Managing Process Change? Easy as Pi (and Petri)

Keith Harrison-Broninski

*There is general agreement amongst Business Process Management practitioners that a **formal foundation** for process support is needed, analogous to relational theory for databases. However, attempts to provide such a foundation are currently in a stalemate between proponents of two different theories: Petri nets and the pi-calculus. This paper has two aims.*

*The first aim of the paper is to try and move the debate onwards by showing how you can – and need to – use both abstractions in conjunction to model different aspects of those processes that are centered on human activity. In particular, each approach is used to represent a different aspect of **process management**. A different formal technique, the Z notation, is recommended for analysis of work activities themselves.*

The second aim of the paper is to deal head on with another Business Process Management issue, one of perhaps more general interest – process change. We show how the approach recommended for modeling of management activity allows change to the design and structure of a human-driven process to be safely controlled, even if it happens on a daily basis, and takes place from within the process itself.

There is a generic business problem that may seem rather abstract, but whose resolution is recognized by practitioners of process orientation as essential. How can we underpin business processes in a *formal* way?

Of course, any process modeling technique is formal, in the sense that it describes the methodical application of principles. However, in this paper we mean something very specific by *formal*: How can we apply the principles of *mathematics* and *logic* to the complex, multi-faceted, ever-changing flow of activity inside an organization? Solving this problem is important not only for theoretical, but for practical reasons.

Work support technologies have traditionally suffered from being complex and expensive to implement. Maintenance is even worse. Moreover, and just as importantly, how can you really know whether the systems you have put in place will do what you think they should do? Do they implement the diagrams you drew up at the start to represent processes? Even if they do that at the start, what happens when processes change – and those processes center on human activity, that we call *human-driven*, inevitably and constantly do change?

These issues are not just of concern to IT staff, but are visible at the board level – concerns such as feasibility, cost, resourcing, and operational quality, are fundamental to proper management of the enterprise. Moreover, with the wave of recent corporate scandals and consequent appearance of new regulations, *transparency* is a major concern for senior executives. For example, in the USA the Sarbanes-Oxley Act of 2002[i] places responsibility for financial accounting squarely on the shoulders of company board members, who may even face a custodial sentence if they do not get it right. Similar statutes are likely to appear in other parts of the world.

As a result, there has been increasing interest in the provision of a *formal foundation* for process analysis. In fact, the nature of this foundation – in particular, whether it should be graphical *Petri*

nets, or a more recent, algebraic approach to process description known as the *pi*-calculus – has been the topic of much study among leading process practitioners.[ii]

But why would such a foundation solve the problem of change in human-driven processes? How is this kind of stuff genuinely relevant? Surely, it is just something for those with brains the size of a planet to discuss at conferences and to write highbrow papers about.

To justify the importance of such theoretical work, and locate it with respect to industry initiatives, Robin Milner (inventor of the pi-calculus) and Ole Høgh Jensen write of the latest effort to provide a formal underpinning for computation (a graphical approach called *bigraphs*, that deals with the relationship between locality and connectivity – between *where you are* and *what you can access*):

"The long-term aim of this work is to provide a model of computation on a global scale, as represented by the Internet and the World Wide Web. The aim is not just to build a mathematical model in which we can analyze systems that already exist. Beyond that, we seek a theory to guide the specification, design, and programming of these systems, to guide future adaptations of them, and not to deteriorate when these adaptations are implemented. There is much talk of the vanishing ubiquitous computer of the future that will obtrude less and less visibly in our lives, but will pervade them more and more. Technology will enable us to create this. To speak crudely, we must make sure that we understand it before it vanishes.

This will only be achieved if we can reverse the typical order of events, in which design and implementation come first, modeling later (or never). For example, a programming language is rarely based thoroughly upon a theoretical model. This has inevitably meant that our initial understanding of designed systems is brittle, and deteriorates seriously as they are adapted. We believe that the only acceptable solution, in the long run, is for system designs to be expressed with the concepts and notations of a theory rich enough to admit all that the designers wish.

The arrival of ubiquitous mobile computing provides an opportunity for this, simply because it is new enough for its languages and implementation techniques not to be entrenched. Another reason is that concurrency theorists have anticipated mobility and have some structures to offer for new languages. Thus designers and analysts may come to speak the same tongue."[iii]

The reference to *mobility* at the end of this passage is crucial. Many people involved in the development of process support systems focus on the importance of allowing processes to change their definition *during their enactment*, via the exchange of information among participants about the process itself. In particular, the participants can use this information to change the connections within the process – the parties to an interaction, for example, or the systems that co-operate to supply a service.

The provision of mobility is a complex technical matter, bringing with it potential problems that are hard to predict. Hence, giving process support systems a formal mathematical basis makes sense – such a basis can then be used to deduce properties of the system's behavior, and help ensure not only that the properties will operate as intended, but also that no unforeseen security risks will be introduced. Just as importantly, if we do it right, a formal basis will help clarify how such advanced features as mobility can be managed – not just *permitted*, but also *monitored* and *controlled* simply and effectively.

However, supplying such a formal basis is harder than it seems. Milner and Jensen are saying above that, if computer systems are to be successfully provided with a formal basis, this basis must be introduced *during development of the* systems – not applied afterwards, papering over

the cracks where the systems do not quite match the theory. Deductions made from theory about a system that only partially matches it are going to be misleading, not reassuring. This means that you need to understand all the relevant mathematics before you can even start building a new form of process support system – you cannot just hire an academic to sort it out afterwards.

In many ways, the poster child for providing a mathematical foundation to computing is the relational database, now the main and ubiquitous storage method for enterprise data.

In the early days of data processing during the 1960s, data was stored in *flat-file* format: each piece of information was kept as a single record in a file. A file of customer orders, for instance, would contain customer information such as address and phone number *in every* record; no matter how many times the customer had previously dealt with the company, the information was repeated for every order record in the file.

This made for files that were unnecessarily large and took up lots of disk space (which was expensive at the time). Worse, such data was hard to maintain: What do you do if a customer changes his or her address? Data management was costly, complex, and time-consuming.

The problem was solved by the *relational model*" for data – which appeared *before* the software to implement it. In 1970, Dr. E. F. Codd published a paper entitled "A Relational Model of Data for Large Shared Data Banks," introducing a set of rules that eliminated the need to store redundant data. These rules were the starting point for relational database theory, that not only made money for companies such as Oracle and IBM, but also saved money for the rest of the world.

Many people feel that process support in the early 21st century is in a parallel situation to data processing in the 1960s: It is too costly, complex, and time-consuming. As a result, they feel that a similar formal underpinning is necessary to sort out the problems. Where should we look for such an underpinning?

As described above, the two main contenders to date are Petri nets and the pi-calculus:

- Petri nets are a graphical technique based on the movement of *tokens* from one *place* to another in a network. The connections between places, known as *transitions*, control the movement of the tokens. An example Petri net is shown below:

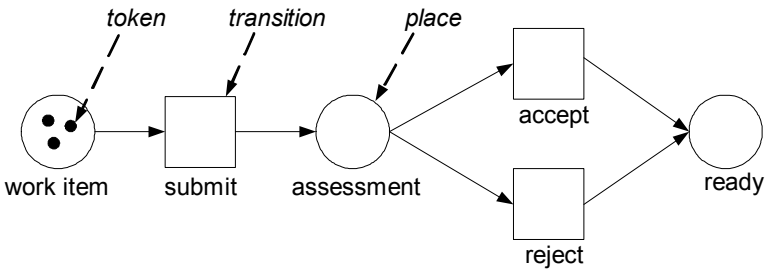

Figure 1. An example Petri net

- The pi-calculus is based on the notion that a computing system is composed of independent, communicating objects, known as *automata*. Things that happen inside these automata are termed *actions*, which move an automaton as a whole from one *state* to another. An automaton can send messages to another automaton via a channel connecting them. Each end of such a channel is known as a *port* – one end is considered to be for output (labeled with an overbar) and the other for input. A port can itself be sent from one automaton to another,

which enables the connections in the system to change during execution. The pi-calculus is an algebraic theory, but it is possible to depict automata graphically, as shown in

-
- Figure 22. : Example pi-calculus automata. Here we essentially see the same situation as depicted in Figure 11, but the upper automaton represents work *performance*, and the lower automaton work *assessment*:
-

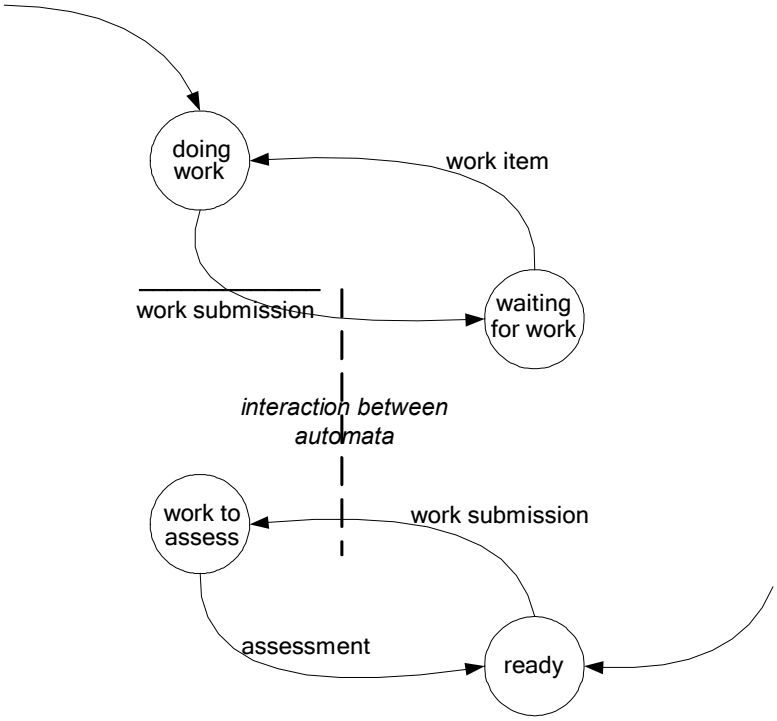

Figure 2. Example pi-calculus automata

Both Petri nets and the pi-calculus have inspired several variants that extend the original theories to add functionality or to cater to specialized situations. Hence, one approach taken to date is to seek specific variants of one or both that meet the needs of a process support system in terms of its formal underpinning.

These pages are not the place to take up this discussion in depth. However, it is possible that the theory of human-driven activity offers a new approach to formal description of processes, one that allows the two competing formalisms to be combined – perhaps in conjunction with a specialization of one or both theories. The essence of this approach is to

- Reconsider the basis on which formal theory has been applied to processes; and
- Recognize that, for such a complex problem, we may need not one formal technique, but several – and more than one way of applying them.

When a formal theory is used to describe a computer system, or any other real world activity, it is done so by creating an *abstraction*. This simply means that each object in the formal theory is identified with a specific sort of object in the real world. There will always be more objects in the real world than in the formal theory, so some real world objects will have to be left out. However, if you choose your abstraction wisely, this does not matter. In fact, this is the *strength* of abstraction. By simplifying the world, you can see more clearly what is going on and deduce

things that were not obvious beforehand – see the forest for the trees, in a way.
So, what abstractions have been used when applying formal theories to business processes? In general, it is always the same one. An activity in the theory (a Petri net *transition*, or pi-calculus *action*) is mapped straight onto a work activity in the real world. The real world activity may be a simple, atomic task (a calculation, for example) or something very complex (such as assembling a car from component parts), but either way, the principle is to identify something that takes place in the abstraction with some work that takes place in the real world.

What if we were to change this, and try a new abstraction? In particular, the principle of *separation of control* proposed for human-driven processes allows us to distinguish *management control* (day-to-day facilitation of human activity, carried out as part of the process itself – ongoing resourcing, monitoring, and process re-design) from *executive control* (exercise of authority over the process, via determination of its primary Roles, interactions, and deliverables).

Process evolution can be implemented under management control via the establishment of consensus among certain participants, based on *how they would proceed from now* on – We term such a consensus an *agreement*. People decide to do things a certain way in the future (which, we suggest, is best documented and shared via a Role Activity Diagram), and then they go away and do some work. After a while, the same cycle repeats: A group of current participants meet, decide how to evolve the process further, depict and share this agreement, and then go away and do some more work.

We propose that a way to move forward with the provisioning of human-driven process support with a formal basis is to map theoretical activities (Petri net transitions and pi-calculus actions), *not* onto real world work items – at least when the processes concerned are human-driven rather than machine-driven – but onto the means by which such processes are instantiated and changed. The process moves forward not in terms of tasks executed, but in terms of *how the organization of work evolves*.

This approach places mobility at the very heart of process theory. Process evolution is the building block, not a by-product, of formal process description.

In addition, it seems eminently possible to support this approach to formal description of processes with either Petri nets or the pi-calculus. More importantly, we need both, since one allows us to reason about *management control* and the other about *executive control*:

- Petri nets can be interpreted as not modeling the flow of activity, but modeling instead the activities of *management control*. A Petri net place represents a stable process definition. A Petri net transition depicts the distribution for approval or implementation of an agreement made by process participants on future process change, which potentially moves the running process from one definition to another, either to refine it or alter it. This is depicted below in Figure 343: . A Petri net showing management control:

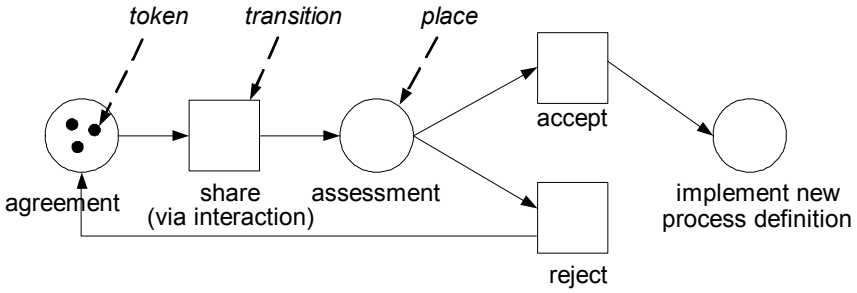

Figure 3. A Petri net showing management control

▪ A pi-calculus automaton can be taken to represent a self-contained process network – a group of Roles, co-operating via interactions and responsible both for management control and for the work itself. Pi-calculus actions equate to process changes; in a sense, a pi-calculus action summarizes the impact of the individual Petri net transitions described above (distribution, approval, and implementation of an agreement on change). This gives us a higher-level view of process change, abstracted away from the individual agreements made, in which ports show the fundamental mechanisms of *executive control* – how communication is effected between the executive sponsor who initiated a process and the lead Role of the process itself. Mobility in pi-calculus terms then equates to the implementation of executive control, and its transfer from one Role to another – for instance, when authority over a process is granted or delegated. Example automata are shown below in Figure 464.: Executive control depicted via the pi-calculus:

▪

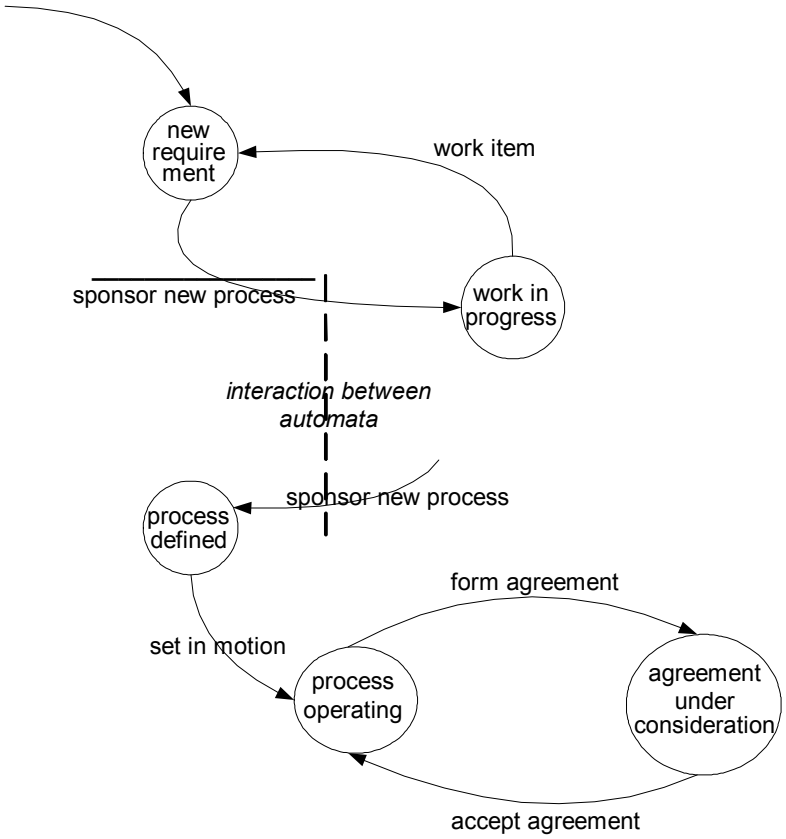

Figure 4. Executive control depicted via the pi-calculus

These abstractions are fundamentally different from any applied to date, and offer the chance to grab the bull by the horns – the ongoing management of processes that inevitably change. Change may be necessary on a frequent basis, under management control, or occasionally via intervention from above, under executive control. In both cases, we need a way of reasoning about change, if we are to deal with it properly.

This change of abstraction – from work activity to management activity – may offer a way forward that deals with critical issues, and, in so doing, resolve a dilemma by uniting two powerful forms of reasoning. But it also leaves a gap. How, with this approach, can we reason about the work activities themselves? Surely, this is important too. After all, a process is only there in the first place in order to *get things done.*

The problem to date has been one of terminology. Computer scientists have a tendency to think about human work activities as if they were steps carried out by a computer program. This leads to process descriptions that are framed in terms of *sequences of tasks*. Both Petri nets and the pi-calculus, for example, allow the history of a process to be examined in such a way. However appropriate this may be to automated or mechanistic processes, this is unfortunate when it comes to human processes, because the end result is that powerful and elegant mathematics are devoted to analyzing work in a way that is ultimately of little meaning. We do not think or act like machines, so what value is there in analyzing our behavior as if we were machines?

We need an approach to modeling work activities that is closer to the true basis on which we decide how, when, and what to do. It may not be possible to capture all this in a mathematical formalism, but that does not necessarily matter. As discussed, the virtue of an *abstraction* is that, by leaving some things out, general truths become clear. So what abstraction is the most useful for human activities?

We propose that such an abstraction should be based on *business rules*. In general, humans do something if and when it seems necessary, or just sensible – regardless of what they just did previously, or originally intended to do:

- On an ongoing basis, we re-evaluate the world around us, and make a judgment about what we should do next.
- Having done something, we and our supervisors then look at the results and make a decision about whether the work has value. If it does, we keep it; otherwise we discard it.

These aspects of human behavior correspond closely to the use of *preconditions* and *postconditions* that we propose for the application of Role Activity Diagrams to human-driven process modeling. A precondition *enables* an activity. If the precondition is true, the work can be carried out whenever desired. A postcondition *validates* an activity. If the postcondition is false on completion, all the work must be undone. Is there a similar mathematical formalism available?
A particularly appropriate approach is the language *Z*.[iv] Z was originally intended for formal descriptions of computer systems. It is based on set theory and mathematical logic, and it is generally used to help prove that high-integrity, safety-critical systems conform to their original specification. However, Z happens to provide exactly the foundation we need for the description of human activity.

In particular, while Z has the ability to describe task sequences, there is no need to *constrain* activities to follow one another in a particular order, using techniques of control flow (if-then-else tests, loops, and so on). Rather, you can use Z to describe for particular activities the specific conditions under which they may be executed, and how the state of a system is changed on their completion – *when you may do things, and what effect they will have*.

In Z, schemas are used to describe both static and dynamic aspects of a system.

The static aspects include
- the states it can occupy; and
- the invariant relationships that are maintained as the system moves from state to state.

The dynamic aspects include:
- the operations that are possible;
- the relationship between inputs and outputs; and
- the changes of state that happen. [v]

An example of Z notation is given below. The *schema* shown in
Figure 585 is the formal description of an activity that submits a new white paper for publication on a Web site.

SubmitPaper

∆WhitePapers content?	:	TEXT
topic? : KEYWORD		
topic?	E	keywords
whitePapers' = whitePapers U {content? → topic?}		

Figure 5: Z schema showing submission of a white paper

The declaration, ∆*WhitePapers,* indicates that the activity is describing a state change to the *domain* of white papers. The declaration implicitly makes several variables available for use:
- *whitePapers* – the current list of white papers
- *whitePaper*s' – the list of white papers after the activity has completed
- *keywords* – the current list of subject areas with which the Web site is concerned
- *keywords'* – the list of subject areas after the activity has completed (not used above).

After the declaration of a domain come the declarations of the two inputs to the activity: *content?* (a document containing the new white paper) and *topic?* (the subject area of the white paper). Input names conventionally end in a question mark.

The part of the schema below the line starts with a *precondition*, stating that the subject area of the new paper must be one with that the Web site is in general concerned. This must be true in order for the activity to take place. Otherwise it cannot proceed.

The schema concludes with a *postcondition*, stating that the new list of white papers must include the submitted paper, mapped onto its subject area.

Due to its background in the most rigorous aspects of computer system development, Z has not traditionally been viewed as a process modeling technique. It is ironic that, in moving to an understanding of how human-driven processes are deeply unlike machine processes, we find that the most suitable formal basis for describing human work activities is one originally conceived

specifically for the most highly constrained computer systems.

However, used in an appropriate way – e.g., as a *declarative* technique based on pre- and postconditions – Z allows us to describe human work activities in precisely the manner that people conceive of them.

Moreover, we can use Role Activity Diagrams to visualize processes described with Z. We need to interpret the diagrams appropriately in order to incorporate Z constructs – to remove the original basis in Petri nets, and impose instead a stronger approach, more suited to processes that involve human activity. The elaboration of a Role Activity Diagram-based approach to graphical depiction of human-driven processes is beyond the scope of this article. Full details are given in the forthcoming book, *Human Interactions*, to be published in February 2005. (See http://www.mkpress.com/hi/).

To conclude, we have seen how a full description of human-driven business processes can be made in formal terms, as a basis both for analysis and for system development. Our approach uses Petri nets and pi-calculus, in conjunction, for the description of process evolution and its management – together with the Z notation for the modeling of detailed process activity.

Our focus in presenting this set of abstractions is on the management of ongoing process design change. Such change is a natural part of human interaction, and we therefore propose the formal basis described here for application to *human-driven* processes. By contrast, *machine-driven* processes change in more limited and tightly controlled ways. Hence, it may be more appropriate to apply different formal bases to machine-driven processes – We do not make an assertion either way. Our specific proposal is that, when it comes to the analysis of business activity that is centered on humans, the three techniques of Petri nets, pi-calculus, and Z notation can all be used in conjunction to provide a sound formal basis – a basis that deals not only with behavior, but with how that behavior changes over time. Such a basis is essential for process management in order to permit an enterprise to *guarantee how its operations are being carried out* – for instance, to provide financial transparency of the kind now required by statute in some jurisdictions.

The next step is further elaboration of these principles, and investigation of their ramifications. We encourage those interested in pursuing the necessary research to join the web forum **Role Based Process Support** (www.smartgroups.com/groups/roles), which serves as a focal point for collaboration on relevant work.

Notes

[i] The Sarbanes-Oxley Act was signed into law on 30th July 2002, and introduced highly significant legislative changes to financial practice and corporate governance regulation. It introduced stringent new rules with the stated objective: “to protect investors by improving the accuracy and reliability of corporate disclosures made pursuant to the securities laws.” It also introduced a number of deadlines, the prime ones being

- Most public companies must meet the financial reporting and certification mandates for any end of year financial statements filed after November 15th 2004 (amended from June 15th).
- Smaller companies and foreign companies must meet these mandates for any statements filed after 15th July 2005 (amended from April 15th).

The act is actually named after its main architects, Senator Paul Sarbanes and Representative

Michael Oxley, and, of course, it followed a series of very high profile scandals, such as Enron. It is also intended to "deter and punish corporate and accounting fraud and corruption, ensure justice for wrongdoers, and protect the interests of workers and shareholders" (Quote: President George W. Bush). The Sarbanes-Oxley Act itself is organized into eleven titles, although sections 302, 404, 401, 409, 802, and 906 are the most significant with respect to compliance (Sarbanes Oxley section 404 seems to cause most concern) and internal control. In addition, the Act also created a public company accounting board. Perhaps one of the most remarkable aspects of this legislation, however, relates to its profile. It is very much in the public and media arena. The focus is certainly intense in this respect, creating yet another clear motivation for compliance. There is simply no escaping it! (http://www.sarbanes-oxley-forum.com/

A more detailed overview of the Act is available at:
http://www.aicpa.org/info/sarbanes_oxley_summary.htm

The Act itself can be downloaded in PDF form from:
http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_reports&docid=f:hr610.107.pdf

[ii] An article in October 2003 extolling the virtues of Robin Milner's pi-calculus as the way forward (Smith, H., and Fingar, P., 2003, "Workflow is just a Pi process," http://www.fairdene.com/picalculus/workflow-is-just-a-pi-process.pdf) was closely followed by two responses (van der Aalst, W., "Pi calculus versus Petri nets: Let us eat 'humble pie' rather than further inflate the 'Pi hype'", http://tmitwww.tm.tue.nl/staff/wvdaalst/pi-hype.pdf, and Pyke, J., Whitehead, R., 2003, "Does Better Math Lead to Better Business Processes?" http://www.wfmc.org/standards/docs/better_maths_better_processes.pdf). The debate then spread across the Business Process Management industry generally, via many other articles as well as Weblogs and mailing lists.

[iii] Jensen, O., Milner, R., 2003, "Bigraphs and mobile processes," University of Cambridge Computer Laboratory

[iv] Spivey, M., 1988, "The Z Notation: A Reference Manual," Prentice Hall

[v] Spivey, M., 1988, ibid.

Keith Harrison-Broninski is the CTO of Role Modellers Ltd. (khb@rolemodellers.com), He is the author of ***Human Interactions: The Heart and Soul of Business Process Management*** (Meghan-Kiffer Press, Feb. 2005) from which this article is adopted. For more information, see http://www.mkpress.com/hi

