Social Network Analysis and Interactive Device Design Analysis

Harold Thimbleby and Patrick Oladimeji
FIT Lab — Interaction Laboratory, Swansea University
Swansea, Wales
harold@thimbleby.net, p.oladimeji@swansea.ac.uk

ABSTRACT

What methods can we use to help understand why users adopt certain use strategies, and how can we evaluate designs to anticipate and perhaps positively modify how users are likely to behave? This paper proposes taking advantage of social network analysis (SNA) to identify features of interaction. There are plausible reasons why SNA should be relevant to interaction programming and design, but we also show that SNA has promise, identifies and explains interesting use phenomena, and can be used effectively on conventionally-programmed interactive devices. Social network analysis is a very rich field, practically and theoretically, and many further forms of application and analysis beyond the promising examples explored in this paper are possible.

Categories and Subject Descriptors

H.5.2 (D.2.2, H.1.2, I.3.6) [**User interfaces**]: Theory and methods; D.2.8 [**Metrics**]: Performance measures

General Terms

Design, Human Factors.

Keywords

Interaction programming, social network analysis, graph theory, network center.

1. INTRODUCTION

The book *Press On* [25] summarizes the value of network analysis in the design of interactive systems. Network analysis is a well-established mathematical field with many applications; it covers issues from connectivity to optimization. Recently attention has been focussed on what is called social network analysis (SNA), partly because of the discovery of interesting social network properties (such as short path lengths) but also because social network analysis highlights the special role of actors (e.g., people) and relations (e.g., friendships), and has thus drawn the attention of human and social scientists to the possibility of analytic research in fields that are otherwise overloaded with hard-to-analyze data. One of the advantages of social network analysis is that precise analytic methods can be applied to networks of thousands or millions of actors, something that would be quite impractical (and unreliable) to do by conventional methods.

EICS'09, July 15–27, 2009, Pittsburgh, Pennsylvania, USA.

Social actors, that is: people, organizations, nations, and so forth create and maintain relations with each other in order to achieve various purposes. Thus, for example:

- Nations relate to other nations by importing and exporting goods.
- Organizations relate to each other through product supply chains or by invoicing and payment.
- Terrorists relate to each other using (what they hope are) secret communications.

The idea of social network analysis (SNA) is that the structure of the network of relations between actors can, with analysis, identify "interesting," "important" or "critical" actors — and SNA has been applied with success to all of the examples above, and more. Of course what is "important" for one application may not be for another, but the point is that appropriate structural properties of the network of relations can identify humanly-important properties: this will be crucial in our analysis below since we may be comparatively ignorant of the psychology behind the networks we will analyze — the evidence is that this does not overly matter. For example SNA has been applied very successfully to the world wide web: each web page represents what its human author wishes that web page to relate to, and the resulting relation creates a network of billions of pages. The SNA of this network generates many insights [2] *without* analyzing the motives or planning of the authors. Later in this paper we shall give examples based on interactive systems, but one may note immediately that neither the size of the relation is an impediment to analysis nor is there any requirement to know the "psychology" of the user or designer.

1.1 The potential relevance of SNA to interaction design

A textbook social network analysis (SNA) application is to identify central terrorists, who then become a target for elimination with the intention that the terrorist network will become less efficient, possibly dysfunctional [9, 17]. Without SNA, it would not be obvious who the central terrorists are because the network is sparse and too large; moreover, the network is dynamic and therefore analysis has to be rapid if it is to be timely. Automatic SNA is essential.

Analogously, if a user does not know about a central mode or state in an interactive system for some reason, their behavior and use of a system will be inefficient (and in the case of medical or other safety-critical devices, their behavior will be dangerous). The direct analogy is that the user model is a network, and the device they are using is also a network. An interactive device clearly "works" and in principle has a known model, but the user's model of it for various reasons may have lost, ignored or eliminated bits

of the model. If those missing or broken parts of the user model are central to the operation of the device network, then the user's activities with the device will be dysfunctional to some degree.

This analogy only makes sense, however, if there is a device network, and that the relations in that network are important in some way to the use of the device. In fact, an interactive device and the user's model of it can both be represented as state machines [25], and these are exactly networks amenable to SNA. The state machines may be transition systems, menu hierarchies, etc.

Device models can be analyzed with SNA just because they can be represented as networks, but why should SNA measures reveal anything significant about interaction? Consider this: as a designer builds a device they add relations between existing states, intending to add functionality or to allow a user to make greater sense of the device, or for other reasons. New features have to be related to existing features, else they will be inaccessible. More profoundly, designers will add new features in places in the device network that "make sense" — whether this sense is based on their intuitions or based, more solidly, on empirical evidence. The claim of SNA is that this sense will reveal itself in the network structure, for instance in the various measures we will explore in this paper.

The design of devices creates networks over time, and is directly analogous to the creation of social networks over time, each step being a social actor making relations with other actors, and hence creating or extending the network. In interaction design, the "actors" are the device state, and the "relations" are the possibilities of action as designed into the system and chosen by the user in order to make it work. As with social networks, individual relations may not be very interesting as they tend to be obvious, like pressing the Off button puts the device in the state *Off*; but what *is* interesting is what can be deduced from the whole structure of the network of relations, and what user behaviors these structures help to explain. This is what SNA reveals.

1.2 Contributions and benefits

This paper will show that SNA generates interesting insights for designers and system evaluators and therefore should be considered another resource in the interaction designer's toolchest; in particular SNA provides a complementary view on design that can give designers new perspectives. Drawing on the field of social network analysis identifies key states in interactive system design, and gives a very good understanding of certain critical interaction issues. More generally, we show that SNA can provide wide insights into design; social network analysis is a rich and well-established field offering an enormous range of intellectual tools. For designers, network analysis can help analyze and design *for* effective user behavior, or to design *out* certain unhelpful user behaviors. A major contribution of this paper is to show that this is indeed a rich resource to mine for the purposes of interaction, and moreover that it is a practical and cost-effective approach.

One of the contributions of the paper is that the sorts of analysis proposed are not just insightful but are also easy to perform. The paper also shows how arbitrary interactive devices can be readied for such analysis: social network analysis assumes that some "network" is available to analyze. This paper therefore goes on to show that such networks can be obtained in a routine way, indeed in a way that also contributes to improved quality control of device production — an important contribution of this paper particularly for safety-critical domains. Once device development and SNA analysis tools have been set up appropriately, obtaining a network and analyzing it is trivial — and remains trivial even throughout iterative design as the device changes its specification. Put more strongly: it is arguable that if a device cannot be set up appropriately then it must be badly designed. It would certainly be difficult to defend such a design, particularly in the context of any critical regulatory evaluation.

There are many interactive devices that SNA can be applied directly to, but one might argue that some current interactive systems are so complex and so dynamic that they cannot be modeled easily as a network. Even if we concede that parts of systems are dynamical and do not have a discrete notion of "actor" or "state," as soon as a user models the interaction there is already a network: it will be composed of rules like "when u, if I do v then w happens" or similar — and these are examples of the sort of relation we discuss below. Furthermore, if the user's model is *not* a network in some sense, it is not at all clear what the user is supposed to have modeled nor what their model is! In short, SNA can certainly address issues relevant to users and interaction even though some aspects of some interactive systems are not amenable to the approach — and these aspects are generally details of the system that a normal user does not need to understand (for example, in a gesture-controlled game, the user does not worry how graphic rendering is achieved). We shall discuss infusion pumps below: their user interface is shown in this paper to be amenable to SNA, even though we make no attempt to understand their drug delivery models (infusion pumps control drug delivery to patients, and some pumps use sophisticated metabolic models).

A final argument is based on Daniel Jackson's "small scope hypothesis" [14]: he claims that many design defects can be found with small models, and large fully-detailed models are generally not necessary. In interaction design, we can strengthen this by saying that if a large network model was required, then the user would certainly not understand it. In summary, if the user model of an interactive device cannot be represented as a manageable network — if the device is not amenable to SNA — then it is not clear how the user is supposed to conceptualize interaction.

1.3 Validity

SNA is based on graph theory (also called network theory [4]), a branch of applied mathematics, so its underlying validity is assured by its mathematical validity; its construct and descriptive validity have already been established in the social domain [28], and the present paper argues and shows that the validity generalizes to interaction. Further, SNA is valid in the very important sense that it is *scrutable*: this paper has an associated web site where all results reported can easily be checked, or new models entered and studied.

As will become clear, SNA can quickly identify or prioritize interesting design features that can then, for example, be explored with conventional usability methods. To give a simple example: SNA could accelerate heuristic evaluation by informing evaluators of anticipated critical design features. Conversely, a heuristic evaluation that identified design issues that did not correlate with insights from SNA would raise very interesting questions that would beg explanation — whether in terms of the specific device, domain and tasks, or in terms of the appropriateness of the methodologies themselves. But, in addition to argument and evidence of insightful application to case studies, it would be nice to provide empirical support to justify using SNA proactively more generally. Using SNA thus raises questions of notions of validity — yet validity is not often discussed explicitly (but see [5]) so, unfortunately, it would be a burden beyond the scope of this paper to provide a discussion in depth.

An interesting view on validity in the interaction design field is presented by Wixon [29] who argues that in the real world of applied usability, validity means maximizing beneficial effects on the design of a product with least effort — he argues that conventional

validity concepts in the literature do not make sense in the applied context. We shall see that SNA delivers design insights with essentially no effort direct from a working interactive system: that is, validity in Wixon's sense. A simple tool (which this paper briefly describes) could simply list and prioritize design issues of the sort that SNA characterizes. As these issues are fixed, and possibly other issues introduced (!), rerunning the tool lists remaining problems. In this sense, SNA is valid because it is so quick and easy, and it is interesting because it highlights interaction design issues that have not previously been considered.

This paper primarily explores the importance of device states using so-called centrality measures, but to the extent that SNA is relevant to interaction programming — we argue it is, because centrality helps explain non-trivial user strategies — this paper opens up many further potential research areas, some of which will be raised in section 7 below.

2. MOTIVATING CASE STUDY

The following anonymized discussion is based on a real incident involving a real device.

A patient is kept anesthetized during a hospital operation, usually by using an automatic drug delivery system. A typical device will use a computer-controlled motor to squeeze a standard syringe loaded with an anesthetic drug. Before an operation, the syringe pump is programmed with the patient's weight and other details, and then the pump takes over the delivery of the drug. Many devices use sophisticated algorithms to deliver the right dose to maintain the patient in an appropriate level of anesthesia throughout the operation, thus saving the anesthetist continually adjusting the drug delivery rate manually.

During long operations it may be necessary to replace the syringe when it becomes empty. Observing long operations I have been surprised that the anesthetist may switch the syringe pump off and on again after changing the syringe, effectively rebooting it. I interviewed 5 anesthetists, and all thought this procedure routine. The anesthetists have learned or mislearned something about the devices. Why? Does their behavior make sense?

Switching the device off and on again has the consequence that the pump discards its patient model, and the anesthetist then has to monitor and manually adjust drug delivery after the change-over. In particular, they have to manually deliver a bolus, a burst of drug, to catch up with the loss of drug during the change-over. In a routine operation, the anesthetist will be easily able to manage (i.e., doing what they did before automated drug delivery systems) but in an emergency, their workload may be too high to cope without adverse consequences.

Such events are not normally reported, because the reporting systems are set up to report adverse clinical incidents. Typically, rebooting a device has no clinical consequences; it is a "normal" operating theater event, so it is not reported. Hospital management, regulatory agencies, manufacturers and designers are unlikely to get much feedback to improve future design.

It is thus critically important to understand why users (e.g., anesthetists) might choose to behave like this, to understand what features of devices or training make this an apparently rational strategy, and, if possible, to identify general design features that can lead to potentially counter-productive or dangerous behavior, or on the other hand, to identify design features that might avoid or guard against such counter-productive behavior.

3. DEFINITIONS AND PREVIOUS WORK

Generally graph theory, matrices or algebra are used to represent networks, and then social network analysis, SNA, is done routinely by programs designed for the purpose. It is beyond the scope and purpose of this paper to review the mathematics of SNA, but we will briefly outline some accepted concepts of importance (see [28] for the classic introduction; [9] for an excellent popular introduction; and [8] for a survey). The analysis is easy to automate and in practice no mathematical knowledge is required to use SNA — a designer could use an analysis tool to obtain the most important states in his design where these measures are just reported as summaries or numbers with the designer or analyst not needing to understand exactly how they were worked out from the design in question.

Graph	Social network	Device
vertex	actor	state
edge, arc	relation	transition

Table 1: Correspondence between property names of graphs, social networks and interactive devices.

One form of importance is to be "central" in the network.

The social networks represented by research papers and their relations have been widely explored. Using the relation "coauthors with" (rather than "cites" or "reads") and using HCI data from the period 1982–2003, Ben Shneiderman, William Buxton, Gavriel Salvendy, James Foley, Jakob Nielsen, John Carroll ... are (in decreasing order) the most central actors in the authoring community [13], as measured by betweenness centrality (a term defined below). Here, the analysis of a network of 22,887 papers with 23,624 authors automatically identifies people that would be widely recognized as leaders of their field — a very loaded psycho-social concept — even though the data was not analyzed for meaning, such as impact factors; in fact the discussion in [13] takes pains to distinguish between coauthorship and contribution to a field (e.g., somebody can contribute significantly to a conference without publishing in it). In short, the network has grown a structure that the actual or human importance of actors in it and the properties that SNA uses to measure importance correlate well.

The simplest measure of centrality is to count how many other states a given state in the device is related to. This is called the **degree** of the state. Although degree is very easy to measure, it does not always give a good measure of importance — in SNA for instance, an actor related to many unimportant actors may not be as important as one related to fewer but more important actors. In other words, degree is not weighted and does not look beyond immediate relations to the rest of the network structure.

Many social relations of interest in SNA are **undirected**, for example one may know that two terrorists communicate but it may not be possible to know or one may not be interested in which way the communication is directed. In contrast, all relations considered in this paper are **directed**, meaning that the relationship state A has with state B need not be the same as the relationship B has with A; this corresponds to the natural behavior of user actions changing the state of a device from one state *to* another — a directed relation. We therefore distinguish between in- and out-degrees, which separately count the relations to and from a state.

The in-degree center of many interactive devices will be the state *Off*, since it generally has an in-degree equal to the number of states, which is obviously maximal. A flashlight, however, typically has other states with equal in-degree: for example if it can only be on or off, then the state *On* will have equal in-degree to *Off*, and the in-degree center is the set {*On*,*Off*}.

States are related through other states, and we define the distance $d(A, B)$ between states A and B as the count of the least number of intermediate steps (transitions, button presses, etc) that can relate them. Note that $d(A, B)$ makes no assumptions on how we reached A, as this is implied by the definition of state. In other words,

$d(A, B)$ is the length of the shortest path from A to B that works under any circumstances. In particular, $d(A, A)$ is usually zero, and $d(A, B)$ is infinite if there is no way to get from A to B via any sequence of user actions. For a pushbutton device, $d(A, B)$ is simply the count of the number of buttons to press to get the device from state A to state B in the most efficient way.

In general, $d(A, B)$ need not be equal to $d(B, A)$. For example, $d(A, \mathit{Off})$ is typically 1, but returning to state A from *Off* generally requires several steps, so $d(\mathit{Off}, A) \neq d(A, \mathit{Off})$. Indeed, it is unusual for $d(A, B) = d(B, A)$ for all pairs of states, although if the device has an Undo button (as usually designed) then $d(A, B) = 1 \Leftrightarrow d(B, A) = 1$ for almost all states (e.g., *Off* excepted as a rule, since usually Undo will not undo an Off action).

For brevity in this paper we assume transitions are unweighted: this means that all user actions are "equal" but this is only a first approximation. In general we may wish to emphasize certain actions for the purposes of some analysis, or de-emphasize some actions whose behavior we are not for the moment interested in. For this paper, though, there is the additional issue that the more complex the analysis, such as weighting transitions permits, the easier it may be to contrive certain sorts of result; methodologically, by taking all transitions as unweighted we cannot "cheat" and get results that depend more on the weights than the SNA. That is, in this paper $d(A, B)$ is a simple count of transitions from A to B, rather than the time, cost or importance of that route. Section 7.3, Further Work, below, discusses weighting further.

Next, the **Sabidussi** measure of centrality for a state A counts how many related states A has at distance 1, then (because they are twice as far away) half the number at distance 2, then a third at distance 3, and so on. Equivalently, the Sabidussi measure for a state A is $\Sigma 1/d(A, B)$ summed over all other states $B \neq A$. In SNA, this measure of centrality is useful where relations *through* other actors are relevant, for instance as would occur if the relations involved financial dealing. A high Sabidussi measure for a state A means the device has been designed (accidentally or deliberately) so that the state A is a good place to start from to get to other states: for example, we would expect standby to have a high Sabidussi measure. Conversely, we define the **reverse Sabidussi** measure to measure how easy a state is to get *to*; for a state A the reverse Sabidussi measure is $\Sigma 1/d(B, A)$ summed over all other states $B \neq A$.

The **Jordan** definition of centrality also looks at the entire network. Jordan centrality can be intuitively understood by imagining the network drawn on a sheet of paper; the center in this sense is then simply the center of the drawing. Most efficient routes across the network will go through this center, and the center is also the best place to start any exploration of the network, because it lies closest to all of it. It turns out that most networks are not easily drawn on a sheet of paper and the intuitive sense of distance, that seems so obvious on paper, is meaningless if the network cannot be drawn correctly (put technically, the metric of some networks cannot be embedded in the Euclidean plane). The Jordan definition can be made more rigorous and apply to device graphs and all network structures, as follows.

Any particular state will have various minimum distances d to other states; hence the **eccentricity** of a state is defined to be the greatest d (i.e., greatest minimum distance) measured from that state to all other states. Generally, a graph of states will have various eccentricities, one for each state. The **radius** of the graph is the smallest eccentricity any state in the graph as a whole has; conversely the **diameter** is the largest eccentricity. (Note that eccentricity is a property of individual states, but radius and diameter are properties of graphs.)

With these preliminaries established, the Jordan center of the network is defined as those states whose eccentricity is equal to the radius of the network. In other words, the center states of a network are as close as possible to every other state as the particular graph allows.

Lastly we consider the **betweenness** notion of centrality, which is defined as follows. If a state C lies on a shortest path of relations between A and B then C is said to be between A and B. For all shortest paths from A to B (there may be many each of length $d(A, B)$) we count how often C is used, and then divide by the total number of shortest paths from A to B. This is a measure of the betweenness of C for A and B; it is the probability that C is used on a fair choice of an efficient route from A to B. To find the network betweenness for C, we consider every possible A and B in turn and calculate the individual betweennesses. It follows that states with a high betweenness are important because they are on many efficient connections.

There are very many other ideas in SNA. One further example of note is the sort of importance Google uses in its page rank algorithm [18]. Here, web pages are related to each other by the links that people (and computers programmed by people) choose to put in them to other pages; Google then uses a technical way to measure page importance, based on the network of links, and then ranks search results listing most important pages first.

3.1 Previous work

The preceding section summarized previous work in the SNA field. Previous work in networks (but not SNA) and relevant to interaction includes covers interactive systems as matrices [23], graphs [10], and as stochastic matrices [21]. These are all types of network directly amenable to SNA. Other authors have considered Statecharts [11] (which provides an exposition of the Citizen Quartz Multi-Alarm III), Petri nets [1] (which have also been used for direct manipulation user interfaces [15]) and other sorts of transition networks; these can all be converted to the type of networks needed. All such methods are routine to implement and are often used for combining rigorous analysis with simulations of devices that can be evaluated with users [20].

However a limitation with these methods is that there is no user model as such; they are simply device models. Another approach is to assume that the user has a strategy that helps them navigate the device network to use it. Information scent [19] is one such approach, and is based on an ecological model of predators seeking prey, users seeking information to "consume." A disadvantage of these approaches is that they rely on somehow rating the user's targets: what information do users want and how much do they want it? Although this can be done implicitly (e.g., by page ranking or other centrality measures), it is an approach much more suited to content-rich interaction, such as web navigation (where pages generally have metadata or are authored explicitly with known content and purpose) rather than device state interaction where there may be millions of states that cannot be specifically rated.

Benyon and Höök [12] introduced the idea that interacting with computers is navigation in information spaces; for the case of the web, the information space, the social space, and the interaction itself are very similar — a theme that has been taken up very productively in CSCW [13].

4. APPLYING SNA IN MEDICAL DEVICE CASE STUDIES

4.1 Infusion pumps

The Graseby 3400 is a typical syringe pump widely used in hospitals, in operating theaters and intensive care units. A realistic in-

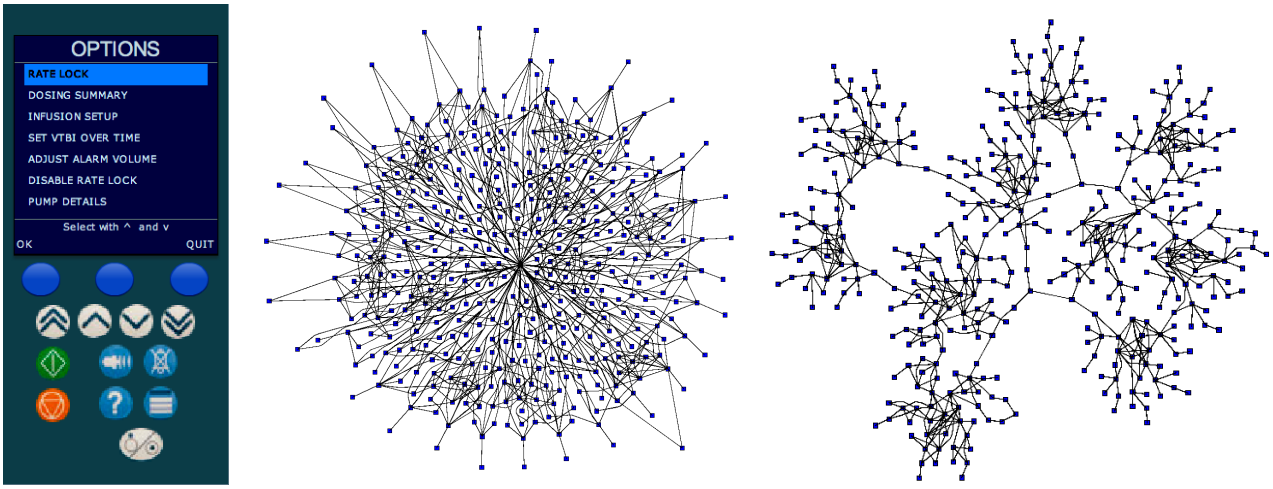

Figure 1: Simulation of the Alaris GP pump and two graphs generated automatically from it. The graph on the left is the full device, and the subgraph on the right has the *Off* state and all arcs connected to it deleted: the graphs (abstracting number values) have 415 and 414 states respectively. The subgraph on the right clearly shows clusters of states that correspond to various modes in the device. When viewed interactively, a user can zoom and scroll to explore the graph, but in this paper the diagrams are not meant to be studied in detail, but rather to illustrate the model complexity.

teractive Java simulation of (most of) it has been constructed [26], and this is used here to provide a network for SNA. The pump has a numeric keypad, and for the present SNA we choose to ignore all number entry (it would have made the network very large to distinguish each different number): the network therefore represents an abstract device, where we are not interested in the actual drug dosages as such or other numerical details but where we are interested in the overall structure of the interaction regardless of those details. This nicely mirrors how a user would think: how a device is used in general is a separate concern from what numbers are used for any specific patient. However, we will not in this way get any insights into number entry (although from other types of analysis we already know this is a problematic area [26]).

We find that *Off* is the most between state, thus showing that rebooting the device is a sensible procedure: it gets the device to an important central state. *Off* is also a top-rated Sabidussi center, so by going through *Off* is easier for the anesthetist to do anything else they want to do, and this is evidently what anesthetists have learnt from their use of the device (and other devices with similar centers). Unfortunately, going through this center obviously switches the device off, and for many devices pressing Off will result in the device discarding its patient model, as described in the introduction to this paper.

Interestingly, after *Off*, *Standby* is the next-most between center for this device. *Standby* (along with *Review settings*) is the Jordan center. The "near miss" of *Standby* to be the betweenness center could be taken to suggest that the design of the Graseby could easily be improved by reducing the relative betweenness of *Off*.

Improving this design now, one that is already in production, is infeasible; it would entail a product recall, retraining, and the dangers of two variations of a device in the field (and variations that differ in critical details). Instead, note that SNA could have been used *before* device deployment, and it would have uncovered the same design issues. In other words, SNA has not only shed light on user behavior, but also (in hindsight from the case study) shown itself to be a useful addition to the designer's repertoire.

Clearly a designer should identify central states, and then ensure that the sensible paths to those states do not discard patient information. For example, there ought to be an easier way of getting to *Standby* rather than via *Off*. Perhaps such devices should have a button called Ready or Standby that transitions to a central state (in fact, *Standby*) without discarding patient data or other internal state information. (This is a different concept than current On buttons support; typically pressing On does nothing to a device that is already *On* — it does not get a device to any standby state if it is not already *Off*.)

Also, the SNA of the Graseby 3400 suggests that designers need to be clear which centers have significance for the tasks the devices are intended for. A mixture of empirical evidence (such as that described in the introduction) and conventional usability investigations will be needed.

We have also analysed the Alaris GP infusion pump using SNA. A very detailed network model for the Alaris pump was obtained using techniques described in §6.1 (see figure 1). The states *Off*, *Dosing Setup* and *Set Rate* are the most between centers of the device; out of the 1,679 transitions, 363 are towards the state *Off*, for example. (When the pump is set to infusing, it cannot be switched off without first stopping the infusion.) In general, it was easiest to get to the *Off* state and easiest to get from the *Set Rate* state to any other states.

4.2 Ambulatory infusion pump

The Graseby 9500 is an ambulatory infusion pump, an automatic pump, battery-powered and small enough for a patient to wear and walk around while using it. Ambulatory pumps are useful in such situations as pain management in childbirth. Typically they would be used for providing a pain killer into the epidural region of the spine.

The model for the Graseby 9500 is taken from [10]. Note that although [10] did not describe SNA (which is the contribution of the present paper), it did discuss the use of scale free networks in interactive device analysis. An abstract network model of the Graseby 9500 ambulatory pump has 54 states, so it is a small but non-trivial device; see figure 2. (The level of abstraction is important; this model treats all numbers as equivalent, so, for example,

number entry is performed in a single state, and numeric keys are not identified in transitions.)

Both the Jordan and Sabidussi center of this device is standby, and the center based on out-degree is the two states *Standby* and *Reset*. Measured on betweenness, the center is standby, but then (in decreasing order of betweenness): *Continuous infusion*, *Reset* then *PCEA BOLUS* (patient-controlled epidural analgesia).

In normal use by a patient, the device will be in continuous infusion, so this is an obvious central state. *PCEA bolus* is the state the patient will put the device into to gain some extra painkiller. The idea is that overall the drug dose is lower, but extra drug can be provided as and when the patient chooses. Clearly, *PCEA bolus* is a sensible central state.

4.3 Design for emergency

The case studies above focussed on how users behave given various devices. We conjecture some parts of a device are "central" to the user experience, or important in other ways, and a designer should identify such centers of concern and reason about them; such an approach gives insights to and enlightens the behavior described in the case study. Equally interesting is using SNA to focus on how devices should behave, given that we know that certain features in the task domain are important — and therefore should have a close relation with the centers SNA identifies.

Many interactive devices have an emergency state that the user must be able to reach quickly under appropriate circumstances. Car brakes, fire extinguishers and resuscitation devices are all examples. In the previous case studies in this paper, important states are ones that a user goes through often (betweenness) or gets from easily (e.g., standby). Here, for managing emergencies, we want important states to be easy to get to. We can achieve this by designing important states appropriately, and by using SNA to assure ourselves that a design is easier to use in the intended sense.

The getting-to importance can be estimated by in-degree. Unlike out-degree (which is necessarily limited by the number of buttons), in-degree is only limited by the number of states, which can be much higher. Thus in-degree is a more discriminating measure.

A modern hospital bed is an example of a device with an emergency state. In normal use, various buttons can be used to adjust the posture of the bed; a patient may need to be flat with elevated legs or sitting up. Some beds can adjust matress hardness and have very sophisticated movements to change the patient posture continually to minimize the risk of bed sores. Such beds typically have control panels with upwards of ten buttons: they are non-trivial interactive devices. Yet however flexible the bed design is, if the patient requires cardiac resuscitation (CPR) it is crucial for the bed to harden and flatten quickly. The CPR mode should be top by in-degree and by the reverse Sabidussi measure.

The Graseby 9500, while having no "emergency mode" as such, illustrates the difference between the forward and reverse Sabidussi measures. The most central to-state is *Infusion*, and the most central from-state is *Standby*. In other words, the Graseby 9500 has been designed so that it is easiest to get anywhere from *Standby*, and easiest to get it to *Infuse*, which is the most common state for the device to be in when it is in use.

5. APPLYING SNA IN CONTRASTING DOMAINS

It would be a sufficient contribution to make headway in medical device design alone, but we now show that SNA has broader applications. The case studies briefly described below demonstrate how readily SNA analysis can be performed and identifies interesting features in user interfaces from different domains.

In the case studies in this section, we will repeatedly make the point that SNA insights and other design properties (such as affordances) should be mutually consistent, or their inconsistencies should be justifiable in terms of the user's task or the domain, or that the design should be (or should have been) modified to achieve that consistency.

The following subsections briefly describe a wide range of devices from a variety of domains. Some are well designed, some have problems. SNA provides accurate insights into design in these very different areas.

5.1 Personal video recorder

The JVC HRD580EK is a typical video-cassette recorder. Although VCRs are passé in terms of their consumer-appeal, the HRD is basically similar to and at the same level of complexity as current DVD, MP3 players etc. The advantage for this paper of using the HRD as an example is that it is a real device and that its definition has been explored extensively elsewhere [25]; it therefore provides a useful benchmark.

The two states most central according the Sabidussi metric are *Pause recording but stop in 90 minutes* and *Pause recording but stop in 120 minutes*. These two states are the places where a user would find it easiest to do anything, yet they aren't particularly useful states. The same two states are also highly rated by the betweenness measure.

The most central states, by the Jordan definition, are: *Stop recording in 90 minutes*; *Stop recording in 120 minutes*; *Pause recording but stop in 90 minutes*; and *Pause recording but stop in 120 minutes*. In other words, under three different definitions of importance, little-used and somewhat obscure states are central. The device has been badly designed in the sense that more useful or frequently used states should have been central.

By the out-degree definition of centrality, *On with tape in* (i.e., the standby mode) is the most central state. This looks like a sensible choice of central state, but conversely it means that in all other states relatively few buttons work (because the out-degree of all other states is less). In other words, most buttons on this device are highly state-dependent, and usually don't do anything. This is counter-intuitive, since many buttons (e.g., Pause, Rewind, etc) cannot work in the standby mode, so there is a reason to think that standby would not be a central state by out-degree! The evidence is that this device is badly designed.

5.2 Digital multimeter

The Fluke 114 is a typical digital multimeter, with a user interface that is a mixture of buttons and a knob. The knob can be turned into 7 positions, corresponding with the main modes of the device (e.g., being off or measuring volts, ohms, etc). The network of the Fluke has 425 states and 4,250 edges (there are 10 possible actions in each state).

The affordance of the knob makes this an interesting device to study. For example, one cannot physically move from knob position 1 to knob position 3 without going through knob position 2 (which happens to switch the device off).

The most central state using the betweenness measure is *MilliVolts DC* (followed closely by *MilliVolts DC with the backlight on*), yet milliVolts is the mode achieved when the knob is in position 5; there are four modes anticlockwise from it and only two modes clockwise from it. In other words, *MilliVolts* is a center of the abstract network, but not of the physical user interface, so if all states are used (about) equally, the user will be doing more knob-turning than necessary. Arguably the physical design for the central knob position should have better-matched the SNA center of the network.

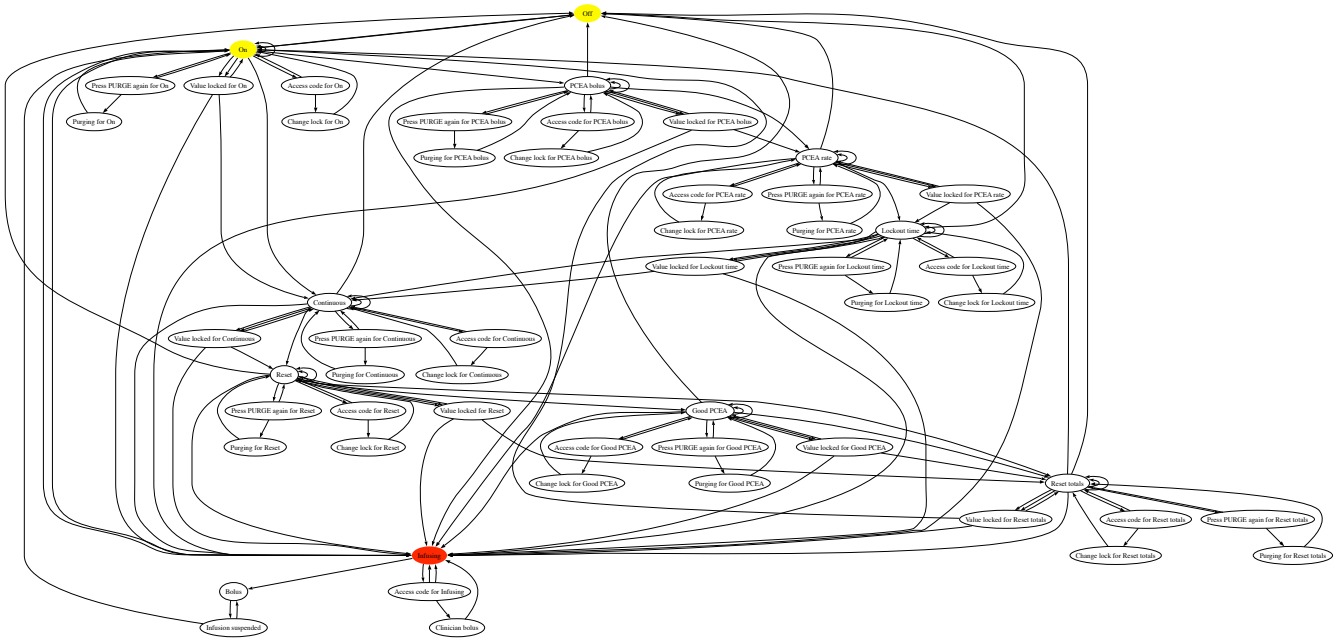

Figure 2: The Graseby 9500 graph used in this paper. The network was constructed automatically from a detailed Java simulation of the Graseby 9500, but in general a manufacturer or device developer would use an actual implementation, not a reverse-engineered simulation. As with figure 1, the graph has had to be reproduced for this paper at too small a scale to be useful other than to give an indication of the complexity of the model.

5.3 Wrist watch

The Accurist FS00 type chronograph is described in its user manual as a very simple 3 state device (as with the Graseby 3400 discussed above, the underlying numbers, here times, are ignored in this SNA). The different measures of centrality identify different states, but *Time measurement* is common to all measures — which is what a chronograph is for. For such a simple device it might have been tempting to design it as a complete graph, in which all states would have been equal, with no center. Evidently, the FS00 is well designed in this respect.

5.4 Cordless drill

A cordless drill is a simple interactive device compared to typical microprocessor-based systems. Many of the constraints of interaction are imposed not by program but by physical design. Nevertheless drills have interesting user interfaces: for example, most drills have interlocks so they cannot be changed from running forwards to running in reverse immediately without going through the intermediate state of switching off.

The de Walt DC925 is a three-speed drill (i.e., it has gears with positions 1, 2, and 3) with a 26-position torque clutch, a drill mode, and a hammer mode; represented as a state network (ignoring motor speed) it has 360 states, so it is non-trivial for our purposes (we take this model from [25]). By Sabidussi, Jordan and betweenness centrality measures, the center is the two states:

- *Locked, clutch set to 13, gears set to 2* and
- *Locked, clutch set to 12, gears set to 2*.

This is indeed what one would expect from experience of the domain or from a visual inspection of the device.

Out-degree is not a good indicator for this device, however, as in many states the user can do equally many things. In any gear 1 or gear 3 state, the user can only stay in that gear or move to a gear 2 state, but in a gear 2 state, the user can change to gear 1 or gear 3 — they have more options; thus all maximal out-degree states are where the device is in gear 2. (We discussed the alternative use of in-degree in the emergency example above.)

In summary, everything that SNA is telling us about this device is consistent with its obvious physical affordances. The device appears to be well designed for efficient interaction.

6. INTERACTIVE DEMONSTRATIONS

In many of the examples above, we repeatedly said the designer should have done something to improve the design. It would appear crucial, then, that designers are supported by tools that can do SNA efficiently, otherwise designers will be unaware of potential improvements, and any design changes may not lead to the desired outcomes. Fortunately there are many SNA tools [28] which can be used, as well as analytic graph visualizers/editors: visone (visone.info) and yEd (www.yworks.com) being typical examples. However, understandably, these general purpose tools do not view SNA as a component part of interactive systems design.

The book *Press On* [25] proposes a simple interactive device design framework based on JSON, the JavaScript Object Notation. The book has a web site, mitpress.com/presson, that allows a selection of existing and new devices to be analyzed (see under Resources > JSON). For the purposes of the present paper, this web site has been extended to include some SNA measures (e.g., including using [3]), and it can be used to directly confirm all the specific findings reported in this paper.

JSON as used in [25, *op cit*] is an explicit model of finite state specifications of interactive devices, and therefore to use it assumes there is already has an explicit model of the device to be analyzed. This approach suits the pedagogic goals of *Press On*, but does not suit most development work, which may wish to be more flexible than an explicit JSON framework readily supports. The next sec-

tion discusses how a network model can be obtained from a conventional program.

6.1 Technical issues: obtaining a network model

Finite state machines are low-level and unnatural to program. Instead of using a finite state specification to design, run or analyze a device, it can be done the other way round: a finite state network model can be derived automatically from an existing ordinary device program. This approach, which we now briefly describe, does not require any problematic "tooling up" in order to be used; a manufacturer can continue to program and implement interactive devices however they already do so (in anything from Flash, JavaFX to C#), and the network is then derived from those programs automatically and in a straight forward way.

We assume that the device implementation can be executed not just by the user (e.g., by pressing buttons) but also programmatically. We simply need to know the set of user actions that are permitted and to be able to generate events in the program that correspond to those actions: this can be defined explicitly or obtained by reflection. If we do not know how to do this, it is arguable the implementation is too obscure for a user, certainly too obscure to serve a role in technical authoring of user manuals or training material.

With a user-event interface, a simple network-generation tool can then "run" the user interface just as a user would. Additionally, the generator must be able to identify states. Again, if we cannot specify how a generator can identify states, it is very unlikely we know what states a user should recognize. Typically the identification can be based directly on the state of device indicators (e.g., if the On indicator is not lit, the device is in the off state). The actual device states are probably at too fine a level of detail to be of interest), and the generator must use an abstract model of states: for example, for analysis purposes we may only be interested in a single abstract state "entering a number" rather than all the possible numbers that might be entered.

These requirements define the components of a simple API that any existing interactive program should be able to implement easily. With the API, a simple state space search can explore the state space and record it for network analysis. Finally, the generator will be more efficient if there is a "state space jump" so it can transition the device to any required state — otherwise the state space search has to be depth first, which requires an Undo or equivalent, or based on Chinese postman tours [22]. A state space jump feature is not normally implemented in user interfaces but, as before, an inability to implement one would raise questions about the design of the system. One might wish to use SNA for part of a device but not all of it: for example, in a complex application such as a word processor, one might build a network to analyze the dialog box interactions in isolation, say, and ignore the text editing features altogether.

For many devices there will not be a straight-forward relation between the device state space and the state space accessed through the API; this is for exactly the same reason that a device implementation may have much finer notions of state than a user needs to be concerned with. For example, many devices track real time to the second, but although this technically increases the state space by a factor of thousands, the user for almost any task is uninterested in the actual time. The API would therefore conceal this part of the state space implementation from the generator. As before, if we cannot decide what to encapsulate in the API, then it is unlikely that the user interface has been properly defined.

6.2 Worked example

The digital multimeter model used above (§5.2) to illustrate SNA properties was created from an ordinary interactive program that provides an accurate simulation of the Fluke 114 DMM. The program was then extended to include a generator, as described above. The generator created a 425-state network directly. Importantly, the original device program was written in an unrestricted way, with no concessions to SNA or any other form of analysis. A full description and conventional evaluation of the multimeter is provided in [24].

6.3 Quality control

With an API defined, a generator can construct a network that specifies the device. Defining the API introduces a useful level of rigor in the device implementation — it is possible that for complex devices or domains there may be several APIs each delivering different views of the device.

It is possible that the choices made in the API are not consistent with the actual device implementation, whether or not they are consistent with the device specification. The divergence of an implementation from a specification is an important design problem to identify.

The generator can help quality control here simply by checking the consistency of the network it creates against the API. For example, having derived a network, the network can be checked for basic properties such as strong connectivity. Then the network can be walked concurrently with the device; any discrepancies are errors. The walk can be done systematically [22] or randomly, which may help identify less systematic errors. The text *Press On* [25] gives many such suggestions.

7. FURTHER WORK

SNA opens up many promising avenues for further investigation; we list here a few possibilities (which to consider in depth would take us beyond the scale and scope of a single paper).

7.1 Clustering

This paper has shown the productivity of SNA but only identified single states; SNA can also identify structure, such as clustering [4, 28]. Clustering could have many uses: for example, many user interfaces, such as mobile phone menu trees, cluster states, so the state clustering as measured by SNA should correspond to the user interface's clustering.

7.2 Usage networks

The cases we have considered are all based on a static network that defines the total behavior of the interactive device. A different approach would be to record the behavior of a user operating a device (either experimentally or by using black boxes). Now the network is an actual user's network, and it can then be used the same way to determine what that user effectively rates as important or unimportant. Such analysis might result, for instance, in improved user manuals and training (to improve the match between designer importance and user importance) or, after a device is in the field, to prioritize user retraining.

7.3 Weighted and special relations

We used unweighted relations throughout for simplicity. This allows us to measure user activity (using d) but more generally, we may want weights such as the time a user takes to achieve goals, or how much it costs. This is a generalization that is readily handled by SNA, but is beyond the scope of this paper.

The SNA used throughout this paper has assumed that relations are homogeneous. In reality, devices may have different sorts of user action that should be considered differently; pressing Undo is

the most obvious such action, as well as forms of commit or other actions that are not reversible (such as "pulling a gun trigger").

In our discussion of medical devices, the SNA did not distinguish between actions that have a clinical effect and those that do not. For example, while a device is infusing it is generally possible to review its settings without interrupting the infusion; this part of the network clearly has a different (clinical) status to the parts of the network that change dosages or stop infusions.

We did not distinguish the different sorts of user a device can have. The Graseby 9500 (the ambulatory pump) is designed to be used by a patient, a nurse, and a supervisor, and it has codes to lock out parts of the interface from unauthorized users. The patient has limited control of the device; for example, how many times the bolus button is pressed, it will be constrained to only deliver a limited amount of drug in a given time interval. The supervisor can change the limits, and the nurse can probably only set the baseline rate within other limits set by the supervisor. Other devices may have technician controls that are for servicing or for specializing the device to particular medical specialities. These are all clearly crucial design details for the domain.

On the other hand, the great merit of SNA as used in this paper has been that design insights were obtained from no domain knowledge being built into the network. This makes SNA useful for early design stages; further work should obviously explore the contribution SNA can make at later design stages, when empirical and other information is available about use, including the type and significance of user actions.

7.4 Global centrality

This paper has concentrated on the measures and uses of centrality for individual states in a network. For a designer, this may be problematic, as potentially every vertex needs to be analyzed before there is empirical evidence to focus the analysis more efficiently. There are global measures of centrality [4], and if centrality is important for a design, then iterative design could first aim to increase a global measure, rather than searching for particular states when the states that are important for users have not yet been identified.

7.5 Why or which devices are easy to use

There has been considerable interest in SNA research to determine why actors can find efficient routes from actor to actor in a social network [16], the "small world" problem. This is analogous to why users can efficiently find (if they can) the functionality they wish to use. However, research such as [16] has been based on networks embedded in Euclidean spaces, and the generalization to interactive systems networks such as this paper introduced is a new topic.

7.6 Theoretical analysis

The devices explored in this paper are conventional pushbutton style user interfaces. Few such interfaces have undo. It is possible that undo would systematically change SNA properties — although it is possible that undo makes (or should make) a uniform change to the network that cancels out in most SNA properties; for example, undo is unlikely to change any measure of centrality. It is clear that an opportunity for future work would be to explore styles of interaction, such as undo, cancel, escape. . . , theoretically rather than by analysis of the networks of particular devices.

7.7 Evaluation

SNA is clearly part of the armory of the interaction programmer and will certainly highlight design issues, such as we have discussed throughout this paper. However it might be argued that SNA has not been evaluated for application in HCI. For example, it is possible that for some application (say, a security device) or for some feature (say, undo) that SNA measures such as centrality do not correlate with central concepts for the user. Given the very general and broad foundations of SNA, it would indeed be very exciting to make such a discovery: it would either help refine SNA or, more likely, draw a clear distinction in HCI about types of interaction or types of feature. HCI is currently very lacking in any theoretical structure to the discipline.

8. CONCLUSIONS

This paper has introduced and explored a new subfield of human-computer interaction research, namely applying SNA (social network analysis) theory to interaction. Although originally motivated by a usability question in interactive medical devices, we used the techniques to analyze a wide range of existing systems and gained interesting insights into their design and use. As well as providing insight into use and possible use of complex systems, social network analysis has many further advantages, including:

SNA uses network properties to draw rigorous conclusions about complex relations that would not easily be recognized, if at all, without it. Interactive systems can be expressed as relational networks, and hence SNA can in principle identify illuminating properties and design issues of interactive systems.

Richness This paper has not had space to explore anything but a token part of SNA; but what we have explored is very promising. The richness, rigor and diversity of SNA will surely provide many opportunities for further research. Indeed, this paper may create a two-way flow of ideas: interactive device networks are a different sort of network than sociologists typically study, and insights from interface design may stimulate social analysis too.

Familiarity to CHI community The basic ideas of SNA are widely used and familiar in the CHI community, but for social rather than interaction analysis (see for example [13]). As this paper shows, extension to interaction is easy and productive, and the community's familiarity suggests deeper results will be found.

Proactive This paper gave examples of how SNA explains user behavior retrospectively based on interactive system networks. It follows that designers, manufacturers and developers should use SNA proactively to anticipate user behavior, and to ensure that the network properties identified are conformant with usage requirements. The examples given in this paper suggest that non-conformance could have serious (but avoidable) consequences in safety critical areas.

Established foundations SNA is an established discipline with rigorous and well-defined foundations. This paper has shown how a well-defined approach can be applied in user interface design; it therefore "imports" existing skills and ideas unchanged. There is already a huge literature that can be drawn on, and many further insights going beyond this preliminary paper's contributions can be confidently anticipated.

Scientific credibility SNA is a rigorous approach to studying networks of social significance; more generally network analysis [4] is the underlying mathematics that applies to any network. The concepts and measures this paper used, such as network centrality, are scientifically credible.

Reproducibility SNA is a method that relies on the existence of networks, and networks can be easily shared in the research community. This paper used a range of examples, all of which are available on the web sites mitpress.com/presson. Readers of this paper trying to understand, duplicate or extend this work can start with the same data and programs, and can check them or devise and evaluate new methods with them.

Incrementality The SNA approach can be applied in an existing product design and manufacturing process; although new insights

can be obtained, it is not necessary to start over to use the ideas. Indeed, it was gratifying that the SNA examples provided in this paper could so easily be built on a range of earlier interaction programming work that had not anticipated it.

The main practical impediment to research in this field remains gaining access to appropriate networks of device specifications and of user behavior. Most devices are proprietary, and their networks can only be obtained after a lengthy reverse engineering process. (Networks that are not obtained this way are subject to criticisms of not being realistic.) It is arguable that medical device designs should be subject to rigorous regulatory approval, and this must require access to their networks. It may be hoped, then, that advances in manufacturing processes, regulation and interaction research can go hand in hand.

The *Press On* web site has been extended for this paper to handle device networks in various standard formats (e.g., XML, JSON, Dot, Mathematica) and to provide SNA analyses of them, or for other devices if their specifications are uploaded. All algorithms referred to in this paper are available on the site and are available in JavaScript.

Acknowledgments

This work was supported by UK Engineering and Physical Sciences Research Council grants [EP/F020031, EP/F059116]. The author is a Royal Society-Leverhulme Trust Senior Research Fellow and acknowledges this generous support. George Buchanan, Paul Cairns, Jeremy Gow and Michael Harrison provided invaluable feedback on earlier versions of this paper.

9. REFERENCES

[1] Bastide, R., and Philippe A. Palanque, P. A. Petri net objects for the design, validation and prototyping of user-driven interfaces, Proc. IFIP TC13 Third Interational Conference on Human-Computer Interaction, 625–631, 1990.

[2] Bonato, A. *A Course on the Web Graph*, Graduate Studies in Mathematics, 89, American Mathematical Society, 2008.

[3] Brandes, U. A Faster Algorithm for Betweenness Centrality, Journal of Mathematical Sociology, 25(2), 163–177, 2001.

[4] Brandes, U., and Erlebach, T. (eds.) *Network Analysis: Methodological Foundations*, Lecture Notes in Computer Science, 3418, Springer-Verlag, 2005.

[5] Cairns, P., and Cox, A. (eds.) *Research Methods for Human-Computer Interaction*, Cambridge University Press, 2008.

[6] Card, S. K., Pirolli, P., and Mackinlay, J. D. The cost-of-knowledge characteristic function: display evaluation for direct-walk dynamic information visualizations. Proc. ACM CHI, 238-244, 1994.

[7] Carroll, J. M., Rosson, M. B. and Zhou, J. Collective Efficacy as a Measure of Community, Proc. ACM CHI 1–10, 2005.

[8] Chakrabarti, D. and Faloutsos, C. Graph mining: Laws, generators, and algorithms. ACM Computing Surveys 38(1):2, 2006.

[9] Devlin, K., and Lorden, G. *The Numbers Behind Numb3rs*, Penguin Books Ltd., London, England, 2007.

[10] Gow, J., Cairns, P., and Thimbleby, H. Automatic Critiques of Interface Modes, Proc. 12th International Workshop on Design, Specification and Verification of Interactive Systems, Springer Lecture Notes in Computer Science, 3941, 189–200, 2005.

[11] Harel, D. Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming, 8, 231–274, 1987.

[12] Höök, K. Benyon, D. and Munro, A. J. eds. *Designing Information Spaces: The Social Navigations Approach*, Springer, 2003.

[13] Horn, D. B., Finholt, T. A., Birnholtz, J. P., Motwani, D., and Jayaraman, S. Six Degrees of Jonathan Grudin: A Social Network Analysis of the Evolution and Impact of CSCW research, Proc. ACM Computer Supported Cooperative Work, CSCW'04, 582–591, 2004.

[14] Jackson, D. *Software Abstraction*, MIT Press, 2006.

[15] Keh, H. C., and Lewis, T. G. Direct-manipulation user interface modeling with high-level Petri nets, Proc. 19th ACM Computer Science Conference, 487–495, 1991.

[16] Kleinberg, J. The Small-World Phenomenon: An Algorithmic Perspective, Proceedings 32nd ACM Symposium on Theory of Computing, 163–170, 2000.

[17] Krebs, V. E. Uncloaking Terrorist Networks, First Monday, www.firstmonday.org/Issues/issue7_4/krebs, 2002.

[18] Langville, A. N. and Meyer, C. D. *Google's PageRank and Beyond*, Princeton University Press, Princeton, New Jersey, USA, 2006.

[19] Perolli, P. *Information Foraging Theory*, Oxford University Press, Oxford, England, 2007.

[20] Thimbleby, H. Analysis and Simulation of User Interfaces, Proc. Human Computer Interaction 2000, BCS Conference on Human-Computer Interaction, XIV, 221–237, 2000.

[21] Thimbleby, H., Cairns, P. and Jones, M. Usability Analysis with Markov Models, ACM Transactions on Computer-Human Interaction, 8(2), 99–132, 2001.

[22] Thimbleby, H. The Directed Chinese Postman Problem, Software—Practice & Experience, 33(11), 1081–1096, 2003.

[23] Thimbleby, H. User Interface Design with Matrix Algebra, *ACM Transactions on Computer-Human Interaction*, 11(2), pp181–236, 2004.

[24] Thimbleby, H. User-centered Methods are Insufficient for Safety Critical Systems, USAB'07—Usability & HCI for Medicine and Health Care, Springer Lecture Notes in Computer Science, 4799, pp1–20, 2007.

[25] Thimbleby, H. *Press On: Principles of Interaction Programming*. MIT Press, Boston, Mass. USA, 2007.

[26] Thimbleby, H. Interaction Walkthrough: Evaluation of Safety Critical Interactive Systems, Proc. 13th International Workshop on Design, Specification and Verification of Interactive Systems, Springer Lecture Notes in Computer Science, 4323, 52–66, 2007.

[27] Thimbleby, H. and Gow, J. Applying Graph Theory to Interaction Design, Proc. Engineering Interactive Systems 2007/DSVIS 2007, Springer Lecture Notes in Computer Science, 4940:501–518, 2008.

[28] Wasserman, S. and Faust, K. *Social Network Analysis*, Cambridge University Press, Cambridge, England, 1994.

[29] Wixon, D. Evaluating Usability Methods, *ACM Interactions*, 29–34, July/August, 2003.