

A Formal Systems Approach to Machine Capture, Representation and Use of Activity Context

Dan Diaper

DDD SYSTEMS, Bournemouth, BH11 8SZ, U.K.
ddiaper@ntlworld.com

Abstract

Communication bad spots on a commuter's train journey provide an example of context for the activities of a work system (user and devices) attempting to transfer large volumes of data. A simplified set theoretic based approach illustrates how the activities, their context, and device context capture can start to be analyzed. It is argued that context is not significantly harder to model than other, human aspects in a general systems analysis.

Introduction

This paper is the second of a pair prepared for the 2011 AAAI Activity Context Representation Workshop. The first paper, 'Defining and Representing Activity Context for Systems Analysis', summarizes the author's formal Simplified Set Theory (SST) approach and the use of his PentaVenn diagram. This second paper uses these in a modest, partially worked example to explore the contexts of an activity and how a formal approach can aid systems analysts by providing structured guidance.

The Train Connectivity Scenario

The scenario is that a user with one or more digital devices, in England, are on their regular commuter train. At several locations on the train journey, external network connections are poor or absent, perhaps because of tunnels/cuttings or the absence of cell phone masts. Being a regular commuter, the user has captured, i.e. learned from experience, the train route's environmental context, that there are predictable spots where external communication is often difficult or is impossible. Knowing this, the user adjusts what they do to work around these spots. In particular, the user avoids starting their frequent, large file external transfers when the train is approaching a communi-

cation bad spot. Concerning solutions, it is assumed there are too many remote sites using too many different platforms for it to be feasible to globally change systems software to allow interrupted data transfers to recommence.

The Unassisted User Scenario

Starting with the above scenario, first the user has to identify their problem, that too often they attempt long data transfers (e.g. 5-20 minutes) to a remote site that fail due to connectivity bad spots on their commuter train route. The user already knows when, if the train is on time, the bad spots will occur. Britain's trains are not noted for their punctuality and they are deemed on-time within a window of ten or so minutes, so just using the train timetable to predict bad spots is not feasible. Over a number of journeys, the user attempts to find journey landmarks that precede the bad spots by a few minutes ("a few" being less than the predicted time for file transfer). Some landmarks might be easy to identify, e.g. just after a station stop, whereas others may be just glimpsed through the window. Ideally, landmarks need to persist as, if the user is working, then they won't be looking out of the window, and even an easily identified landmark might be passed in seconds. Such persistence is rare, so what the user will do is choose a series of visual landmarks that precede a bad spot so that they come to recognize the area when they look out of the window before they perform a major remote site access.

Being human, most of the user's activity is perceptual and cognitive, learning then recognizing from memory where the train is relative to communication bad spots and estimating how long before these are likely to be reached. The user might compile a short list as an *aide memoire* or, being computer literate, they produce a map of their route and mark the bad spots and their approaches (Figure 1).

In Figure 1, the width of the bars represents the duration of the journey that will be affected by communication bad spots (black) and their approaches (grey), e.g. bad spots b and c have broader approach areas to represent that trains

are usually going faster here. As a memory aid, the users' hard copy map could be credit card sized, but, if its main purpose relates to using a laptop, then the map can be displayed on screen, which the user might do in a small window when the laptop is started on the train.

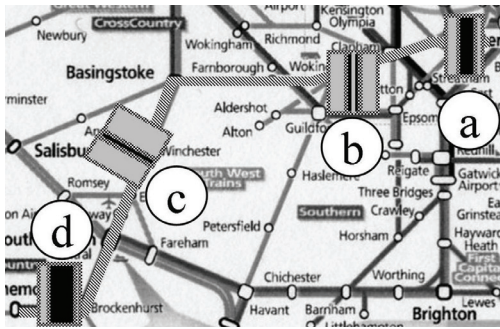


Figure 1. User's communication black spot map showing four spots (a-d) in black and their approach areas in grey.

This is as far as one user might go, but the project is to develop an application to support users' remote file handling and to remove the considerable cognitive effort that a user would have to make investigating and developing the map, and ameliorating the still considerable cognitive load associated with its use. Unassisted users will make errors, because they are busy working, so might forget the context completely or forget to look out of the window in time, or they may fail to recognize where they are as one landscape may look much like another in poor visibility conditions, e.g. heavy rain, fog, or dirty windows in low sunlight, and, of course, at night.

Systems Analysis

General systems analysis is different from many other approaches in HCI and software engineering because of its emphasis on the context in which activities occur. The contrast might be between a user-computer dialogue model and the more general systems model of an activity. In the latter, using the PentaVenn diagram and SST (Figure 2), the work system set $A=\{WS\}$ consists of a user and their digital devices (e.g. laptop, 'phone, PDA). The Application Domain set $B=\{AD\}$ is the remote site and there are two interface sets, $C=\{Input\}$, from the WS to AD, and $D=\{Output\}$, vice versa. The fifth set is $E=\{Context\}$.

An obvious first step is to consider normal communication between WS and AD. Analysts need to understand and define each intersection and the communicative relationships (paths) through the diagram, e.g. sending an email to a remote site, a five intersection path is:

$$(1) = WS \setminus *$$

'The user and device compose an email.' $\in (1)$

$$(7) = \{WS \cap Input\} \setminus *$$

'The user's device sends an email.' $\in (7)$

$$(3) = Input \setminus *$$

'External network transmits email packets.' $\in (3)$

$$(10) = \{AD \cap Input\} \setminus *$$

'The AD computer receives an email.' $\in (10)$

$$(2) = AD \setminus *$$

'The AD computer and user displays/reads email.' $\in (2)$

In the other direction the path is: $(2) \rightarrow (11) \rightarrow (4) \rightarrow (8) \rightarrow (1)$. This is for one-way communication and there are two other paths for the simultaneous case, e.g. a mobile 'phone call, where (13) , $\{Input \cap Output\} \setminus *$, replaces the middle intersections, either (3) or (4) . Already nine of the upper fifteen intersections have been populated, if not yet done well.

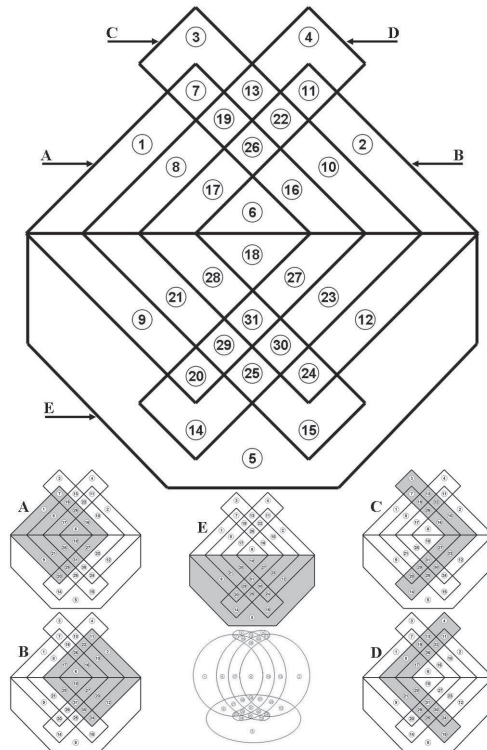


Figure 2. PentaVenn diagram of 5 sets (A-E) and with the 31 intersection areas numbered.

An easy way to populate the remaining six areas is to interpret them initially with respect to people's knowledge/beliefs of the data transmission activity:

$$(26) = \{WS \cap AD \cap Input \cap Output\} \setminus *$$

'The knowledge/beliefs the WS and AD users share of how their communication channels work.' $\in (26)$

This then allows a separation into knowledge/beliefs about either input or output, and either WS or AD users' separate ones, because, of course, to have shared beliefs must mean that they are also held by each individual, who may also hold different, unshared knowledge/beliefs as well:

$$(16) = \{WS \cap AD \cap Input\}^*$$

'The knowledge the WS and AD users share of how the input channel works.' \in (16)

$$(17) = \{WS \cap AD \cap Output\}^*$$

'The knowledge the WS and AD users share of how the output channel works.' \in (17)

$$(19) = \{WS \cap Input \cap Output\}^*$$

'The WS user's knowledge of how their communication channel works.' \in (19)

$$(22) = \{AD \cap Input \cap Output\}^*$$

'The AD user's knowledge of how their communication channel works.' \in (22)

It is an analyst option whether to then add similar elements to areas (7), (10), (11) and (8), but one may choose not to do so because: (a) these areas have already been defined with respect to device function, which isn't a sufficient reason for not adding alternative definitions; and (b) because most people know little about network connectivity and assume an input-output reciprocity, which may or may not be true depending on which technologies are used, so it might be decided to reset (16) and (17) to empty (\emptyset). Similarly, if for simplicity here, we assume IO reciprocity, then we may also set the one-way transmission outside the WS and AD intersections, (3) and (4), to empty.

Area (6) in the PentaVenn's top half has not been considered and it can represent other shared knowledge types:

$$(6) = \{WS \cap AD\}^*$$

'Shared domain knowledge of the WS and AD users.' \in (6)

This should automatically lead analysts to adding the individual WS and AD users' domain knowledge to (1) and (2), respectively.

The resulting model can now be broken down into several parts. First, the IO paths (1)→(7)→(13)→(10)→(2) and (2)→(11)→(13)→(8)→(1), with the note that (13) is external to both WS and AD. Second, users' knowledge of their two-way communication, both their individual knowledge, (1) and (19) and (2) and (22) for WS and AD, respectively, and the communication knowledge they share, (26). Third, the WS (1) and the AD (2) have individual domain knowledge and some that they share (6).

Finally, areas (7) and (11), and (8) and (10) concern devices and so don't involve the users and thus are not germane to the context initially captured by the WS user and which we wish their device to capture and use. Removing these four areas leaves just seven intersections in the top half of the PentaVenn diagram.

Adding Context

While the analysis so far has been cursory and heavily truncated, it is sufficient to start considering the focus of this paper, which is activity context, represented in the lower half of the PentaVenn diagram. Using the diagram's vertical symmetry in Figure 3, there are 14 areas of interest

in white and empty intersections, for current purposes, are in grey. Area (5) is also in grey as this, Context*, represents context irrelevant to the system of interest.

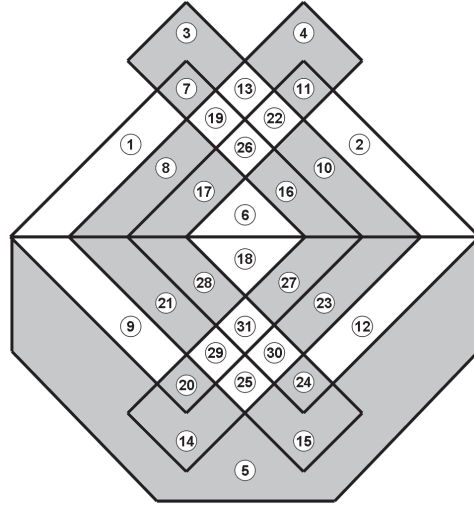


Figure 3. PentaVenn diagram of the current system of interest with the activities and their mirrored contexts in white.

It should be obvious that area (9) will contain much context information, but (29) is equally critical and can best be understood by starting with (25). As area (13) concerns communication external to WS and AD, so (25) is the network's context, i.e. the actual communication bad spots. While there may be locations, e.g. tunnels, where there is no external connectivity, it is common knowledge in the U.K. that different mobile networks have differing coverage, (as well as devices having differing capabilities to deal with weak signals). Consequently, to be more general, the context of "actual communication bad spots" must have additional context elements concerning different networks:

$$(25) = \{Input \cap Output \cap Context\}^*$$

'Complete communication bad spot locations' \in (25)

'Bad spot locations for different networks.' \in (25)

The context of the AD, (12), is that it is remote from the WS and the initial scenario's assumption was that it's non-mobile. Further analysis might consider when both WS and AD are mobile and either or both suffering bad spots.

$$(12) = \{AD \cap Context\}^*$$

'AD is geographically remote from WS.' \in (12)

'AD is non-mobile (temporary assumption.)' \in (12)

Staying on the AD side, area (30) concerns the AD's general bad spot context knowledge, i.e. independently of any particular WS. Area (31) is the shared history, if any, of bad spot problems between WS and AD, just as (18) is the shared history of the WS and AD working together, excluding the bad spot context separated out in (31):

$$(30) = \{AD \cap Input \cap Output \cap Context\}^*$$

'AD's general knowledge that bad communication spots can occur.' \in (31)

(31) = {WS∩AD∩Input∩Output∩Context}*

'WS and AD shared history of their bad spot communication problems.' ∈ (31)

(18) = {WS∩AD∩Context}*

'WS and AD shared history excluding their bad spot communication problems.' ∈ (31)

This leaves (1) and (29), where this section started, with the general WS context and the WS's bad spot context:

(9) = {WS∩Context}*

'The WS is on a train.' ∈ (9)

'The WS knows the train's route.' ∈ (9)

'The WS knows their current location.' ∈ (9)

'The train is on-time or not.' ∈ (9)

'File transmission time can be calculated.' ∈ (9)

(29) = {WS∩Input∩Output∩Context}*

'The WS knows the location of communication bad spots.' ∈ (29)

'Approaches to bad spots can be recognized.' ∈ (29)

'Time to bad spots can be calculated.' ∈ (29)

The (9) and (29) elements all arise directly from the original scenario. The first three of (9) are straightforward, specifying a thing in relation to the WS: train, route and location. The fourth is a shorthand for the thing 'train timetable', but, as discussed below, the critical thing is train progress. The fifth element might just be a binary decision by the user, e.g. "It's a big file to send.", which should trigger the user to check their location for approaching bad spots.

Area (29) deals with communication bad spot context. The critical WS issue is whether there is time to transfer data before reaching a bad spot. Thus, (29)'s last element uses all the other ones to calculate this. Failure of any of the other elements will mean this calculation fails, e.g. if the WS isn't on a train, then train tunnel bad spots will not be an issue and, obviously, the route and current location must be known by the WS. The train on-time element is about journey progress, e.g. if, unscheduled, the train's been stationary for some minutes, and particularly if this has happened several times, then the user might reasonably gamble that they have time to do a large data transfer with the AD, even though close to a bad spot.

While sufficient for illustration, a more detailed analysis, of specific devices, networks, etc. would generate more detailed elements. It should be recalled that all 15 upper half PentaVenn diagram areas can have assigned meaningful elements, so could have context elements in the lower half. The one-way communication to and from the WS and AD probably needs analysis and, to make the software general, that both WS and AD are mobile. This leads to scenarios where the AD is actively sending data, in contrast to the WS downloading, and whether the AD and WS negotiate before transmission, i.e. the AD needs to know if the WS has time before a bad spot to receive the data.

Context Capture

The set {Context} specifies context, whereas much of the original scenario is about how the user captured it. If the device is to (mostly) replace the user, then it should capture, and then use, equivalent context information. Context capture can be added as another set using SST, and although all five PentaVenn diagram's sets have been used, the upper half of the diagram can be reused to represent capture while keeping the lower half for specifying what is captured. This maintains Figure 3's white/grey symmetry and the convention adopted is that the upper half numbers will be preceded by CC, e.g. (CC1) = {WS∩ContextCapture}* as the mirror image of (9) = {WS∩Context}*.

Starting with (9) and (29), then the non-calculation elements specified can be rewritten:

(CC1) = {WS∩ContextCapture}*

'The device captures that the WS is on a train.' ∈ (CC1)

'The device has captured the train's route.' ∈ (CC1)

'The device captures the WS's current location.' ∈ (CC1)

'The device captures whether the train is on-time or not.' ∈ (CC1)

(CC19) = {WS∩Input∩Output∩ContextCapture}*

'The device has captured the location of communication bad spots.' ∈ (CC19)

'The device has captured the location of approaches to bad spots.' ∈ (CC19)

A difference between these elements is their use of "captures" versus "has captured". Both (CC19) elements must be captured before they become relevant, i.e. it's too late if a bad spot is reached before its location is captured and used. Necessarily, the (CC1) route element must be captured so as to relate this to the (CC19) elements.

Device intelligence and autonomy are major, related design issues for context capture. For example, 'How does the device know that it and the user, i.e. the WS, are on the train?' A uncontroversial design assumption is that the device will have a GPS system which the device can interrogate to place itself on the train's route. While this seems much simpler than what the unassisted user had to do, looking out of the train window and so forth, just geographical location, no matter how accurate, comes at a cost of other context information that the user has access to and the device does not. For example, the user is at their departure station and turns on their laptop device. The device knows it is in the station, but does not know that it is on a train, e.g. they may be on a platform seat or in the platform buffet, and if on a train, it may not be the expected one. A very simple context capturing device might check its location and capture its missing context by asking the user if they are on a train and, if so, which one. A slightly more sophisticated one might have programmed rules that check time and location and assumes it's the usual train until it

gets contrary information, e.g. the train is late departing the station.

At this level it is disputable whether this is really Artificial Intelligence (AI), but the advantage is that the device does not have to disturb the user. The device can then, invisible to the user, monitor route progress until external data transfer is attempted. This has to involve area (1), not context but the user and device's work, and transfer time must be calculated. Here capture is needed from the work system, WS*, and then passed to the context element: *'File transmission time can be calculated'* \in (9)

The calculation could be as crude as just exceeding a set file size, or it could be based on claimed network bit rate and a more advanced system might have context information about bit rates being achieved during the current or past activities near the locale. Since (19), (29) and (CC19) concern network knowledge, then elements concerning bit rates can be added to these as in the manner above.

Train routes and timetables might be easily captured by a device, perhaps as a semantic web application, which still leaves the problem of identifying *en route* communication bad spots. While a bad spots map might be available from the web, the context that different networks (and different devices) have differing bad spots has already been mentioned. This might lead to designs that involve the device learning, and that usually means quite serious AI.

At the moment, most symbolic AI systems are relatively poor at learning, i.e. substantially they are human programmed. In contrast, learning is the strength of neural networks, but currently at the expense of being computationally resource hungry, and/or often slow, and generally perceived as too exotically leading edge for most applications. On the other hand, and whatever AI used, a device that learned about its own bad spots and their approaches would be a superior design. It could even create its own training examples during journeys, without actually transferring files. Whether it could learn about delays (perhaps 20% of journeys), and even of trains going faster after delays, is probably at the edge of current AI capabilities.

Finally, if developers were to take the network connectivity context capture by AI design route, where a device continuously monitored its connectivity and learned from it to produce some form of stochastic, predictive map, then it might have much more general applications than just to train journeys, or even just to journeys by other means of transport. Just which is beyond the scope and concerns of this paper and so is left the imagination of designers.

Personal Observations

The scenario used in this paper was written as a AAAI Workshop discussion document several weeks before the paper was started. I'd just haphazardly generated scenarios using an informal user-computer interface model, which is

how the user's map (Figure 1) arose. That such a SatNav interface as I first imagined was replaced by other designs following the systems analysis above is one, generally rare, piece of concrete evidence of how doing an analysis can change, eventually, understanding the system of interest. What follows are a few, more subjective, observations on how using a formal systems modeling approach helped me. Overall, I am sure that having SST and the PentaVenn diagram made me think differently about the system.

What's invisible to readers of papers such as this one is the complexity of doing the analysis and, in particular, all the alternative models that were tried and rejected. Although I've used SST for a dozen years, it was primarily the PentaVenn diagram that I used when constructing models. My basic approach was to generate part of a scenario and write a note of it directly on the (colored) diagram. Then one can use the diagram's symmetry to investigate other intersection areas which might be involved. For example, thinking about two-way simultaneous communication (e.g. telephones), then this involves area (13) outside the WS and AD. However, if WS or AD use a conference 'phone that allows either microphone or speaker to be active, but not both simultaneously so as to prevent auditory feedback, then while this doesn't affect (13) it does affect areas where the WS and AD and their interfaces intersect, which ones depending on who is using the conference 'phone. Also, people adjust their behavior to such 'phones, although they may or may not be conscious of their adjustments. A major advantage of using the PentaVenn diagram was to keep me on track as to the WS and AD as the mobile and remote systems, respectively, and not to slip into a user-computer interface model.

While admittedly only poorly specified as text elements, adding the context and capture parts of the analysis was much easier than building the first, acceptable version of the upper half of the model. This ease was due to being able to exploit the diagram's symmetry so that context could be specified with respect to the non-context mirror elements. For example, if (13) is external two-way communication, then its mirror (25) must be the actual communication bad spots and this can then be contrasted with the WS user's knowledge of such bad spots (29).

The author has often commented (e.g. Diaper, 2001, 2004) that some "typographical" errors are inevitable with large (task) analyses. An advantage of a formal system is that these should be readily detected. Such was the case here, where I'd transposed two numbers on the PentaVenn diagram which were detected as soon as they were used.

Discussion and Conclusions

What can the author of these two papers hope to achieve? Also, what can't possibly be achieved? Starting with the

latter, negatives, to get them out of the way quickly, first there are several omissions, including the confessed lack of a literature review. Thus there is no discourse on the many ways that people have thought about context, which may be a bonus as the overall conclusion that the author has reached, as a direct consequence of working on these two papers, is that context is not especially more complex than many other things in systems models. The roles of methods in the software industries is much omitted and, entirely and unfortunately, also is any discussion of the advantages of formal methods, and their limitations, of course. The description of SST is very incomplete, for example, neither paper deals with set elements beyond their textual descriptions and there is no attempt at describing how SST and the PentaVenn diagram can be used as part of a method, although Diaper (2000) explicitly provides such a method (SST4SM – SST for Systems Modeling, later referred to as the SAM - Systems Analysis Method). This last omission may not be a problem in that one thing the author always knew about these papers was that they could not, and would not try, to convince people to use his methods and formalisms. At the very best, a few people might be sufficiently interested to contact the author and find out more.

Without denying other possible negatives, what these two papers do achieve are several very positive and, the author hopes, useful things. First, the analysis of the two examples in the pair of papers, no matter how cursory and incomplete, is a call for clarity of thought. The personal observations subsection above is important to combat the impression that such general systems analyses are straightforward, even for experts. Apart from the easy to make error of slipping into considering user-device dialogue scenarios when these are inappropriate, the logical completeness of an SST model and its PentaVenn diagrams forces a structured thinking approach on analysts, making them fill in other intersection areas than the ones they start with. This then encourages generating alternative, related scenarios and, importantly, a rationale for generating them.

In these two papers, the Dowell and Long (e.g. 1989) WS and AD model, and the author's extensions to this (Diaper, 2004; Diaper and Sanger, 2006), is important as illustrated are two genuinely different possible perspectives, the one where users and devices operate together in the WS to perform work in the AD, and the user-computer dialogue model where users and their devices are separate, in the WS and AD, respectively. Much is often made of the importance (and complexity therefore) of having multiple perspectives in systems analysis, but well formed examples are much rarer.

That complexity and context go together will be a consensus at the AAAI Workshop. This is an error, according to the conclusions reached by the author after preparing these two papers. Modeling context is not really different from modeling the other sorts of things and relationships in

human-computer systems. Indeed, in one sense context is easier to model because, as argued in the first paper, its effects are one-way as it can change other parts of a system but it will not itself be changed. If we are already modeling people, then we are already modeling the most complex parts of any system. Much of such modeling is of intangible things, of thought, memory, beliefs, plans, goals and, always, within social systems, accepting Wittgenstein's proposal that a private language is impossible.

Unlike many software engineering methods which tend to focus narrowly, e.g. on data flows, entities, objects, it is necessary for a general systems approach to be able to model anything. This is not a problem for general mathematical/logical systems as definition is the analysts' prerogative, e.g. in set theory, initially each set is defined however an analyst wishes. The most complex anything to be modeled is still people, and so modeling context will be most complex when it involves people rather than simpler systems such as the weather, or people's models of things, such as of their models of social or economic systems. These papers' message is therefore: '*Context? Do Not Panic.*', because context is like all the other complicated things that are already modeled in a general systems analysis approach.

There is, however, a final caveat, if analysts don't think of it, then it won't be in their analysis. This, of course, is true for any systems analysis approach. Even so, if people don't think of economic, business, social, cultural, ethical or religious contexts, then they won't be modeled and such contexts can be difficult to identify when their effects are pervasive, i.e. where the system is always affected by such a context. In contrast, problems encountered within WS and AD analyses usually makes systems components far more visible and provide a basis for how analysts define and articulate their systems models.

References

- Diaper, D. (2000) Hardening Soft Systems Methodology. in McDonald, S., Waern, Y. and Cockton, G. (Eds.) *People and Computers XIV*. 183-204. Springer.
- Diaper, D. (2001) Task Analysis for Knowledge Descriptions (TAKD): A Requiem for a Method. *Behaviour and Information Technology*. 20, 3, 199-212.
- Diaper, D. (2004) Understanding Task Analysis for Human-Computer Interaction. in Diaper, D. and Stanton, N.A. (Eds.) *The Handbook of Task Analysis for Human-Computer Interaction*. 5-47. Lawrence Erlbaum Associates.
- Diaper, D. and Sanger, C. (2006) Tasks For and Task In Human-Computer Interaction. *Interacting with Computers*, 18, 1, 117-138.
- Dowell, J. and Long, J. (1989) Towards a Conception for an Engineering Discipline of Human Factors. *Ergonomics*, 32, 11, 1513-1535.