Been Made" will depend on that library containing much more

than physical construction specifications.

We need to think about the concept of the Operator's Manual in the same radical way we think about disassembly and recycling. The manual needs to be actually useful in describing how the object is intended to be used and what standard level of background knowledge is expected of the user. Making documentation for a truly broad audience does *not* mean copying the same unintelligible instructions into seventeen different languages.

31 The Challenge of Hidden Feedback Systems

- Non-proportional Feedback
- Binary signaling
- Single Control Paths
- Proportional Control Systems
- Redundancy
- Operational Envelope
- Detecting Systems at their limits
- The pieces all work as designed but the system fails
- Single point failures
- Thermostats with on-off signals. Neurons with firing-rates and action-potentials. Crosstalk as a good thing.
- Multiplexed data and packet data.
- Multi-path redundancy.

I used to teach scuba diving. This led me to a study of the effects of breathing gasses under pressure.

In the early twentieth century, J. B. S. Haldane came to the conclusion that different "compartments" of the body (different tissue types) dissolved and released gasses such as nitrogen at different rates. His work led to the Dive Tables used by the Navy and sport divers today. The tables give empirically derived values for the amount of time that a diver can spend at a given depth and the rate at which he can ascend to the surface without the risk of dangerous bubble formation in the tissues.

In my own analysis, attempting to work from first principles, I was never able to prove that any change in atmospheric pressure was safe. This is not a matter of physics or chemistry, or even biology, per se.

The human body contains many interlocking feedback systems, governing the distribution of oxygen, removal of carbon dioxide, balancing electrolytes and enzymes, maintaining temperature and controlling digestion and excretion. All are the product of billions of years of evolutionary selection for individuals that did not die when a low-pressure weather system passed by.

People die when a biological feedback system

hits its limit and is no longer able to compensate for current conditions. The problem is that these feedback systems are completely hidden from view. The empirical tests conducted on 20-year-old Navy SEALs do not generally apply to 50-year-old overweight smokers in a dive class.

In my youth, a friend of mine studied swatting flies. He discovered that a fly sitting on a table is very good at evading a hand. The fly's reflex makes it turn away from the attack very quickly. If, however, there are two attackers the fly is not nearly so successful.

Therefore, if you hold both hands about six inches apart, parallel to the direction the fly is facing and an inch or so above the table, and then clap above the fly, you will almost invariably hit and stun the fly.

The fly takes off when it senses movement, either visually or through air currents. It will fly straight, turn left or right, or (interestingly) even backwards, but this attack defeats all those options.

The fly's reflex is similar to the one that kills so many armadillos in the road.

The near-sighted armadillo would likely be safe if it just sat still. But when it detects the car it jumps up into the passing undercarriage.

---

32 The Challenge of Intellectual Property

Intellectual property issues will become more important as Replicator technology becomes more widespread. The fact that all of society will be driven by patterns exchanged as simple data streams means that the users need to be informed about the sources and reliability of that data. Some method of rewarding creative individuals who improve the patterns must also be provided.

Credit vs. Control. It seems to me that two incompatible requirements have been bundled into the current implementation of copyright and patent law. The desirable goal that an inventor should be able to profit from his creativity is inexplicably coupled with the idea that he should have absolute control over that which he has invented.

A really popular invention may generate lots of revenue. I do not understand why this should result in either a greater or lesser *percentage* of that revenue reaching the inventor.

- Trademarks, Patents, Copyrights, and Trade Secrets

Patent law should include "Must License" provisions to allow independent use of discoveries or techniques. The all-or-nothing approach currently used, along with its arbitrary expiration dates leads to many abuses. The definition of appropriate compensation for a license should be based on an independent assessment, not the whim of the patent holder. The ability to

hold an invention hostage stifles the ability of creative individuals to develop and improve their ideas and leads corporations to costly and unnecessary reverse engineering or independent development.

- Copyright is not appropriate when everything is based on prior work and the originality of a single individual is insignificant in the scale of things.
- Disney and the RIAA *vs.* the world.
- Single-bit changes and derivative works. Introducing noise. Re-mixing, creativity and copyrights.
- Trademarks and Replicator Patterns.
- Ego and Collaboration
- Issues of Quality
- Issues of Extensibility. Patterns must provide a foundation for future development and improvement. No aspects may be hidden.
- Replicator Usage is a Privilege not a Right.

I tried to watch *Alien* on my cable channel's Fright Festival. What I saw was not *Alien*, but rather a derivative work: an original compilation by a nameless marketing director. Short clips from *Alien* interspersed

with thirty percent randomly selected advertising, periodic banners telling me that watching *Alien* now is not as important as the fact that *Swamp Thing* is coming up next. And a ubiquitous television channel logo.

If I were Ridley Scott I would be mightily offended. At no time did my vision for *Alien* include the juxtaposition of H. R. Giger art and *Girls Gone Wild*.

Advertising this show as *Alien* is false on two counts. First, it is not *just* Ridley Scott's *Alien*, as we are led to expect. Second, claiming it to be Ridley Scott's work deprives the marketing director of his rights of authorship for the original compilation that is actually broadcast.

In fact, if I had the stature of Ridley Scott, I might be tempted to enforce draconian licensing requirements on my works. No butchery. No editing. No overlays. No advertising. No time compression. And, in this age of high-definition, No Pan-and-Scan. Basically, if you want my work, you will take the whole thing. My way.

Cable television needs the content creators a lot more than the content creators need cable television. Direct sales and the Internet are much more lucrative than residuals for late-night TV.

Interestingly, Georgia O'Keeffe placed unique restrictions in the licensing for her wonderful paintings. She required that all reproductions be smaller than the originals.

This is a brilliantly simple use of the current copyright laws to protect the brand and preserve the wonder of the originals, while also allowing a wider audience to be exposed to her work.

33 The Challenge of Semantic Blindness

- Missed opportunities outside of the scope of Replication
- Opportunities too good to be true. Or too good to last.
- Replicator foundations and evolution. Replicators will certainly get better or more efficient with time.
- Divergent human societies. Cultures connected only by exchanging replicator patterns.

> You can't describe a concept if you don't have words for it.
>
> You tend not to think about things you cannot describe.

Nature never invented the concept of permanence. Everything in nature is either being actively used and maintained or it is being recycled. Corrosion, decay and death are all part of the natural process of recycling building materials.

Nature never invented the wheel. There is no obvious way for a true rotating joint that conveys nutrients and seals off the outside environment to evolve naturally.

Floods, drought and fire are all naturally occurring events. Organisms have adapted to tolerate, use, or even require all three. All organisms manage moisture and transpiration, based on their expected environment. Beavers even pro-actively manage floods. Only man creates, manages and destroys fire.

34 Roadmap to a Replication Future

Selection of the minimum subset of Replicator requirements and functions.

Redesign common objects to be Replicator-friendly.

Identify strategic objects that will not be able to be Replicated.

- Quantum mechanics.
- Copies of tokens. Alternate time-lines. Schrodinger's Cat.

There is nothing inherently wrong for me to sell multiple tickets for the same airline seat. The world continues with a superposition of two (or more) people believing they will fly on the plane. My customers are happy up until two of them actually show up and try to sit in the same seat. At that point, the wave function collapses and a single reality is restored. One of the cats lives, the others die. My only problem is dealing with the dead cats by explaining the fine print in the contract.

- Replicating money. Currency and Concert tickets.

Likewise, I see nothing inherently wrong with copying currency. U. S. paper money already has unique serial numbers. In this age of telecommunications and cryptography this serial number concept simply needs to be extended.

Modern cryptographic tools allow the creation of un-guessable serial numbers. By this I mean a serial number that is long

and complex enough that no-one could guess it (in a reasonable amount of time) without having seen it.

The Federal Reserve needs to issue notes with cryptographic serial numbers and maintain an on-line verification facility. At any time I can have this master system confirm the validity of my note and exchange it for a brand-new note with a newly created number while simultaneously invalidating my old bill. The key here is that the long, un-guessable number is the real "currency". You could print it on your own printer. You could make backup copies to your heart's content. As soon as the first person *uses* the currency, however, all the copies are no longer valid.

By using a cryptographic aspect to the serial number I can still perform off-line operations, confirming that this is real, U. S. currency. I just do not have absolute confidence that I can cash it in until I can contact the validation system.

This virtual currency is exactly what happens today with a personal check. If I have $100 in the bank, I can still write $100 checks to each of five different people. The first one to the bank wins.

Making the cryptographic serial number un-guessable is important, not only to allow some degree of off-line verification, but to prevent trial-and-error theft of other people's cash. This is similar to having a set of rules that can be used to make up new words. The rules can be used to determine if something looks like a real word. But you still need a dictionary to be sure that the word exists, and what it means. The short serial numbers used today are an open invitation to forgers who simply print bills with numbers similar to a known good bill.

For example, one could use a sixty-letter-and-digit serial number

in place of the current eleven-character format. This could give you a 300-bit value that would include a public-key signature to prove authenticity.

Now, suppose you want to send money to your daughter at college. Just read the number off of a bill from your purse to her over the phone. Western Union, eat your heart out!

You would want to be careful about leaving your cash laying around. Someone could steal it just by copying the number. But the same is true with modern currency - if you leave it laying around it will probably get stolen.

You could carry your entire bank account on an encrypted thumb drive. And leave a backup copy at home. If it gets lost or stolen, you lose nothing. And the "bills" that you spent before losing the drive are just "no longer valid" on the backup copies. The encryption makes the lost drive useless to anyone who finds it.

Very few people can remember a sixty-character value after just glancing at it. Lots of people, however, can remember the five to nine digits on the end of your checking account number. I am surprised that more so-called identity theft isn't perpetrated by simply printing bogus checks on account numbers you see at the grocery store.

Doing bill-by-bill verification against a master database would not be inherently more difficult than current credit card verification. There would be a higher volume of simpler transactions, though.

Something similar to this is already in use by the U. S. Postal Service. Various companies offer "print your own postage" programs. These generate a barcode that is scanned by the post office to verify payment of postage. The barcode can be

printed or copied any number of times but the first time it is used (scanned at the post office) the copies become invalid.

Now let's see how this idea stacks up.

The Bureau of Engraving and Printing might or might not like it. They do not actually get to print anything anymore and get turned into database administrators. On the other hand, they were fighting a losing battle against the counterfeiters, anyway.

North Korea would hate it. No more super notes.

The Secret Service would love it. No more super notes.

Banks would love it. They do not have to deal with cash anymore.

Retailers would love it. A simple scanner could read the "bills" that customers could print on ordinary paper. Just like handling coupons. And no theft or robbery problems.

Drug dealers would love it. No more unwieldy suitcases of cash.

The ACLU would hate it. Big Brother might track the serial numbers.

Your daughter would love it. Instant party time.

This Verify-Invalidate-and-Replace strategy could be applied to almost any value token. Airline tickets. Concert Tickets. E-Commerce specie. Each scenario would require a method for

dealing with fraudulent sales, but detecting the fraud would be greatly simplified.

A variation of this single-use verification and replacement strategy could be used to replace personal identifiers such as Social Security numbers, credit card or bank account numbers. Using the ability to make duplicates invalid would prevent most of the current forms of identity theft.

Note that what I am talking about here is not one of the elaborate cryptographic security schemes developed in academic circles. Yes, it is a variation of a public-key system. But, no, it does not require users digital signatures or encrypted transactions (beyond basic network communication security such as SSL). It is not invulnerable, but it also does not suffer from the key management and key revocation issues that plague those academic systems.

The generally temporary nature of a particular token ("bill") and the possibility of local "trusted issuing authorities" fit in with my general theme of Replicator Technology. Each community, country or colony could have its own Issuance and Verification center. This provides the local, autonomous operation that I look for. And a simple communication link with other authorities could provide exchange of payments and allow honoring of another colony's money.

- Trademarks and the security of Replicator Patterns.
- Cryptographic Certification.

III Essays

I Have A Cat

On The Failure Of Capitalism

On Inventory Management

What Is A Source File?

Malevolent Social Engineering In Open Source Software

35 I Have A Cat

My cat is diabetic. When he was diagnosed, the vet said that grocery-store pet food is almost certain to cause diabetes. Let us examine the reasoning.

Dogs and cats evolved as carnivores, and as such have metabolisms adapted to a high-protein diet in support of a fairly active lifestyle. When kept as pets they are granted access to essentially unlimited amounts of food that takes little or no effort to obtain. And their activity is restricted by being confined indoors or in a small yard.

Protein is generally expensive, so pet food manufacturers tend to add fillers in the form of carbohydrates. This led my vet to refer to these products as 'diabetes in a bag'.

In order for the pet food manufacturer to stay in business, customers need to buy their products. Therefore *anything* that might trigger a pet's natural finickienss is carefully suppressed. And flavor enhancers, salt and other substances are added. As far as the manufacturer is concerned, the pet must keep eating this brand at all costs.

It is also necessary for the pet owner to be satisfied with the apparent quality of the food. He expects nothing less than absolute consistency year after year.

It is also bad form for the pet food to have bugs in it. It used to be that finding the occasional weevil or beetle in a bag of dry pet food was more-or-less expected. It has been years since I saw any bugs in the pet food. And if I spill some on the floor in the garage? Still no bugs. Hmmmm. Could it be that

there are insecticides in the pet food? No labeling requirements. Ignorance is bliss.

All of this actually works. The life expectancy of a pet is perhaps twice as long as the same animal in the wild. In the wild, death is usually from exposure or predation. Pets die of organ failure and tumors. Overall, this is probably a good deal.

And pet food is readily available, reasonably priced, and generally safe in the short term.

To manage my cat's glucose levels, my vet recommended Purina Veterinary Diets DM Dietic Management Feline Formula cat food. This stuff is cat food with less carbohydrate content. It is also *prescription* cat food. It is just food with added things (that were not necessary in the first place) left out, but it is still *only* available by prescription.

And it costs $5.00 per pound. My cats eat better than I do.

My cat requires insulin injections. He weighs fourteen pounds and only requires two units of insulin twice a day. Feline insulin is only 40% as potent as human insulin because smaller quantities are needed and the dose needs to be a certain size to measure accurately with a syringe.

Therefore, syringes with different markings are used for 40U insulin than the ones used for 100U human insulin. Naturally, these less-common 40U syringes are more expensive.

One can measure any desired *actual* amount with either syringe. You just have to be able to do the simple math to convert the proportions. Even mentioning the use of an alternate syringe to a Health Care Professional is likely to cause an apoplectic attack. There is "the right way to do it" and that is the *only* way to do it. Because that is what the instructor said in *Doctoring 101*, and because if you give any other answer on the Standardized Certification Exam you are likely to risk your license.

The Prozinc feline insulin is actually a suspension of human insulin and additives that are designed to make it more effective in the feline metabolism. It comes in bottles containing 400 units. This means that one bottle will last me about 100 days.

In general, insulin will last up to 30 days without refrigeration, but much longer if kept properly chilled (*never* frozen). OK. If I am careful, I can get full use out of the bottle.

The insulin suspension will separate fairly quickly when stored. It is imperative that it be properly, gently mixed before withdrawing each dose. Even an occasional lapse here will result in an overdose or underdose in the rest of the bottle that will mess things up badly over the course of the three-month life of the particular bottle.

The insulin bottle has a rubber stopper that is penetrated by the syringe to withdraw a dose. A syringe will be inserted and removed through this stopper 200 times. This is a lot of wear on the single stopper. Even if one is careful to keep the surface clean, over time there is no telling what foreign material gets pushed into the bottle along with the needle.

Each time a dose is removed a comparable quantity of air must be injected into the bottle. This is accomplished by pulling the

syringe plunger back in the air, sticking the needle into the bottle, injecting all the air from the syringe into the bottle and then withdrawing the proper dose of insulin.

This process takes whatever is in the air (pollen, mold spores, dust, etc.) and injects it into a previously sterile media. And is repeated at measured intervals two hundred times.

Yet another reason to keep the bottle carefully refrigerated. And my fingers crossed.

I have alluded to several problems that could be solved if proper attention were given to the entire process.

1. Insulin is one of the simplest proteins and should be very easy to synthesize.
2. There should be a clean and easy way to measure the concentration of proteins in solution.
3. There should be a simple method of detecting foreign material in a solution.
4. There should be a realistic way of creating pet food to a specific recipe, in quantities and at a cost that makes small-scale, local production possible.
5. The regulatory environment that allows pet food to be available 'by prescription only' is another egregious example of abuse of intellectual property rights.

36 On the Failure of Capitalism

The free market economy worked reasonably well in a 20th century world composed of multiple, competing nations and corporations. The 21st century represents an entirely new challenge. The unprecedented globalization and multinational aspect of every facet of the economy has led to a complete lack of competition in every important aspect of society. This lack of competition removes the key protective feedback required to make capitalism a functional economic concept.

Caveat Emptor

All of the platitudes that our parents and grandparents tried to instill in us as guidelines for dealing with the world are rendered fallacious in light of insufficient knowledge. All modern advertising and promotion actively prevents the customer from knowing what he is buying. This allows the seller to redefine the transaction at his own discretion. The customer is unable to compare offers from different suppliers on any basis other than marketing propaganda.

Obfuscation Wins

Electric and telephone companies, insurance companies, banks, automobile manufacturers all make their money by knowing more about their business than the customers. These organizations are the only ones in a position to accurately compare the prices of their products, but they make their money by preventing accurate comparisons. It is easy to make money by simply lying. You can use smoke and mirrors to cause your employees and customers to buy into, and spread, the lie. And your competitors fan the fires of "freedom of choice" by adding lies of their own.

Literally no one knows the cost of a cell phone, a car, or a shirt at WalMart. Every item comes with so many hidden costs, marketing gimmicks and sales incentives that it is impossible for customers, store employees, store managers or company shareholders to know anything at all about the actual cost of goods. Without this information it is impossible to know if the management decisions and product selection are reasonable. This argument does not even address the larger issues of personnel and overhead that play a major role in the profitability and sustainability of the retail environment.

Outsourced Middlemen

Virtually any aspect of modern business can be separated from the core business flow and outsourced to a third party. Economic legerdemain can make this seem like a good idea. The voice taking the order at the drive-up window of a fast food restaurant can come from a call center in Bangalore. A company's management should realize that well-meaning ideas such as this are fundamentally flawed.

The overriding business case is that excessive outsourcing destroys the accountability required to understand the operation of your own business. You no longer have the ability to examine the true costs associated with your operations. With every organization trying to maximize their profit at all costs, each one will lie, cheat and steal to achieve that goal. Now you have given every outsourced operation a reason to misrepresent themselves in an attempt to grow their own business at your expense.

Mob Rule

The reason that there are so many high-profile scandals involving Utilities, Finance, Government and the Press is that the leaders in these organizations form a closed society, feeding each other self-congratulatory stimulus that incentivises more and more egregious behavior. Their actions are completely divorced from outside, objective reality. The legality or morality of their actions are buried under the avalanche of self-importance and the compulsion to take the bit in their teeth and run faster toward the precipice.

The intelligence of a mob tends to be lower than that of any individual. The behavior of these corporate mobs may lead to the ultimate failure of the global economy.

Cascade Failures

The world financial markets are so ill-conceived and mis-designed that they are ripe for a catastrophic cascade failure. Once the dominoes begin to fall there are no mechanisms to prevent total collapse. Propping up failed micro-economies with loans that (objectively) can never be repaid is nothing more that a paperwork band-aid covering a still-festering pustule. Making the balance sheet look good from some arbitrary perspective just delays the inevitable.

If some nation is deemed Worthy, aid should be given by more prosperous economies with no strings. If that nation is able to grow and develop, it will become a useful member of the world community. It is not necessary for the wealthy nations to engage in balance-sheet self aggrandizement while keeping an oppressive boot on the necks of the poor. Economically successful partners represent a truly valuable return on investment that does not show up on any balance sheet.

The clarity that would be generated by simply writing off these "loans" would represent a needed breath of fresh air in all our economic calculations. Draw a line in the sand. Evaluate the reality of the situation as it stands now. Do not continue to rely on dubious promises of payment extracted under duress from failed economies.

An objective look at the world's major economies would make for much more rational decisions. Major aspects of the U.S. economy are not sustainable. Take, for example, the automotive industry. The entire business model requires selling more vehicles, and therefore enticing more people to sign on to the idea. But cars last longer, the population declines, and operating costs

escalate.

I argue that virtually no one today actually wants a car. They want a video game console with leather seats. They would leave off the car part if they could. They do not want to buy it gas. They do not want to insure it. They do not want to sign away their income to get it. They do not want to sit in traffic with it. The only reason that people buy cars, even today, is that they have been convinced that there is no alternative.

Soon the customer base will decide, *en masse*, that they really do not want or need a new car - despite ever-increasing marketing hype.

Without a rational plan, this will likely come as a Big Surprise to many investors. And their reliance on idealistic projections with no basis or objective feedback may prove catastrophic.

The Chinese economy is another case-in-point. Everyone seems to believe that China is set to become the dominant economy in the 21st century. I find this hard to believe. There is so much smoke and mirrors work involved in evaluating their business practices that I think that any reliance on such information is suspect. Objectively, they have stepped up from a third-world economy by purchasing their place at the table of world powers. In order to do so, they are selling their labor and resources for pennies on the dollar. They can do this because they have a large population and geographic area. But what they are doing is not in any way sustainable. It shows no understanding of the shortcomings of 19th century industrialism.

Demand for Chinese goods may sag, whether due to natural disaster, economic turmoil or even a moral refusal to purchase goods manufactured with what amounts to slave labor. Their

reliance on the inflated promises of the financial wizards may well be catastrophic in the face of even a modest slowdown.

Feedback Limits

I have mentioned the loss of feedback that caused the economy to lose its equilibrium. The equilibrium in question was the comfortable *status quo* of the late 20th century. However, there are other, larger, feedback systems that will take over. Even after a catastrophic failure, nature reaches a new balance - possibly in a completely unexpected configuration. The fundamental constraints of nature provide the ultimate fail-safe.

Adapt or Die

No matter how much wishful thinking and pious hand-wringing we engage in we must deal with the objective realities. We cannot legislate success, or vote to repeal natural law. We must act to cushion the impact of the failure of the current economic model, not continue propping up failed policies with ever-more-vehement rhetoric.

Buyer Rules

It is amazing all the clever (doubletalk) financing and (mumble) incentives and (ahem) volume discounts that a supplier will dream up if they are told, in advance, exactly how much the customer will pay. The buyer must simply be realistic and refuse to be swayed by ANY escalation of the real bottom line. Absolutely no contracts for the future – purely pay-as-you-go.

Implementing this philosophy would solve the debt crisis facing our schools. Very simply, we take the amount of current tax revenue and apportion it using exactly the same percentages as last year. Salaries, maintenance, utilities and supplies. All get the same PERCENTAGE of the revenue that they did before. Extremely fair. The electric company no longer dictates prices to the government. Union workers no longer get arbitrary raises. The customer (school district) takes back control of their own operation. If the suppliers (employees and utilities) want the business, they are responsible for making it work. If they do not want the business, there will be other, more efficient suppliers who will be able to step up. No lay-offs or building closures. No fear-mongering or threats. No uncertainty. Just across-the-board adjustment in payments.

What about the employees and suppliers? Won't they bear the burden? No, they do not have to. This is the ultimate trickle-down. They will realize that they, too, can just say "NO!". They do not have to be at the mercy of ever-increasing costs. They can actively regulate the prices that they pay – not simply settle for a choice between bad and worse. This will re-introduce the balance that is missing when inflation is controlled only by arbitrary decisions by an elite few individuals, based on

unverifiable currency market reports.

This would also work for ordinary consumers, IF….

Contracts and Lawyers

The sheer length of all modern contracts is an admission that the seller is pushing a product that the buyer does not want, will not like, or cannot afford. Examples include cell phones and credit cards. There is nothing reasonable or sustainable about these lock-in deals. If the seller had a valuable product and he knew the customer would be satisfied there would be no need for draconian legalese. The seller should always strive to keep the customer happy. A happy customer will be glad to pay. If the deal doesn't work out (for whatever reason) all of these deals should allow the parties to walk away amicably. Maybe the next deal will work out better.

Objectively, a home mortgage is nothing more than a rent-to-own contract in which the landlord (bank) doesn't even have to maintain the plumbing. Just because it is couched in twenty pages of fine print, and it has been marketed since the days of "Leave it to Beaver" as the American way, does not make it a reasonable deal. The fact that the deal is fundamentally unreasonable is the reason that there is so much business for the courts today.

In all these cases, I propose that "the deal" be limited to what can be written on a single sheet of paper. I will do this, if you will do that. If it doesn't work out, we walk away. No recriminations. No credit scores. No lawyers. No unhappiness. Maybe we can make a better deal the next time.

I guarantee that the phone companies would be more responsive if every customer that they lost was a Big Deal, and not just part of business-as-usual churn. And if the customers all left because marketing hype turned out to be lies, maybe the advertising

would be more responsible.

Single page "contracts" that are easily understood would actually reduce the costs of virtually all aspects of doing business. Even though some deals do not work out and represent a cost, it is hard to figure how those costs to a well-run business could possibly exceed the total cost of the legal system as it stands now. There is nothing sustainable about making your own customers the adversary from day one.

I do not believe that it is necessary to "Kill all the Lawyers" as Shakespeare so famously suggested. It should be sufficient to simply let them starve to death.

37 On Inventory Management

Abstract

Managing the availability of goods is key to the sustainable future of civilization. Making better use of materials and resources, eliminating needless duplication, and improving reuse and recycling can improve the lives of individuals, families and communities. This essay examines the flow and availability of goods from the standpoint of inventory management systems. I give examples of current systems, look at an ideal goal, and propose steps that could lead to both immediate and long-term benefits.

> **This essay is a work-in-progress and will be updated periodically. Other related essays concerning image processing and object recognition will be posted as they reach maturity.**

Background

In my youth, my father had a machine shop and lab in the five-car garage area of our house. We parked the cars in the driveway. As I grew up, I spent much time exploring the boxes, bins, cabinets, shelves and assorted containers, learning about the objects within. I learned to use the tools, built projects and conducted experiments. As an inquisitive nine-year-old, I had examined almost every object and attempted to divine its use. I read the Newark and Grainger catalogs while I fell asleep. I had a strong vocabulary and could describe and name most any tool or part.

In particular, I could accurately describe the location of almost any item in the shop. Many items had multiple homes, as duplicates were encouraged and often grouped by project instead of into simple bins. I could clean a work area and put tools, parts and equipment back in their normal places. I could disassemble and reassemble things ranging from toys to lawnmower engines.

I was, in effect, an inventory manager with skills superior to any "professional" system in existence today.

Let us examine the features and requirements of inventory management and suggest techniques that might bring the capabilities in the high-tech world up to the level of a small child.

Requirements

Very simply, we must keep track of objects in time and space.

We must have some general idea of what we mean by an "object", and ways of recognizing and remembering properties. This implies a data entry system with a method of rapidly assigning properties to objects. These properties may be descriptions from catalogs or data sheets, observed properties such as size or color, and arbitrary manually entered information.

Tracking, in its simplest form involves only "Get this object from here and put it there" concepts. Manual forms of data entry, and simple scanners might be sufficient as a first step. An automated system would probably observe an area and recognize objects as they enter and leave.

We should be able to answer questions like:

- "Where is the nearest ...?"
- "How many ... do we have?"
- "How long have we had ...?"
- "Where has ... been stored?"
- "Is ... safe to handle?"
- "Does ... need to be right side up?"
- "What does ... attach to?"
- "How does ... need to be stored?"

- "Do we need to order more … when this is used up?"
- "Is … more expensive than …?"
- "Is there anything special about …? Is it rare or valuable or dangerous or fragile?"
- "What is …? What is it used for?"

The system must be so easy to use that it will be part of everyday life. An assistant that can answer accurately when you wonder where you left the car keys. A retail checkout system that does not need barcodes.

The overall system must be tolerant of bad or conflicting data. Over time everything should be generally self-correcting.

Do not necessarily require a "Parts in Bins" organization. There may be preferred locations so that tools tend to wind up in tool boxes but it should not be carried to extremes.

The database should:

- allow for object identification,
- retain arbitrary properties,
- track current location and location history,
- group objects during storage or use,
- allow for assembly and disassembly of composite objects

Visual Object Recognition

Current barcode scanners beep when they successfully scan an object. As far as I am concerned, a proper scanner will only beep when it sees an object that it does NOT recognize. I.e. we should eliminate the unnecessary confirmation noise. Identification should be so accurate and so routine that the only thing that should need the user's attention is the true exceptions.

> **A forthcoming essay will focus on the requirements of visual object recognition systems**.

There is a range of requirements from the most basic detection of visual features within a background of clutter all the way through the comprehensive integration with a central object-location-tracking database. This is required to ensure accurate identification of a _particular _object, not just the _kind _of object.

Selecting the pencil laying on the notepad in front of you is almost always preferable to selecting an identical pencil from the pencil holder. The history of the object is as important as its location, and, in general, history requires the combined recognition and tracking of multiple visual sensors.

In a world of ubiquitous, distributed visual recognition systems such as foveal cameras, each camera develops a learned history of particular features that compose and are associated with particular objects. The different histories ("experiences") of each camera means that their library of recognition templates will be unique. And yet, we want to be able to assign the same "identity" to objects as they move from one camera's area to the next. This

implies that there should be an "object template description" that is both compact and sufficient to (more or less) uniquely identify a particular class of object. This data is what would normally be communicated with the central object-location database, and with other nearby cameras to aid in tracking particular objects from one station to the next.

Consider: trying to locate a particular individual using the cameras in a shopping mall. Start with a general description such as "short, fat guy in a red suit". This is actually a LOT of information expressed very succinctly. It lops out most of the objects from your recognition database and allows attention to be devoted to the most likely suspects. Maybe a candidate is seen from one point of view and you add to the description: "he has shiny black boots". Motion tracking and adjacency ensures that this is the same individual. You are building a more complete description. Another view: "He has a white beard". Multiple observers watching from different cameras share the ability to casually recognize these high-level features and need ONLY the general location and compact description to be reasonably assured of success.

Modern Examples

The inventory at a WalMart retail store is intended to be in near-constant motion. Trucks with assorted merchandise arrive at the back doors. Products are rapidly distributed to essentially arbitrary locations within the store for presentation to customers. Customers roam the store selecting desired items. Items thus selected are scanned, purchased and removed through the front doors. Approximate item-counts are maintained by using a "delivered minus sold" algorithm, but this becomes so inaccurate over time that periodic physical inventories and complete overall reorganizations of the store are necessary.

If I visit a hardware store I usually expect to be able to find a knowledgeable employee and say something like "I need a bigger one of these", or "This wore out and pieces broke off. Do you have any more?", or "I need to mount this on a brick wall. What do I use to do it?" The employee is expected to be able to recognize my object and its use, match it against items in his experience using arbitrary criteria, and give me a meaningful response within a few seconds.

Typical large companies manage warehouses and stock rooms with bins, shelves, cabinets, etc. and try to ensure that all like objects are collected in one place. This facilitates locating desired items, counting stock, providing an appropriate storage environment and ensuring that replacements are ordered in a timely manner. Frequently in-house part numbers are created and assigned to the storage locations to help with this process. Unfortunately, there is often a many-to-many relationship which allows many different vendors to supply the product that winds up in a particular bin, and the same exact product

may need to be stored in different locations due to convenience or necessity. The computerized inventory system most likely attempts to enforce an idealized "one part number, one location, one quantity" paradigm. More advanced or customized systems tend to be increasingly unwieldy due to special cases and exceptions and the need for more operator training.

My wife appears to have a diametrically opposite approach. I have often observed an apparent equal probability that a particular item will be in any of the boxes, bins, shelves, cabinets, closets, and drawers in the house. This is actually not an outlandish situation if you have a good memory and are able to accurately and rapidly communicate enquiries and responses: "Where's the glue?" "Elmer's is in the bin under the bed. Contact cement is on the top shelf of the linen closet."

On the International Space Station inventory items tend to be stored in bags within bags. In the micro-gravity environment there is generally no need for rigid containers. Objects can be stored compactly in collapsible bags, packed into storage spaces, and gently secured against air currents and slight accelerations using bungee cords. Inventory access problems usually revolve around trying to figure out "which bag?", "where?", and "how do I get to it?". This frequently involves lengthy conversations with the ground controllers who are responsible for trying to record ongoing activity and look up records from past operations.

Homer Simpson has a garage full of lawn and garden equipment. It is all labeled "Property of Ned Flanders". The traditional view of this satire is that Homer is a thief who never returns items that he borrows. I, however, would contend that Ned has simply found a way to store his excess inventory in Homer's garage.

Other Recognition Techniques

In these essays, I focus on visual recognition systems. As a child, I was not so limited. We had recently moved, so much of the material in the garage was packed in various boxes. One day I was going through a box of vacuum tubes – ancient electronic components that are kind of like fist-sized, glass transistors. All of these tubes were wrapped in newspaper packing material. As I picked up one of the wrapped objects, I knew immediately that it was not a tube. It was a bottle containing about a half-pound of mercury. In fact, I knew that it was mercury before I unwrapped it. And I had no prior knowledge that we even had a container of mercury.

This is an example of what I consider a proper an object recognition system in operation. The object was manipulated by a sensitive tactile handling system. The manipulation system safely transitioned from working with glass containers of vacuum to glass containers of mercury. The feedback from the system instantly provided object-density estimates. Movements allowed me to detect that the object was not only NOT solid, but that the contents were fluid. Silence during the manipulation allowed me to deduce that this was not a jar of washers or nuts. The fact that, during rotation, the center of mass did not shift as one would expect with a granular fluid also helped narrow the identification.

The tentative identification became clearer as the bottle was unwrapped. During this short time the entire system changed. Dropping a newspaper-wrapped vacuum tube is a non-event. Dropping a bottle of mercury is a whole different matter. Even in the days before California-inspired paranoia concerning

heavy metals, a child could be concerned about making a Big Mess. My casual attitude became much more focused. My grip became firmer. My posture became more stable. In short, the discovery led me from idle curiosity to attentive excitement in just a few seconds. All without a single visual cue.

Implementation

In light of this background, I contend that inventory management should be an automated, continuous, interactive process. By "interactive" I mean that the inventory management system should physically interact with the items that it is managing, much as I did as a child, or as the hardware store employee does to become good at his job.

This would allow the updating of inventory data to be treated as a routine maintenance operation instead of an inefficient, disruptive quarterly or annual event. Managing objects in boxes (or boxes of objects) is only sufficient if there is a complete prior understanding of the actual, individual objects.

Applications

Although I am describing this as Inventory Management, there are many applications. The Inventory that we are managing need not be simply nuts and bolts. For example:

- Identify people and track their movement
- Production operations in a manufacturing facility. Time and Motion studies.
- Ensuring "Appropriate Redundancy" of tools and supplies. Not too many and not too few.
- Transportation, Cargo and Freight operations.
- Restaurants, Food Services and other Just-In-Time manufacturing
- Produce tracking for food safety
- Infrastructure Maintenance - Buildings and Utilities
- Construction Industry - On-site Manufacturing and assembly
- Health Care and Pharmaceuticals
- Records Management - Customers, Patients, ISO 9000, etc.
- Libraries and Collections

And maybe we turn the whole thing around. Make a geolocation system by mounting the cameras on some of the objects and use

them to watch the surroundings. No more reliance on a fixed infrastructure. Recognition of objects and places are just two sides of the same coin, going into the same database.

Notes

Keep track of items in time and space.

Object identification - data entry, description, photo(s), size, mass, etc.

Object tracking - manual / automated

Object status -

storage in bag, box, etc.; conditions (temperature, etc.)

usage - quantity is partially used (count in / count out) liquids, aerosols, etc.

assembly - object becomes part of something else

disposal / recycling / disassembly - including damaged or incomplete items

movement - new object / new location

Object query -

find nearest

find totals

find expired (drugs, milk, etc.)

find history - objects / locations

Must be so easy to use that it is ALWAYS used for Get and Store operations

Bill of Materials - Object associations or groupings

Nesting - recursive objects within objects

Inventory - continuous update / verification of object info when any storage location is accessed

best if automated

Important to detect unexpected objects

Automated recognition.

Database -

Object identification

Object history

Photographic recognition

Introduction process - controlled observation and examination

Multiple views and lighting

Unique feature extraction

Associate with similar objects

Establish photographic scale and allow scale invariance

Record markings or other identification features

Do not require arbitrary categorization.

Let the recognition engine make its own categories or groups.

Do not expect perfect identification

“Narrowing it down” should be good enough

Combine with location history to complete the identification

Do not necessarily require “Parts in Bins”. Items can be anywhere.

Preferred storage locations may help ensure (toolboxes, etc.) are properly stocked

Locations <–> Parts should be a Many-to-Many relationship

Tolerant of bad / conflicting data. Generally self-correcting.

Examples

Hardware store

Borrow a cup of Sugar

Craigslist

Ned Flanders
Tracking Santa Claus

Maintain orientation. Don’t spill it.

Disassembly -

What is in it. Hazardous?

What is this part of?

Survival inventory. Motors contain coils of wire…

Expiration dates. Use oldest first vs. Use freshest first. Frozen bread anecdote.

Object history and current status. A maid that always moves the dishes from the table to the dishwasher and THEN to the cabinet.

Automated explorer manipulates objects to conduct inventory and cataloging. Dangers include hazardous chemicals, high

voltages, heavy/unstable objects, sharp tools, firearms, fragile objects, falls from high places, rotating machinery, buttons, switches and knobs, insects and pets.

Current systems and limitations

Hospitality industry

Large number of identically furnished rooms

Maid service touches every object daily

Common maintenance, purchasing and disposal operations.

Apartments, Condos and Tract homes

Many redundant objects

Progressively less commonality

Seasonal storage

The maid knows:

Clean the nightstand

Leave the lamp

Wash the dishes

Do not wash the paperback book

No communication regarding object location or availability

You have to ask for it before the system will tell you anything

System should be proactive and push appropriate information to you in advance of need

CraigsList

Arbitrary descriptions are a problem
Locations are hidden until a query is made

38 What is a Source File?

Abstract

Source Files are ubiquitous in the world of software development. Little thought is given to their core technology concept, which is now more than fifty years old. The transition from a desktop-PC to tablet-based computing environment represents a sweeping paradigm shift for software developers. Rethinking the true requirements of software from the Source level onward will allow more modern tools to enter the development arena. This essay presents an analysis of the situation from a historical perspective and proposes new methodologies for use in the future.

Introduction

The concept of the Source File is so fundamental and so ingrained in the lives of software developers that there is seldom any thought given to its true function.

Historically, the program source was simply the human-readable form of input to a language compiler, assemble or translator. The most important aspect of Source Code was that it be able to be fed to the language processor without generating errors. This meant that virtually everything about the process of program creation was geared toward simplifying the task *for the compiler*. Combined with the early data entry methods - Hollerith cards , teletype machines and paper tape - the idea of a program source became established.

And now we find ourselves, fifty or more years later, *expecting* program source to look like it could be printed on a teletype machine. Eighty columns (or so). Fixed pitch fonts. Nothing but ASCII characters.

The scope of programming requirements has changed radically from the early days of computing.

Programming languages and methodologies have evolved in many directions - some useful, others not so much. Our largest projects still suffer from the "simplify it for the compiler" mind-set. A prime example is the ever-present header files used by most modern programming tools. These redundant and hard-to-maintain files are used to provide descriptions of the required linkages in modular programming. Newer programming environments attempt to deal with the header/linkage problem with added features or conventions that allow the tools to handle

much of this drudgery. Handling the header problem is only one of many steps that need to be taken to convert from a "what is easiest for the compiler" to a "what is easiest and least error-prone for the programmer" mentality.

What Are Source Files?

A Source File is a collection of text (variously known a syntactic symbols or tokens) that are either:

- **Human language**, as in comments
- **Directives**, usually for the top-level language processor that uses the particular Source
- **Programming Language**, the statements or syntactic constructs that we think of as the program itself
- **Literals**, as in data which will be manipulated by the program but which is encoded in some fashion and stored within the Source.

Note that String Literals almost always contain some form of data that could be described as Source Code. For example, the classic "Hello, World!" program used in beginning programming classes is a program which manipulates data in another language - in this case, English. Most string manipulation in modern programs exists solely for building elements of another programming language, which will be fed to another language processor.

In addition, many Source Files today contain sections written in many different programming languages. For example, an HTML file might include CSS, JavaScript, HTML and English. Modern text editors attempt (with varying degrees of success) to help sort all this out for the developer. Highlighting, auto-completion and various warnings are used to help prevent the (sometimes

spectacular) errors that result from feeding one language into the processor that is expecting another.

> I will make a brief sidebar to rail against the very poor implementation of what is referred to as "internationalization". If a modern application needs to be relevant to a world-wide audience it is expected that the developers isolate all locale-specific aspects into separate "resource" files, to use localizable operating system calls that can never be evaluated on the developer's system and use character sets and string manipulation tools to support character sets display, and keyboard modes that also cannot be tested on the developer's equipment.
>
> Internationalization should be inherent in the program development process - not scabbed-on to a final product. International test cases should always be visible to the developer and the operation and aesthetics of the product should be visible in a simple and robust manner at all times. To achieve this, the adaptive, multi-lingual keyboards on tablet computers, as well as the world-wide collaboration techniques discussed here become critically important.

Consider the notational nightmare that results from trying to pass a literal CSS property name to a JavaScript function that is being invoked from an HTML event handler. The nesting of escaped quotation marks converts a (relatively) simple concept into something that requires deep understanding by the

programmer. And is thus virtually impossible for independent developers to modify or verify.

Unfortunately, one only gets help from the editor in the static situation such as the hand-written web page. If the program needs to dynamically generate code (such as SQL queries) it becomes much more difficult to test, debug and verify. There are virtually no tools that address this dynamic code creation problem, although it is one of the most common tasks performed today. Many *ad-hoc* build-a-string techniques are used, but never with any consistency or robust certification.

How Are Source Files Used?

Source files seem to be used in four distinct but interrelated ways.

- Design

In the Design phase the source is used as part of the collaboration and documentation for the project. When properly done, this provides a good part of the framework that multiple developers use to ensure accurate understanding of the requirements and to convey details of the implementation as it progresses.

- Edit

The Edit phase is where most human interaction with the source occurs. This is where much current effort in man-machine interaction is focused. Making better editors with greater knowledge of program internals and the possible intentions of the programmer has been the goal since the first Integrated Development Environments (IDEs) were created.

Big, fancy IDEs that need big, fancy displays are moving in the wrong direction. As the computing world moves away from the PC-centric toward the tablet-based future, our interaction will become much more focused. The display-and-keyboard will adapt to the operations that we *really* want to perform, and will retreat from the overwhelming offering of everything we _might _want to do. Adaptive keyboards, touch screens and more compact, high resolution presentations provide opportunities to completely rethink the developer interface.

- Build

Ultimately, the purpose of the program source is still to create a working application program. This means that the Programming Language parts of the Source must be submitted to a language processor. In many cases, this is not a trivial operation. Many separate functions or modules must be combined, their linkages validated and a final product produced. This build process can be time-consuming and can result in errors that are the ultimate responsibility of diverse members of the development team.

Expecting the build process to be performed on individual desktop computers is something that needs to be reevaluated as we move into the 21st century era of cloud storage and cloud computing.

- Test

After a successful build developers generally expect to test the new application. This may require transferring the newly created code to a test environment, which may be a particular hardware device or a software simulator. There should exist a suite of test cases for the "finished" application. Although some things can be automated, in many cases manual interaction and visual aesthetics will be important factors. Handling these in a consistent manner, and ensuring that full testing is actually performed for each build or release candidate is a serious problem in the development process. Maintaining and documenting both known-good and known-bad test cases is of critical importance and can be overwhelming for complex projects.

The source-level run-time debugger is one of the great advances in PC-based computing and is a major feature of all Integrated Development Environments. Unfortunately, the multiple- programming-language nature of modern programs limits the actual usefulness of the Debugger. While I occasionally use a debugger for certain compute-only functions, and neophyte programmers really benefit from the capability, I believe that the problems of client-server interactions, dynamic scripts and cross-platform compatibility are more important challenges that cannot be addressed by a simple debugger. Therefore, the debugger capability should not be viewed as being of critical importance when evaluating future development tools and environments.

I believe that every function or statement should have available an easily-accessible library of test cases. This would be sort-of like using a current debugger to run to a breakpoint and then be able to examine and modify the data structures as they are processed.

Knowing the working-set (data structures that are accessed or modified) used by by any particular function is of critical importance to verifying the correctness of a program. In general, the compilers know this, although sometimes it cannot be determined until run-time. Unfortunately, this critical knowledge is never made available to the developer.

The test-case library would take the place of ancillary test programs used during the development process. Currently, such test programs are created, as-needed, by individual developers. They are never documented, maintained or shared. Even worse, they are often discarded once the developer feels that his function is “working”.

The concept of Assertions in various programming languages are used to catch errors at run-time when selected data does not match expectations. I would extend the Assertion concept by allowing the capture of a function's working-set - at run-time - and saving it as an addition to the test case library. This allows the collection of robust sets of real-world data. When the function is modified or replaced with code that should be equivalent, these collected test cases can provide input to an automated verification process.

What Does the Future Hold?

"Keyboards" should be viewed as "Token Selectors" instead of letter-by-letter token builders. IDEs try to do this in some cases, but fail because of (1) the multiple-language problem, and (2) the "word-ish" nature of tokens needed to convey meaning to people.

A single font size doesn't fit all. Nested structures could be displayed with smaller fonts, so you could literally zoom in to move deeper into the code. Current tree-collapsing displays give an essentially useless all-or-nothing display.

Need more screen space? Get a second iPad. Properly done, this could not be more expensive than multiple monitors and should be much more useful in the general sense.

All source should reside in the cloud. Current Version Control Systems make much ado about "Checking Out" and "Committing" changes to modules. This makes it (intentionally) impossible for multiple developers to share work on individual modules. Storing all development trees in the cloud and changing the checkout process from a "download it to my computer" to a "mark this group of changes as pending" in the cloud concept would be a great improvement.

All Build operations should be performed in the cloud. There should be no need for any compute-intensive operations at a user's desktop. This allows high-powered, dynamically-allocated resources to be brought to bear on what should be an independent background task.

What Could Replace Source Files?

Given the ways current source files are used, the multiple-language problem, the need for collaborative work, and the trend away from desktop computing it seems obvious that we are actually referring to a Database application. We have been using the computer's simple file system to store and access our programs, even though that has never been a particularly appropriate technique.

What is needed is a Cloud-based database storage system which is accessed by multiple Very Thin Clients (read iPad Apps) that provide the user interaction during the software development process. This inherently allows world-wide collaboration among the development team.

The compile and build process is performed as a Cloud-based service which provides additional elements to the Database. These elements would include error and diagnostic information, compiled code, linkage information and target application code. Automated test cases could be run and results verified. All these capabilities would be available to every development team member, without having to have separate instances of hardware, tools, environments, etc.

I have some very specific ideas about how such a specialized software development database would be structured, implemented and accessed. Of course, it has nothing to do with relational databases, SQL, or traditional Client-Server techniques.

I Am Requesting Some Feedback

For now, I am interested in feedback from as diverse a set of developers as I can manage.

Specifically, please let me know how you develop you code.

What types of applications do you develop?

What hardware do you use?

What size screens do you need?

What applications do you need to run? Editors? Compilers? Debuggers? Simulators? Browsers? Documentation tools? Email? Chat?

How many things do you try to do at once? How many things do you absolutely need to do at once?

If you were going to work using a tablet computer, what would you see as advantages?

What would be disadvantages, even given your idea of a perfect tablet-of-the-future?

What do you think would be impossible to do using a tablet?

39 Malevolent Social Engineering in Open Source Software

Abstract

A hypothetical attack on core open source software technologies is presented. Extreme danger lies in the fact that these potentially compromised core technologies may be incorporated into an almost unlimited number of different application programs, unknowingly created and marketed by unrelated organizations which may be completely unable to determine if they are distributing malware. Mitigation strategies are discussed, although none are anticipated to be effective.

Background

Open source software is an increasingly important method of developing modern applications and tools. In many cases the collaborative work of different authors provides for new features and qualified review that would be impractical for any corporate effort. The wide availability, ease of use, and inherent peer-review of open source packages makes them tremendously appealing to virtually all developers.

Unfortunately, it is this very collaborative nature and peer-review that opens the door for social manipulation and creating the illusion of quality and safety while masking malevolent software.

The term malware is usually used to mean intentionally malicious software designed to compromise a target system. From the user's perspective there is little difference between a system that fails due to deliberate machinations and one which fails simply due to buggy software. Accidents and intentional attacks have the same effect. In this analysis I treat both cases equivalently.

Social engineering is a term associated in the public's mind with spreading computer viruses via email. Disguising a threat with some desirable or benign coating (a picture of Martina Navratilova, or a valentine from a secret admirer) causes the user to circumvent the computer security system. A threat that causes a panic reaction can thwart common sense: "Your computer has a VIRUS! Click here to fix it." or "Your account has been suspended due to suspicious activity. Click here to sign in and review your transactions."

Social engineering in the open source software development community can take many forms. Popular tools or packages, a friendly author in a forum, beta testing opportunities, web-based code snippet libraries - all can be the source of code which may fail to receive the scrutiny that it should.

Modern software systems are far too complex for any individual or organization to adequately evaluate, monitor, test or verify. Malevolent code can be incredibly compact - sometimes requiring only a single character.

Examples

> Let me begin by emphasizing that there have been no known *intentional* examples of the use of these attacks to date. The examples describe accidentally introduced bugs in actual software that could be devastating if carefully placed by a knowledgeable adversary.

Single-Bit Date-Time Bug

The firmware for an access control system was found to contain a single-character typographic error which had the effect of rendering day-of-week calculations inaccurate. The error was discovered during testing, only when employees were unable to enter the building on the first Monday of April. Supervisor access was not restricted. Troubleshooting revealed that the system believed that the date was part of a weekend. The erroneous line of code is reproduced below.

```
DB  31,28,31.30,31,30,31,31,30,31,30,31  ; Month Lengths
```

This example is particularly noteworthy because the error (substitution of a period for a comma) is actually only a single-bit error in the ASCII encoding of over 300K bytes of source code. The keys on the keyboard are adjacent, and the difference in visual appearance of the characters is minimal (and was barely discernable on the displays being used).

The error was not discovered during initial development testing because it was dependent on future values of the real-time clock. Code review did not catch the error because reviewers focused

on verifying the “important” things - in this case the sequence of numeric values, and took for granted the punctuation. The assembler software failed to indicate an error because an obscure syntactic convention allowed the overloaded period character to be interpreted as a logical AND operation between the two integer values.

The line between accident and malice can be quite fuzzy. This ambiguity can allow a knowledgeable adversary to obscure an attack by hiding it among hundreds of lines of well-written, clean code. Furthermore, since the malicious nature of any error can easily be explained as human error, the attacker remains free to try again if discovered. The peer-review process may even be commended for finding and correcting the error, while giving the adversary additional information to improve the next attack.

Intel FDIV Bug

The Intel FDIV bug was an error in the floating point division algorithm in certain versions of the Pentium processor. Apparently, the actual underlying error was confined to five cells in a lookup table that were unintentionally left blank.

The effect was that software running on these processors would occasionally receive computational results which were in error after the fifth decimal digit. Subtle errors such as this are extremely difficult to detect - in fact it took a skilled number theorist with great tenacity several months to isolate and demonstrate the problem.

In any case, it would have been far easier to have certified the processor correctness at the design stage by ensuring that the

lookup tables were computed and verified by multiple independent sources prior to production. In fact, by the time the bug was publicized, Intel had already produced processors using the same algorithm which were free of the bug.

Thomas R. Nicely discovered and publicized this bug during 1994, and in December of that year Intel recalled and replaced all affected processors. In his analysis of this situation, he concludes:

> Computations which are mission critical, which might affect someone's life or well being, should be carried out in two entirely different ways, with the results checked against each other. While this still will not guarantee absolute reliability, it would represent a major advance. If two totally different platforms are not available, then as much as possible of the calculations should be done in two or more independent ways. Do not assume that a single computational run of anything is going to give correct results—check your work!

The legacy of this bug is that, fifteen years later, many development tools for Intel-based software still include conditional code relating to the recommended work-around for this hardware error. The mentality that justifies this as "Oh, that's always been there", or "It doesn't do anything, but it can't hurt" is a symptom of a larger problem. Unnecessary code is always harmful. At the very least it allows extra opportunities for undetected corruption to occur. It fosters an un-critical mind set in the reviewer.

Code paths that are never executed can never be tested. Their

presence in modern production code should be considered suspicious. Hiding malware in such untested but ubiquitous code potentially allows for its wide distribution. Dormant code such as this needs only a suitably crafted trigger to affect all compromised systems.

This five-value error caused an economic impact of $500 million to Intel in 1995, and is still being felt in unquantifiable ways today.

> As the Iranian nuclear program found out, to its own detriment, it is never a good idea to run unverified industrial control software on your black-market enrichment centrifuges.
>
> The precision bearings tend to eat themselves for lunch.

What if the table errors in the floating-point algorithm were not as blatant as being left zeroed out? What if the tables contained a carefully selected number of random errors? And were embedded in the floating-point unit of a counterfeit GPS chip? And that chip happened to find itself in the terminal guidance system of an opponent's missile? And that the only effect was to change the rate of successful position updates from 100 per second to one per second? And the CEP (circular error probable) for the missile went from 3 feet to 3000 feet?

This kind of thing could win or lose a war.

And how could one ever expect to detect such a deeply-embedded, subtle attack?

How much would such an attack cost?

Would it be worth it for an adversary to try?

NASA End-Of-Year Protocol

NASA space shuttles use a voting system of three active and two standby computers for flight control operations. Each of these systems is intended to be essentially identical, and each runs identical software. The intention is to detect and mitigate hardware failures, as these are deemed to be the most likely source of problems during a mission lasting two weeks or so.

Even so, flight rules prevent any shuttle from flying on New Year's Day, since it is well recognized that the operating software cannot be positively certified to operate correctly when the year changes. This is especially true when the shuttle orbiter is viewed as a small part of the much larger system involving communications, tracking, navigation, and planning systems which are geographically distributed throughout the world. Ensuring that every component of this worldwide network will be free of anomalies when the year changes is viewed as an insurmountable problem and an unnecessary risk.

McAfee Automatic Update

On April 21, 2010 McAfee Software released an update to its anti-virus software which incorrectly identified legitimate `SVCHOST.EXE` operating system files on Microsoft Windows XP systems as the W32/Wecorl.a virus. Affected systems were locked in an endless reboot sequence and required manual intervention in the form of a local data load by a knowledgeable person to recover.

At least one police department instructed its officers to turn off their patrol car computers to protect them from the McAfee update. It is unclear why every patrol car should have been running anti-virus software in the first place. Much greater security and performance could be gained by closing the department's network and installing proper protection at the gateways.

Anti-virus software is by its very nature a social engineering phenomenon. The threat of malware and the lack of confidence in our legitimate operating systems and software has led us to the perception that we must install software which slows performance and causes unpredictable and non-deterministic behavior under normal circumstances. The fact that perfectly good, working systems can have their behavior altered by anti-virus updates on a daily (or perhaps hourly) basis is, in itself, a source of great concern.

The fact that updates are allowed to proceed in an automated mode may be acceptable or even desirable for consumer products. For dedicated applications or mission-critical systems there is little justification for automatic updates.

Adobe Flash Player

Much controversy attends the question of Adobe Flash player and HTML 5 features on mobile devices such as iPhone. It has been claimed that the Flash player is buggy, a resource hog, and responsible for many system crashes. The Flash player is a proprietary piece of software implementing a proprietary standard. It is difficult to understand why the open source community, principally revolving around the Android operating system, seems to be more vocal in their support of Flash than

Apple, who champions the open source HTML 5 standards.

In reality, the controversy appears to be an example of social engineering, designed to allow a proprietary standard to maintain dominance in an evolving marketplace.

It is true that MacroMedia (now Adobe) filled a real and important need by developing Flash in an era when no standard mechanism for animation or user interaction with computers existed. The time has come for such ad-hoc early forays into user interfaces to yield to more mature, carefully designed systems that incorporate the best features discovered so far and meet the requirements of modern systems.

Proprietary systems will always be more vulnerable than open systems due to the limited resources and unknown business priorities of the controlling company.

Zune 30GB Music Player Leap Year Bug

On December 31, 2008, all Microsoft Zune 30GB Music Players failed during the boot sequence. The software that failed was the Real-Time Clock driver firmware for the Freescale Semiconductor *MC13783 Power Management and Audio* chip. Near the end of the boot process, the driver was called to convert the internal Days and Seconds representation of the current time into Year, Month and Day. On the 366th day of the year, the year-conversion loop would fail to exit, thus causing the device to hang permanently at that point. The work-around was to allow the batteries to run completely down and to wait until the next day to restart the device.

The problematic driver software was contained in the `rtc.c`

source file provided by Freescale Semiconductor to customers of its products. The `ConvertDays()` function was missing an `else break;` statement which would have correctly terminated the loop. Using the normal formatting conventions adopted by Freescale, this would probably have added two lines to the 767 lines in this file.

A second function in this same file, called `MX31GetRealTime()`, uses exactly the same loop structure for year conversion and includes diagnostic message outputs, apparently intended for verifying the calculations. In the day 366 case, this code would output the (incorrect) message "ERROR calculate day", and then break the loop. In other words, if Freescale's own diagnostics had been used to test the code there would have been a single suspicious message among a flurry of output, but the diagnostic code would not have hung. If the real code had been tested or simulated on the correct date, the hang would have been discovered.

Note that the chip in question is called a "*Power Management and Audio*" chip. Page 2 of Freescale's Data Sheet lists 17 features for this chip, including battery chargers, regulators, audio amplifiers, CODECs, multiple audio busses, backlight drivers, USB interface and touchscreen interface. The Real-Time Clock is item 13 of 17 on this list.

It is clear that this is an example of a catastrophic bug in a "trivial" function, buried deep within mountains of code implementing "important" features. This code was provided by a trusted supplier. The features of the chip are so complex (and proprietary) that users (in this case, Microsoft) have little alternative but to accept the supplied code without exhaustive or critical examination.

Sony Root Kit

In 2005, Sony BMG Music released over 100 titles of music CDs that surreptitiously installed rootkit software on user's computers running Microsoft Windows. The alleged purpose of this rootkit was to provide copy protection for the music, but in actuality provided cloaking technology and a back door for malware. Prior to legal action and the eventual recall of all Sony CDs with the XCP technology, over 500,000 computers were compromised.

The corporate mindset at Sony that viewed their own consumers as an enemy, stark terror in the face of declining sales, and a total naivety concerning computer technology left them vulnerable to manipulation by groups selling Digital Rights Management software.

In the case of XCP, it also demonstrated that anti-virus services can be manipulated simply by the choice of names used by the malware. Because it was being distributed by a giant corporation and was covered by the aura of anti-piracy claims, the anti-virus services spent more than one year allowing the infestation to grow. This despite the fact that, in all respects, the software behaved maliciously by (1) being loaded from a music CD, (2) replacing system files, (3) cloaking registry entries and (4) conducting clandestine communications with a BMG host computer.

A Tirade Against Digital Rights Management Software

Digital Rights Management software may be viewed as malware, in that its purpose is to selectively block access to certain data or programs using arbitrary and unexpected rules. Any software that behaves differently on one machine than another, or that works one day and not the next, should be viewed with great suspicion.

DRM software is operationally indistinguishable from malware. Test and verification of DRM software is, by its very nature, difficult for its own developers. In addition, the presence of DRM features on a particular system makes the performance of that system essentially impossible to certify.

Any software that cannot be backed up, restored, and made fully operational at an arbitrary point in the future should not be allowed in a professional development environment. Software that includes timeouts, or that requires contact with a validation server is not reliable. Any software whose continued operation is subject to the corporate whims of third parties is fundamentally unsafe.

Programs that include behaviors that are dependent on hardware identity (station names, MAC addresses or IP addresses), date - time values, random or pseudo-random numbers, and cryptographic codes are inherently difficult to verify. If at all possible, these features, where required, should be carefully isolated from as much of the production code as possible.

Since there can be no universal guarantee of network connectivity or the continued operation of a central server (such as a

licensing server), I would argue that any software that implements "time bomb" behavior or otherwise deliberately ceases to function if it does not receive periodic updates should be banned.

Experience has shown that DRM software is generally ineffective in achieving its stated goal, and causes undue hardship to legitimate users of the product. Development efforts would be much more productive if they were directed toward improving the experience of all users, instead of trying to restrict some users.

Physical Damage to Memory

In the late 1960's the DECsystem 10 used core memory for its primary storage. There existed a memory diagnostic program designed to find errors in this core memory array. The diagnostic proceeded to repeatedly read and write sequential locations. It was found that this diagnostic would almost always find bad locations - even in known good arrays - and that entire rows would be genuinely bad after the diagnostic ran. Investigation proved that the continuous cycling of the three Ampere (!) select current pulses were physically burning out the hair-thin select lines in the array.

The memory design engineers had known of this possibility, but discounted it as a failure mode because the system was equipped with a semiconductor memory cache that would prevent repeated operations on the same address. Naturally, the designer of the memory diagnostic included instructions that explicitly disabled the cache.

Forty years later, our most modern portable devices use high density NAND flash memory as their storage mechanism of

choice. Flash memory relies on the storage of small quantities of electric charge in tiny cells, and the ability to accurately measure that charge. In order to store new values in this type of memory, entire pages must be erased and then sequentially written. The 16GB flash memory used in the iPhone 4 (for example) stores multiple bits in each memory cell using different voltage levels to distinguish values. The ability of these cells to reliably store and distinguish bits begins to degrade after only 3000 page erase cycles. Elaborate hardware and software mechanisms exist to detect and correct errors, and to provide alternate memory pages to replace failed areas. In order to achieve acceptable production and operational yields and longevity, modern error correcting systems are typically capable of correcting 12 or more bit errors in a single block. Furthermore, wear-leveling algorithms attempt to prevent excessive erase/write cycles on individual pages.

Unfortunately, the memory management algorithms both in Samsung's memory controller and in Apple's iOS4 are proprietary. Not only are the specifications of the individual subsystems unknown, but the interactions between the two are cause for concern.

NAND Flash memory suffers from a mode in which repeated reads can indirectly cause adjacent memory cells to change state. These changed cells will trigger the error detection and correction mechanism and be generally harmless. It is unknown whether there is a threshold where a large number of bit errors in a page will cause that page to be moved or rewritten, and possibly even marked as bad. The possibility exists, therefore, that simply reading flash in a pathological manner may result in additional hidden erase/write cycles, or possible additions to the

bad block table.

It is also unknown how bad blocks are reported from the hardware to the operating system, and it is unclear how the file system will respond as the available known-good storage shrinks. Meaningful studies or empirical results are difficult to achieve because of the statistical nature of the underlying failure mode, the number of levels of protection, and the differing implementations of different manufacturers and products.

All systems should collect and make available absolute, quantitative statistics on the performance of these error detection and correction methods. We can have no real confidence in a system if we do not know how close we are to the limits of its capabilities. One thing is certain: "It seems to be working" is a recipe for disaster.

It is not beyond the realm of possibility that suitably malicious software could clandestinely bring virtually every page of the system's flash memory to the brink of ECC failure and then wait for a trigger to push the system over the edge.

This would be an example of software that can physically damage modern hardware, and leave the user with no recourse but to replace the entire device.

Analysis

It would be preferable for the designers of development tools to strive toward the smallest possible set of features for the use of programmers. By concentrating on the most frequently needed operations and making them clear and predictable the review process will be simplified. Obscure or infrequently-used features should be only invoked with great fanfare. Long keywords or elaborate syntactic requirements will draw attention to the fact that this code is not "business as usual" and deserves careful scrutiny.

Vulnerabilities, Exploits and Triggers

Traditionally, malware such as trojans, worms and viruses have relied on some vulnerability in a computer system's design, implementation or operation. Logic errors, unchecked pointers and buffer overflows are examples of vulnerabilities. In general this vulnerability is independent of the exploit, or actual malware, specifically written by an attacker. Once introduced into a vulnerable system, the malware may require an additional trigger event to begin malicious execution. This allows infection of multiple systems to proceed undetected until a particular date, or remote command, causes the nefarious code to spring forth. The trigger will always appear in the form of data within the infected system.

In the present analysis, the distinction between the vulnerability and exploit may appear to be blurred. A sufficiently knowledgeable adversary may subtly introduce the entire body of malicious code into a large number of different application

programs by patiently corrupting core technologies. Using the definitions above, the actual vulnerability is the software design methodology itself, and the exploit could be virtually any piece of commonly used core software.

My primary thesis involves the social engineering that could be used to corrupt otherwise benign and robust software systems. A secondary topic involves the acquired vulnerabilities that have evolved in software development "best practices". This involves using hardware and software features because "that's the way you do it", without any critical reexamination of whether those features actually make any sense in the year 2010, or in the application being developed.

Several of these "evolutionary vulnerabilities" are readily apparent.

1. The use of core open source frameworks by many completely unrelated applications.
2. The programming style that allows and encourages interleaving of distinct objectives within "tight", "efficient" or "multipurpose" functions.
3. The use of needlessly compact source notation without redundancy or cross-checks.
4. The practice of allowing access to every data structure that a function MIGHT need to use without explicitly stating that access to a PARTICULAR structure is desired.
5. Allowing the use of unnecessarily similar variable and function names.
6. Operator overloading.
7. Implied namespaces and namespace obfuscation.
8. Conditional compilation mechanisms
9. The inherent untestability of supporting multiple platforms.

10. Unchecked and unconstrained Pointers.
11. The Stack.
12. Loops that do not look like loops - callbacks and exceptions.
13. Dynamic code creation and execution - interpreters
14. Portable devices that may operate unmonitored for extended intervals.
15. Assuming that individual developers are experts in multiple programming languages.
16. The vulnerability of different programming languages to naive mistakes.
17. The lack of common version control systems among developers.
18. The lack of a global cross-reference checking facility.
19. The lack of inherent range and bounds checking at runtime.
20. The lack of a central revocation authority.
21. Automatic update systems themselves.
22. The lack of a common threat analysis and notification system.
23. The lack of a mechanism to track the installation of application programs in consumer devices.
24. The lack of a mechanism to notify consumers of potential threats.
25. The vulnerability of critical infrastructure to denial-of-service attacks.
26. Trusted Software Developer Certificates that may be easily be circumvented by simply supplying that Trusted developer with malicious tools.

The Stack As An Unnecessary Vulnerability

Since the 1960's the use of a stack-based architecture has been considered a requirement for computer systems. The stack provides a convenient storage area for function parameters, return addresses and local variables. It inherently allows for recursion. It makes exceptions and hardware interrupts easy to implement. It minimizes memory use by sharing a single, dynamic area.

In the world of formal logic, recursion often represents an elegant and compact technique of explaining a complex operation. In the world of computer software it is almost always a serious mistake. There are a few cases in which recursion provides an elegant solution to a problem, but I contend that the risks of allowing universal recursive operations far outweigh the few instances in which any real benefit is derived. Anything that can be done by recursion can be done by iteration, and usually in a much safer and more controlled fashion.

In the absence of recursion, the maximum calling depth can always be computed prior to execution of any given function. In the best case, this could be done with a static calling-tree analysis by the compiler or linker. In the worst case, the program loader must handle calls through dynamic linkages, and the loader must perform the analysis. Knowing the possible calling tree implies that the actual maximum possible memory requirement can also be derived. It thus becomes unnecessary to specify arbitrary stack space allocations. Programs can be treated in a much more deterministic manner.

The fallacy of Mixing Data and Code Addresses - Modern hardware implements a single stack for each executable unit. Programs use machine instructions to load function parameters

and local variables into memory in the allocated stack area. Call and Return operations use a program address placed in the same stack area. This shared allocation is the vulnerability used by most "Arbitrary Code Execution" exploits. It is completely unnecessary for the return address list to share a memory segment with function parameters and local data. If this "conventional wisdom" were to be thoroughly reexamined, virtually all buffer-overrun exploits would be eliminated at the hardware level. Data could still be wildly corrupted, but the flow of program execution would not be accessible to an attacker.

The fallacy of Necessary Recursion - The vast majority of functions in a modern application have clearly defined, static calling trees. These functions have no need for any recursive features, and any recursion indicates a flaw. The fact that modern languages automatically allow and encourage recursion means that recursion is an Error-Not-Caught in almost all cases. It does not seem unreasonable to require that recursion (both direct and indirect) be indicated by some affirmative notation by the programmer.

The fallacy of Saving Memory - The lack of static calling-tree analysis and the assumption of recursion means that arbitrary-sized segments are allocated to the stack. Arbitrary allocations are always erroneous and lead to the mistaken impression that the software is reliable. No one actually knows how close a system is to a stack overflow situation. The presence of unnecessary memory allocation is a waste of resources and leaves a memory area where undetected malware can reside.

The contention that the stack architecture saves memory is one of the elementary explanations of the appeal of the stack. This might be true if the alternative is a naive implementation

in which all function parameters and locals were concurrently allocated from global memory. Calling-tree analysis can be used to allocate parameter frames statically, and yet use only an amount of memory identical to the worst case of the actual calling pattern.

The fallacy of Hardware Interrupts - In order to achieve any degree of security, modern systems always switch stacks when a hardware interrupt is encountered. Thus, it is not necessary that more than a rudimentary allocation be made in the application memory space.

The fallacy of Dynamic Stack Frames - Virtually all modern code computes parameters and pushes them onto the stack prior to a function call. The functions allocate space for local variables by further adjustments to the stack pointer. These dynamically-allocated stack frames are a source of needless, repetitive code that could be eliminated in many cases by static frame allocation and intelligent code optimization. Again, static calling-tree analysis is used to determine the required allocation of these frame areas.

The fallacy of the Memory Dump - It is assumed that memory dumps can be a useful tool to allow crash analysis and code verification. In reality, the use of the stack architecture and its immediate reuse of memory areas for consecutive function calls means that the internal state of any function is destroyed shortly after that function exits. If the stack frames were statically allocated the system would tend to preserve parameters and local variables after the completion of any particular function. The implementations of exception-handling functions (or the dump facility itself) could easily be marked to use frames outside the normal (overlapping) frame area.

The open source development community is an ideal place to implement advanced compiler / linker / loader technology that revises the calling conventions used by modern software. Every application that operates unexpectedly when the calling conventions are changed is an application that was most likely harboring design fallacies that had been unrecognized. Consider this an opportunity to radically improve all open source software with a single paradigm shift.

Hardware and software systems have grown mostly by accretion over the years. The goal has almost universally been expediency: make it run fast and get it done now! Little thought has been given to mitigating common sources of error, except in academic circles.

Much effort goes into testing, primarily to validate the interoperability of various software modules or systems. In general the goal is to ensure that changes made to a new version do not break features of a previously certified application.

In the biological world, organisms develop resistance to antibiotics through exposure. Malware - whether accidental or intentional - will grow and thrive at the boundaries of the test cases. Such malware may spread in a benign form for long periods, only to be triggered into an active form by a possibly innocuous event.

Recommendations

It has been demonstrated that it will be essentially impossible to exclude the accidental or deliberate introduction of malicious behavior into software during its development and maintenance.

Therefore, instead of trying to control humans and their behavior, it would seem reasonable to treat the software itself as the adversary. If every line of code, piece of data and linked module was considered a threat it might be possible to develop high quality threat abatement tools that would have a better chance of success than other approaches.

The open source community is the perfect place to develop such mitigation strategies. Proprietary software development efforts lack the resources, and tend to hide, deny and fail to document vulnerabilities. Open source developers have the opportunity to take both white hat and black hat roles. Adding test cases that succeed or fail in different implementations is a valuable contribution to the robustness of any software. Such continuing development of both code and validation cases should be the norm. Improvement should be continuous and incremental, without the need for monthly "Critical Updates" or other disruptive strategies that are unevenly applied and of questionable effectiveness.

1. Software development methodology
   a. Require the Designer to provide complete natural-language functional specification document for all software systems, modules and functions, as well as example test cases.
   b. Require software to be written exactly to specification by at least two independent development groups, none of which were the Designer of the specification. Preferably this will be

accomplished in different programming languages.

c. Disallow direct communication between independent development groups.

d. Resolve ambiguities and conflicts between implementations by changes to the specification document, incorporated exclusively by the Designer.

e. Require each development group to provide test cases which are not shared with other development groups.

f. Provide each development group's software to a Validation group which is not privy to the specifications. The Validation group runs

i. Stress tests with all known test cases,

ii. Stress test with random inputs,

iii. Stress tests with random structures and data types.

iv. Stress test with all supported operating environments.

v. Expect all results to be identical from each group.

(1) This implies detecting all changes to global memory and confirming that they are allowed and intended.

(2) include range and sanity checks for all returned values.

g. Validation group will record all resource utilization, including speed, memory usage, and external communication.

i. Resource utilization, including external memory and references must be identical.

ii. Every failed validation must be documented and traced to its origin. The nature of the original error must be identified and shared. Repeated problem areas should be studied and mitigation methods developed.

h. One implementation will be chosen for production use, perhaps based on speed, compactness or programming language. The alternative implementations will be available for

validation testing of higher-level modules.

i. New features and future versions will start with changes to the specification by the Designer and will end with comparison of recorded resource utilizations.

i. Any changes in resource utilization from one version to the next, especially global references, must be properly confirmed.

2. Stick with one set of development tools. Do not change the core library that your developers use every time a new release comes out. Validation and version control are needlessly complicated if third-parties can randomly revise any pieces of your software.

3. Use a version control system that captures every piece of software, tool, source file, header file, library, test file, etc. necessary to build and test each release candidate.

a. Build the final release version on an independent system with a clean OS installation using only the files extracted from the version control system.

b. At the very least, when the inevitable disaster strikes it will be possible to identify the versions of your software that are affected.

4. Develop a runtime linkage system capable of swapping out implementations of a particular function or module on the fly.

a. In the verification process, this would allow the verification system to generate random switches between implementations and ensure continued correct operation of the system.

b. In the operational case, normally only one implementation of each function would be distributed. This mechanism would allow for the distribution of software updates into running systems without requiring a reboot in many cases.

> "What I tell you three times is true."
>
> The Hunting of the Snark
> - Lewis Carroll

These suggestions may seem onerous, especially to small developers. This type of approach can easily be implemented using only four individuals: Designer, (2) Developers and a Validator. These roles may be traded for each different module or feature of a project. Far from increasing effort or time-to-market, it could be argued that the improved documentation, cross-training and more robust final product actually reduce overall development effort. New employees can be of immediate use and can be rapidly integrated into the corporate or community structure by assuming any one of the roles without the need for a lengthy training period.

Converting software to another language or porting it to different hardware will be greatly simplified by the comprehensive documentation and test cases inherent in this method. Identifying the ramifications of bugs (detected by whatever means) will be more comprehensive and rapid if the development tools allow easy generation of a list of all software and modules that use a given feature.

Afterword

I have used most of these sidebar boxes to describe disasters caused by reality reaching out and grabbing the unwary. The world is not a safe place. Mortality or extinction may be closer than even the most paranoid realize. It is also true that when people are very good at what they do, and are very careful, it is possible to achieve virtually unimaginable things.

I once visited the McDonald farmhouse at the Trinity Test Site near Alamogordo, New Mexico. This is an uninsulated wooden, pier-and-beam house typical of the early twentieth century. I could look down through the cracks between the floor boards and see the dirt below.

In this house, in 1945, the absolute top scientists of their day, given the essentially unlimited resources of the Manhattan Project, gathered to hand-assemble the world's first atomic bomb.

References

- "What Birds See" by Timothy H. Goldsmith, *Scientific American*, July, 2006.
- "The Joy of Cooking" by Irma S. Rombauer, et.al.
- **www.wikipedia.org** - Always valuable for cross-checking. Valuable links to many of these topics.
- **www.google.com** - Should be high on your list for researching any topic.