2012

Artificial Agents, Cloud Computing, and Quantum Computing: Applying Floridi's Method of Levels of Abstraction

Marty J. Wolf
Bemidji State University

Frances Grodzinsky
Sacred Heart University, grodzinskyf@sacredheart.edu

Keith W. Miller

Recommended Citation

Grodzinsky, F., Miller, K.W., Wolf, M.J. (2012). Artificial agents, cloud computing, and quantum computing: applying Floridi's method of levels of abstraction. In Hilmi Demir (Ed.), Luciano Floridi's Philosophy of Technology (pp. 23-41). Springer. doi: 10.1007/978-94-007-4292-5_2

Chapter 2
Artificial Agents, Cloud Computing, and Quantum Computing: Applying Floridi's Method of Levels of Abstraction

M.J. Wolf, F.S. Grodzinsky, and K.W. Miller

2.1 Introduction

In his paper "On the Intrinsic Value of Information Objects and the Infosphere," Luciano Floridi asserts that the goal of Information Ethics (IE) "is to fill an 'ethical vacuum' brought to light by the ICT revolution, to paraphrase Moor" (1985). He claims "IE will prove its value only if its applications bear fruit. This is the work that needs to be done in the near future" (Floridi 2002). Our chapter proposes to do part of that work. Initially we focus on Floridi's Method of Levels of Abstraction (LoA). We begin by examining his methodology as it was first developed with J. W. Sanders in "The Method of Abstraction" (Floridi and Sanders 2004) and expanded in "The Method of Levels of Abstraction" (Floridi 2008b). Then we will demonstrate the general applicability and ethical utility of the method of levels of abstraction by considering three different computational paradigms: artificial agents, cloud computing, and quantum computing. In particular, we examine artificial agents as systems that embody the traditional digital computer (modeled as a single Turing machine). This builds on previous work by Floridi and Sanders (2004) and Grodzinsky et al. (2008). New contributions of this chapter include the application

M.J. Wolf (✉)
Department of Mathematics and Computer Science,
Bemidji State University, Bemidji, MN, USA
e-mail: mjwolf@bemidjistate.edu

F.S. Grodzinsky
Department of Computer Science and Information Technology,
Sacred Heart University, Fairfield, CT, USA
e-mail: grodzinskyf@sacredheart.edu

K.W. Miller
Department of Computer Science,
University of Illinois Springfield, Springfield, IL, USA
e-mail: miller.keith@uis.edu

H. Demir (ed.), *Luciano Floridi's Philosophy of Technology: Critical Reflections*,
Philosophy of Engineering and Technology 8, DOI 10.1007/978-94-007-4292-5_2,

of the method of levels of abstraction to the developing paradigm of cloud computing and to the nascent paradigm of quantum computing. In all three paradigms, we emphasize aspects that highlight ethical issues.

Our focus throughout is on the levels of abstraction that are most relevant to computing professionals. What are the consequences of each paradigm, and how should computing professionals approach that paradigm to maximize benefits and minimize risks to the public? What virtues of computing professionals are most relevant to each paradigm? And do these paradigms significantly affect the responsibilities associated with the design, implementation and deployment of computing artifacts? As we consider each – artificial agents, cloud computing and quantum computing – we develop multiple LoAs to form a gradient of abstraction (GoA) for each of the systems under consideration. In our final analysis, we tie together the GoAs, observing their similarities and differences.

2.2 Floridi's Theory

The notion of observables is central to the application of Floridi's Method of Levels of Abstraction (Floridi 2008b). Observables are interpreted, typed-variables. A collection of observables forms a level of abstraction. Different collections of observables give rise to different LoAs. A collection of different LoAs that focus on a particular system or feature forms a gradient of abstraction (GoA).

In each of the systems we will apply Floridi's theory by identifying observables and determining the relationships that hold among the observables. We will identify multiple LoAs and compare and assess the corresponding systems. Our assessment focuses on the relevance of Floridi's theory to computing professionals as they consider questions such as: What are the consequences of the type of system? How can computing professionals approach the system to maximize benefits and minimize risks to the public? What virtues of computing professionals are most relevant to this particular system? How does the system affect the responsibilities associated with the design, implementation and deployment of these computing artifacts?

2.2.1 Levels of Abstraction

For Floridi, a LoA qualifies the level at which a system is considered and informs the discussion of such a system. When we analyze a system, we do so from a particular perspective or level of abstraction. This often results in a model or prototype that identifies the system at the "given LoA". Floridi refers to this as the system-level-model-structure scheme: "Thus, introducing an explicit reference to the LoA makes it clear that the model of a system is a function of the available observables, and that it is reasonable to rank different LoAs and to compare and assess the corresponding models" (Floridi 2008b).

When developers understand the particular LoA under which a system is being built, the discussion of the analysis and design of the system and eventually its

realization can be more productive. Floridi (2008b) asserts that "[t]he definition of observables is only the first step in studying a system at a given LoA. The second step consists in deciding what relationships hold between the observables." He defines this as the concept of system "behaviour." A *behaviour* of a system, at a given LoA, is defined to consist of a predicate whose free variables are observables at that LoA. The substitutions of values for observables that make the predicate true are called the *system behaviours*. A *moderated LoA* is defined as a LoA together with a behaviour at that LoA.

There can be many LoAs applied to the same system; a helpful distinction is that of a Gradient of Abstractions. "A *Gradient of Abstractions* is a formalism defined to facilitate discussion of discrete systems over a range of LoAs. Whilst a single LoA formalizes the scope or granularity of a single model, a GoA provides a way of varying the LoA in order to make observations at differing levels of abstraction" (Floridi 2008b).

To effectively work with LoA's and GoA's, Floridi has created a Method of Abstraction. The steps of the method consist of: (Floridi 2008b)

- First, specifying the LoA means clarifying, from the outset, the range of questions that (a) can be meaningfully asked and (b) are answerable in principle. Knowing at which LoA the system is being analyzed is indispensable, for it means knowing the scope and limits of the model being developed.
- Second, being explicit about the LoA adopted provides a healthy antidote to ambiguities, equivocations and other fallacies or errors due to level-shifting.
- Third, by stating its LoA, a theory is forced to make explicit and clarify its ontological commitment. The ontological commitment of a theory is best understood by distinguishing between a committing and a committed component. A theory commits itself ontologically by opting for a specific LoA. A theory becomes ontologically committed in full through its model, which is therefore the bearer of the specific commitment.

We have seen that a model is the output of the analysis of a system, developed at some LoA(s), for some purpose. So a theory of a system comprises at least three components:

(i) an LoA, which determines the range of available observables and allows the theory to investigate the system under analysis;
(ii) an elaboration of the ensuing model of that system
(iii) the identification of a structure of the system at the given LoA.

2.3 Artificial Agents

In an earlier paper, we used the Method of Abstraction to analyze the ethics of designing artificial agents (Grodzinsky et al. 2008). In that paper we identified two different levels of abstraction, LoA1, which refers to a user's view of what is often called an "autonomous system," and LoA2, which refers to the designer's view of

that same system. We extend those notions in this paper to refer generally to the user's view and to the designer's view of each system under consideration. That is, LoA1 is a set of observables available to a user of a system and LoA2 is a set of observables available to the designer of a system.

In that paper, we focused on LoA2 and described a model of computation whereby artificial agents could exhibit traits that at LoA1 appeared similar to, if not indistinguishable from, human traits we call learning and intentionality. This exploration of the interaction between these two LoAs demonstrated that if the designer failed to consider an expansive enough set of observables at LoA1 to be given consideration at LoA2, the designer might miss certain ethical responsibilities that arise at LoA1. If the designer is focused on low-level observables (LoA2) such as the changing of the value of a variable or the changing of the sequence of operations carried out by the artificial agent, the designer may well get the code for the agent "right." However, the observables properly associated with LoA1 take on new importance when the designer is producing an artificial agent that appears to be learning or demonstrating intentionality. We demonstrated that these sorts of agents are more prone to unpredictable future behaviors and are capable of emergent behaviors not initially programmed by the developer. Thus, we concluded that a designer of artificial agents is under an increased burden of care. That burden requires a thorough examination of observables at LoA1 and their implications. Once those are understood, the designer must consider the GoA, the interface between LoA2 and LoA1 and design the system (an LoA2 endeavor) in such a way as to minimize the risk of undesirable behaviors at LoA1.

In this paper, we are still interested in LoA1 and LoA2, but we also introduce a third Level of Abstraction: LoAS, where the "S" stands for "society." LoAS is the set of observables available to an observer of society. This set of observables consists of those social structures and relationships that are prevalent in the functioning of an information society. At LoAS, observables include a set of variables that describes the characteristics of entities that are or could be affected by a piece of software: descriptive observables concerning individuals, businesses, and governments are all possible members of the set. Questions that might be addressed could be, e.g., If individuals are among the buyers, is there a particular demographic that dominates the buyers? The set of observables at LoAS might be available to a user of a software system. It might be, however, that some observers at LoAS will have access to certain research that is not typically available to a user or even a designer of software systems. It might be that LoAS observables are largely disjoint from the observables typically considered at LoA1 and LoA2.

Our ethical analysis at LoAS focuses on the changes in the users from using the software, and on the changes in others because of the existence of the software in society. Our observations at LoAS are concerned with identifying not only the changes in individuals, but also the cumulative effect of these changes to larger groups and organizations, effects that may be attributable to the software, or to the software combined with other sociotechnical factors (Johnson and Miller 2009).

One particular GoA of interest is the combination of LoA2 and LoAS. The designer might be looking at the demographic in deciding who the users of the

system are and their values (see work on Value Sensitive Design, such as Friedman 1996). For example, in designing e-voting software, the developers had to consider the user interface for able-bodied users, and for those with disabilities due to infirmities and age. In the state of Connecticut in the United States, for example, several interfaces were tested at several sites, to see what potential users actually preferred. The secretary of state contracted with a team of University of Connecticut engineering faculty "to provide advice to the state regarding new voting technology and to assist in the certification and acceptance testing of the AccuVote Optical Scan voting machines…" (UCONN 2010). This team conducted pre-election and post-election audits of the memory cards used in the machines. Once these cards are programmed the integrity of the vote falls upon the precinct polling personnel (LoAS). Misinterpretation of instructions, failure to conduct pre-election tests, inadequate training of precinct personnel all led to problems that were unlikely to have been anticipated at LoA1 or LoA2. Concerned with fair voting practices, Connecticut is using several safeguards to verify the accuracy of the election outcomes. In another example, developers of social networking sites like Facebook and Twitter did not accurately predict the impact of these products on the communication habits of the users when the products were launched.

LoAS can frame earlier work by social psychologists, sociologists, and society and technology scholars. In the early 1990s Chuck Huff developed a social impact statement for software developers based on an idea of Ben Schneiderman's. Huff encouraged software designers to "find out the social impact of the systems they design in time to incorporate changes in those systems as they are built" (1996). Cast in our terms, Huff was encouraging developers to consider LoAS as they manipulated a program at LoA2.

The Embedded Values approach of Friedman and Nissenbaum concerned itself with the ways in which biases emerge in computer systems. These authors examined preexisting biases of the individual or organization, technical biases and emergent biases which arise when "the social contexts in which the system is used is not the one intended by its designers." For example, an ATM that relies heavily on written instructions may be deployed in a neighborhood with an illiterate population (Friedman and Nissenbaum 1996). If designers are aware of biases (at LoA2) that have significant impacts at LoA1 and LoAS, they can use that awareness to design systems that avoid problems. An analysis that incorporates LoAS could be an effective method for managing emergent biases.

In his piece Moral Methodology and Information Technology, Jeroen van den Hoven states, "We need to give computers and software their place in our moral world. We need to look at the effects they have on people, how they constrain and enable us, how they change our experiences, and how they shape our thinking" (2008:50). He asserts that,

> We are now entering a third phase in the development of IT, where the needs of human users, the values of citizens and patients and some of our social questions are considered in their own right and are driving IT, and are no longer seen as mere constraints on the successful implementation of technology (van den Hoven 2008:60).

One theorist who has embraced this concept is Phillip Brey. Brey's Disclosive Ethics reveals embedded values in IT systems (2010). His theory concerns itself with the question: "Is it possible to do an ethical study of computer systems themselves independently of their use by human beings?" (Brey 2010). His answer is basically no. He espouses Disclosive Ethics as a method in which different parties responsible for the design, adoption, use and regulation of computer technology share responsibility for the moral consequences of using it, and in which the technology itself is made part of the equation (Brey 2010:53). The GoA of LoA1, LoA2 and LoAS suggests a formalism that could address Brey's concerns.

We contend that the addition of LoAS to the method of levels of abstraction is consistent with Floridi's desire to formulate "an ethical framework that can treat the Infosphere as a new environment worth the moral attention and care of the human inforgs inhabiting it" (Floridi 2010:19). LoAS consolidates the concerns of those working on embedding values in design and those concerned with the effect of technology on society. It expands Floridi's method beyond the levels of designer and user and includes society in the mix.

A plausible criticism of using LoAS is that LoAS adds nothing to the existing work described above, and merely muddies the water with new (superfluous) terminology. We disagree. Our contention is that the idea of LoAS is a concept that unifies, rather than obscures, the underlying commonality in the existing work of Huff, Friedman, Nissenbaum, van den Hoven, Brey, and others. The similarities in their work derive, at least in part, from the high level of abstraction (as compared to LoA1 and LoA2) at which they work. Their different emphases can be seen as a consequence of their different choices of observables at LoAS.

In addition to providing a framework for better understanding existing work at the sociotechnical level, LoAS also helps integrate work on technology and society at the different levels LoAS, LoA1 and LoA2. When ethical analysis at these three different levels is perceived as being in competition, or at odds with each other, unnecessary conflicts can arise. If work at these different levels is seen as similar analyses, recognizably using the same fundamental concepts, but using different observables, we are convinced that a more effective coherence can be perceived and refined. This theoretical coherence could, and we hope will, lead to practical negotiations and agreements between academics and practitioners who will be better able to understand, together, the important differences and similarities at LoAS, LoA1, and LoA2. In the next sections, we explore how these levels of abstraction can be used in concert to examine carefully the ethical significance of three computing paradigms.

2.4 Artificial Agents and Mapping Table Processing

Floridi and Sanders originally presented a notion of a transition system to describe the internal actions of an agent (2004). In Grodzinsky et al. we developed a more detailed description of the transition system and made important distinctions regarding

the burden of care borne by those who design artificial agents (2008). We include a brief description of it here to give readers a sense of how the concept of theoretical computational machines complements Floridi's notions of LoA, and how technical details of an artificial agent's implementation can have significant impacts for LoAS. Readers are referred to the original work for a more detailed presentation.

Our model closely follows the Turing Machine model of computation and includes a large mapping table with a mechanism for mapping inputs and the current state to a next state and output values. In any practical situation, the mapping table is prohibitively large, though finite. The table is a model for the programming (and therefore the design) of the agent. We explored two variations of the model in which the agent had the ability to modify part of its mapping table. In the first, the agent can modify any part of the table that defines the intelligent agent's behavior during its execution; in other words, the agent can self-modify. In this variation, the agent can add new entries to the table, delete entries from the table and modify entries that exist in the table. Its execution proceeds as in the original case, except when the table fails to contain a valid mapping. In this case, the agent is forced to stop. An agent with a table with this variation (called "fully modifiable") has enough power to render itself useless by introducing changes that force it into a state for which the table contains no mapping. Note that it is also possible for an agent with such a table to add an entry to the table that would duplicate an existing entry except with different outputs or a different next state. A table with multiple identical entries except for the next state seemingly exhibits nondeterminism,[1] since the same input/state would have two different output/state mappings. Although the steps outlined above are deterministic, the choice of which of the two mappings might indeed be arbitrary since the possibility of multiple mappings are not explicitly dealt with in the fundamental behavior of the agent.

In the second variation (called "modifiable"), the mapping table is divided into two parts: in one part, the mappings can be modified; in the other part, the mappings cannot be modified. In other words, some parts of the mapping table are protected from self-modification by the agent. Since the mapping table governs the entire operation of the agent, the designer may wish to prevent the artificial agent from carrying out certain modifications like the one mentioned above. Thus, the designer may opt to protect the entries that govern self-modification from self-modification. While this idea has a certain appeal, especially from the perspective of designing the mapping table in such a way that a modifiable agent always behaves properly, we showed that the modifiable variation can readily promote itself to a fully modifiable machine.

This argument suggests that there is no absolute distinction between a modifiable agent and a fully modifiable agent. However, the two models give different perspectives on the ethical intentions of the designer, even if the designer's intentions may eventually be thwarted.

[1] Note that we are referring to computational nondeterminism. Computational nondeterminism is a theoretical construct that allows a device to be in two or more states simultaneously, with each state experiencing independent sets of inputs and producing independent sets of outputs. This notion is not to be confused with the philosophical notion of nondeterminism.

We can expand on our earlier work, which emphasized LoA1 and LoA2 for artificial agents, by examining LoAS for artificial agents. A natural question at LoAS is about the consequences on society as artificial agents become increasingly common; one possible and ethically significant consequence is that many jobs previously held by people will be done by artificial agents. The use of machines to replace human employees is nothing new, but the sophistication of modern artificial agents may result in people being displaced in jobs that used to be considered immune from automation. At LoAS, we could examine observables such as the number of artificial agents deployed in, for example, care of the elderly or as receptionists; then we could examine the number of humans in jobs in this same area. Next, we could see if people displaced from jobs in these areas found work elsewhere. Finally, we could examine what groups of people had gained from the increased use of artificial agents (perhaps corporations and employers), and what groups of people had lost from that same use (perhaps former employees) (Grodzinsky et al. 2009). Another LoAS consideration might include benefits and risks from the increased use of artificial agents; in health care, for example, an observable might be accidental deaths among the elderly. Perhaps the use of artificial agents to care for the elderly would, overall, reduce such deaths; perhaps not.

Paying attention to LoAS observables during and after artificial agents are designed, developed and deployed should help computer professionals build artificial agents that are more likely to benefit people, and less likely to harm them. At LoA1, the people who directly use a computing artifact are directly in focus; at LoAS, people who are affected by, but do not directly use, an artifact are also in focus.

2.5 Cloud Computing

Cloud computing is a computational paradigm that has been of concern in recent years because of its apparent differences with a more traditional desktop paradigm. In cloud computing, many of the complexities of computing applications and infrastructure are hidden behind abstractions afforded by the Internet. Information system developers already use this level of abstraction to create virtual local area networks and virtual servers which abstract the complexities of the underlying computer network. In cloud computing, we take this a step further and use the cloud to make “an entire data-center’s worth of servers, networking devices, systems management, security, storage and other infrastructure, look like a single computer, or even a single screen” (Fogarty 2009). One concept that has arisen with the advent of cloud computing is “software as service” (SaS). Rather than run software that is present on the computer under the direct local control of the user, SaS requires the user to submit his/her data to the owner of the SaS system; the SaS system carries out the computation using hardware and software that is completely owned by the vendor and then returns the result to the user. Issues of legacy applications, interoperability and accessibility all affect the user of SaS who may or may not be aware of what is happening to his/her data at all times.

The paradigm for computation that most users have experienced since the advent of the personal computer includes the user as owner of the hardware and software that is holding and manipulating the user's data. Typically at this level, the software itself is opaque to users, except for software where the source code is freely available, e.g. free software and open source software. The user is familiar with his or her data and its meaning, and by the locality of the media, controls access to the data – at least to a first approximation. The user decides which software to install on the computer, which programs get access to which data files and how long they get that access.

The cloud computing paradigm brings a different set of access and control features. For example, it is quite possible that the data is no longer stored on hardware owned by the user, but stored "in the cloud." Both Facebook and Google docs are early examples of this kind of service, and now many other providers have entered, or are planning to enter, the market. Another distinction with cloud computing is that the software that manipulates the data is not necessarily present on the same device that is used to access or compute the data. Instead, the user submits data to a software service, the service carries out the computation on its hardware with its software and returns the result to the user. A search executed by a commercial search engine is a common example of this protocol. Only the search query and the search results are ever local to the user; the algorithms and data necessary to carry out the search are owned by the search engine company, and are located on its servers. Google is a prominent example of this kind of "in the cloud" service. To the user, the observables are the same: click on an icon, the program runs and a result is produced. Thus at LoA1, the user may not even be in a position to distinguish SaS from a traditional program. At LoA2, there are significantly more observables in the cloud computing paradigm, visible to a developer but not to a user. Everything from web addresses to the type of compression make a difference at the designer's level of abstraction.

There are also regulatory issues and control issues that impact the user who may or may not have the means of supplying the supporting data if it is stored in the cloud. An analysis of cloud computing at the LoAS level would include issues of trust between cloud providers and customers; issues of control, security and confidentiality, standardization attempts, and consequences of the outcomes. In each of these issues, knowledge of hardware and software are not sufficient; instead, people, institutions and events would have to be taken into account. While these issues do involve technical details at LoA2, they are driven by human values that may be reflected in observables at the LoAS level; thus, forming a GoA is helpful.

Observables at LoAS can be used to gather empirical data useful in making an ethical analysis. For example, it was initially thought that cloud computing could be used to reduce overall energy consumption, but some scholars now dispute that claim (Berl et al. 2010). The data necessary to test claims about energy consumption and the cloud would be available in LoAS observables.

We briefly consider Facebook's role as a cloud-based document storage service provider as an example. Among other things, Facebook stores users' pictures. In the typical flow of operations, the user has complete control over who gets to see the photos and how long the photos remain with Facebook. Facebook has a fiduciary

relationship with the user in which it agrees to show the photos to only those people the user has identified and to delete the photos when the user asks for them to be deleted. Of course, there are no assurances that Facebook complies with these sorts of requests. This is especially true for any backup copies of the photos that Facebook may have made to maintain a high quality of service level.

Facebook's recent difficulties with users unhappy about its policies, illustrates that social forces can influence technical decisions (either proactively or retroactively). Issues of privacy and confidentiality will be played out on the LoAS level as cloud computing becomes an increasingly competitive marketplace. It may be that ethical behavior and good business will coincide when users gravitate to vendors that treat their users with respect. People's trust for cloud computing (LoA1 and LoAS) will be affected by whether cloud computing providers are trustworthy stewards of users' data. Users will have to trust cloud providers in order to be comfortable giving up a large measure of control over their data and processes and should choose vendors wisely. Therefore, as with artificial agents, we contend that computing professionals (at LoA2) should pay careful attention to LoAS observables as part of the development process.

Cloud computing, more specifically SaS, presents a potential ethical impact on the Free and Open Source Software (FOSS) communities. When the Free Software Foundation developed version three of the GNU General Public License (GPLv3) there was controversy surrounding provisions dealing with SaS. As a result of those controversies the provisions addressing SaS were removed from GPLv3 and included in a second, companion license, the Affero General Public License (AGPL). Our analysis of GPLv3 and the AGPL identified a piece of software where the observables at LoA2 were the same, yet the observables at LoAS were very different and had a different social impact (Wolf et al. 2009). Depending on how the software was deployed, the developer was under different legal obligations regarding the release of modified source code. That is, in one scenario, the developer was required by the AGPL either to share the modifications with the community, or in another, seemingly ethically equivalent scenario, the developer was under no legal obligation to share. Our interest here, however, is in showing how analysis of LoAS raises the question of the impact that SaS will have on the sharing ethic that is prevalent in FOSS communities.

2.6 Quantum Computing

When considering quantum information, there are two different aspects that are of importance. One is the notion of quantum computation and the other is the notion of quantum information transfer. As a practical matter, both are currently feasible. A quantum computer that factors an integer has been built. That integer is 15 (Blatt 2005:244). ID Quantique offers a quantum computer that uses quantum principles to generate truly random numbers (rather than pseudo-random numbers common in classical computers) for 1,000–2,500 €. The same company offers a quantum computer embedded in a quantum networking device – a product that implements

secure classical information transfer using both quantum and classical means (see: http://www.idquantique.com).

The quantum network device is an important example, since it uses a combination of quantum techniques and classical (non-quantum) encryption algorithms to transmit secret data. Researchers and others have made claims such as: "Quantum cryptography makes an absolutely safe communication possible for the first time" (Weinfurter 2005:166). The physics of the quantum mechanics ensure that should an eavesdropper "listen in" on the communication, both the sender and the receiver will know that the communication has been intercepted. Yet, European researchers have recently demonstrated that with easily obtainable components they can remotely control a key component of the system and obtain the data in the communication and remain undetected (Lyderson et al. 2010). Clearly, there are ethical problems lurking in the development of, and understanding of, quantum computing.

General quantum computing and quantum teleportation are in the research stage and barring an unexpected breakthrough will not be used or available in a general sense for quite some time. However, much is known about the nature of quantum information and fundamental quantum computation techniques, giving us the opportunity to begin exploration of ethical issues that are emerging along with this model of computation. As in the previous sections we will draw attention to the three different levels of abstraction. However, our main focus will be at LoA2 and how, as in the artificial agent case, quantum developers will carry an increased burden of care. We will find that due to the nature of quantum computation, "quantum developers" seems to include a broader range of people than we normally consider in the development of traditional computing applications.

Next we give an overview of some of the fundamentals of quantum-information processing and transfer. We are especially concerned with two distinctions that make the quantum case different from the classical case: superposition and entanglement. We will then look at three applications of quantum techniques: factoring, searching and cryptography. Once these ideas are presented we will consider the impact these distinctions have on LoA2, and in particular quantum developers. We will conclude this section with an analysis of quantum computation's impact at LoAS. We anticipate that fundamental differences between quantum and classical computation will raise significant ethical issues when users routinely access machines based on quantum computing.

2.6.1 Distinguishing Quantum and Classical Approaches to Computation

Perhaps the most striking difference between classical computation and quantum computation is the way that information is conceived. In classical computation, the smallest piece of information is the bit – either a 0 or a 1 – and it is given a physical realization. Once the bit is given a physical realization, it can be read again and again and it should always yield the same information. Quantum information on the other hand, is stored in a superposition of classical states. That is, to a first

approximation, a single quantum bit (qubit) in a given physical realization is in a probabilistic state where with probability p it is 0 and probability of $1 - p$ it is 1. Quantum computation typically proceeds by repeatedly refining the probabilities of qubits until the probability that they contain the correct answer to a given computation crosses a given threshold, a threshold that is arbitrarily close to 1, but never exactly 1.

Qubits possess another property that distinguishes them from classical bits. Once a qubit is read, the superposition is destroyed and the qubit reverts to being a classical bit. "Read" needs to be broadly interpreted here. It can mean the usual sense of reading a bit. But it can also mean the qubit interacts with any particle or photon that is not part of the intended computation. In such a case, a qubit exhibits the same behavior of having its superposition destroyed.

This destruction of superposition has an important consequence: it is impossible to clone or make a copy of a qubit. The "copy" operation is an important part of classical computation. Yet, it is impossible to copy a qubit (Werner 2005:176). The proof of this statement goes even further. Not only is impossible to copy a qubit, it is impossible to translate all of the quantum information contained in a qubit into classical information. It is impossible to classically describe the state of a qubit in a complete and accurate way. Thus, developers of software that require multiple copies of a particular qubit must anticipate that need, so that multiple qubits can be set up in the same way and have the same computation applied to all of them to prepare them. They can then be used as if they are true copies of each other. However, their quantum nature ensures that they are clearly not.

Closely related to the inability to copy quantum systems, is the inability of a quantum system to carry its identity with it. "We cannot mark a quantum system and then recognize it again" (Esfeld 2005:278). This inability provides challenges to those who develop algorithms for quantum computation systems.

Another important distinction between qubits and classical bits is that qubits can be entangled. More correctly, two quantum systems carrying quantum information can be entangled. That is, they form a single quantum system where all that can be described externally is the state of the relationship between the two quantum systems. The states of the individual subsystems are not knowable without destroying the entanglement. Only the relationship between the two entangled systems is knowable. Thus, in an entangled system, there is no information that is intrinsic to the subsystems therein. When a subsystem in such an entangled quantum system is measured (or read), the entanglement is lost and both the subsystems contained therein lose their superposition and revert to classical information.

We use some standard notation to form an example:

$$(|00\rangle + |11\rangle) / \sqrt{2}$$

describes an entangled quantum system in the following state:

> There are two entangled quantum systems such that should the system decohere, both of the bits will be identical and with equal probability, they will be 0 or 1.

Note that the state implicitly includes the description that under no decoherence scenario will the two subsystems register different bits. Entanglement introduces two additional properties that are important for quantum computation and quantum information transfer systems and challenge usual assumptions about information and computation. The first is that locality of information is no longer required. Once two quantum systems are entangled to form a single quantum system, there is no requirement that they be kept in close physical proximity. Thus, from the notation example, the two subsystems can be separated, one of the two subsystems can then be measured, and without measuring the other, its state can be known with certainty.

Researchers on quantum teleportation systems have recently separated entangled photon pairs 16 km (Jin et al. 2010). These photons were sent through free space, rather than a fiber optics cable. While the current research is obviously experimental, our point is to demonstrate that locality of quantum systems should not be assumed in consideration of ethical concerns.

The final property to consider is that is possible to entangle multiple quantum systems into a single quantum system. Under certain conditions, measurement of a single subsystem can result in either complete decoherence or partial decoherence. Roos et al. entangle three quantum systems in two different ways (2004). Using the notation above, they are:

$$(|000\rangle + |111\rangle) / \sqrt{2}$$

and

$$(|110\rangle + |101\rangle + |011\rangle) / \sqrt{3}.$$

The first entangled system can be described as:

> There are three entangled quantum systems such that if the system decoheres, all of the bits will be identical and with equal probability, they will be 0 or 1.

Note that all of the entanglements and superpositions are lost when any one of the bits is read. The second one is different. Say that the first bit is read, if it is a 1, then the second and third bits retain their coherence in the state $(|01\rangle + |10\rangle) /\sqrt{2}$. If the first bit is a 0, the second and third bits still retain their coherence, but in the state $|11\rangle$. Note that these experiments demonstrate similar behavior when reading any of the three bits in the second entangled system.

2.6.2 Quantum Approaches

Two of the more well-known quantum algorithms are for the factoring problem (given an integer, find its prime factors) and the database searching problem. In addition to their interesting technical attributes, these algorithms demonstrate that practical implementations of "quantum computation" are really a combination of

classical computation and quantum computation. Quantum algorithms typically involve some initial classical computation, preparation of the quantum register where the quantum information is stored, quantum computation, decoherence (reading) of the quantum register, evaluation of the results. Notice that it is trivial to see if a factoring process has yielded the correct answer: multiply the proposed factors and see if the product matches the integer to be factored. If the results are incorrect (say the quantum register reports that 12 and 13 are the prime factors of 143), the process repeats itself. If the results are correct (11 and 13, 11*13 = 143), the results are reported and the computation ends. When quantum algorithms are implemented, the system consists of a classical computer with a quantum subprocessor. The quantum part of the computation requires complete coherence of the quantum subprocessor. During that time, the computation is completely reversible and none of the information in the quantum register is or can be revealed to any system outside of the quantum subprocessor (Blatt 2005:239). It is only when the quantum subprocessor has completed its run that reading can take place.

There is currently no known way to factor integers using a classical computer that works efficiently for large integers, say fifty or more digits. For each additional digit in the integer, the amount of time it takes to determine its factors on a classical computer doubles. This problem is of particular importance in cryptographic systems as their efficacy relies on the assumption that integers cannot be factored efficiently. The factoring problem has been studied by mathematicians and computer scientists for many years using myriad techniques and relatively little progress has been made on producing an efficient algorithm for factoring (or on showing that no such algorithm exists, for that matter).

Perhaps the biggest breakthrough on the problem occurred in 1994 when Shor developed a quantum algorithm for the factoring problem (1994). Shor used a variety of mathematical techniques to transform the factoring problem into a different problem for which quantum computers are well-suited. The quantum computation can actually fail and produce wildly wrong results for the problem, yet Shor's algorithm provably detects these situations and correctly resolves them. With Shor's quantum algorithm, it is possible to factor integers efficiently.

Grover's quantum algorithm for the database searching problem is another potentially significant development (1997). Given an unsorted database of items with unique keys and a search key, Grover's algorithm retrieves the item from the database whose key matches the search key. During this algorithm's quantum computation, it too reaches a correct answer, but again, probabilistically. The algorithm accommodates the possibility of an incorrect response and easily verifies any result that it gives is indeed correct.

As suggested earlier, both of these algorithms are of little practical use today outside of experimental research set ups. For the algorithms to be useful in a general sense, we need the physical realization of quantum registers consisting of more than a few bits. However, quantum registers with just a few bits are quite useful in the world today as part of a secure information transfer systems. Secure information transfer today relies on cryptography: any message to be sent from Alice to Bob is encrypted by Alice, transmitted, and then decrypted by Bob. In order for this process

to work, Bob needs to have the decryption key. Thus, Alice needs to send a key to Bob via a secure medium. Obviously, the key cannot be encrypted, otherwise Alice would need to send Bob a key to decrypt the encrypted key.

Quantum cryptography, or more properly, Quantum Key Distribution, solves this problem in a secure way. Using superposition of photons, Alice transmits the key to Bob. Through unsecured communication Bob and Alice agree on a key based on the photons Alice sent. Quantum properties of the photons ensure that if the photons are intercepted, both Alice and Bob know that the key has been compromised. Once a key is agreed upon, Alice uses the key to encrypt the data and then transmits the encrypted data to Bob. Bob can then decrypt the data. The actual transmission is secure. But, as we will note in the next section, this does not mean that it is impossible for an eavesdropper to intercept the message without being detected.

Quantum approaches to computation and information transmission can take advantage of both superposition and entanglement. Superposition and entanglement are the resources that leads to new possibilities for data transmission and the speed-ups found in quantum algorithms (Werner 2005:183).

2.6.3 Ethical Concerns

During its research stage, quantum computing has already begun to bring to light some ethical concerns. If quantum computing becomes a common, practical technology, we expect significant ethical issues will arise. Although there are important practical speed and efficiency advantages to quantum computing, qubits by their very nature do not register information in the same way that conventional digital memories do, thus challenging some of the most fundamental assumptions of Information Ethics. If quantum computing is to become practical for most users (many researchers believe that it will eventually), it seems likely that the probabilistic nature of quantum memory and computation will be hidden from users. Further, users will likely not even know when the quantum subprocessor has been used to determine a result. Seen another way, when most users enter input, they are not going to include a probability threshold to be used to determine whether an output is correct. Users will want to assume confidently that the output is correct; it will be left to those developing and implementing quantum algorithms to determine the level to set the threshold for correctness. Only someone with an LoA that includes at least some knowledge about the intricacies of quantum computing can make an informed ethical choice about picking the threshold (LoA2). There is power and responsibility in that choice. The autonomy of users at LoA1 is impinged upon when they receive output without knowing about the inherently probabilistic nature of quantum computation.

For some algorithms (such as factoring, described above), a conventional program can easily and efficiently check to see if the quantum algorithm has delivered a correct answer. However, many useful applications of quantum computing in which such “post quantum checking” will not be practical. These applications

include functional versions of NP-complete problems such as the Traveling Salesman Problem. In these cases, the setting of a threshold will carry important ethical weight.

Another concern at LoA2 has to do with claims made by those developing quantum-based devices. Statements such as, "Quantum cryptography makes an absolutely safe communication possible for the first time" (Weinfurter 2005) have the potential to be misleading. While it is true that the laws of quantum physics ensure that the communication of the quantum key cannot be eavesdropped, "absolutely safe communication" has at least two additional requirements. First, the implementation of the system must be done correctly. As Lyderson's team has shown, currently available quantum cryptography has an exploitable implementation flaw (2010). With these systems, it is possible for an eavesdropper to take control of detectors in two different commercial quantum cryptography systems. Second, absolutely safe communication requires the nature of computation to remain unchanged. We argue this point more thoroughly later in this section.

In addition to the concerns about the user's ability to retain autonomy at LoA1, we have concerns about the impact that quantum computation will have on LoAS. Right now, quantum computation is in its infancy, just as classical computation is maturing. An attribute of that maturation process is the relatively stable role computers play in the lives of many people – at home, work and school. People have begun to expect, perhaps unconsciously, that the computer behaves in a particular way. Part of that expectation is based on the way that software is developed. That is, the enterprise of developing software has matured to a point where there are well known guidelines and techniques that help ensure better-developed software. There is much controversy about which techniques are the best and about whether sufficient reliability is typically achieved. However, many scholars agree that significantly effective techniques are available, even if many developers choose to ignore them (For example, see Parnas 2009). Regardless of the particular software engineering technique used, debugging software is part of the software development process.

Quantum computation is fundamentally different from traditional computing in at least two ways: the impossibility of making copies of quantum objects, and the inability to inspect quantum objects part way through a computation. Because of these two important differences, debugging quantum software will be profoundly affected. This calls for special ethical care for those implementing quantum software.

A commonly used debugging techniques is one in which software is stopped mid-execution and the values stored in registers are inspected and, if need be, changed, to better understand the reason the software is not producing the expected results. This technique cannot be used to debug quantum software because any inspection of a quantum register causes it to decohere. A simple variant of this technique that entails making a copy of the quantum register is similarly foiled due to the physical impossibility of making copies of quantum objects. The aforementioned possibility of using multiple quantum registers that are prepared in the same way may prove to be effective. However, the probabilistic nature of quantum computation

should serve as a constant reminder to the debugger of quantum software that no two of the quantum registers can be assumed to be identical as is so often done (and rightly so) with the information stored in classical registers.

The message here is clear. In order to avoid the sorts of ethical concerns that arise from the release of poorly designed and tested classical software, we need to develop methods of designing, implementing, testing and debugging that are appropriate for quantum software. The old ways are insufficient to meet high ethical standards. In a rush to exploit the considerable advantages of quantum computing, it is vital that computing professionals insist on sufficiently mature quality control of quantum applications before they are deployed in situations that might be dangerous for the public.

Looking at quantum computing from LoAS, it seems reasonable to assume that once quantum computers are widely available, the existing classical computers will not all be replaced at the same moment. It is during this transition time that the development of quantum computers has the potential to have a profound impact on all of society. For example, almost all security on the Internet today is provided by some cryptographic means. Today's cryptographic methods rely on the assumption that factoring is computationally difficult (the amount of time it takes to factor an integer doubles with the addition of each digit). As Shor's algorithm demonstrates, a quantum computer can efficiently factor large integers. Anyone who possesses a quantum computer with a sufficiently large quantum register will have the means to break any cryptographic code that relies on today's most popular cryptographic techniques. Thus, even though two parties may engage in a quantum key exchange, an eavesdropper who has access to a quantum computer and intercepts an encrypted message will be in a position to decrypt that message quickly without the having knowledge of the decryption key.

While it may be possible that new secure communication techniques will be developed for communication between parties who possess quantum computers, we are concerned about a "have's" and "have not's" situation developing. In a communication scenario where quantum computers are available to some but not to others, those with sufficient funding to obtain a quantum computer will be able to ensure their secure communication, while the rest will not. Not only will these entities be in a position to ensure their secure communication, they will also have the computational ability to decode any communication they intercept. This clearly creates a power imbalance. Furthermore, this threatens to disrupt much of the e-commerce system currently in place.

Thus, should large quantum computers be developed, the first to develop them will be in a position of power over those who do not. Since it is impossible to determine the virtues of the first developer *a priori*, a prudent practical and ethical argument can be made for the development of systems that will ensure secure private communication between parties that do not possess quantum computers, when there are those in the world who do possess them.

In addition to the ethical challenges inherent in quantum computing, we also contend that quantum computing presents a fundamental theoretical challenge to

Floridi's Information Ethics. Floridi and Sanders discuss the nature of an act by an actor *a* to a patient *p*:

> Evil action = one or more negative messages, initiated by *a*, that brings about a transformation of states that (can) damage *p*'s welfare severely and unnecessarily; or more briefly, any patient unfriendly message (Floridi and Sanders 2001:57).

It is important for our purposes at LoAS to note that the patient *p* in Floridi and Sanders' formulation may be human, biological but not human, or artificial.

We contend that this definition of evil means that the probabilistic nature of quantum computing may be considered fundamentally evil, or at least not entirely commendable. Quantum computing introduces an inherent uncertainty. Such uncertainty can sometimes be managed (as in Shor's quantum factoring algorithm), but that does not remove the objection that quantum computing is, at its core, less certain that traditional computing. If less certain, then it can be argued, it is less good in Information Ethics (Floridi 2005).

2.7 Conclusions

In this chapter we have approached three cases using Floridi's Method of Levels of Abstraction. It is clear to us that this method offers a usable framework in the analysis and development of software applications. The addition of LoAS provides us with an added dimension which addresses the direct and indirect effects of software on society. The three levels that we have chosen to define, LoA1, LoA2 and LoAS, are clearly applicable to Artificial Agents and the emerging paradigm of Cloud Computing. The use of the method with Quantum Computing demonstrates its effectiveness even with nascent notions of computing. The challenge for Quantum Computing developers is to find a way to address the ethical concerns that the intrinsic nature of quantum computing presents at all three levels of abstraction, and the challenge to IE theorists is to address how or if quantum applications fit into their conception of the Infosphere and IE. We have begun part of that work; there is much more to be done.

References

Berl, A., E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Dang, and K. Pentikousis. 2010. Energy-efficient cloud computing. *The Computer Journal* 53(7): 1045–1051.

Blatt, R. 2005. Quantum information processing: Dream and realization. In *Entangled world: The fascination of quantum information and computation*, ed. J. Audretsch, 235–270. Weinheim: Wiley-VCH.

Brey, P. 2010. Values in technology and disclosive computer ethics. In *The Cambridge handbook of information and computer ethics*, ed. L. Floridi, 41–58. Cambridge: Cambridge University Press.

Esfeld, M. 2005. Quantum theory: A challenge for philosophy! In *Entangled world: The fascination of quantum information and computation*, ed. J. Audretsch, 271–296. Weinheim: Wiley-VCH.

Floridi, L. 2002. On the intrinsic value of information objects and the infosphere. *Ethics and Information Technology* 4(4): 287–304. doi:10.1023/A:1021342422699.

Floridi, L. 2005. Information ethics, its nature and scope. *Computers and Society* 35(2): 3. June 2005.

Floridi, L. 2008a. Foundations of information ethics. In *The handbook of information and computer ethics*, ed. K. Himma and H. Tavani, 3–23. Hoboken: Wiley.

Floridi, L. 2008b. The method of levels of abstraction. *Minds and Machines* 18: 303–329. doi:10.0007/s11023-008-9113-7.

Floridi, L. 2010. Ethics after the information revolution. In *The Cambridge handbook of information and computer ethics*, ed. L. Floridi, 3–19. Cambridge: Cambridge University Press.

Floridi, L., and J.W. Sanders. 2001. Artificial evil and the foundation of computer ethics. *Ethics and Information Technology* 3: 55–66.

Floridi, L., and J.W. Sanders. 2004. On the morality of artificial agents. *Minds and Machines* 14(3): 349–379.

Fogarty, K. 2009. Cloud computing definitions and solutions. http://www.cio.com/article/501814/Cloud_Computing_Definitions_and_Solutions?page=1&taxonomyId=3024. Accessed June, 2010.

Friedman, B. 1996. Value-sensitive design. *Interactions* 3(6): 16–23.

Friedman, B., and H. Nissenbaum. 1996. Bias in computer systems. *ACM Transactions on Computer Systems* 14(3): 335.

Grodzinsky, F.S., K.W. Miller, and M.J. Wolf. 2008. The ethics of designing artificial agents. *Journal of Ethics and Information Technology* 10(2–3): 115–121. doi:10.1007/s10676-008-9163-9.

Grodzinsky, F.S., K.W. Miller, and M.J. Wolf. 2009. *Why turing shouldn't have to guess*. Asia-Pacific Computing and Philosophy Conference, Tokyo, October 1–2, 2009.

Grover, L. 1997. Quantum mechanics helps in searching for a needle in a haystack. *Phys Rev Let* 79(2): 325–328.

Huff, Chuck. 1996. About social impact statements. http://www.stolaf.edu/people/huff/prose/SIS.html. Accessed September, 2010.

Jin, X., J. Ren, B. Yang, Z. Yi, F. Zhou, X. Xu, S. Wang, D. Yang, Y. Hu, S. Jiang, T. Yang, H. Yin, K. Chen, C. Peng, and J. Pan. 2010. Experimental free-space quantum teleportation. *Nature Photonics* 4: 376–381. doi:10.1038/nphoton.2010.87.

Johnson, D., and K. Miller. 2009. *Computer ethics: Analyzing information technology*, 4th ed. Upper Saddle River: Prentice-Hall.

Lyderson, L., C. Wiechers, C. Wittmann, D. Elser, J. Skaar, and V. Makarov. 2010. Hacking commercial quantum cryptography systems by tailored bright illumination. *Nature Photonics* 4: 686–689. doi:10.1038/nphoton.2010.214.

Parnas, D.L. 2009. Document based rational software development. *Know-Based Syst* 22(3): 132–141.

Roos, C.F., M. Riebe, H. Häffner, W. Hänsel, J. Benhelm, G. Lancaster, C. Becher, F. Schmidt-Kaler, and R. Blatt. 2004. Control and measurement of three-qubit entangled states. *Science* 304(5676): 1478–1480. doi:10.1126/science.1097522.

Shor, P. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th annual symposium on foundations of computer science*, ed. S. Goldwasser, 124–134. Los Alamitos: IEEE Computer Society Press.

University of Connecticut (UCONN). 2010. http://www.engr.uconn.edu/votercentertechnology.php. Accessed October 25, 2010.

van den Jeroen, Hoven. 2008. Moral methodology and information technology. In *The handbook of information and computer ethics*, ed. K. Himma and H. Tavani, 49–67. Hoboken: Wiley.

Weinfurter, H. 2005. Quantum information. In *Entangled world: The fascination of quantum information and computation*, ed. J. Audretsch, 143–168. Weinheim: Wiley-VCH.

Werner, R.F. 2005. Quantum computers – The new generation of supercomputers? In *Entangled world: The fascination of quantum information and computation*, ed. J. Audretsch, 169–201. Weinheim: Wiley-VCH.

Wolf, M.J., K. Miller, and F.S. Grodzinsky. 2009. On the meaning of free software. *Ethics and Information Technology* 11(4): 279–286. doi:10.1007/s10676-009-9207-9.