What Should Be Automated?

The Fundamental Question Underlying Human-Centered Computing

Matti Tedre

University of Joensuu, Department of Computer Science and Statistics
P.O.Box 111, FIN-80110, Finland
+358 50 362 3922

matti.tedre@cs.joensuu.fi

Abstract

In 1989 the ACM task force on the Core of Computer Science argued that "*What can be (effectively) automated?*" is "the fundamental question underlying all of computing". The task force's view of computing was a machine-oriented one; the task force recognized the theoretical, empirical, and design-oriented aspects of computer science. The question "*What can be effectively automated?*" indeed draws some fundamental limits of automatic computation.

However, since the 1980s there has been an ongoing shift away from the machine-centered view of computing, towards a human-centered view of computing. In this paper I argue that human-centered computing necessitates a perspective shift in computer science. I note that the central question of machine-centered computing fails to recognize the driving issues of human-centered computing. I argue that in all branches of human-centered computing there is another fundamental question that should be asked: "*What should be automated?*"

Categories and Subject Descriptors

K.4.2 [**Computers and Society**]: Social Issues; K.m [**Miscellaneous**]

General Terms: Design, Human Factors

Keywords

Human-centered computing; fundamental questions; ethical questions; normative questions

1.   Introduction

One of the most influential figures in the early development of computer science, George Forsythe, argued in 1969 that "*The question 'What can be automated?' is one of the most inspiring philosophical and practical questions of contemporary civilization*" [7]. That question was adopted as the name of a report of the NSF Computer Science and Engineering Research

Study (COSERS) in 1980 [1]. The question "*What can be automated?*" emphasizes the very foundations of computation – it asks, in a very general form, what can be, in principle, automated with any kind of machinery.

In 1989 the *ACM Task Force on the Core of Computer Science*, headed by Peter Denning, published their oft-quoted report *Computing as a Discipline* [5]. They defined the discipline of computing as "*the systematic study of algorithmic processes that describe and transform information*". Denning's task force noted that the theory, analysis, design, efficiency, implementation, and application of algorithmic processes belong to the field of computing, and they posed what they called the fundamental question underlying all of computing; "*What can be (effectively) automated?*"

Computer scientists agree, at least in principle, on the task force's definition of computing as a discipline as well as its "fundamental question underlying all of computing". Although the discipline of computing has diversified greatly since 1989 (see, e.g., [4]), the fundamental question has rarely been questioned (with the exception of, e.g., [18] which is an analysis of Denning et al.'s fundamental question, taking into account issues of effectiveness, reliability, and ethics). In this article I take a look at the constituents of the "fundamental question underlying all of computing", sketch the trends that brought about human-centered computing, and argue that those trends necessitate a review of Denning et al.'s fundamental question.

2.   What Can Be Automated and How?

In the January-February 1985 issue of *American Scientist*, Peter Denning defined computer science as "*the body of knowledge dealing with the design, analysis, implementation, efficiency, and application of processes that transform information*" [3]. By noting that computer science does not deal with calculation only, Denning arrived at what he called the fundamental question underlying all of computer science, "*What can be automated?*"

In addition to his one fundamental question underlying all of computer science Denning listed eleven topic areas of computer science and outlined a number of fundamental questions asked in each topic area. Denning had 50 questions altogether. However, in 19 out of the 50 fundamental questions that Denning mentioned, the question deals with *how* instead of *what*. Denning's 50 fundamental questions include questions such as "*How can large databases be protected from inconsistencies generated by simultaneous access [...]?*", "*How can the fact that a system is made of components be hidden from users who do not wish to see that level of detail?*", and "*What basic models of intelligence are there and how do we build machines that simulate*

them?" [3]. Later Denning et al. [5] changed the question to "*What can be (effectively) automated?*"

The theoretical question "*What can be automated?*" is the central question in computability theory, which studies what can be, in principle, computed with any kind of machinery. The theoretical question "*What can be effectively automated?*" is one of the central questions in computational complexity theory, which studies the amount of resources, such as time and storage, it takes to solve different kinds of computational problems. Both of those two theoretical questions are fundamental to empirical research on computation and computers, and crucial to the design and implementation of computing systems. However, it has been argued that since the 1980s the focus in computing research has been gradually shifting away from the machine and automation towards the uses of computers, the context of computer use, the activities of end users, and the group interactions of end users [8, 19]. That is, it has been argued that there has been a clear shift from *machine-centered computing* towards *human-centered computing* [19]. Consequently, the attention devoted to the social implications of computing has continued to increase.

2.1   Theoretician's and Practitioner's Questions

Answers to questions that ask *what* are different from answers to questions that ask *how*. Generally speaking, natural scientists tend to ask *what* and *why*, and engineers tend to ask *how*. The *theoretician's questions* "*What can be automated?*" and "*What can be effectively automated?*" divide all possible processes into those processes that can be (effectively) automated (according to some criterion of *effectiveness*) and those processes that cannot be. The theoretician should not be concerned about the specific technologies that might be used to automate processes, be the technologies electronics, optics, pneumatics, or magic [6]. Although the question of computability is indeed one of the central questions in computer science, the rest of this article predominantly focuses on what can be *effectively* automated.

For the theoretician's questions there can be straightforward yes/no answers. Conversely, there usually are many correct answers to the *practitioner's question* "*How can process* p *be automated?*" For instance, implementations *a*, *b*, and *c* can all automate process *p* and be optimal in their use of space, time, and other resources, or they can all be non-optimal in different ways. Because there is not always a single optimal way to automate process *p*, one of the central problems for a practitioner becomes, "*How does one choose between a number of different implementations that all automate process* p *but that all have non-optimal aspects?*"

There is no generic solution for the practitioner's (meta)problem above. If there is no optimal implementation for automating process *p*, the choice of implementation depends, among other things, on the practitioner's preferences, experience, and proficiency. Note, however, that although notions of significance are ontologically subjective matters, they can still be epistemologically objective in the sense that all people in a given community can feel the same way about the significances of aspects of phenomena. Naturally, the consensus may be weaker or stronger, depending on the issue. For instance, there is often a trade-off between the accuracy of the result of a computation and the time that the computation takes. There may be a very strong consensus that accuracy is more important than computing time in a program that calculates the flight path of a planetary probe; yet there may be lesser consensus about whether accuracy or speed is the most significant aspect of a web service that finds driving routes through Europe.

2.2   Why Can Some Processes Be Automated?

Theoreticians often approach the question "*What can be effectively automated?*" through more general questions such as "*Why can some processes be effectively automated?*" or "*What kinds of properties are typical of processes that can be effectively automated?*" The theoretician's mode of working often starts with proving that a single process can be effectively automated and then generalizing the proof to a class of processes that all can be effectively automated because they share some essential similarities.

For the practitioner, theoretical questions about single processes "*Can process* p *be effectively automated?*" are still important: only if the answer is yes, the practitioner goes on asking, for instance, "*How can process* p *be automated?*" or "*Which implementation automates process* p *most efficiently?*" (In the latter question the practitioner often has to define which aspects should be optimized at the cost of others). But the practitioner, too, can ask general questions such as "*What kinds of problems do certain implementations automate best?*" or "*What kinds of implementations are generally best for problems like f(x)?*"

Although the terms *effective* and *efficient* are often used interchangeably, there is a certain difference between them. Generally speaking, *effective* means something that can produce a desired effect without unnecessary effort whereas *efficient* has a connotation of something having a good input/output ratio. The emphasis on effectiveness and efficiency is one of the central issues for the practitioner [5]. There are two reasons for that emphasis: theoretical and practical.

Firstly, there are theoretical considerations about effectiveness, that is, tractability. If a problem is considered to be intractable, the practitioner needs to employ very different implementation strategies than if the problem is considered to be tractable. Secondly, there are practical considerations of efficiency. Many proponents of the engineering-view of computer science (e.g., [2, 9, 14, 20]) have noted that the practicing computer scientist needs to be aware of real-world concerns such as operationality, usability, and the maintainability of implementations. The practicing computer scientist often needs to take into account time frames, budgets, project management, material and human resources, cost-effectiveness, and resource consumption. One of the central questions for the practicing computer scientist is "*How can process* p *be automated efficiently?*" Kimmo Raatikainen has noted that the question "*How can process* p *be automated reliably?*" is as important as the question of efficiency [18].

All the questions presented in this section are concerned with *what* can be effectively automated, *why* can some things be effectively automated, *how* can one automate those things, and how to achieve *effective* and *reliable* implementations. Most computer scientists work in areas where those are the central questions. Although those questions may crystallize the computational core of computer science, in the following section I consider how the recent characterizations and definitions of computer science; which have increasingly extended computer

science towards human, social, and ethical issues; have brought new questions into focus.

3.   Human-Centered Computing

In their 1989 report *Computing as a Discipline*, Denning's task force noted that in the discipline of computing, science and engineering cannot be separated because efficiency is one of the fundamental issues in computing [5]. Denning et al.'s brand of computer science focuses on the *theoretical* and *technical* limits of machine computability, and its fundamental question regards *what machines can do*. This view, still dominant and widely accepted in 1989, could be called *machine-centered computing*.

Ed Lee, in the *34th IEEE Computer Society International Conference*, San Francisco, presented a definition of computer science that was much different than Denning's definition, yet Lee's and Denning et al.'s definitions were both presented in 1989. Lee wrote, "*Computer science is the field of human endeavor that includes the study, the design and the use of machine based data processing and control systems to enhance peoples' ability to study information, perform work or explore reality*" [13]. In Lee's opinion, the focus of *human-made* computer science is the *human*. He wrote, "*Neither a computer nor the teaching of computer science has any value or meaning outside of its impact on people*" [13].

Lee's definition of computer science places the human, not the computer, at the core of computer science. In other words, computer science has no intrinsic value, but it only acquires value through its effects on people or society. This view could be called *human-centered computing*. Human-centered views in many branches of computing such as human-centered multimedia implicitly or explicitly incorporate the view that technology has no intrinsic value but the value of any technology is measured by its impact on people. It can be argued that human-centered computing began in the 1980s (when the focus of interface design shifted from programmers to end users (see, e.g., [8])), yet it can also be argued that the shift from machine-centered computing to human-centered computing is still largely uncertain and incomplete.

There are a variety of concepts that are similar to human-centered computing; for instance, *new computing* [19], *end-user computing* [15], and *user-centered technology* [10]. Those terms have their strengths but also have their weaknesses. Ben Shneiderman's term *new computing* emphasizes a shift from *old computing* (which is about what computers could do) to *new computing* (which is about what users can do). Although the content of Shneiderman's new computing is rich and human-centered, the term *new computing* itself is vague. (What is new computing anyway? Today's new computing is tomorrow's old computing.) The terms *end-user computing* and *user-centered technology* both focus on the human aspects of computing, but *user* in both terms implies that people are active in their interaction with computing technologies.

Instead of being consciously active *users*, people are nowadays often *not active* in their interaction with computing technologies. Note that a large technological shift happened after Denning's report in 1989, in the period between the early 1990s and mid-2000s: Increased affordability, miniaturization, integration, and interoperability of information and communication technology took computing machinery from the desktop to the pocket, from cable-bound to wireless, from rare to ubiquitous, and from shared to private (cf. [11]). In addition, during the period between the early 1990s and mid-2000s the amount of technology increased, its forms diversified, and information and communication technology gradually became an integral and commonplace part of many people's lives in industrial countries. Computing technologies have pervaded everyday life; people use computing technology when they turn off their alarm in the morning, heat their breakfast, drive their cars to work, collaborate with their colleagues, call their friends, and entertain themselves [11]. But computing technologies are often active also when people do not even know they are there: switching lights on and off, regulating room temperatures, monitoring people's moves and traffic, and so forth.

Although the ubiquity and the number of technologies have increased, the need for people to be active users of computing technology has not increased to the same extent, because new computing technologies often work without people having to be aware of them (cf. ubiquitous computing). Because very often people are in contact with computing technology without knowing it, and because the verb *use* seems to connote some kind of conscious intention to use technology, I prefer the term *human-centered computing* to the terms *user-centered computing* and *end-user computing*. (I agree that the choice of the term is debatable – see, e.g., [17].)

4.   The Ethical Question

Technological development has always entailed questions of whether certain things should be automated or not. A prominent philosopher of technology, Carl Mitcham, noted that skepticism about technologies could already be seen in the ancient myths of Prometheus, Hephaestus, and Daedalus and Icarus [16]. Mitcham wrote that the attitudes about technological development have ranged from Ancient skepticism to Enlightenment optimism to Romantic uneasiness [16]. New media and communication systems have always been especially suspect to suspicion. However, many technologists set aside ethical and social concerns from their work by arguing that science, technologies, or technological development are neutral or value-free.

Neither the theoretician's question "*What can be effectively automated?*" nor the practitioner's question "*How can process* p *be automated reliably and efficiently?*" include, explicitly or implicitly, any questions about *why* processes should be automated at all or if it is *desirable* to automate things or to introduce new technologies. The theoretician's and practitioner's questions belong clearly to machine-centered computing. When the central concern is "*What machines can do*", ethical issues can be set aside by arguing that machines have no conscience, that technologies are value-free, that theories are neither good nor evil.

A shift from machine-centered computing towards human-centered computing inevitably brings along new questions altogether. When the central concern is "*What people can do*", the whole gamut of ethical and social questions cannot be brushed aside. Those questions include "*What should be automated?*", "*Should process* p *be automated or not?*", "*Why should process* p *be automated?*", "*When should process* p *be automated and when not?*", "*What individual or societal consequences does automating process* p *have?*", and perhaps even meta-questions such as "*How can we know what should be automated?*" Ethical and social questions pervade all fields that purport to be *human-centered*, such as human-centered multimedia. The questions include, for instance, "*Where to employ new types of multimedia and why?*", "*What individual and social consequences do new*

multimedia systems have?", and "*Are the changes that new multimedia bring about desirable?*"

Ben Shneiderman argued that the key questions of human-centered "new" computing are "*not whether broadband wireless networks will be ubiquitous, but how your life will change as a result of them*" [19]. Changing people's lives is certainly not a mere technical matter but it entails ethical questions about what is *desirable*. The questions of machine-centered computing are *descriptive questions*, questions about "what is", but human-centered computing entails *normative questions*, questions about "what ought to be".

5. What Should Be Automated?

In contrast to Denning's fundamental question underlying all of computing, "*What can be (effectively) automated?*", the modern version of the fundamental question underlying all of (human-centered) computing might be, "*How can one effectively automate processes that can and should be automated?*" This question includes the practitioner's question, "*How can process* p *be automated?*", the theoretician's question, "*What can be effectively automated?*", and the human-centered question "*What should be automated?*" All three aspects are necessary, but they lie in different domains of knowledge. The first one belongs to the domains of engineering-oriented and empirical computer science, the second one belongs also to the domain of theoretically-oriented computer science, and the third one belongs perhaps to
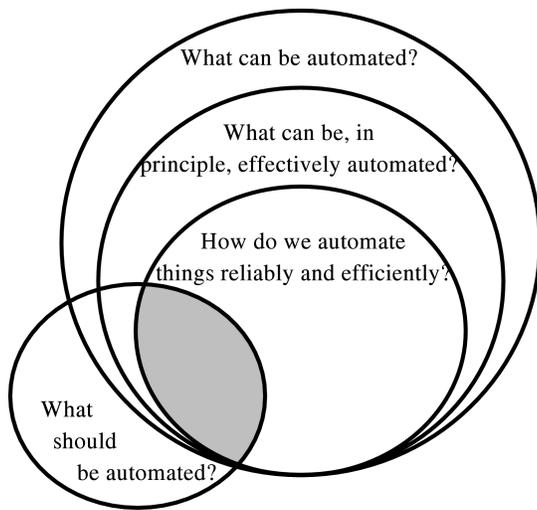

Figure 1: Four Fundamental Questions in Human-Centered Computing

the domains of social sciences and applied philosophy (Figure 1).

The theoretician's questions, the practitioner's question, and the human-centered question are portrayed in Figure 1. All those questions are important themselves and research in non-intersecting areas is important. However, whereas in machine-centered computing the production of *effective* and *reliable* computing and communications machinery takes place in the area delineated by the question "*How do we automate things reliably and efficiently?*", in human-centered computing the *responsible* production of *useful* computing and communications machinery takes place in the intersection marked with light gray.

In order for human-centered computing to really respond to human needs, it is not enough to consider only technical and

ethical questions. The different branches of human-centered computing also require an understanding of the needs, wants, hopes, expectations, and wishes that people have about technology. And they also require an understanding of the fears, threats, and anxieties that people have about technology, as well as other challenges and barriers to technology. Naturally, humanities and social sciences play an important role in human-centered computing.

If the field of computing is shifting towards human-centered computing as it has been described, and if the theoretical-methodological-conceptual toolbox of computer science turns out to be inadequate for human-centered computing, tools need to be borrowed from other disciplines. In Denning et al.'s machine-centered computing there are three separate, but irrevocably intertwined, branches: *theory*, *modeling*, and *design* [5]. In human-centered computing there is one more branch, called, for instance, *social studies of computing* (see, e.g., [12]).

6. Whose Concern Are the Social And Ethical Issues?

The debate about the inclusion of sociocultural topics under the term *computer science* has sometimes been harsh. The status and position of social studies of computing seems to be a charged, emotional issue. Those who argue that social issues do not belong to computer scientists are sometimes labeled as irresponsible technophiles, and those who argue that social issues belong to computer scientists are sometimes labeled as idealistic do-gooders.

Although the status of philosophical, sociological, psychological, anthropological, and all other kinds of scholarship can be disputed in machine-centered computing, philosophical, sociological, psychological, and anthropological studies clearly belong to human-centered computing. The more practical goals a branch of computing has, the stronger the need is for socially and ethically informed research and production. What is *not* clear in human-centered computing is whether social, ethical, political, or cultural issues should be the concern of *computer scientists* or of some other group. One might argue that it is not computer scientists' task to study things like social actions, cultures, the human behavior, or ethics. However, computer scientists do have expert knowledge about the possibilities and limitations of computing technology, and that expertise is indispensable in social studies of computing.

Another difficulty in arguing that the contemplation of ethical issues concerning technology is not the task of computer scientists comes from the fact that computer science is not detached from society. The products of computer science are used in society and its scientific activities are made possible by society. Receiving funding from external sources entails ethical questions. Any effects on society and individuals entail ethical questions such as "*Who is affected by technology?*", "*How they are affected?*", and "*Are such effects desirable?*" Ethical and professional issues are nowadays largely acknowledged as a legitimate branch of computer science, as seen in, for instance, the ACM Computing Classification System 1998 or Computing Curriculum 2001.

Clearly, neither human-centered computing nor social studies of computing would eliminate the need for technological experts – computer scientists play a central role in all development of computing and communication technologies. But in human-centered computing the questions are not only about

machines; the questions are about people and what people can do. The chances are that the methodological, conceptual, and theoretical framework of computer science, as described in, for instance, Denning et al.'s report [5], turns out to be insufficient to deal with the issues of social studies of computing. That is, the framework of computer science may turn out to be insufficient for selecting, recording, understanding, explaining, analyzing, or predicting phenomena in the field of human affairs. Co-operation between experts from different disciplines is necessary.

It indeed seems unreasonable to expect computer scientists to become experts in sociocultural, ethical, economic, or other issues outside of computer science. Mastering computer science is hard enough as it is. It is equally unreasonable to expect people from other disciplines to become experts in computer science. Experts, specialists, and professionals have their areas of expertise and they should do what they know best. Instead of scores of broadly trained bricoleurs, human-centered computing requires a working multidisciplinary combination of experts from different fields.

Many aspects of human-centered computing and social studies of computing are currently wide open. Although the significance of human-centered computing has increased since the advent of personal computing in the late 1970s, it is not yet clear what human-centered computing really is. It is not certain at all whether human-centered research should belong to academic computer science in the first place. It also seems that philosophers, sociologists, psychologists, anthropologists, and all other kinds of scholars are equally reluctant to adopt human-centered computing as their own.

Perhaps human-centered computing and other human-centered branches of computing do not need to spawn new interdisciplinary fields, but perhaps *human-centered computing* should work as an eclectic, multidisciplinary umbrella term. That is to say, maybe *human-centered computing* should not become a branch of computer science, but maybe it should become a concept that people use to systematize, categorize, and collect other concepts.

7.   Conclusions

It is certain that there is increasing interest in human-centered issues in the field of computing; especially lucid this trend is in practically-oriented branches of computing. Yet it is uncertain whether human-centered computing is or should be a part of computer science or if it should be subsumed by some other discipline – or if it should *not* be subsumed under any single discipline. Regardless of where human-centered computing is located in the academic landscape, focusing on the human in computer science research inevitably brings forth a number of ethical and social questions that have not been important in machine-centered computing.

In human-centered computing concerns about social and cultural responsibility, responsiveness to people's needs, the consideration of social and cultural consequences, and sensitivity to human expectations and anxieties limit the production of computing machinery as much as the machine-centered questions of efficiency and reliability. The ethical questions of human-centered computing are certainly not any easier than the technical and theoretical questions of machine-centered computing. But no matter how one approaches the new problems that a human focus brings forth, a shift from machine-centered computing to human-centered computing inevitably changes the question from "*What can be automated?*" to "*What should be automated?*"

References

[1] Arden, Bruce W. (ed.) (1980) *What Can Be Automated?: Computer Science and Engineering Research Study*. MIT Press: Cambridge, MA, USA.

[2] Brooks, Frederick P., Jr. (1996) *The Computer Scientist as Toolsmith II*. Communications of the ACM 39(3): pp.61-68.

[3] Denning, Peter J. (1985) *The Science of Computing: What is computer science?* American Scientist 73(1): pp. 16-19.

[4] Denning, Peter J. (2003) *Great Principles of Computing*. Communications of the ACM 46(11): pp.15-20.

[5] Denning, Peter J.; Comer, Douglas E.; Gries, David; Mulder, Michael C.; Tucker, Allen; Turner, A. Joe; Young, Paul R. (1989) *Computing as a Discipline*. Communications of the ACM 32(1): pp. 9-23.

[6] Dijkstra, Edsger W. (1987) *Mathematicians and Computing Scientists: The Cultural Gap*. Abacus 4(4): pp.26-31.

[7] Forsythe, George (1969) *Computer Science and Education*. Proceedings of IFIP Congress 1968. August 5th-10th 1968, Edinburgh, UK: pp. 92-106 (Volume 2).

[8] Grudin, Jonathan (1990) *The Computer Reaches Out: The Historical Continuity of Interface Design*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People. April 1.-5., 1990, Seattle, Washington, United States: pp.261-268.

[9] Hartmanis, Juris (1993) *Some Observations About the Nature of Computer Science*. In Shyamasundar, Rudrapatna K. (ed.): "Lecture Notes In Computer Science vol. 761": pp.1-12.

[10] Johnson, Robert R. (1998) *User-Centered Technology: A Rhetorical Theory for Computer And Other Mundane Artifacts*. State University of New York Press: Albany, NY, USA.

[11] Kamppuri, Minna; Tedre, Matti; Tukiainen, Markku (2006) *Towards the Sixth Level in Interface Design: Understanding Culture*. Proceedings of the CHI-SA 2006, 5th Conference on Human Computer Interaction in Southern Africa (ed. van Greunen, Darelle). January 25th-27th 2006, Cape Town, South Africa: pp.69-74.

[12] Kling, Rob (1980) *Social Analyses of Computing: Theoretical Perspectives in Recent Empirical Research*. ACM Computing Surveys 12(1): pp.61-110.

[13] Lee, Ed (1989) *Some Suggestions on a Computer Science Undergraduate Curriculum*. Proceedings of the COMPCON Spring '89: 34th IEEE Computer Society International Conference: Intellectual Leverage. Feb.27-Mar.3, San Francisco, CA, USA: p.366.

[14] Loui, Michael C. (1995) *Computer Science is a New Engineering Discipline*. ACM Computing Surveys 27(1): pp.31-32.

[15] Mahmood, Mo Adam (2002) *Advanced Topics in End User Computing*. Idea Group: Hershey, PA, USA.

[16] Mitcham, Carl (1994) *Thinking Through Technology: The Path Between Engineering and Philosophy*. The University of Chicago Press: Chicago, USA.

[17] Norman, Donald A. (2005) *Human-Centered Design Considered Harmful*. Interactions 12(4): pp. 14-19.

[18] Raatikainen, Kimmo (2004) *Issues in Essence of Computer Science*. Unpublished manuscript. Available at https://www.cs.helsinki.fi/u/kraatika/Papers/IssuesInEssence OfComputerScience.pdf

[19] Shneiderman, Ben (2002) *Leonardo's Laptop: Human Needs And the New Computing Technologies*. The MIT Press: Cambridge, Mass., USA.

[20] Wegner, Peter (1976) *Research Paradigms in Computer Science*. Proceedings of the 2nd international conference on Software engineering. October 13-15, San Francisco, California, USA: pp.322-330.