Position paper for NRC workshop on information technology literacy, January 1998, unpaginated.

What Adults Should Know about Information Technology and How they should Learn It

Mary Shaw

Computer Science Department
Carnegie Mellon University

January 1998

Public literacy in information technology (IT) has three important aspects: appreciation of important *fundamental concepts* underlying computing and information, *functional competency* in personal computer use, and *citizen literacy* – critical ability to interpret IT results and to hold informed opinions on public issues. Here I concentrate on the latter two. For these, the ability of the concepts to give pragmatic guidance about dealing with the world is more important than the purity, elegance, or detailed correctness of the concepts. The most important target audience is current adults. Since they have, for the most part, left the school systems, we must find creative ways to reach them.

The natural sciences offer some useful examples. Many people seek to understand some of the deep results such as relativity, string theory, and genetics. Often, they do so out of intellectual fascination – to appreciate the elegance of the theories rather than because of their everyday utility. At the same time, most people rely on simpler, more mundane models for guidance on how things in the world work. These include, for example, some simplified version of Newtonian mechanics and qualitative gas laws. These pragmatic models shape the way people manipulate their immediate environment (how hard to push a child on a swing) and make decisions about public events (how much faith to place in claims about speed and collision forces).

I begin with some observations about the demographics of computer use and about the pragmatic models that people use to deal with natural phenomena. Then I list some of the important concepts in each of the three categories above. I close with suggestions about how IT, and hence the body of relevant IT concepts, is changing and a plea to target literacy efforts at adults as well as at children.

The problem with computers is their success

Over the past decade, the demographics of computing have changed dramatically. Computers are now widespread, nearly ubiquitous. Most personal computers are used by people who neither have, nor want, nor (should) need, extensive special training in computing. Business computers are often in the hands of information users, no longer under exclusive control of a centralized information systems department. Instead of gaining access to IT only through professional intermediaries, vast numbers of people are responsible for their own computing – often with little systematic training or support. This *disintermediation*, or direct connection of users to their IT services, has created new problems for IT system designers.

If all these computers are to be genuinely useful, their owners or handlers must be able to control the IT services effectively. They must understand how to express their problems; they must be able to set up and adapt the computations; and they must have reasonable assurance of the correctness of their results – without actually writing programs. This must be true across a wide range of problems, spanning business solutions, scientific and engineering calculations, and document and image preparation. Furthermore, owners of personal computers must be able to carry out the tasks formerly delegated to system administrators, such as configuration, upgrade, backup, and recovery.

The means for this disintermediation have been the availability of affordable hardware coupled with application-targeted software that produces information and computing in a form that the end user can understand and control. The IT carriers – the "killer apps" – have been spreadsheets, the world-wide web, integrated office suites, and interactive environments such as MUDs. To a lesser degree, Visual Basic has enabled people with minimal programming experience to create useful software that fits their own needs. These applications have become winners in the marketplace because they put a genuinely useful capability in the hands of people with practical problems.

The problem of supplying computing to real people is not yet solved, of course. We are still plagued by bad documentation, incompatible formats, configuration problems, and generally inscrutable systems. I regularly marvel that so many people do succeed in configuring and upgrading their own computer systems. From the anecdotal evidence, it's easy to conclude that everyone who productively owns a personal computer either is a highly-trained IT professional or knows one well enough to ask for favors.

Good-enough models

Most of the general public gets along quite adequately with imperfect models of natural phenomena. Most people can operate appliances without understanding electricity and magnetism, drive cars without understanding internal combustion and gear ratios, and tune radios and TVs without understanding signal propagation and processing. They can, for the most part, avoid the inherent hazards of the devices. Many people can even make minor repairs to these devices with only hazy understanding of the phenomena.

The models that guide people in using these devices are often vastly simplified and sometimes outright wrong. They are

generally simple, though not always internally consistent
often qualitative and inaccurate in detail
rooted in direct experience
often based on a physical analogy
explanatory, predictive, and often operational
generally good enough for practical purposes

Examples of models that are simply wrong but "work" anyhow include

Centrifugal force (Many people believe that an object traveling in a curved path exerts an outward force)
Thermostats (Most people act like they believe either (a) the thermostat is an off/on switch or (b) the lower (higher) you set the thermostat, the faster it gets cool (warm). Few people actually treat it as the means of providing the setpoint for a feedback loop. Nobody freezes (roasts) because of this misunderstanding, though.)
The water-pipe model of electricity (In this model, voltage is analogous to pressure, current is analogous to flow rate, resistance is analogous to the difficulty in overcoming narrow or

obstructed pipes. For example, http://www.repairfaq.org/ uses this analogy to help explain home appliance repair.)

Not all analogy-based explanations provide good models, of course. A recent example in which the analogy doesn't seem to help is the use of an island to explain copyright law at Copyright Bay (http://www.nmjc.cc.nm.us/copyrightbay/). The information at this site may be fine, but the island-harbor-reef analogy doesn't do much to help explain or relate the elements.

While I won't go so far as to suggest that we invent false models, I do think we should be relaxed about the details. It's very important to get lots of people onto the right track and prepared to refine their understanding incrementally. It is much less important to get a few people in command of subtle details. We should seek advice on what types of model work for people and take this into account when we plan what to teach.

Citizen literacy concepts

These are the concepts related to critical evaluation of IT in public settings: evaluating the credibility of claims about IT, understanding the risks and benefits of adopting IT, and so on. IT literacy should imply deep enough understanding to hold an informed opinion on topics such as

Claims that installing internet connections in all public schools will enhance the quality of education
Risks and benefits of electronic commerce
Various proposals for protecting electronic intellectual property
The possibility for U.S.-based regulation to control international electronic business (e.g., gambling)
Risks of aggregating personal information in electronic rather than paper form (e.g., data mining, targeted personal investigation); the possible benefits (e.g., targeted marketing)
The relative merits of technologies and regulations proposed for preventing adult content on the internet from reaching children
Appropriate and inappropriate situations for either explicit anonymity or false-name activity
The merits of the FBI's desire to have digital wiretap capability
The tradeoff between security risk and business advantage in the export (or not) of strong encryption
Credibility of environmental models (the ozone hole, global warming, El Niño, etc)
Unsolicited commercial e-mail, aka spam
A proposal by company management to automate part of the business that was manual until now
A claim by a creditor that "our computer can't do that"

Some of these can be addressed by analogy to the physical world, but in other cases the analogies are misleading – most notably when close human surveillance is absent, when identification of other parties is impractical, and when the scale of time and quantity are radically different from physical scales. Some of the concepts required for this type of IT literacy are

Elementary security: kinds of threats, kinds of countermeasures, differences from physical security
The internet as a communication medium; the world-wide web as a distributed document repository; the paucity of quality control; the transience of much of the information;
Implications of availability and scale (e.g., of personal information); how a few orders of magnitude can make a significant qualitative difference

Dynamic nature of documents, including possibility of continual update, absence of a definitive original, opportunity for manipulation of images
Social norms in an electronic community; nature of risks (e.g., rarely physical) and of enforcement (e.g., largely ineffective other than as peer pressure)

Functional competency concepts

These are the concepts that allow people to set up and control their own computers, computations, communications, and bodies of information. They include both models to explain IT and specific skills pertinent to the care and handling of computers.

Hardware configuration: processor, internal and external storage, modems, "boards"; what the associated numbers mean – people should at least distinguish properties for which bigger is better (e.g., memory sizes, modem speeds) from properties for which smaller is better (e.g., access times)
State, persistence, and transience: Files, and documents and applications as special kinds of files; where to find them (e.g., "I made the memo in Word" doesn't mean that the memo will later be found "in" Word); backup and restoration; caching
Generality: reasoning about all the cases at once
Correctness: analysis, testing, and reasoning about IT
Nature of computation: discrete computation, algorithm as recipe, GIGO
Standard interface conventions: the common subset of Mac/Windows conventions about desktop, windows, customary menus and their entries, cut-paste-drag, etc
Applications as distinct entities with different capabilities (e.g., you can't do photo editing in a spreadsheet); association of file types with applications; integration as smoothing the differences
Compatibility of formats, versions, platforms, devices; version control; managing upgrade paths
Distributed information on the internet: e-mail, netnews and WWW (both as consumer and producer); standard conventions such as FAQs and netiquette; scale (size of online community, relations among file size, bandwidth, contention); page caching; evaluating validity of information found on web; difference between pull and push; personal control over what content is viewed
Security: virus checking, what information to worry about protecting

Fundamental concepts

These are the concepts related to the intellectual foundations of computing, communications, and information. The public should understand these for the same reasons they should understand the foundations of physics or biology. Since I expect most respondents to focus here, I'll just list some topics:

Algorithm, abstraction, information hiding, types/objects/inheritance
Reasoning about correctness – testing vs. simulation vs. analysis vs. verification
Reasoning about performance – complexity analysis vs. simulation vs. measurement
State, persistence
Events, concurrency, synchronization
Discrete vs. continuous phenomena
Scalability
Limits of computation

Setting Priorities

It's easy enough to list things that people "ought" to know. However, curriculum design is at heart a resource-allocation problem, with content (however measured) as the scarce resource. The situation is even worse for adults who have left the educational system: it's harder to get their attention, and you get less of it. In response to a challenge from Al Aho, here are my top 10 choices for the information technology concepts that responsible adults should know. Each concept elaborated with some concrete examples of content that might be included and a rationale; note that the important part is the core concept, not so much the possible specifics. Skills should follow along, providing examples that make the concepts real and memorable. This list does not address the question of how much a high school graduate should know about each concept – I'd have to say, "enough to be useful, but constrained by available curriculum space."

1. Anatomy of computing and information systems
    - Computer at processor – memory – peripheral level
    - Software at the decomposition granularity of file system/operating system / data base / application / suite
    - Networks at the node (client, server, peer) / link level
    - Properties such as capacity, latency, bandwidth
    - Sufficient depth to understand search, privacy, security
    - *Rationale:* Provides basis for obtaining and using computing and information services, including owning a computer.
2. Nature of electronic content as an economic good
    - Intrinsic value of information and structure, and added value of added structure (value of editorial and evaluation contributions)
    - Evaluation of sources and of content
    - *Rationale:* Provides the basis for engaging in electronic commerce and in debate of public issues; also provide basis for personal decisions about creating electronic products
3. Abstraction and generalization
    - Information hiding, problem partitioning, possibility of loosely-coupled decomposition
    - Generality – reasoning about many cases at the same time; finding generalizations about collections of particular examples
    - Separation of concerns
    - Hierarchy and composition
    - *Rationale:* Provides the basis for believing that you can solve problems by finding and combining independent parts.
4. Representation
    - Symbolic computation, representation, syntax, syntactic manipulation
    - Types, objects, naming, addressing, indirection
    - The physical symbol system hypothesis (i.e., that symbolic computation systems provide a necessary and sufficient basis for intelligent action)
    - Role of appropriate representation in problem solving
    - Possibility of representing your information in a way that supports the manipulation you want
    - *Rationale:* Provides the basis for believing that computers are good for a wide range of subtle and complex tasks – especially tasks that are not obviously computational
5. Persistence and transience
    - Mutability of electronic documents
    - Locality, ownership, sole vs. shared access
    - Delete vs. expunge vs. caching vs. re-retrieval
    - *Rationale:* Provides the basis for understanding discrete phenomena, the world-wide web, backups
6. Large-scale information processing

   - Structure of information, relations, types, entities
   - Access and retrieval; formulating queries effectively
   - Searching and browsing
   - Evaluation criteria (is it specificity and coverage, or something like that?)
   - *Rationale:* Provides the basis for finding information of interest in large, heterogeneous, distributed bodies of information
7. Scale
   - Scalability, nonlinearity
   - Limits of computation (very, very basic introduction to computability and decidability – just enough so they know there's an issue – if they can handle the Barber of Seville, they can handle this)
   - Implications of scale, especially in the absence of human-time-scale oversight
   - Reasoning about performance
   - *Rationale:* Provides the basis for understanding limits of computing power, enlightened skepticism, appropriate expectations, and (with Persistence and Information processing) risks of information aggregation
8. Operational procedures
   - Algorithm, protocol, and other precise operational descriptions
   - State, state transition; discrete vs continuous phenomena
   - Reasoning about correctness
   - Interface as the set of operations a component will respond to
   - Compatibility of interfaces; interoperability
   - *Rationale:* Provides the basis for understanding software as a sequence of discrete steps that operate on data
9. Time, sequence, and concurrency (builds on #8, Operational procedures)
   - Sequencing, serial operations
   - Concurrency, synchronization, events
   - Atomicity, deadlock, starvation
   - *Rationale:* Provides the basis for understanding computation and information processing in distributed concurrent systems
10. Modeling (builds on #3, Abstraction and Representation)
    - Model as a simplified description that captures essential detail, validation of model, limiting use of model to questions pertaining to the aspects that are faithfully modeled
    - Simulation and prediction based on models
    - *Rationale:* Provides the basis for understanding the use of models for real systems and the limitations of such use

Future developments

It's always risky to predict the marketplace. One place to look for ideas is the relation between the people who use computing and the computing they use. One trend is toward reducing the degree of hands-on control the individual user maintains. We have already seen substantial disintermediation – that is, increasing numbers of people have direct access to their computers and software. At present, their computing is dominated by individual interactive computations; this allows them to monitor the results as they go. It is more challenging to set up standalone processes that run unmonitored. This requires generalization – description of policy for an open-ended set of computations, not just manipulation of instances. We can see small beginnings of such independent processes in mail filters, automatic check payments, and the demons that select articles from news feeds. What will be required to enable large numbers of users to set up autonomous software agents with larger responsibilities? At what point will large numbers of people trust the internet and electronic commerce mechanisms enough to carry out individual transactions? How about enabling an autonomous software agent to carry out a series of transactions? What about composing your own agent from a number of available components?

A second development trend is toward increasing each user's interaction with other people. Original personal computers were isolated devices serving individuals. Addition of (intermittent) telephone connections supported sporadic communication such as e-mail, newsgroups and classical World-Wide Web use (i.e., pull technology responding to individual user clicks). Permanent network connections, as they become widespread and affordable, will allow push technology to provide information (and entertainment) as it becomes available from its purveyor. Given enough bandwidth, it will also support real time interactions as required for cooperative work, business, and multiparty games. We can expect one of the side effects of this trend to be a fusion between computing and entertainment. This will, of course, require infrastructure development in bandwidth, 3D display, intellectual property protection, and electronic commerce – the usual stuff of software computer science research. Beyond that, though, what new capabilities will the consumer need? What will be required to make entertainment both interactive and multiparty? How can individuals become producers as well as consumers of computer-based entertainment?

A third development trend is away from the full-function general-purpose computer. It currently takes three forms: (a) network computers, essentially thin-to-the-point-of-anorexia clients that allow whoever's responsible for the server to worry about system maintenance; (b) palm-tops, PDAs, and upscale pagers that trade functionality for size and weight; and (c) embedded computers specialized to a particular application such as the dozen or so in your car or the one in your microwave oven. This trend moves away from the need to understand IT; the IT functions are embodied in the product that serves some particular function.

Implementation

The big problem with IT literacy isn't reaching people in classrooms, it's reaching the people who aren't in classrooms and aren't likely to be. This group includes many current decision-makers. However, virtually all the position papers and discussion that addressed delivery at all seemed to assume that IT education would be carried out through traditional schooling – explicit teaching based on a plan for introducing certain ideas at certain ages. A few considered library-based approaches. Some of the discussion mentioned preparation for lifelong learning, but the flavor that adults will attend formal classes. The underlying assumptions of this discussion would lead us to ignore a problem that is currently much more pressing than teaching K-12, namely how to help adults understand IT

We can't afford to wait 30 years for children taught in traditional institutions to turn 40 or 50. The target for IT literacy education must be the public at large, not just school students. This implies that much of the education must be accomplished outside the school system. Classrooms and libraries are "pull" technologies – students come deliberately to the institution, presumably having decided to participate. To reach the adults, we need to figure out how to "push" ideas to the public at large. This will require the expertise of public educators: science museum educators, journalists, writers, game builders, movie producers, and entertainers. Reaching this population will require content organization that is, compared with formal courses, more selective, more meticulous about prerequisite knowledge, finer-grained, situational, and gentle-slope.

Let's also distinguish carefully between teaching and learning. Learning is something a student does. It can take place in the total absence of an instructor. Teaching is creating activities or settings that make it more likely for learning to take place. Still, the strongest correlation between learning and any of the other elements of education is with time on task – the amount of time the student spends attentively engaged in practicing with the content. It follows that finding ways to engage students with learning about IT is crucial. Adult students who seek out courses (the pull case) are often highly motivated. The push case is harder – the delivery has to draw in the target learner. To reach the public at large will require the expertise of, for example, science museum educators, marketing experts, and product designers (the design-for-usability types). Recent publicity campaigns concerning the hazards of undercooked hamburgers and the proper use of ABS brakes come to mind; have they actually been effective?

Rather than asking of each concept "at what age it should first be introduced", it may be fruitful to establish which of the concepts depend on others and hence *in what order* they should be introduced. For example, we might construct a prerequisite tree of concepts. I hope that this will turn out to be shallow and bushy, at least for the concepts of functional competency and citizen literacy. If so, it will ease the learning curve; individuals will be better able to pick up the knowledge they most need, as they need it. This is most critical for adult learners because the curriculum designers have much less control over the order in which an individual will be receptive to topics.

Over the past few years, personal computer products have become easier to install, and their user models have become more consistent (though they have also become more complex through feature accretion). Despite these recent improvements, they are still not simple. It may be productive to enlist the major hardware and software developers (especially application developers) in defining models of IT literacy concurrently with redesign of their own products. That is, perhaps some of the problems of functional competency can be alleviated by design changes in the computer and software systems.