



# Subsumption Ethics in Software Development<sup>1</sup>

By: David H. Gleason

Managing Consultant

IT Quality Solutions

[dgleason@ITQual.com](mailto:dgleason@ITQual.com)

[www.ITQual.com](http://www.ITQual.com)

Copyright © 1999

Rev. 3/18/03

As a technology consultant, I carry four information appliances with me during the business day, including a cellular phone, a Palm Pilot, a notebook computer, and this, small, voice recorder. All have been subsumed into my work routine.

A month ago, my notebook computer was stolen out of my car in New York City.

My wife, my son and I were unharmed.

I am going to try and make three points in this talk:

1. Information systems subsume the ethics of decisions
2. Business software development practices are abysmal
3. Tools are available to improve the process.

As for my notebook, I had a recent backup, and my computer had a password in CMOS – the kind that is required to boot up. Such passwords protect well against a casual hacker, but they can be circumvented.

If someone got through my password, they probably found a few things. Over the years, my computer had subsumed:

- Confidential client records and data
- Everything I have written since 1982, including my personal journal
- My financial records for the last 5 years, including my tax returns
- A database of all my contacts, including my notes on client interactions
- Don Gotterbarn's SoDISizer software
- Three years ago I had created a file with all my passwords and PINS, but then I realized that that was a bad idea, so I had deleted it.

My software system was extensive. I had installed about a dozen programs that I use regularly, and had configured the system to attach easily to three client networks, my own

---

<sup>1</sup> This paper was presented at the Fourth Annual Ethics and Technology Conference at Boston College in June of 1999.

network and the phone. It was also set up to synchronize with my desktop computer. I had continually improved its customization over three years – it had subsumed all my ideas about how to make my work more efficient.

Now software is a funny animal, and Microsoft's software is not simple. Windows 2000 has something like 50 million lines of code in it. No one in the world knows what all those lines do. Some of them were written ten years ago. They have been subsumed into an extremely complex system.

I have a love-hate relationship with this stuff. When it works, it facilitates my work. When it doesn't work, life is stressful. I have a feeling I am not alone.

It took me over 40 hours to get my new notebook computer fully operational. I kept installing software which would cause problems with subsystems that had been working previously. In the end, I figured out the order that I had to install everything in order to make it work.

Subsumption makes the installation order important. You must install the network drivers before you install MS Office, because you want Office to recognize your Internet configuration. You must install the Office Service Release before other software, or it won't run. You must uninstall McAfee virus scan before you install Norton Anti-virus, and so on.

I discovered after I had done all this that my network card was incompatible with one of my clients. When I tried to install a new one at the end of the process, it caused all sorts of problems. It seems that the first network card had left some of its configuration behind when I removed it.

That configuration information had been subsumed into the system. It was buried layers and layers deep in the software. The only thing I could really do was clear the disk and start over.

Subsumption ethics is the process by which decisions become incorporated into the operation of information technology (IT) systems, and subsequently forgotten. IT systems, by nature, repeat operations over and over. If those operations have unethical impacts, the system will continue to execute them anyway. Unlike a human operator, there is no point in the cycle where the machine pauses to ask, "Should I do this?"

Subsumption in general is the process of building larger components from smaller ones. Small components are developed and tested, and once they are working reliably they are subsumed into larger systems. This is the enabling technique of object oriented programming.

The larger systems, in turn, are subsumed into still larger systems. Once components, subsystems and applications are operating, the subsumed process becomes invisible and *unavailable* to the user, what James Moor calls the "invisibility factor."

Systems seem like they should be extremely malleable. People tend to think that changes to software should be easy because programming is just a set of instructions, and not like a building made up of hard materials. However, the principle of subsumption makes it clear that

changing base components is like moving building foundations, and can require changes to entire systems. The Y2K problem, for example, is a result of subsumed date processing. There are thousands of layers of subsumption in a typical computer system.

There are four axioms of subsumption ethics:

- A. Information systems subsume design, policy and implementation decisions in programming code and content. Code segments and content become “subsumed objects.” While it is demonstrable that systems are built from subsumed components, it is less easy to show exactly how decisions are subsumed. This axiom posits that the decisions themselves, including many subtle factors, are incorporated into systems operation.
- B. Subsumed objects have determinate moral value. Anecdotally, we can see the moral value of subsumed objects. A windowing system that can only display certain colors, thereby excluding users with certain visual disabilities, has a negative moral value for those users.
- C. Subsumed objects have a high “invisibility factor.” Subsumed objects are invisible to most users. It is not possible, for example, to know all the calculations that mortgage eligibility software might use without seeing the source code. Such software could systematically discriminate without a user’s knowledge.
- D. Subsumptive complexity increases over time. As systems are developed, components become subsumed more and more deeply. For example, purchasers of automated toll booth data subsume it into much larger databases, which might incorporate home ownership, family and estimated income information.

Most developers understand that components become subsumed into larger and larger systems. However, attention to the impact of subsumed objects is frequently overlooked. This paper argues that the impact-oriented issues raised by subsumption ethics can have a significant affect on the success of software projects.

A few years ago, I led a software development project for a large bank. When we took it over the project was already a problem. I walked in to a meeting with the vice president and a dozen others, and outlined my plan for completing the software. I told him it would cost \$70,000.

He stood up, slammed his fists on the table, pointed his finger at me, said, “you have \$70,000 and not a penny more,” and walked out of the room.

I found out later that the project was a year late, \$250,000 over budget, and that the VP’s job was hanging in the balance.

We finished the software as planned, and even made a little money on it.

While I’d like to think that it was my skill, there was only one reason the project succeeded: we had an outstanding programmer, who, in fact, ended up getting hired –subsumed – by the

bank. He paid attention to the fact that everything he did early on would be buried in the code when he got done.

Inattention to subsumption inevitably leads to projects from hell. This is because bad decisions become subsumed, and are extremely difficult to undo later on. One project I worked on failed because we chose Power Builder as the programming platform when we should have used a faster and more robust tool. By the time we realized our mistake, it was too late.

85 percent of software projects in the US run over time or budget. The average software development project overshoots its schedule by half, and some three quarters of all large systems are 'operating failures' that either do not function as intended or are not used at all.

This might not be such a big deal, except that software runs medical equipment, automobiles and airplanes.

There are many methodologies available for software development. Most developers use none of them.

The Software Engineering Institute describes 5 levels of software development skill in its "Capability Maturity Model," from ad-hoc development to optimized. The following chart shows our current status:

	LEVEL	% OF US DEVELOPERS
1	Initial (Ad Hoc)	75%
2	Repeatable (Stable, controlled, project management)	15%
3	Defined (Codified processes rigorously followed)	9%
4	Managed (Process measurement, quality targets)	<1%
5	Optimized (Continuous process improvement)	<1%

Both the bank and Powerbuilder projects were at level 1.

Subsumption ethics applies to the software development process. It also applies to the process by which an organization moves up the CMM. Organizationally, processes and learning are subsumed as the organization improves.

Simon Rogerson and Don Gotterbarn are currently developing another tool, called the "Software Development Impact Statement" (SoDIS). It is a process applied to software project management plans and work breakdown structures (WBS). The process is intended to identify ethically significant issues before software design is underway. In particular, SoDIS seeks early adjudication of issues that would require rework of the product after delivery.

So far this paper has said little about ethics, although ethical issues are implied throughout. It is unethical (not to mention illegal) to steal a computer out of someone's car. I don't even want to think about what someone might do with my stuff. It is unethical to have software projects fail. It is unethical to require users to have 10 years of experience to get their computers working properly.

Fortunately, a code of ethics has recently been adopted by the ACM and IEEE that begins to set parameters. The *Software Engineering Code of Ethics and Professional Practice* provides an ethical framework within which to evaluate the impacts of subsumed objects as they are planned. *Software Development Impact Statement* provides a tool to evaluate subsumed objects as they are developed. The *Capability Maturity Model (CMM)* describes how companies can leverage their processes. Subsumption ethics can be used to further inform such methodologies.

The value of these considerations is impact-aware software development that meets specifications and is delivered on-time and within budget while requiring minimal revision after implementation.

Do I need four information appliances? Probably not. I know I am pushing the technology envelope, but they make my work a lot easier when they're running.

The technology is exciting and enabling, but our software development and reliability record is abysmal. This is true both for the custom database for the 3-person office and the 50 million line Windows 2000. Excellent, reliable software is the exception rather than the rule.

When my car was broken into, I was glad that my family was OK. If someone uses my information for ill, I will deal with it. If someone writes code that crashes my airplane, I will have a problem.