Chapter 1

Perspectives on Social Computing

CONTENTS

Computing is increasingly called upon to move outside of the computer. The blazingly rapid advances in computer science have enabled applications that could not have been conceived of even a couple of decades ago. In particular, modern applications of computing pull us into the real world of people with all its attendant messiness.

In computational terms, social entities are not only autonomous but are also *heterogeneous*, meaning that they are of distinct construction. Heterogeneity at the level of programming language or operating system or information representation is important for the purposes of realizing implementations but is more an artifact of the way we program today than of any fundamental aspect of the concerned social entities. In addition, heterogeneity at the level of the meaning of the information produced and

accessed by social entities, including the semantics and pragmatics of that information is deeply ingrained in the nature of the social entity. Dealing with this heterogeneity calls for not just technical solutions but technical solutions supplemented by a computational modeling of the social structures (internal) and relationships (external) of the parties involved.

Related to autonomy and heterogeneity is the notion of *dynamism*, which refers to the continual changes in the meaning of information as well as of the social structures and relationships. Dealing with these at a computational level is a nontrivial challenge and involves a combination of innovations in architecture that reflect social relationships as first-class abstractions.

Table 1.1 describes an easy way to distinguish the three properties mentioned above.

Table 1.1: Characterizing openness.

Property	**Definition**	**Impact**
Autonomy	Independent action	How will a party act?
Heterogeneity	Independent design	How will a party represent information?
Dynamism	Independent configuration	How can a party come and go?

These properties taken together describe what are called *open* IT environments. Sometimes, in addition, the *scale*, that is, the potential for large numbers of parties in a number of overlapping simultaneous transactions and relationships, is included in openness. An important point from our standpoint is that whereas traditional computer science deals best with closed environments, modern IT environments are open. To handle such environments presupposes flexible approaches and arms-length relationships.

Social computing is interesting as a computing discipline because it provides us with representations and techniques—and a new mindset—for dealing with openness.

1.1 Microsocieties and Sociotechnical Systems

We turn to human interactions for inspiration. In essence, we can treat each information environment as a *microsociety*. The specification, analysis, and enactment of such a microsociety using social abstractions is what social computing is about.

Conceiving of, representing, and reasoning about microsocieties gives us a handle on talking about sociotechnical systems (STSs) in a high-level, meaning-rich,

and reusable manner while remaining grounded in computer science. In our conception, an STS is a system of two or more autonomous or social entities who interact with each other through or about one or more technical entities. The social entities include people, communities, teams, and organizations; the technical entities include computers, networks, IoT devices, data stores, and software. This conception of STS has the benefit of being a social abstraction that can be computationally applied.

It is worth pointing out that our conception of an STS lies between the conventional conceptons of STSs. One of those conceptions is concerned with the application of technologies in societies and user communities where there is no computation or reasoning by the participants about their interaction, just the sociologist's view of how the nature of those interactions may have changed as a consequence of new technology having been introduced. For example, an STS investigation may point out that the advent of computers in medicine has led not to a reduction in time spent by physicians on record keeping but an increase in the time spent. Sometimes the term sociotechnical systems is used to talk about Society as a whole or a large business organization responds to the introduction of Technology. For example, researchers may be concerned with the effects of introducing email on productivity or the effects of introducing social media on social distance. Those are valuable concerns but largely removed from the present setting. Here we are not interested in Society and Technology in the large but instead in computational models that relate to decision making and actions by the social entities. For example, we may be interested in computationally representing and reasoning about whether Alice sharing Bob's health records with Charlie violates some relevant social norm that applies to Alice.

The second conception of an STS is of anywhere we see people (as social entities) interacting with computers or mobile phones (as technical entities). This view doesn't require there to be an social effect and does not even require there to be two participants. Clearly, one could argue that a user playing Solitaire on his computer in his spare time is a weak example of social computing at best.

There are clear reasons for accommodating openness—namely, that the parties concerned demonstrate the properties of autonomy, heterogeneity, and dynamism and their interactions reflect the subtleties of these properties. However, in addition, there is a major architectural reason to support these properties. Treating the concerned parties as autonomous, heterogeneous, and dynamic leads to cleaner interfaces with better modularity and encapsulation. Thus, we can apply social computing principles even in settings where we know autonomy, heterogeneity, and dynamism to be limited. For example, we might architecturally model an enterprise as if its various divisions were fully heterogeneous and its staff fully autonomous though they

might not be so. The benefits lie in reducing the fragility of an architecture and thus future-proofing a sociotechnical system.

Interestingly enough, business environments meet our criteria for social computing. This is because what characterizes business environments is that they involve the interactions of autonomous parties. What differentiates social computing (as defined here) from traditional computing is its emphasis and focus on the interactions and autonomy. More generally, we see other environments, such as scientific collaboration or even academic computing that demonstrate the interactions of autonomous parties. We can therefore pull them into the scope of social computing and understand them from the perspective that we apply throughout this volume.

The unifying theme is that in social computing, we seek to build and compute with abstractions that at the core reflect intuitions of human social interactions.

1.2 Understanding Social Computing

Another way to think of social computing is that like any other academic discipline it does not have sharp boundaries. Therefore, a way to approach understanding social computing is to consider a number of putative applications or platforms for computing that involve a social element. We can abstract out their important features and determine what features are more important or more decisive in determining if an application or platform is social.

Table 1.2 lists some major platforms that demonstrate social features. Arguably, the first prominent social computing application is email. Email is an important test case for clarifying our concepts. Clearly, email is used to achieve social interaction. However, the computational support in email is primarily concerned with low-level details. Specifically, an email platform provides *(a)* a way for drafting and sending messages; *(b)* a way to store and retrieve messages, including threading conversations; and *(c)* a way to suggest people to communicate with based on prior conversations However, the social structure is primarily in the minds of the users. That is, the social state is largely epiphenomenal as far as the email platform is concerned.

The next few entries in Table 1.2, Google search and Netflix movies, are social only to the extent that they bring together information produced by multiple people to solve a problem of interest to other people. For Google search, the estimation of the relevance of a search result is based in part on hyperlinks created by people. For Netflix, movie recommendations are based on similarities of ratings of movies given by various users. In these cases, the hyperlinkers and movie raters do not interact with each other though their insights are aggregated together.

Table 1.2: Platforms and applications with a social flavor.

Platform	Problem	Key Feature
Human communication		
Email	Facilitate communication	Conversations, contacts
Blogger.com	Support conversations	Feeds, posts, comments, tags
Who . . . Millionaire?	Answering questions	Voting
Selection and recommendation		
Google	Ranking search results	Hyperlinking
Netflix	Recommend movies to users	User profiles, ratings
Amazon	Help users in product selection	Reviews, comments, ratings
Social media		
Twitter	Find tweets on a topic	Common hash tags
Facebook	Services for apps	Social network
Collaboration		
Wikipedia	Create a free encyclopedia	Revision history, talk
Quora	Find information on something	Questions and answers
Reddit	Select top stories	Feeds, posts, comments
Stack Overflow	Question answering	Posts, comments, ratings
Crowdsourcing and human computation		
reCAPTCHA	Recognize text in images	User-provided content
ESP game	Determine photo content	User-provided captions
Mechanical Turk	Perform tasks	Market, HIT
Iowa Electronic Markets	Predict election outcomes	Dynamic futures pricing
External purpose		
KickStarter	Select fundable projects	Projects, rewards, backing

1.2.1 Recommendation

Several other of the applications in Table 1.2, e.g., in Amazon review ratings, make the interactions among people explicit and use them as a basis for rating the reviews (and reviewers) that rate products. Some, like Stack Overflow, apply a similar idea to producing and rating answers to questions that users pose. Reputation or status is an important motivator in these applications. The users come together to interact for a product or question but have no long term association.

Facebook goes further in that it exploits the somewhat stable social relationships between users, especially friendship, to further guide their interactions and produce

customized recommendations for content that the users may find valuable.

1.2.2 Crowdsourcing and Human Computation

Recent years have seen an increasing attention directed toward socially inspired approaches for finding and judging information. Crowdsourcing involves using people to provide or vet information. The intuition is that what a *crowd* knows is usually better than what an individual knows, and with greater certainty. For example, to find the population of Brazil, we can query a (large) number of people, and use their median answer as an effective approximation.

reCAPTCHA is the common application we all encounter when we try to access some service on the web, e.g., to create a new account. reCAPTCHA addresses two problems. The first problem is that when there is no other authentication, e.g., because one's account isn't set up yet, it is possible for attackers to use software bots to request the service. For example, attackers may create spurious accounts or access free services to benefit themselves or harm others. reCAPTCHA serves as evidence that the requester is human because it involves completing a task (such as image recognition) that is easy for people but difficult for software. The service already knows the answer and a user establishes his or her credentials as a human by providing that answer. The second problem that reCAPTCHA addresses is how to acquire ground truth or labels for a problem for which a clear answer is not known. This problem could be identifying words scanned in from old publications. Because reCAPTCHA presents a composite problem to the user as a challenge, the user does not know which is which. Each user has an interest in proceeding with whatever he or she wishes to accomplish, and therefore generally complies by providing a useful answer to each part of the challenge. The answers of many users are collated to determine what the scanned words really are.

A similar process occurs with the ESP game where users are asked to create photo captions, and points are awarded and captions used when independent users happen to agree on the terms used. The above approaches, though intriguing, are highly limited. In particular, they replace intelligence by an estimation of the majority answer. In some cases, the estimation can be through statistical methods, as in the reCAPTCHA example. In other cases, the estimation can be through incentives for the participants to agree, as the ESP game demonstrates.

Subsequent approaches expand from the above into settings where there are explicit incentives. For example, Amazon's Mechanical Turk web services support the execution of human intelligence tasks by people. This approach broadens the set of possible applications; well-known examples (`http://aws.amazon.com/mturk/`)

include natural language translation, media production, and data cleansing.

These approaches have a common structure in that they are centrally conceptualized. One party decides to have a problem solved and offers financial or other incentives to persuade members of a crowd to solve it. Such approaches can work when we can treat the members of the crowd as external entities, safely understood as a service, and which we can largely assume are disinterested except for the incentives we offered them. Further, the people work in essence independently of each other. To summarize, traditional approaches are centrally driven (orchestrated), treat the people as independent and disinterested, and rely upon consensus or majority view to determine the answer they offer.

1.2.3 Knowledge Management

Knowledge management is an example of a social application. Since the earliest days of knowledge management in computer science, researchers have considered how to build platforms that users could adopt in order to sustain knowledge management. The idea behind knowledge management is to find a way to support capturing the knowledge that people (especially employees of a company) have so that such knowledge could be reused by other employees of that company. In other words, knowledge would propagate from the employees toward the central storage.

Knowledge management in its typical rendering ends up as a software module through which the participants communicate, including providing their ideas or hunches, fielding user queries, and storing and retrieving information. That is, the social nature of the knowledge is mostly left outside of the computational realm and purely in the minds of the participants.

1.3 Toward a Scorecard for Social Computing

Social computing, like any other intellectual discipline, would not have a crisp definition of what's in and what's out of its scope. Building upon the intuition expressed in our working definition of social computing as computing of and with social relationships, we can identify some key features that indicate social computing.

Here, we focus on the core of a problem—how it is modeled—as opposed to architectures of particular solutions that may have been or could be produced. In other words, we seek to understand what the problem itself calls for regardless us potential solutions to it, which may make additional assumptions.

Table 1.3: Key distinguishing features of social computing applications.

Meaning	Examples
	Autonomy
What is the extent of independence of the various parties?	• Who may initiate a computation? • Who selects the participants?
	Strategy
Are the parties interested in the outcome?	• Do the parties interact repeatedly? • Do the parties learn and might useful outcomes emerge? • Is it a majority or a minority game?
	Interaction
How do participants interact with each other and with any requesters not closely involved in the computation?	• How rich or complex are the interactions? • Can a coalition be formed? • Is the nature of the work negotiable? • How is the service engagement governed?
	Meaning
How much of the social state is represented computationally?	• Is the meaning of a social relationship or interaction represented? • How deep is the meaning of the social state? Does it relate to the intimacy participants feel toward each other? • Does the represented meaning involve human concerns, such as emotion and personality? • Is the meaning explicit so that the participants can extract it and reason about it?
	Time and Structure

. . . Continued

Meaning	Examples
How is the work structured?	• Is it a one-off task? • Who selects the performer and assigns the task? • Who completes the task? • Who evaluates the task and how? • Are the results produced continually?
	Originality
How much originality or insight does the work require?	• Does it require original thinking? • Does it encourage consensus thinking or novelty? • Does it promote creativity?

These features or dimensions can help us distinguish diverse forms of social computing and when they occur together they indicate a stronger form of social computing than when they fail to occur.

The following questions provide an indication of the nature of the various features. Answers to these questions indicate whether and to what extent an application or architecture may be considered social. These questions can be used as a basis for what we might call a *social computing scorecard*—a higher score corresponds to a more central instance of social computing. We won't go into numeric scores, however, because such scores would be quite ad hoc unless one had a specific narrow purpose in mind.

- *Autonomy and organization:* What is the extent of independence of the various parties? Specifically, who may initiate a computation? Who selects the participants? Is the work carried out collaboratively between two or more participants? How complex are the social relationships between the participants?

- *Strategy:* Are the parties interested in the outcome? Specifically, do the parties interact repeatedly? Do the parties learn and might useful outcomes emerge? Is it a majority or a minority game?

- *Interaction:* How do participants interact with each other and with requesters? Specifically, can a coalition be formed? Is the nature of the work negotiable? How is the service engagement governed?
- *Meaning:* How much of the social state is represented computationally? Specifically, is the meaning of a social relationship or interaction represented? Is the meaning explicit so that the participants can extract it and reason about it?
- *Time and planning:* How are the work and interactions structured? Specifically, is it a one-off task? Are the results produced continually?
- *Human intelligence:* Does the interaction involve human judgment, for example, in regards to producing creative ideas?
- *Computing:* Is the main contribution centered on computing?

Let's consider the last question above as a segue into a deeper discussion of computing. Computers are used everywhere so the mere use of a computer to solve a problem that relates to some task in sociology or psychology would not rate as a social computing. For example, one may carry out data analysis to discover the age, education, and wealth distributions of the population of Wake County in North Carolina. This may be a valuable computational exercise but I would place decidedly outside of social computing. If were instead to compute these distributions for Facebook users, the problem would not quite change. Now it is possible that in the latter case there is something about the data and how it is obtained that would make it interesting from a social computing perspective. But it would be more compelling if weren't merely reporting facts but applying such facts in further reasoning, e.g., in making higher-quality friend recommendations on Facebook or in the physical world.

1.4 Social Computing as Computing

Another direction in which it would help to expand our working definition of social computing is into a characterization of a formal model for computing. Such a characterization can provide a foundation for further investigations.

To do so, let us consider the following two leading and complementary abstract models of computation established in the literature. Likewise, we can understand social interactions in corresponding terms.

The first idea is of an *automaton* or machine that maintains a state and that describes transitions from one state to another; the machine could be deterministic

or not and finite or not. Examples of such machine models are finite state automata, push-down automata, Turing machines. A possible application of such a model is in describing how a vending machine may operate. For simplicity, imagine that this machine vends only one type of product, a candy bar, and only takes one type of coin. Such a vending machine would have states corresponding to two counters: one for how many candy bars it has and one for a payment within a transaction. Suppose it takes two coins to buy a candy bar. The vending machine would also have a coin counter with states corresponding to whether there it has received zero, one, or two coins since its last transaction. Suppose initially that the machine contains 100 candy bars and zero coins. When a customer feeds the machine coins, the coin counter increases from zero to one to two. If the machine is ever in a state where it has received two coins and it has one or more candy bars, it would dispense one candy bar, reset the coin counter to zero and reduce the number of candy bars by one. (It would also save the coins and a practical machine would let a customer take back a coin, but let's disregard such details.) If the machine is ever in a state where it has received two coins and it has zero candy bars, it would return the coins, reset the coin counter to zero and leave the number of candy bars unchanged. To describe good behaviors, we would specify the accepting or final states of the vending machine: each state where the coin counter is zero and the number of candy bars is between zero and 100 is an accepting state.

The second model is based on a formal *grammar* for describing a formal *language* treated as a set of *sentences* constructed out of elementary *tokens*. We can define a language (and grammar) for our vending machine example. A token in this setting is coin (meaning a coin being inserted), bar (meaning a candy bar being dispensed), return (meaning two coins are returned to the customer). Specifically, the language for the vending machine contains sentences such as these: *(a)* coin, coin, bar; and *(b)* coin, coin, return. There is a grammar for the language—a set of rules by which to generate the sentences in the grammar—corresponding to each machine. The grammar for the vending machine language would be quite simple because the machine is quite simple. In general, machines and grammars can be a lot more subtle.

Although there is a correspondence between grammars and machines, there is a fundamental difference. A machine describes how to achieve specified functionality. A grammar describes a machine's interactions with its environment. That is, the grammar-language view focuses on what the interaction is and the machine view focuses on how to realize it.

The above concepts can be lifted quite naturally to the social setting. Whereas the state of the vending machine captures the numbers of candy bars and coins, a *social*

machine would be built on social states. Specifically, we can model the social state of a microsociety as a set of social relationships among the relevant principals. A social machine would transition from one (social) state to another based on relevant events that describe how the social relationships progress as the concerned parties interact with each other. Likewise, we can imagine a *social grammar* as describing how the concerned parties interact in terms of their potential communications among principals understood at the social level.

For example, we can understand a part of Facebook's functionality in social terms. Imagine we have only three users, Alice, Bob, and Charlie. Initially, all have accounts but no one is anyone's friend. If Alice requests to be added as Bob's friend, Bob receives her invitation. If he accepts it, they become each other's friends. If Bob rejects it or does nothing, they do not become friends. Likewise, Alice and Charlie can become friends. Once Alice is friends with both Bob and Charlie, Facebook may generate a friend suggestion for Bob and Charlie to consider adding the other. If either of them takes up the suggestion, the same friendship process ensues.

We might describe the Facebook social machine as having states, each of which describes who is whose friend and whose friendship invitation is pending with whom. The initial state shows no one as anyone's friend. When Alice's invitation to Bob goes out, the state changes to one where Alice's invitation is pending with Bob. Once Bob accepts it, the state changes to one where Alice's invitation is not pending but Bob and Alice are friends. In this view, Facebook's suggestion has no effect on the social state. One could create an alternative social machine where it does.

In terms of the formal grammar, we would characterize what communications may take place between Alice, Bob, and Charlie. Some possible sentences in the associated language would be these: *(a)* invite(Alice, Bob), accept(Bob, Alice), . . . ; and *(b)* invite(Alice, Bob), invite(Alice,Charlie), accept(Charlie, Alice), accept(Bob, Alice), . . .

In conceptual terms, then, we can understand social computing as computing where the objects of the computation are based on social state. That is, we can understand social computing in terms of automata that manipulate social state or in terms of a grammar whose content applies to communications understood at the social level. Notice that we understand communications in social terms, not in terms of the bits transmitted or even the format adopted for a message though such details are important in realizing a practical system. But such low-level details apply equally to communications between software programs. In contrast, it is the high-level, social understanding of communications that justifies our treating interacting parties as autonomous and the ensuing computation as social computing.

The foregoing discussion is meant only to make the point that we can formulate

social computing in classical computing terms. The devil is in the details—of how we construct and manipulate social machine states and how we model and realize social communications. We will pick up this thread again and get into these details in Chapter 13. Until then, let's treat it merely as an abstract way of thinking about social computing that characterizes as a form of computing.

Smart et al. [2014] describe a somewhat traditional take on social machines, wherein the social machine is characterized by interactions among people that are facilitated by computers. The computers are essential to the enterprise but the computational representations are purely operational: all the social meaning is invisible in the computational representations and is represented entirely in the minds of the (human) participants.

With Amit Chopra, I have developed an alternative approach called *Interaction-Oriented Software Engineering* that characterizes social machines in terms of the social protocols that they support [Chopra and Singh 2016]. Each protocol is specified in terms of the social relationships among the participants and how those social relationships progress as the participants interact. In this way, the social meaning is central to the computational effort, and the explicitness of the meaning supports greater independence from implementation than is otherwise possible.

1.5 Historical Remarks

Why should anyone care about social computing? The last few decades have seen information and communications technology (ICT) work its way into virtually every domain of human society. Increasingly, ICT brings people together—sometimes in enhancements of interactions people have had for millennia and sometimes in ways that lack any precedents in the pre-ICT era. In either case, interactions among people are a crucial way in which ICT is applied, whether it be in business, education, or entertainment.

Social computing provides us a way of realizing the value and promise of ICT, and is the natural next step in the progress of computer science.

References

Amit K. Chopra and Munindar P. Singh. 2016. "From Social Machines to Social Protocols: Software Engineering Foundations for Sociotechnical Systems." In *Proceedings of the 25th International World Wide Web Conference,* 903–914. Montréal: ACM, April.

Paul Smart, Elena Simperl, and Nigel Shadbolt. 2014. "A Taxonomic Framework for Social Machines." In *Social Collective Intelligence,* edited by Daniele Miorandi, Vincenzo Maltese, Michael Rovatsos, Anton Nijholt, and James Stewart, 51–85. Computational Social Sciences. Springer.