

Kinetic Process Graphs : Building Intuitive and Parsimonious Material Stock-Flow Diagrams with Modified Bond Graph Notations

Jacques LeFèvre^{1,2,3}

¹ Equipe de Recherche en Génie Industriel, Ecole Centrale de Lille, France

² IDEA.SIM LTD, Harrow, UK

³ Centre for Measurement and Information in Medicine, City University, London, UK

Mail Address : Ecole Centrale de Lille, BP 48, Villeneuve d'Ascq, F59651 France

Phone : 00 33 3 20 33 54 46 (Secret. 54 00), Fax : 00 33 3 20 33 54 18 - Email : lefevrej@libertysurf.fr

Key Words—*Stock-Flow Diagrams, System Dynamics methodology, Kinetic Modelling, Bond Graphs*

ABSTRACT : This paper presents a new graphical modelling language extending the stock-flow (SF) diagrams used in System Dynamics (SD). It introduces a new kind of graphs, the «*Kinetic Process Graphs or KPGs*». A KPG groups in a single flow-process some mechanisms representing flows modelled separately in a SF diagram but *cognitively* linked in our mental models as belonging to a single process using and providing material resources of different kinds. KPGs contain SF-diagrams as special cases and every KPG may be transformed in a SF equivalent. However for many processes in which several *material-like resources* are transformed into and/or transform each other (e.g. systems in business dynamics, environment, biology...), a KPG is usually more *parsimonious in arrows* and thus less visually complex than its corresponding SF diagram. It is also closer to our cognitive or mental models and thus more intuitive. In consequence, for this class of problems, KPG models are often easier to obtain, explain and maintain than SF diagrams and frequently suggest new vistas on a given problem. Section II discusses the current tools of SD (causal loop and stock-flow diagrams) and surveys quotations by leading SD scientists suggesting that these tools need some basic improvements. The rest of the paper presents KPGs which generalise the tank-pipe metaphor underlying SD by using a chemical reaction metaphor allowing a more natural representation of resource transformation. KPGs import some ideas from the method of Bond Graphs used in engineering to represent energy-based systems. Since SD is basically a kinetic theory, these BG concepts have nevertheless to be strongly modified. We briefly discuss the implementation of KPGs in the library KPG-LIB™ for the bond graph modelling and simulation software 20-SIM™ (see footnote 1). Finally, we present very briefly three advanced aspects of KPGs : array-based KPGs, fuzzy processes and adaptive graphs and we list some significant examples developed in our consultancy work in business dynamics, socio-economy and biology. We conclude by showing where exactly KPGs belong in the family of modelling languages and by discussing our current research.

I. INTRODUCTION

Since 1990, I have used System Dynamics (SD) in scientific research, in academic teaching and in a consultancy business. During that period, I have developed mid and large scale SD models in biochemistry, physiology, regional development, governmental policy analysis and business dynamics. Like many SD users, I often had the feeling that the graphical representation of SD models was not as intuitive and parsimonious as it should have been. This was most obvious in biochemistry, a field in which I was using two methods: SD to develop kinetic models of metabolic pathways and another graphical methodology from engineering, the Bond Graphs (BG) [1,2] to build energetic models of the same pathways. Despite their greater level of detail, the structure of the BGs was easier to explain to biochemists than the Stock-Flow (SF) diagrams of their SD cousins. It was therefore tempting to ask if a purely kinetic method merging the main ideas of SD with some features of BG would not provide SD models with appreciable gains in simplicity and intuition. However, SD is a kinetic theory and BG are based on energy conservation. Direct integration of both methods is thus impossible and many adaptations are needed. Would the resulting method still present intuitiveness and simplicity ? This paper presents the results of the work started in 1995 to answer these questions. Its result is a new SF graphical modelling tool called « *Kinetic Process Graphs (KPG)* » implemented in the library KPG-LIB for the software 20-SIM.

¹ This paper mentions the following trademarks : KPG-LIB (from IDEA.SIM, Harrow, UK), 20-SIM (from ControlLab, Enschede, NL), STELLA (from HPS-Hannover, NH, USA), Powersim (from POWERSIM, Bergen, Norway) and Vensim (from Ventana – Harvard, MA, USA).

Until now, KPGs have been presented only to audiences specialising in simulation methodology [3,4] and engineering modelling [5] and the goal pursued here is to submit their main concepts to the SD community for discussion. KPGs extend the stock-flow diagrams of SD by replacing the classical tank-pipe metaphor by a generalised chemical reaction called a «(N,M)-process » which admits the tank-pipe metaphor as a particular case for N=M=1. To change, even partially, the metaphor on which a whole research community is thriving is certainly a high risk business which has to be strongly justified. Therefore section 2 briefly reviews the existing SD methods and discusses quotations from leading SD authors suggesting that some improvements are needed². Our new (N,M)-process metaphor is then introduced and I suggest that its use may give significant benefits in simplicity and parsimony of the models for many SD applications ranging from biochemistry to socio-economy and business dynamics. Section 3 uses then simple biochemical examples to introduce heuristically the main KPG notions and to show how they complement the current SD methodology. The rest of the paper is devoted to a more complete presentation of KPG, introducing successively its main concepts and notations, some advanced concepts needed in the development of large scale models and my current research directions. Finally the conclusion discusses how KPGs relate to other modelling methods: graphical continuous methods like block diagrams, bond graphs and classical System Dynamics, object oriented equation-based continuous methods and discrete event methods.

2. WHY DO WE NEED TO MODIFY THE STOCK-FLOW NOTATION OF SYSTEM DYNAMICS?

The two main graphical modelling tools of System Dynamics (SD) are « *Causal Loop or CL* » diagrams and « *Stock-Flow or SF* » diagrams. They emerged almost fully defined from the seminal work of J.W. Forrester [6-9]. Over the years impressive progresses have been made in their application methodology (e.g. mental model elicitation and refinement, group modelling, policy modelling, validation, micro-worlds...) and obviously in their supporting software. However, their basics have remained almost unchanged (see their original definitions in chapter 8 of [6] for SF and in p.257 of [9] for CL). Despite and perhaps because of this longevity, the use of CL and SF diagrams in real world modelling is not without its problems, repeatedly discussed in the SD literature and briefly surveyed hereafter.

Let us see first what current authors think about CL diagrams (see [7] for a brief history). Initially Forrester did not use them (there is no CL-diagrams in [6-8]). Then he started to use them sparingly and only as explanatory aids (see for instance [7], p.257). Soon their apparent (deceptive ?) simplicity led many people to use them also as pre-modelling tools for interdisciplinary works in which SF diagrams were seen as too technical³. However, from the beginning, many people recognised the limitations of CL diagrams for modelling. To mention a goldsmith's opinion, let us give here a quotation from Forrester [11]:

« Causal loops do not provide the discipline for thinking imposed by level and rate diagrams in SD. I do not use them as the beginning point for modelling. Instead I start from identifying the system levels and later develops the flow rates that cause those levels to change. Sometimes I use CL-diagrams for explanation after a model has been created and studied. For a brief presentation to people who will not try to understand the real sources of dynamic behavior, CL diagrams may be a useful vehicle for creating a general impression of the subject. »

Similar opinions have been expressed by many experts ([1], [11-17]). Although not accepted by all, the lesson emerging from their papers is that the use of CL diagrams for modelling is not conducive to good SD practice. Therefore it seems that, in their current form, they should only be used in the early phases of a project as tools for the elicitation of simple aspects of the underlying mental models and, at the simulation stage, to help in introducing some general aspects of the observed dynamics. In both roles they may and should be improved and automated by using for instance artificial intelligence methods like semantic attributes, fuzzy cognitive maps and qualitative simulation. However, due to the uncertainty generated by this kind of methods for high dimensional problems, we conjecture that the usefulness of CL diagrams will not scale up well. Their improvement, although much needed, is therefore not my priority.

² All the quotations made in this paper are a bit shortened. However, I think that their messages are true to the originals.

³ We focus here on System Dynamics and thus computer modelling and simulation are crucial. We will thus not discuss the qualitative use of CL diagrams and archetypes by the « System Thinking movement ».

On the contrary, it is what Richmond calls operational thinking or «*Thinking Physics*» [14], i.e. getting a clear view of the structure of the stock and flows of a system which is the «*Key Step*» in building a good SD model [14] and it is often impeded by the use of CL diagrams. In this perspective, SF-diagrams become thus the main SD tools. Initially, stock-flow models were built directly in code (e.g. Dynamo). With the advent of graphical programming, it was realised that a code-based approach was limiting drastically the communication with domain specialists and model users. Thus many current SD packages (e.g. Stella, Ithink, Powersim, Vensim, see footnote 1) implement stock-flow models graphically, i.e. as SF-diagrams. This notation has been quite stable over the years having got only some additions (conveyors, queues, ovens, co-flows, sectors) since the early seventies. It has thus acquired the status of a de-facto paradigm and almost every system dynamicist takes it for granted. However, a disturbing voice may be heard from B. Richmond, the very father of Stella and Ithink. In a 1994 paper, he wrote an assessment of the SF graphical language [14]. Putting together and shortening some of his sentences, his message reads:

« *Operational thinking represents the essence of SD... one does it when one gets at the core SF (stock-flow) infrastructure... We must thus focus on understanding the impediments to internalising the SF language... The first is the abyss between a mental model and the associated SFD. The second is the visual complexity of SFDs which often look like a spaghetti of stocks, flows, converters and connectors... We must find ways of parsing them out into more bite-sized chunks... We must squarely face up to the challenges inherent in bringing about a major paradigm shift and in improving our language.* » (Note : underlining by the author of the current paper)

We find a message hinting also at a lack of intuitiveness in a recent text from Forrester [13] : «*The level-rate-feedback structure in SD is the fundamental structure of real social and physical systems...The process of SD modelling must still be baffling to those who are new to the subject.... There is still much room for very constructive research on the process of converting information from the real world into SD models.* »

Supported by Richmond's view but being more radical, we claim that, for mid to large scale systems, SD-diagrams are often too complex to be of much help in interdisciplinary model formulation, understanding and modification (we call this « *the visual complexity trap* »). We also claim that the difficulties felt by many users to draw even simple SD diagrams show that these diagrams do not correspond sufficiently well to our cognitive representations of the systems modelled (this is « *the mental abyss* »). Clearly, improving the SF notation to decrease these two problems must be our first priority.

To better understand what is wrong with SF diagrams, let us discuss first the visual complexity trap. Richmond's drastic opinion is confirmed by a brief examination of the SD papers describing large models. Often, their authors present only the global structure of their models. This is clearly inadequate since then the reader cannot really understand and check the work presented. Other, specially with very large models, revert to equation-oriented languages and, in some cases, do not even mention their links with SD. Again this is unsatisfactory. Finally, a few papers present the SF diagram completely but then fall in the complexity trap. In all cases, the result is «*cognitive overload* » which, for the reader, leads to misunderstanding and rejection and, for the modeller, to oversimplification or a very difficult life.

Obviously, the more complex a model is, the more we feel this *complexity trap*. A possible answer to that problem is thus to limit our use of SD to simple models. This has been a trend in some recent SD works focusing almost exclusively on very simple models well suited for general insights or introductory teaching but often too oversimplified for more realistic applications. Excessive focus on such simple models give to outsiders and would-be users and customers the wrong feeling that SD may only be used for elementary modelling. It creates thus an attitude of rejection in some professional circles (e.g. among many engineers or hard scientists). On the other hand excessively detailed models miss the point of good system thinking : to see at the same time both the forest and the trees by adopting what Richmond calls the « *intermediate vantage point between specificity and breadth* ». Thus we have to focus on models of « *intermediate complexity* » and, as Forrester tells us, the definition of what this means clearly depends on the goals and intended uses of the model [18]: « *For interacting with mental models, small models have clear advantages over large ones. However, the appropriate size of a model depends on the amount of time and effort available to study it. In a half day, all we can do is to consider a few variables. For a detailed study, with months available, models can be of a far wider scope.*»

Thus we need both simple models used as sources of metaphorical, preliminary or general insight and mid or even large scale models for more comprehensive prospective work. These more complex models are obviously subject to the full complexity trap. Their maze-like SF diagrams are difficult to use. Even the model builders themselves have troubles when, after a few months spent on other projects, they need to modify a large model. Similarly, it is quite difficult to use a complex SF diagram to understand the underlying model from scratch and to see its assumptions and its structure. Finally using a SF diagram to explain a complex system to non specialists (students, managers, customers, general public) is just asking for trouble. Array variables, modularisation and hierarchical structures have been introduced and go a long way to improve communicability. However, if we use them too much, we end up with another form of the complexity trap : in trying to get a global view of a strongly modular and hierarchical complex model, we need to shift so frequently from one level to another that, again, the global view fades away and remains out of focus. Obviously, nobody expects a faithful graphical model of a truly complex system to be simple. However, our methods should not make this representation more complex than strictly needed. Since SF diagrams should be communication vehicles between people, there is clearly room for improvement.

Richmond's first impediment, the mental abyss separating our mental models and SF diagrams is more subtle to perceive. As this author explains in [14], this abyss does not appear exclusively in large models but even in small ones since the difficulty starts with the first stocks and flows drawn on a piece of paper when starting a study. To isolate the problem, let us consider a typical small model, the classical prey-predator « PopDynam » model used in the « Getting Started » brochure of Stella™ [19]. Due to its educational role, this SF diagram has probably been chosen for its intuitiveness, good compromise between simplicity and complexity and intrinsic interest. However, let us ask ourselves if we can use it directly as our main communication tool to explain basic ecology to complete newcomers to SD and ecology. Let us consider for instance Fig.1 which gives a slightly modified version of a figure found in this brochure [29].

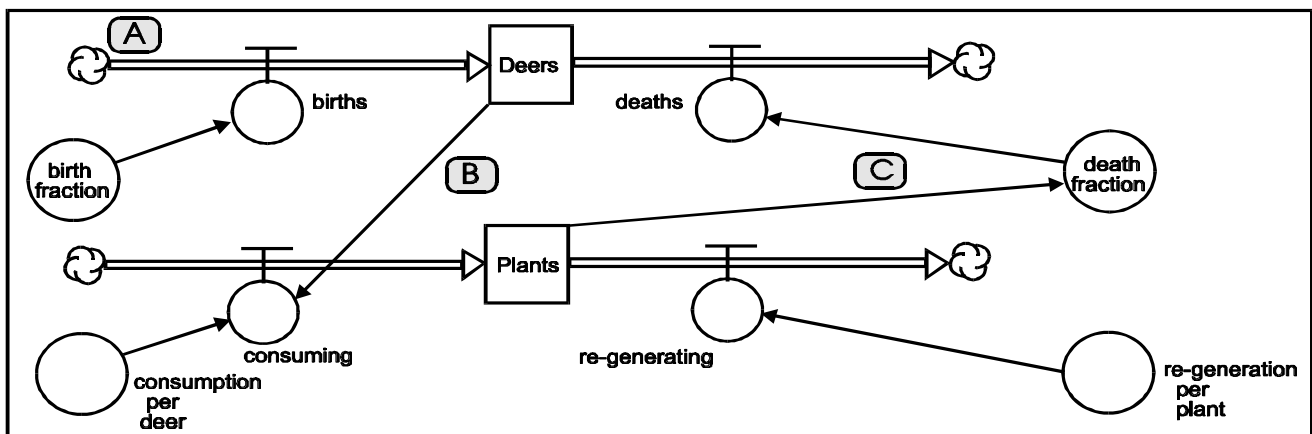


Fig.1 : A SF diagram modified from an introductory SD text [19]

This SF diagram looks crystal clear to every system dynamicist worth his or her salt. However, I tried to use it and similar models with newcomers to ecology (students in secondary education, medical students, average citizens) to introduce them to simple notions like exponential and limited population growth, dynamic carrying capacity, links between nutrition, metabolism and reproduction, prey-predators... Then I asked them to express their feelings about the diagrams themselves. In general, they found them acceptable. Indeed, to avoid the complexity trap, I presented only simple and classical ecological models like the one above and Volterra-like models. However, they also felt a bit puzzled by their structure, saying things like :

A 13 year old student pointing to source A (Fig.1): « What is the source of deers ? »

A biology student showing the signal B (Fig.1): « You told us that dark lines are signals and do not transport materials. But then look: eating plants is a signal. This is strange. Deers eat plants, they do not receive signals from them. To come to that, they do not receive signals from plants to die (point C)... I understand what you mean but... »

A biologist : « Ok, I can use your diagrams, this is not a big deal but to me, they are just another way to describe the equations. For instance, they do not fully capture what I mean when I speak about material flows in metabolism... »

To avoid a bias due to the particular model considered, I did similar interviews for two other models : a model of a Michaelis-Menten enzymatic process [20] and the CCRS model of human resource management in business dynamics [21]. Although expressed differently, the reactions were quite similar.

Let us now make some remarks :

- In SD we are always told to look at the endogenous causes of a behaviour. It is thus interesting to note that the first naï ve comment points at an exogenous source. We all know that its presence is just a notational trick... but it is indeed the artificial nature of this element which creates a problem for the student. Such dummy sources are endemic in SD diagrams. It seems that they are sources... of confusion and should be eliminated when possible. Obviously, this cannot be done without basic changes to the SF method.
- The two other « gut reactions » are interesting too : these people express what they perceive as unnatural in the SF notation : the representation of material flows (e.g. plants enter into deers) by signals linking entities of different natures. Again modifying that point needs basic changes to the SF method.

At first sight, these two remarks seem anecdotal and we might just ignore them. However, I claim that they reveal something important since they point towards areas of confusion which, if taken seriously, forces us to a complete revision of the basics of SF-diagrams. Indeed, what they really mean is that, if we delete all the auxiliary signal links from a SD diagram, to focus only on stocks and flows, we get a few rather simple disconnected networks having each the form of a tree and usually showing dummy sources (Fig.2).

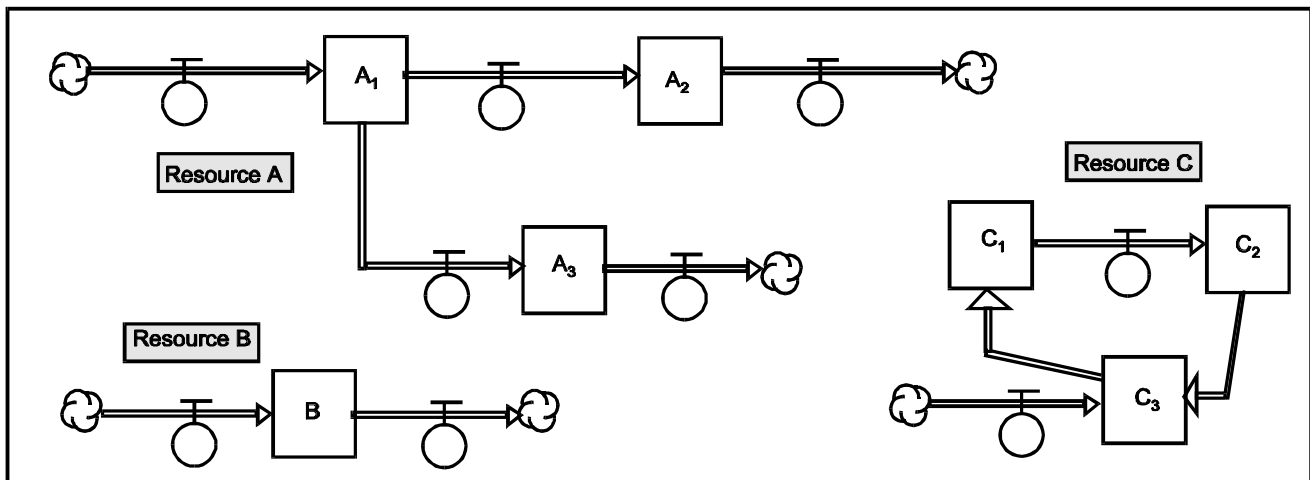


Fig.2 : Simple disconnected trees showing the usual structure of stock-flow networks when the networks of auxiliary signals are ignored.

This is due to the fact that all the entities flowing in a connected set of stocks and flows must have the same nature or in other words, that a flow process may have only one input stock and one output stock. Indeed, in [6] (p.70), Forrester gives what I call the principle of « *material network consistency* » which is the basis if the tank-pipe metaphor:

It should be noted that flow rates transport the contents of one level to another. Therefore the levels within one network must all have the same kind of content. Inflows and outflows connecting to a level must transport the same kind of items that are stored in the level. Items of one type must not flow into levels that store another type.

This assumption seems totally sound and belongs to the very core of SD. However, I will suggest in the next section, that it often gives a decomposition of the networks of stocks and flows which is mathematically correct but too detailed and too close to the equations representing the system to really express the structure of our mental models. I will also suggest that, as far as flow rates and stocks are concerned, this is the basic source of the mental abyss. The following sections will then present the theory underlying a modified notation for SF diagrams eliminating this problem and allowing us to build what we call « *cognitively justified models* ». We will also see that this new notation gives much simpler and « *parsimonious* » diagrams and is thus a partial solution to the complexity trap problem.

3. (N,M)-PROCESSES : A NEW METAPHOR FOR MATERIAL STOCK-FLOW DIAGRAMS

Before to proceed, we must make some remarks limiting the scope of our claims which otherwise could appear overly ambitious.

First, our new graphs will be compatible with the existing SD framework which has proven its worth time and again. This compatibility requirement has two consequences :

- We keep the basic pillars of Forrester's thinking : flows, stocks, intertwined causes and effects, delays and feedback loops remain central to determine the dynamic behaviour of a system and we still focus on endogenous reasons for this behaviour before to suppose that it comes from exogenous factors.
- The new method does not extend the scope of SD. Its goal is only to provide clearer models for the same class of systems. It is easy to show that SF-diagrams are generic for systems described by delay-differential or even functional equations. Any model developed with the new graphs will thus have a SF equivalent and the translation of the new model to a SF model will be algorithmic. Finally, it exists many cases in which the current SF-diagrams are perfectly satisfactory. In these cases, our new method will reduce naturally to classical SF-diagrams.

Secondly, the extent of our modifications is limited. SF-diagrams interconnect what Forrester calls physical networks and information networks [6]. Physical networks deal with flows of matter-like quantities like people, money, goods or species which satisfy a conservation principle. Hereafter we will call them « *generalised matter or GM-networks* ». Information networks represent signals. They have thus no conservation property. We will call them « *generalised information or GI-networks* »⁴. Stocks (called levels by Forrester) exist not only in GM-networks but also in GI-networks. Indeed as soon as it is dynamic, i.e. has a memory, a GI-network contains integral operators which are equivalent to stocks. For instance entities like awareness level, motivation or knowledge may be represented by accumulations and thus by stocks. However GM and GI-stocks are different, a fact which has important consequences for our work :

- We distinguish carefully between both kinds of accumulation by using different names and symbols for GM-networks (stocks) and GI-networks (levels). Although making this distinction clearly in its principles, standard SD does not distinguish sufficiently between both at the level of its notations. This fact, in addition to creating some confusion for non specialists, may be one of the reasons why the representation of GM-networks has remained a bit under-developed in modern SD.
- Our representation of GI-networks will be very similar to the one used in SD. Indeed, considering the huge variety of SD applications ranging from biochemistry to biology and management, I do not think that there is much to gain in trying to find a set of totally general structuring principles for GI-networks⁵.
- The thrust of the effort reported here is to find such a set of general structuring principles for GM-networks by focusing on matter conservation which is valid not only in SD but also in physics and chemistry. The question is thus to see if we can benefit from the more structured approaches to modelling used in these fields. They have two conservation principles (matter and energy) and a method like bond graphs implements them directly in its structure. We, in SD, cannot use energy conservation as a general concept. Indeed, energy, although existing in all systems, is not a useful concept for most applications of SD which, to use a physical terminology, are purely kinetic. We have therefore to abandon all the physical modelling concepts representing energy-based aspects. It remains then a core of physical modelling ideas dealing with matter conservation. Until now, they have not been interwoven in the fabrics of SD. We claim that there is much to gain by doing so and, taking really and deeply Richmond's advice on the need for « *Thinking Physics* ». This is exactly what the remaining parts of this paper do.

⁴ The term « *generalised* » is needed for both networks. It is only from the aggregated viewpoint of our models that people or animals may be considered strictly as matter. Similarly, the information network contains entities like a degree of motivation or an anxiety level which are seen as information in our models but are probably much more complex than that.

⁵ Although, as shown by Morecroft, that may very well be possible and fruitful in specific fields of applications like business dynamics in which management and decision theories may for instance provide some general background to policy modelling.

Biochemical kinetics seems to be the proper field in which to go shopping for new ideas to improve the representation of SF-diagrams by focusing on matter conservation. Indeed, biochemists spend their working life in analysing complicated set of reactions which they see as networks in which matter flows and is transformed. The first task on our agenda is thus to compare their descriptive tools with SF-diagrams⁶.

We start by remarking that the kinetics of a simple mono-molecular reaction is naturally described by a SF-diagram. This is shown in Fig.3-A where we see, in panel A, a reaction $R>$ from A to B with a rate given by the kinetic equation $dB/dt = -dA/dt = F(A,B,y)$, y being an external signal like for instance the temperature or PH level. Its SF-diagram is given in panel B and proves our point. We can go further : every set of mono-molecular reactions may be represented by a SF-diagram as demonstrated in panel C which shows the GM-network of the system of reactions $A \rightarrow B$, $B \rightarrow C$, $B \rightarrow D$, $C \rightarrow A$, $A \rightarrow E$, $E \rightarrow F$ with provision of A from outside and loss of D and F to the environment (auxiliary signals not represented).

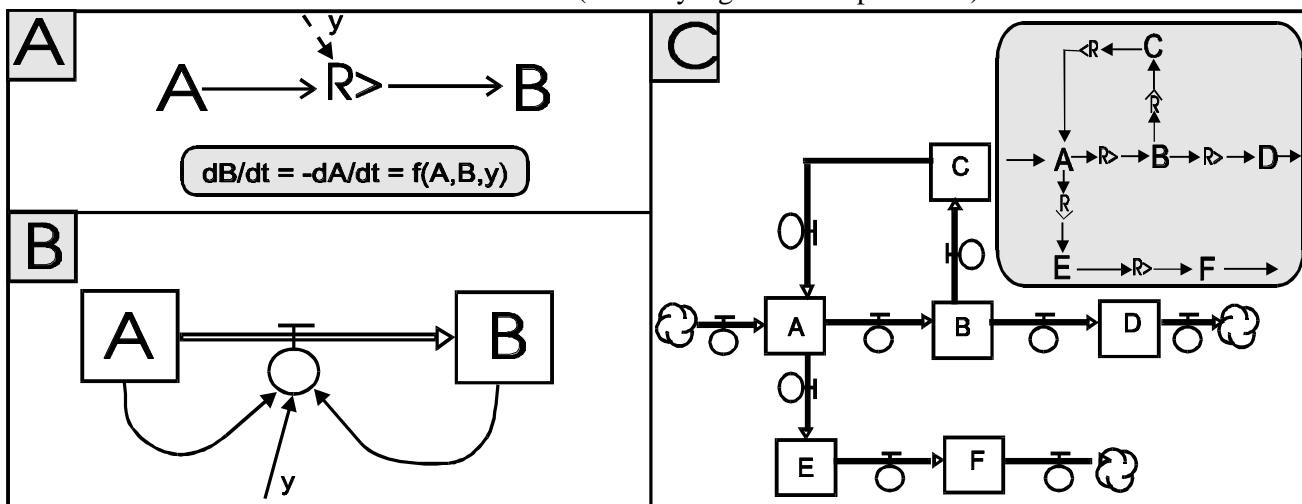


Fig.3 : Representation of mono-molecular reactions by SF-diagrams.

The cycle ABCA, the competition of reactions for using A and B and the organisation or structure of the reactions are obvious at first glance in the SF-diagram but need attention to be remarked in the above list of reactions. This is a minor advantage for such a simple system but it is very important for large sets of reactions with many cycles, branching points, recombination points and disconnected reactions. In fact, to support their discussions, biochemists who routinely study sets of hundreds of reactions with hundreds of chemicals use a specific informal representation called a « metabolic map » which uses symbols like those of panel A to represent their networks of reactions [17]. The metabolic map of the example of panel C is given in its inset. It has obviously the same structure than the SF-diagram. It should thus be clear by now that, for sets of mono-molecular reactions, SF-diagrams are just the fleshing out of metabolic maps with all the details needed to get fully fledged models but with the same high level of simplicity and intuitiveness. If biochemists find mono-molecular metabolic maps useful for qualitative discussions, they will of course accept SF-diagrams very well since they bring dynamics, i.e. life to metabolic maps.

However, mono-molecular reactions are quite trivial. The situation changes drastically when we consider more complex cases like the general bi-molecular reaction $nA + mB \rightarrow pC + qD$ in which n moles of species A and m moles of B combine to produce p moles of C and q moles of D⁷. Its metabolic map (Fig.4-A) uses the same symbols than in Fig. 3-A plus two additional nodes called J and S. It clearly shows that the material flows of A and B get together at point J (join) in a reaction $R>$ which produces and separates at point S (separate) the material flows of C and D. The most salient point in this map is that material flows of different entities A and B join in given stoichiometric ratios at point J and that the reaction separates its products in given stoichiometric ratios into flows of different entities C and D at point S.

⁶ Our heuristic basis is chemical. Thus we cannot avoid here the presentation of some elementary chemical kinetics. If the chemical knowledge of some readers is too rusty to accept comfortably the arguments presented here, they should not despair. They should indeed be able to follow the gist of the argument. Moreover, later sections will redefine these notions without further references to chemistry. They will also present applications to management.

⁷ The « stoichiometric coefficients » n, m, p, q define the atomic proportions of A, B, C, D needed and produced in the reaction.

This is directly in contradiction with the SD principle of « *material network consistency* » recalled in section 2. In fact, SD has no « *direct* » means to represent points like J and S for junction and separation of material flows of different natures. Consequently, the SF-diagram must use a trick to represent them (Fig.4-B). It is made of 4 trivial and disconnected GM-networks, one for each species. Reactants are represented by stocks flowing into sinks and products by stocks filled by sources. The reaction rate (flow) is computed by an auxiliary block R receiving the appropriate signals and imposing it through blocks imposing stoichiometric constraints to all the flow-defining taps. In contrast with its metabolic map (panel A), this diagram presents the flaws discussed before for Fig.1 and 2:

- The material flows are connected only by signals and the GM-network contains dummy sources.
- Conservation is imposed artificially by abstract signals which do not exist in reality.
- It is much more complex (i.e. contains more arrows and symbols) than the metabolic map.

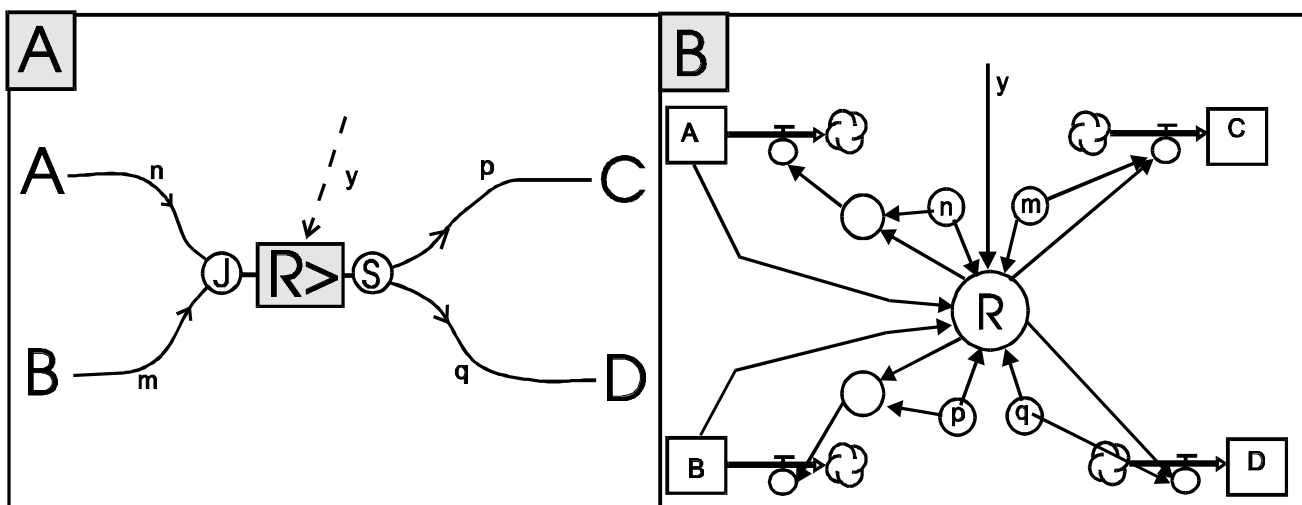
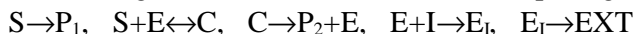


Fig.4 : Representation of a pluri-molecular reaction by a metabolic map and a SF-diagram.

Like in our ecological example, if we try to use the SF-diagram to explain this basic reaction to a student or if we try to convince a professional chemist that it is a description as intuitive as the metabolic map, we will run into trouble. SF-diagrams lack the notions and notations J and S needed for a parsimonious and intuitive representation of a general, i.e. non mono-molecular, chemical reaction with an arbitrary stoichiometry. Some tricks are needed and they do not result in a parsimonious and cognitively justified diagram. For a single reaction, this is still acceptable. We can certainly find our way around the SF-diagram of Fig.4-B. However, for a network of reactions, the SF-diagram becomes a nice example of Richmond's complexity trap. Consider for instance the following biochemical reaction scheme depicting an enzymatic system:



Although a bit more complex, it is very similar to a SD example presented in [20]. Its metabolic map, obtained by combining the maps of each reaction, is given in Fig. 5-A. We will now give an introductory description of this system and we invite the reader to follow it directly on Fig.5-A to realise that this map is indeed a good vehicle for communicating the main aspects of this system. We see a reactant S (substrate) provided from the environment by an input process PI>. It may then follow two reaction pathways R1> and R2> in competition for its use: R1> transforms it in a product P₁ and alternatively, <R2> combines it with an enzyme E to produce reversibly (arrows on the two sides of <R2>) an intermediate product C (complex) which then is decomposed irreversibly by a reaction R3> into a product P₂ while recycling the enzyme E. This enzyme E may thus cycle repeatedly along the path just described, taking at each cycle a molecule of S and producing a molecule of P₂. The number of molecules of E cycling and thus the production of P₂ may be decreased by a reaction R4> binding E to a molecule I (inhibitor) to give an ineffective enzyme E₁ eliminated into the external environment EXT by an output process PO>. This map shows clearly the fate of the various species, the cycle, the two competitions between reactions for S and for E. It is thus a good tool to discuss qualitatively the dynamics of this reaction scheme and I used it many times to teach basic enzymatic kinetics to complete beginners. They always considered it as very intuitive and useful.

The corresponding SF-diagram is given in Fig.5-B. In an effort to make it more palatable, I have drawn it with the same spatial arrangement of the stocks than the one used in panel A. Despite this, if you try to follow the previous description directly on the SF-diagram, you will experience problems due to the fact that it presents the previously described pathologies to an extent making its use difficult for what should be simple discussions. Since the diagram is not very complex, the problem is not its complexity but its lack of intuitiveness. It seems to most users that the metabolic map captures well our mental model of the structure of a reaction network and that a SF-diagram does not do it well. This is exactly Richmond's mental abyss⁸.

Remark that, compared to the sophistication of real biochemical networks, this example is elementary. It has only a few reactions and does not use stoichiometric coefficients. If we were using more complex networks with these two features, we would see that the metabolic map scales up well and remains usable and even mandatory for any serious understanding [22]. However, SF-diagrams would become almost useless. We would experience then the full force of the complexity trap. Like Forrester said (see previous quotation), building or using SD models may indeed appear sometimes quite daunting.

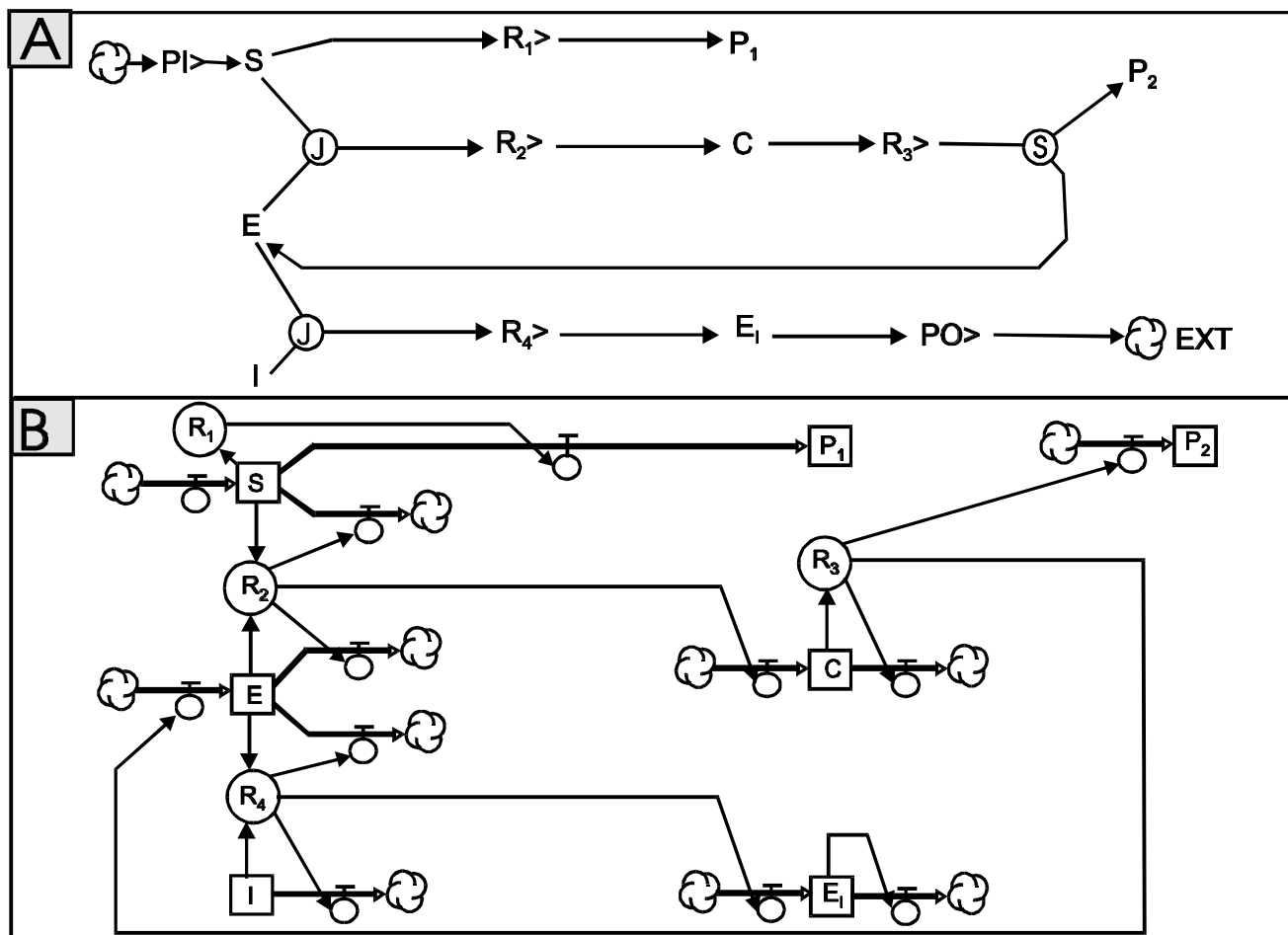


Fig.5 : Representation of an enzymatic reaction scheme by a metabolic map and a SF-diagram

As far as chemistry is concerned, the way forward is obvious, we want our SF-diagrams to be as simple and intuitive as metabolic maps but also to be able to present unambiguously all the details needed to fully define the underlying model and its simulation program. We will show in the next section that this may be done by re-defining taps and by adding to the current concepts of SF-diagrams special symbols dealing with the operations denoted in Fig.4-A by the « Join » and « Separate » points and by the stoichiometric coefficients. However, there is more to SD than just chemistry... and it is now time to look further to other applications. Our goal is to make SD more powerful and not only to get better chemical kinetic models. Will the modifications just suggested be sufficiently generic to cover the whole range of SD applications?

⁸ For instance, you may try, as I did years ago, to convince a biochemist that Fig.5-B is at least as intuitive as Fig.5-A and you will lose some hardly gained credibility. This is in fact what motivated me to develop the Kinetic Process Graphs.

This question may be answered positively by defining what we call a (N,M)-process. It is an (eventually abstract) operation taking at each time stoichiometrically defined flows of N input GM-quantities and acting on them to produce stoichiometrically defined flows of M resulting output GM-materials. Like it is done for flows in SD, the rate of a (N,M)-process is defined in steps per unit time by a formula or algorithm dependent on different variables communicated to the process by auxiliary signals. However, in opposition to SD flows, a (N,M)-process acts on N species and produces M species at each time. The entering and exiting flows of each species are deduced from the rate by multiplication by the stoichiometric coefficients.

(N,M)-processes are similar to but more general than chemical reactions since their rate laws may be much more general than those used in chemical kinetics and since their stoichiometric coefficients, instead of being constants like in chemistry, may depend on all sorts of auxiliary variables. This similarity is made obvious by Fig.6. Panel A defines a (N,M)-process by a chemical-like equation. We call A_1, \dots, A_N its input GM-stocks and B_1, \dots, B_M its output GM-stocks. The process occurs at a rate $R(\underline{X}, \underline{U}, t)$ expressed in number of steps per unit time and dependent in general on several variables : a vector \underline{X} of variables measuring the contents of stocks in the model (i.e. of the stocks participating as inputs or outputs to the model as well as of any other stocks), a vector \underline{U} of external time-dependent signals, and sometimes explicitly of time. At each time, the process computes its rate R and takes flows computed stoichiometrically (see defining equations) from its input stocks to deliver stoichiometrically computed flows to its output components. Stoichiometric coefficients, instead of being constants like in chemistry may be controlled like the rates.

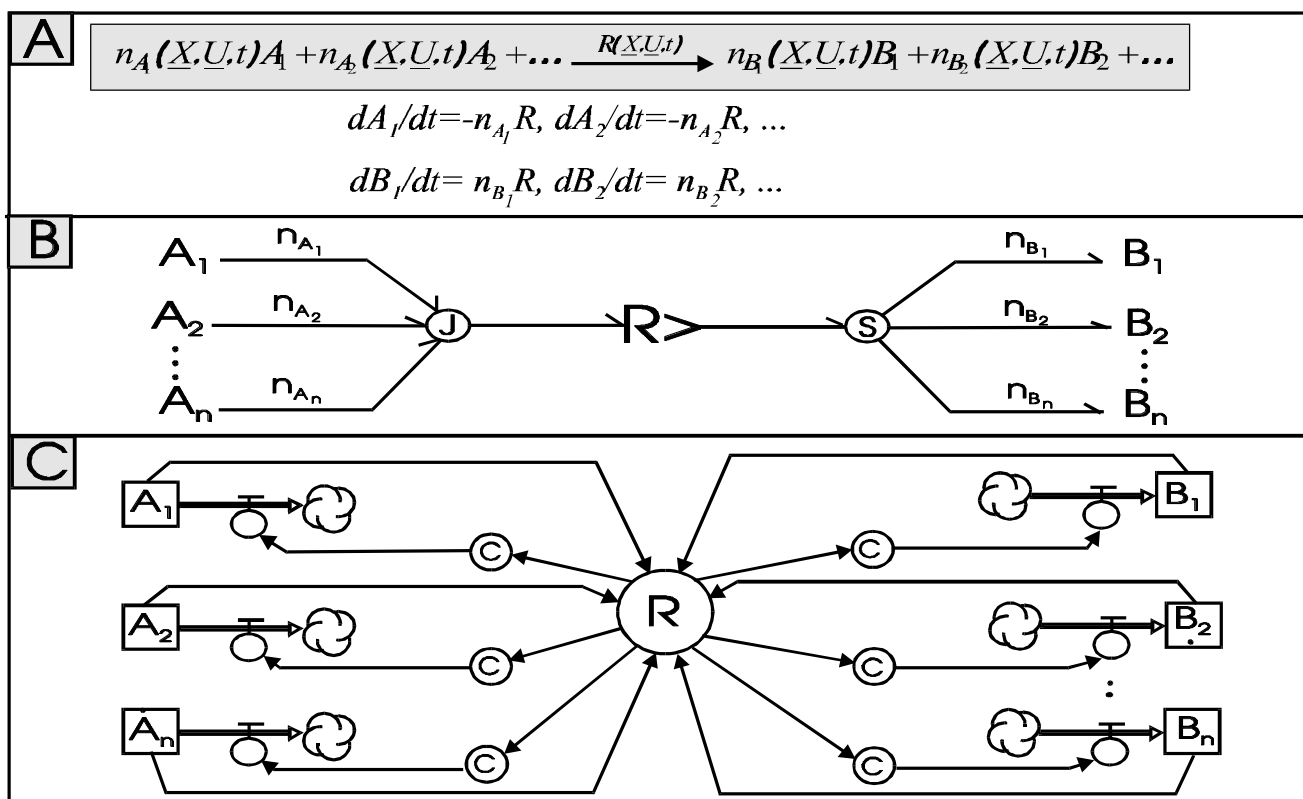


Fig.6: Defining a (N,M)-process by a chemical-like equation (A), a «Process map» (B) and a SF-diagram (C).

The equations of the rates and stoichiometric coefficients are not restricted to simple algebraic forms but may be arbitrary functionals of their arguments. Panel B shows the corresponding generalised metabolic map which we call now the « Process Map PM » and which is quite simple. Its stoichiometric coefficients and the symbols J and S have still to be defined mathematically but we will see later that this does not modify the complexity of the map. Panel C gives the SF-diagram which, as expected, is more complex. It remains to show that (N,M) processes do occur naturally in SD applications and to define more exactly the symbols of their «Process Maps» given in Fig.6-B but if these two points are dealt with, it seems obvious that SF-diagrams incorporating (N,M)-processes represented as simply as in Fig.6-B will go a long way to fulfil our goals of parsimony and intuitiveness.

Firstly, let us see what exactly is the scope of (N,M)-processes. Can we use them « naturally » in discussing non-chemical systems like those studied in SD ? To see this, let us consider the following list of application domains and chemical-like equations describing very informally some (N,M)-processes denoted by $P>$ ⁹:

- a) **Bio-technology** : bacteria+glucose \rightarrow more bacteria+useful products+toxins
- b) **Ecology** : foxes+rabbits \rightarrow more foxes+biomass
- c) **Population biology**: people+food+water \rightarrow more people+pollution+toxins
- d) **Epidemiology** : uninfected people+infected people \rightarrow more infected people
- e) **Management** : equipment+materials+resources+people \rightarrow products+equipment+people+pollution
- f) **Management** : products+orders+resources \rightarrow more resources+provided customers
- g) **Management** : resources+trainees+trainers \rightarrow more trained staff
- h) **Management** : doctors+patients+resources \rightarrow cured patients+ doctors

Admittedly, the views expressed in these processes are too simple to be really useful. However, they are not much more simplified than the ideas which we use generally as starting point for SD models and we can understand their meaning intuitively without further explanations. We can even see directly how naïve they are and how we would modify them or add supplementary processes to go further and get much less naïve models in the form of networks of processes representing our mental models. Despite their simplistic character, we may use them as communication vehicles in our discussions about the systems concerned. For instance if, in interviews conducted to build SD models, we ask people to « Think Physics », i.e. to describe the flows of « *generalised materials* » in their systems, we usually collect sentences like the following:

- (1) *An Australian civil servant* : « In our country, foxes do not predominantly hunt rabbits anymore, they have started to feed on small Australian mammals and they just thrive on them. »
- (2) *An executive from a food company* : « We tried hard to decrease the dependence of production on human resources but then quality went down because people were demotivated »
- (3) *A biochemist* : « In the production cycle of this substrate, the enzyme is clearly not rate limiting. It is the co-factors which control the production rate. »

We might translate them into normal SF-diagrams. However, we can also see them as resulting from an hypothetical underlying « *universal cognitive structure* » presented by our mental models and centred around (N,M)-process. Indeed, we can re-word these sentences as follows:

- (1) « Foxes and rabbits interact in a hunting process which produces more foxes. Foxes also interact with small Australian mammals in a currently emerging hunting process which again produces more foxes. »
- (2) « Human resources, equipment and materials interact in a food production process. Decreasing human resources lead to an increased proportion of low quality products. »
- (3) « Enzyme substrate and co-factors enter a reaction process rate-limited by co-factors. »

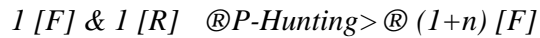
It is now clear that we may represent them as (N,M)-processes. To see more clearly the cognitive nature of this representation, let us denote a stock of a quantity Q by [Q] and a unidirectional process P by $P>$, we may then represent a (2,2)-process by the notation:

$$n[A] \& m[B] \text{ @ } P(r_p(I_1, \dots, I_n)) \text{ > } \text{ @ } p[C] \& q[D]$$

In which n,m,p,q are the stoichiometric coefficients. This is just a rule similar to those used in artificial intelligence and considered by many cognitive theories as basic building blocks for our mental symbolic structures [23,24]. Its semantics is simple. The process $P>$ has a rate r_p of transformation of input resources to output resources; r_p is computed by an algorithm from given « *influence signals* » I_1 to I_n . The process $P>$ transforms packs of input resources in packs of output resources. At each time t, it computes its rate r_p in number of packs transformed per unit time and transfers these packs from its input to its output. An input pack is made of m units of A and n units of B. An output pack is made of p units of C and q units of D. In the following, we will rename the stoichiometric coefficients and call them by the more domain-independent names « *packing and/or unpacking coefficients* ». The As and Bs needed are taken from [A] and [B]. The Cs and Ds produced are sent to [C] and [D].

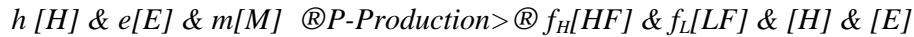
⁹ These definitions are illustrative and informal. Their stoichiometry is specified very incompletely by words like several, more, less... and their rates R are left unspecified. In reality, the rates should be defined by eventually complex algorithms dependent on auxiliary signals and possessing frequently a dynamics and a memory of their own. All these points will be dealt with later.

Using this notation, we can model one of the processes in the ecology example (1) by :



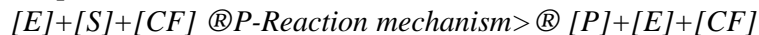
with F = Foxes, R = Rabbits and $n > 0$. In that representation, the rate equation $r = r(I_1, \dots, I_n)$ is left unspecified. Remark that [F] appears on both sides : a fox entering P-Hunting> produces 1+n foxes. This is just a slightly more accurate version of the informal ecology process given above.

In the same way, for the production process (2), the result would be :



with H = human resources, E = equipment, M = materials. HF and LF are high and low quality food. The packing coefficients f_H and f_L sum to 1 and may be variable to express the changing proportion of low and high quality food. Again H and E appear on both sides since their units are recycled after each production cycle. Compare with the informal rule (2).

Finally, the chemical example becomes :



These rules or processes are obviously similar to a chemical reaction $nA + mB \rightarrow pC + qD$ but the kinetic algorithm of P> is usually more complex than the chemical laws and the packing and unpacking coefficients n, m, p, q may be modified by various influences.

With this notation, we have thus established a bridge with the modern theories of cognitive psychology which use similar production rules in much more general settings. Since improving our knowledge on the nature of the mental models with which we perceive our systems is an important objective for SD, it seems that the connection just mentioned should be profitably pursued. However, we cannot really use the rule notation as a basis for SD. Indeed, we would end up with a representation of the model by a large collection of lines like (1) and we would be as confused as before. Due to the multiple appearances of resources in the left and right sides of several lines, it would indeed be very difficult to get a feeling for the global behaviour of the system. This kind of rules may well be used as unconscious or partial building blocks for our mental models but we are also visual animals. What our conscious cognition needs to reach a global understanding of a system is not a set of rules but a diagram summarising them. This is just what the informal metabolic maps described before achieve by connecting together the representations of each reaction. These maps are to reactions what semantic or cognitive maps are to rules in artificial intelligence.

This ends up our qualitative discussion of the motivations and basic hypotheses underlying Kinetic Process Graphs (KPG). The next section will now present them more accurately. We will proceed in two steps :

- Our first step will be to define what we call «*Structural Process Maps or PM*». We will begin by defining the components of an elementary process map like the one shown in Fig.6-B and then we will see how we may interconnect elementary process maps to get the PM of a system. The PM generalises and makes quantitative the metabolic maps discussed above. It allows the representation of the relations between the various stocks and flows of a system and the computation of all the flows entering or leaving the stocks once the external inputs and the rates of its various processes are determined. We use the word « structural » to characterise this map since it summarises all the properties of the system which depends only on its GM or “physical” structure (defined as the interconnections between its stocks, processes and packing/unpacking coefficients) and not on the functions modulating the packing/unpacking coefficients and the specific rate laws or kinetics used to compute the process rates themselves. We will see that, if, at each time, we know the rates of each process and the values of the packing and unpacking coefficients, we can compute the flows everywhere in the PM and thus obtain its state equations.
- Then in a second step, we will show how to compute the process rates and values of packing/unpacking coefficients needed by the PM defined in the first step. This will be done by defining a signal network transmitting to these elements the signals needed to compute their values. We will call these signals « *Influences* »; their network will thus be called a « *GI-network* ». By interconnecting the PM defined in the first step and the GI-network we will get a full KPG.

4. STRUCTURAL PROCESS MAPS : DETERMINING GM-FLOWS FROM PROCESS RATES

Most of the elementary components of our PMs have already been introduced informally in the preceding pages. Table 1 completes their definitions^{10,11}:

TABLE 1 : BASIC ELEMENTS OF STRUCTURAL PROCESS MAPS

NAME	NOTATION	COMMENTS
GM-FLOW BOND		Flow is positive in the direction of the semi arrow and negative in the other direction.
FLOW CAUSALITY MARKER OR CIRCLE		It indicates which side of the bond receive the information on the flow value.
GM-STOCK		$Q=Q_0+INT(\text{sum of flows}_{in} - \text{sum of flows}_{out})$. Must receive causal markers on each bond.
CONSTANT STOCK (ENVIRONMENT)		May have both flow causality; Q is constant independently of the flow given or received
PROCESS RATE		Defines a rate R_p and imposes the flow $f=R_p$ in its input and output bonds. In P , R_p is always positive, In $\langle P \rangle$, it is positive in the direction \gg and negative otherwise.
PROCESS RATE WITH DELAY		Same behaviour than preceding element but $f_{input} = R_p(t)$ and $f_{output} = R_p(t-\tau)$ with a delay τ . Thus some GM-quantity is stored inside (in the process internal store [Q]).
SOURCE		Provides to the left side a flow $f=r(t)$ taken from the constant store.
SINK		Delivers to the constant store a flow $f=r(t)$ taken from its left side.
MODIFIER	1) 2)	1) Receive an information on the value of f_{left} and imposes to its right side $f_{right} = n * f_{left}$. 2) Same definition but $f_{left} = n * f_{right}$.
EQUALITY JUNCTION (J or S)		Receives from one of its bonds an information on a flow value (marker near e) and imposes this value to the other bonds.
BALANCE JUNCTION		Receives information on flow values from all its bonds minus one. Impose on this last bond a flow=algebraic sum of flows on other
REPARTITOR		Receives information on the flow value f (marker near r) and determines other flows by : $f_1=m*f$ and $f_2=(1-m)*f$ with $m \leq 1$.

¹⁰ For advanced applications, we have also defined other elements (multi-bonds and array elements, time varying connectors, external product elements, or-junctions, modular elements) to be described elsewhere.

¹¹ The reader familiar with bond graphs (BG) will notice that these elements are obtained by neglecting some BG elements (R,I, S_e, GY) and stripping the other (S_r, C,0,1,TF) from every property related to power and effort variables. From a BG perspective, the result may, at first sight, look trivial and BG aficionados may wonder if anything useful may be achieved with it? The answer is a definite YES: the whole class of kinetic models and thus of SD models may be represented easily and intuitively.

Let us now describe the symbolic representation and the meaning of these elements :

Flow bond : The transportation of a GM-flow between elements is represented by a « *flow bond* » denoted by a semi-arrow¹². A positive flow is considered to travel in the direction of the arrow and a negative flow travels in the reverse direction. When needed, the name of the flow travelling in the bond is indicated close to the bond on the side opposite to the semi-arrow.

Flow Causality Marker : Consider a bond with its origin connected to a sub-model A and its end (arrow side) connected to a sub-model B i.e. $A \longrightarrow B$. The value of the flow in the bond may only be computed by A or B, not by both since then they could compute different values and the bond would be over-determined. Consider the case in which A has an equation computing the flow value. This flow value may be positive in which case the flow, in addition of being determined by A, physically flows from A to B. On the other hand, the flow determined by A may be negative and then physically flows from B to A. In the first case, A determines the value of flow it is delivering and in the second case, it computes the value of flow it is receiving. In both cases, B has just to comply and physically receive the flow if f is positive or physically give it if f is negative. B therefore cannot use one of its defining equations to compute f. It is given the information on the value of f by A and may use it as a variable in the right terms of its equations to compute some other variable. In other words, B sees the flow value as a cause received from outside , i.e. from A, to determine its behaviour, i.e. what it will compute from it.

We indicate the side which has to comply, i.e. which receives the cause by putting a small circle called a «flow causality marker » on the side of the bond connected to it. Thus in the case just analysed, the notation becomes $A \longrightarrow \bigcirc B$. Obviously, since physical flow directions and causality sending are unrelated, the flow might as well be determined by B while flowing physically either from A to B or from B to A. In that case, A would have to comply by giving or receiving the computed flow. The notation would then be $\bigcirc \longrightarrow$. These causality markers are useful to determine at first sight if a PM is well constructed or not. Indeed, every element will have a defining condition on its flow causality markers. If a model is well constructed, all these conditions have to be satisfied otherwise it contains an invalid connection of elements

GM-stock : Like a stock in SD, this element stores a GM-quantity Q. We denote it by [Q]. It is connected to an arbitrary number of bonds which may be directed towards it or which may point to it. A bond pointing to a store transports a flow which, if positive, will enter the store or will leave it if negative. A bond pointing from the store does just the reverse. At each time t, the store determines its value of Q by computing the flow balance : $Q=Q_0 + \text{integral from 0 to t of the algebraic sum of all its flows}$ ($Q_0 = \text{initial value}$). All the flows in the bonds connected to the store must be determined outside the store, i.e. in the model parts connected to the other side of the bonds connected to the store. Causally, the store must thus receive the values of all its entering or leaving flows. This is indicated by the fact that all its bonds, pointing to it or from it, must have causality markers on their side connected to the bond.

Constant stock : This is a special stock used like in SD to represent boundary conditions from the environment. It is denoted by the SD cloud symbol but we consider it to have a given constant value of the GM which it provides or receive. A constant stock has one or several bonds connected to it. They may point towards the stock or not but in both cases, each bond must have a causality marker close to the constant stock. In other words, it gives or receive flows which have their values determined elsewhere.

Process rate block: This is the cousin of the SD taps. It is connected to an input bond and an output bond. A process rate block has an equation which allows it to determine at each time the rate R_p of the process from information signals received from other places in the model. In this section, we do not discuss this equation nor the representation of the signals needed for its computation. This will be done in the next

¹² The KPG notations used from now on will be quite different from those used in SD even for the elements having an identical meaning in both methods. The reason of this choice is that I want to emphasize the kinship of KPGs with bond graphs in order to be able to minimize the adaptations needed to implement KPGs with a bond graph modelling package.

section. For now, we just suppose that the value of R_P may be computed at each time. The process rate block imposes then the values of the flows in its input and output bonds by the equations $f_{IN}=f_{OUT}=R_P$. It has thus the causality indicated in table 1. We distinguish uni-directional processes $P>$ and bi-directional processes $<P>>$. The equation of a uni-directional process is such that its flow is always positive. The flow occurs thus from « In » to « Out ». This rate process block requires then at all times the value of f from the element connected on its « In » side and delivers f to the element connected on its « Out » side. For bi-directional processes, the flow equation may compute positive or negative flows. The direction of positive flow is indicated by the double arrow $>>$. If f is positive, the flow occurs from « In » where it is required to « Out » where it is delivered. If the value of the flow is negative, the situation is the opposite.

Process rate block with delay: This element verifies the same rules than the simple process rate block. However, its flow conditions are $f_{in}=R_P(t)$ and $f_{out}=R_P(t-\tau)$ where τ is a discrete delay. It has thus the memory of its flow on a horizon τ . The integral of the flow stored during the delay period may be considered as an internal stock of the GM transported in the process. This element is thus partially a process and partially a stock. This is indicated by its symbol which contains both the notation $P>$ and the notation $[Q]$. Instead of a discrete delay we may use special continuous delays enforcing GM-conservation.

Source and Sink : They are composite elements obtained by connecting a constant stock and a uni-directional process rate block. The source delivers a flow given by $f=R_P$ and the sink receives a flow given by $f=R_P$ (cf. causalities). By connecting a constant block to a bi-directional process, we obtain an element which may work as a source or sink dependent on the sign of R (not illustrated).

Modifiers : They correspond to the packing/unpacking coefficients discussed before. A modifier is linked to an « In » and an « Out » bond. It has a parameter n called its ratio and makes between its input and output flow a transformation with two cases. In case 1 (see Table), it receives an information on the value of its input flow and then computes $f_{out} = n*f_{in}$. In case 2, it does the opposite, receiving an information on the value of the output flow, it computes $f_{in} = n*f_{out}$. As we have mentioned previously, n may be variable and may depend in eventually complex ways on signals received from elsewhere. Again, we will not discuss this point here but leave it for the next section. For now we will just suppose that n is known at all times.

The interpretation of the equations given above depends on the notion of packing/unpacking. We have seen that a process works at a rate of R_P steps per unit time and at each time takes packs of various resources to combine them into packs of various products (see Fig.6-A). Let us thus consider such a process taking at each step n_A units of A to give n_B units of B. We represent it by the PM :

$$[A] \xrightarrow[f_A]{(M:n_A)} P \xrightarrow[f_B]{(M:n_B)} [B]$$

R_P

The process P computes its rate and imposes a flow $f=R_P$ of packs to its input and output bonds. On the input side, the modifier $(M:n_A)$ is in the causality situation 2. We have therefore $f_A=n_A*f$. On the output side, the modifier $(M:n_B)$ is in the causality situation 1 and therefore $f_B=n_B*f$. These equations represent the packing in the input side and the unpacking in the output side. We can consider the flow f as made of packs of As on one side and Bs on the other side while f_A and f_B are made of separate units of As and Bs. Packing is made by a modifier with a case-2 causality and unpacking by a modifier with a case 1 causality.

Equality junction : it represents the nodes which we have called previously J (join) and S (separate). An e-junction (e for equality) may be connected to an arbitrary number of bonds with arbitrary orientations. One and only one of its bonds must have its flow causality marker near the e-junction. This bond, called the commanding bond (see the little © in the table) imposes thus its value of flow to the junction. This value may be positive or negative and the commanding bond may point toward the e-junction or depart from it. Then the e-junction imposes to all the other bonds to have a flow value equal to the one imposed by the commanding bond. These other bonds, receiving the information on their flow value, have thus the causality indicated in the table (marker on the side distant from the e-junction) and impose this value of flow to the elements connected to them. We will see later that e-junctions are used mainly in direct connection with (N,M) -processes with $N,M>1$ where they give us the packing and unpacking points needed for each participating GM.

Balance junction : This is the dual of the e-junction. It may be connected to an arbitrary number of bonds with arbitrary orientations. All its bonds except one must have their flow causality marker near the b-junction (b for balance). The junction receives thus the flow values in all these bonds. It computes their algebraic sum, i.e. their balance and imposes the result as the flow value in the remaining bond which is called the commanded bond (see the little ©) and has thus a flow marker on the side distant to the junction. This junction is mainly used to compute explicitly the balance of flows for a stock participating to several processes (see later).

Repartitor : This is a composite element (cf. left side of the repartitor box in Table 1) made of an e-junction connected to modifiers with ratios summing to 1. The e-junction receives a flow f and transmits it to its output bonds. The modifiers, receiving each an input flow f , compute their output flows respectively equal to mf and $(1-m)f$. The flow is thus divided in a given ratio between the branches. In practice, we use the notation given on the right side of the repartitor box (table 1). It is an input repartitor. We may reverse the direction of each bond (but not the causal markers) and we obtain an output repartitor requiring a flow f from two directions. We use also repartitors with more complex conditions than simple equality to 1.

This ends up the first description of our elements. We will see later that each of them has additional features to deal with signal propagation. Right now, to gain some insight on how these elements combine together, we will present a few simple PMs (Fig .7).

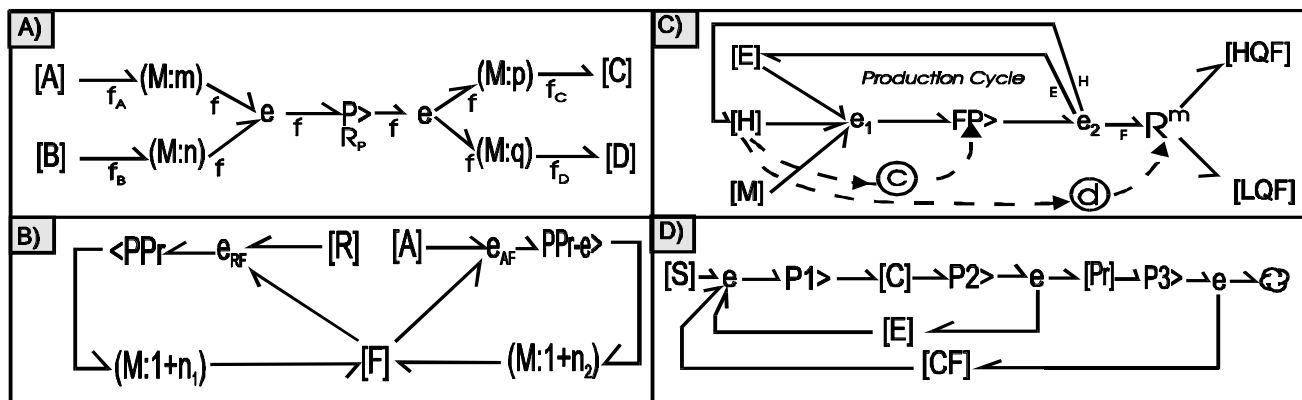


Fig.7 : Process maps of the simple examples used in the text

Our first example is the elementary process $mA+nB \rightarrow pC+qD$ (see also fig.6-B) . We see what we call the « Donor stocks » A and B and the « Acceptor stocks » C and D. We also see the process $P>$ connected on its input and output sides to e-junctions. Modifiers implement the packing coefficients n, m, p, q and are linked by bonds to the stocks and to the e-junctions. The units of A and B needed by $P>$ flow from their stocks to the packing coefficients which packs them into groups (packs) of respectively m units for A and n units for B. These packs flow to the input e-junction where they unite to form combined packs. Each pack entering the process $P>$ is made of a pack of A having m units and a pack of B having n units. This process transforms the input packs in output packs at a rate R_p per unit time. The output packs separate in packs of C and D in the output e-junction and these separated packs flow to their unpacking modifiers where they unpack respectively into p units of C and q units of D. The Cs and Ds flow then into their respective stores. Computation proceeds at each time t by computing first R_p (see next section). Then the flow R_p of packs computed is required from the input e-junction by $f=R_p$. This e-junction transmits an equal requirement of $R_p = f$ packs to each of its input sides for A and B. The packing coefficients then transfers a requirement for $f_A = n*f$ units of A and $m*f$ units of B to the input stores.

Remark that once R_p and eventually, n, m, p, q are computed, this PM is a full computational model of the process. It is thus equivalent to the SF-diagram of Fig.6-C but has still the same simplicity and intuitive character than the simple metabolic map of Fig.4-A. Admittedly, the auxiliary signals needed to define the rate are still missing from the PM which is therefore not strictly comparable to the SF-diagram. However, we will see later that, due to the bond graph-like notations explained in the next section, their inclusion leaves the simplicity and the intuitiveness of the PM unchanged.

Panel B gives the PM of the system of foxes [F], rabbits [R] and small Australian mammals [A] described before (see the interview of an Australian civil servant). The foxes [F] may participate to two processes. First they « get together » with rabbits [R] in the packing e-junction e_{RF} to enter the classical predation process $PPr>$. Due to the modifier in the output side of this process, each fox entering the process generates $1+n_1$ foxes which go back in their stock. The population of foxes grows if $n_1>1$. The foxes may also enter a second emerging process $PPr-E>$ in which, instead of rabbits, they predate the Australian mammals [A] and grow if $n_2>1$. Remark (1) that each process acts on different resources, (2) that the graph contain loops since the two processes deliver a resource F also used in their input, (3) that a competition is obvious between the processes since foxes F participate to one or the other and finally (4) that the model does not contain dummy sources like a « source of foxes » (see Fig.1). On these accounts, the PM given here looks more intuitive and simpler than the SF-diagram. Of course auxiliary signals are still missing from it but, like in the previous example, we will soon see that their addition does not change our conclusions.

Panel C gives the PM of the food production process of section 3 (sentence from a food company executive). Equipment [E], human resources [H] and raw materials [M] are packed together at an e-junction e_1 to enter a food production process $FP>$. The resulting output packs of this process first separate in three GM-flows at the output e-junction e_2 . Flows of equipment and human resources go back to their stock from which they can be used again and again to transform raw materials into food. The third flow exiting from e_2 is a flow of food products which separates in two in a repartitor R. A fraction m of the food produced ($m<1$) goes into a store of high quality food [HQF] and the remaining fraction $(1-m)$ goes into the stock of low quality food [LQF]. By modifying the value of m , the ratio of low to high quality production changes. Again, the map is simple and may be directly used with the managers of the plant to discuss their problem.

As seen in the above story, this problem is that the value of [H] has two effects. First, it influences directly R_p since the more people we have, the more we will be able to run them around the production cycle. However, it also mentions that the ratio m depends on [H] probably due to a low morale or efficiency effect if [H] decreases too much. To represent these two effects we need to add two signals to the PM used until now (full arrows represented with interrupted lines). The first one goes from [H] to $PF>$ and represents the dependency of RP on [H]. The second one, from [H] to the repartitor R modifies m for the low morale or efficiency effect. We will deal with signals in the next section. However in preparation for this section, I have added them on panel C using notations similar to those used in SD. If we have to add many such signals, we will again get a diagram with the spaghetti-like aspect familiar to SD practitioners. The goal of the next section will thus be to find a way of adding signals while minimising the trouble they create.

Finally, panel D presents the PM of the enzyme system discussed in section 2 (the biochemist's sentence). Substrate [S], enzyme [E] and cofactors [CF] get together in a first reaction $P1>$ forming a complex [C] which is transformed by a second process $P2>$ to give a product [PR] and recycle the enzyme. The more enzyme we have to cycle in the production loop, the higher is the production of [PR]. The product [PR] is then used by a third process $P3>$ which sends it into the environment. Since the cofactor [CF] was attached to [S] and [E] by $P1>$, it is present in [C]. Since it is not liberated by $P2>$, it is still attached in some form to the product [PR]. We see this by the fact that once $P3>$ uses [PR], it recycles the co-factor [CF]. Thus if the process $P3>$ does not work i.e. if the product is not used, the cofactor accumulates in a form linked to [PR]. If this occurs, the [CF] stock is empty and no co-factor is available to attach to [E] and [S] in $P1>$. Consequently $P1>$ stops and is rate limited by [CF]. The whole process is strongly reminiscent of a JIT process. Indeed, [PR] is not produced if it is not used in $P3>$ and the cofactor plays the role of the Kan Ban which is just a tag playing not a physical but an informational role in production. It is attached to the product and liberated only if the product is used. The kan ban is then recycled to control the production from its end-use point. Remark that the PM itself is a critical tool in the preceding discussion, a fact which provides a strong argument for our thesis that PMs are excellent tools for SD modelling.¹³

¹³ The PM suggests thus a deep analogy between enzymatic processes and production management. Co-factors were evolved by Nature a million years ago and Kan Ban by Toyota several decades ago. The fact that KPGs suggests this analogy is a strong argument for their intuitiveness. However, we can go further : applied to biochemistry, KPGs show graphically many deep control mechanisms. Co-factors are just one of them. Considering that our body has evolved to be reasonably efficient, could we use biochemistry as a source of metaphors for production management ?

A PM is not only intuitively appealing, it is also a good theoretical tool. Indeed, its state equations are given by the following “*GM- structural flow matrix equation*”:¹⁴

$$\frac{d\mathbf{Q}}{dt} = \mathbf{S}\mathbf{r}^T$$

with \mathbf{Q} an array listing all the contents of the stocks of the model and \mathbf{r} a vector listing all the rates of the various processes. \mathbf{S} is the packing/unpacking connection matrix with rows corresponding to the stocks and columns corresponding to the rates. The element S_{ij} of this matrix is defined by:

$$\begin{aligned} S_{ij} &= +n && \text{if process } j \text{ produces } n \text{ units in stock } i. \\ &= -n && \text{if process } j \text{ consumes } n \text{ units from stock } i. \\ &= 0 && \text{if process } j \text{ do not produce or consume units of stock } i. \end{aligned}$$

If a process is recycling or auto-catalytic, i.e. if it consumes and produces units of the same stock, the corresponding element of \mathbf{S} is the sum of two terms, one for consumption and one for production. Formally, we may define the model “*GM-structure*“ as given either by its PM or by \mathbf{S} . Any information related to the definition of the rate vector i.e. of the functions determining the dependence of \mathbf{r} on stocks and other auxiliary variables values is not GM-structural but kinetic. In chemistry, \mathbf{S} is a constant matrix and formal kinetics study structural properties independently of the kinetic equations (conservation invariants, steady states, cycles, zero-deficiency). However, our coefficients may be modulated, i.e. dependent on the content of the stores and of external variables and we have :

$$\frac{d\mathbf{Q}}{dt} = \mathbf{S}(\mathbf{Q}, \mathbf{U}, t)\mathbf{r}^T \quad \text{with } \mathbf{r} = \mathbf{r}(\mathbf{Q}, \mathbf{U}, t)^T$$

This equation is much more complex and, in general, the study of its global properties may only be done by simulation. However, for local analysis around a given operating point, we may consider \mathbf{S} as « locally constant » and then, the link with chemical kinetics made by PMs makes all the concepts of formal kinetics available for the mathematical analysis of SD models by distinguishing three levels of properties:

- Strictly GM-structural*: valid for any equations and values of the coefficients and of the rates.
- Weakly GM-structural*: valid only for specific equation forms or values of the variable coefficients.
- Kinetic*: considering all the information (structural + kinetic).

As explained before and as the preceding two equations make obvious, the full definition of a model by its PM needs a preliminary step : the determination of its rate vector \mathbf{r} and of its packing/unpacking coefficients i.e. of the elements of the matrix \mathbf{S} . This is done by specifying the equations :

$$\mathbf{r} = \mathbf{r}(\mathbf{Q}, \mathbf{U}, t) \quad \text{and} \quad \mathbf{S} = \mathbf{S}(\mathbf{Q}, \mathbf{U}, t)$$

Where \mathbf{r} and \mathbf{S} may contain arbitrary functionals. Graphically, specifying these equations is equivalent to drawing the previously defined GI-network. Adding it to the PM fully defines a model which we call then a « *Kinetic Process Graph* ». Since \mathbf{Q} is known at each time t (initially by initial conditions and later by integration), the simulation of a KPG is then done by following at each time t a two step procedure :

- Compute $\mathbf{U}(t)$ and use it with the GI network to compute \mathbf{r} and \mathbf{S}
- Use \mathbf{r} and \mathbf{S} to compute $d\mathbf{Q}/dt$ and integrate to get the value of \mathbf{Q} for the next time t of integration.

At this stage, the only step remaining to define a KPG is thus to specify its GI-network. We will do this by using, like in SD, auxiliary signals linking the variables in \mathbf{Q} and \mathbf{U} to the various rates and modifiers. However, we have seen that the GI-network is responsible for the visual complexity of the SD models. We will thus have to introduce some new concepts to obtain maximally simple GI-networks and we will see that again the theory of bond graphs will give us the tools required. This will be explained in section 6 but there is a last point to discuss regarding PMs : all the examples discussed until now have indeed be very simple and, from the point of view of standard SD practice, a bit exotic (chemistry, ecology). Will we observe the same parsimony and intuitiveness in more complex examples chosen in mainstream SD i.e. in Business Dynamics ? As we said before, models of complex models have to be complex but not more than needed. Can we use PMs of mid to large complexity with the same ease than simple PMs ? Our next section answer this question.

¹⁴ The reader will notice that we use the word « structure » to indicate the physical arrangement of stocks and flows. This is different from the usual SD practice which uses the word « structure » to denote the arrangement of causal loops.

5. SCALING UP PROPERTIES OF PROCESS MAPS

To investigate the scaling properties of PMs regarding parsimony and intuitiveness, let us discuss a more extensive example in Business Dynamics^{15, 16}. Its size is probably typical of many studies of mid range in complexity. We will present first a verbal description of the system to model and then show that it may be expressed naturally by a PM.

Problem Description : HOTSTUFF LTD makes high quality « model Ts » better than the « model Xs » of their rival company PLAINSTUFF. HOTSTUFF sells its Ts but can also use them in steps of their own production (e.g. a software company producing OO-development systems or a mechanical engineering company producing lathes used in the production of some of their own critical parts). HOTSTUFF must obviously prime its production by using the Xs of PLAINSTUFF but hopes to gain a competitive advantage by replacing most Xs by Ts to depend less on PLAINSTUFF and to reach leaner production and higher quality. Here are excerpts of HOTSTUFF's CEO's last talk to the board (annotated for further reference):

« As you know, the new production plant for our model Ts is almost fully operational and we have already started production at 60% of our capacity (d). We expect to be at full speed in about 6 months (e). We are thus busy to progressively re-assign to the management of this plant most of the specialists from corporate and other divisions who are now supervising its development and test. Unavoidably, we will loose some people but we hope to keep most of the best ones. Indeed, our investment in development will soon decrease and we will re-invest a major portion of the funds liberated to support plant operation (f). This should allow us to give attractive conditions to these people. A second critical factor for the success of our new venture will be the cumulative boosting of our quality level and the increased productivity due to the smooth but massive introduction in our production lines of our new « model T » tools. This will decrease drastically our production costs since we will not need PLAINSTUFF's model Xs anymore (a) and since we will be able to stop paying their expensive maintenance contract (b). In addition, this will also allow us to re-engineer our lines and get a leaner, more efficient production process (c). Our vision is to reach a dominant position during the second financial year after kick-off. We have thus already started to build up our sales force which should start being operational in 8 months. Its progressive strengthening will generate enough income to break even in about two years, a time at which we expect the whole plant to become self-supported.»

Starting from this text, we can certainly produce a classical SD model but, instead, we give in Fig.7 a PM representing the main processes deduced from the CEO's talk. We see a decomposition in sectors (Procurement, Production, Sales and Marketing, Training and Plant Development) which, to stay in reasonable limits, are all represented by extremely simplified PMs. Let us start with the procurement sector. It shows a feature which has not yet been described. Indeed, all its elements are drawn with unfilled symbols. They denote not scalar elements like those described previously but *vectorial or array elements* manipulating not single GM-quantities and flows but arrays of such entities. We will explain this notation later. For now, let us just say that P-procurement> computes two flows of resources : a flow of model Xs bought from PLAINSTUFF and a flow of various aggregated resources including raw materials. These flows separate at a junction which we meet for the first time, a *U-junction* from which model Xs and resources go in their respective stocks. The procurement process needs an input from outside (the cloud) and an input of money from a store [\$ ®] where ® indicates that it is a « *repeat-stock* », i.e ; a « *conceptually* » unique stock with phantoms drawn at several places just to get a nicer drawing. In this case, the real stock [\$] is defined on the right side of the figure in the Sales and Marketing sector. The symbol D in this stock tells us that it is its defining place and that it is repeated elsewhere. The procurement process is related to the sentences marked a, b, c in the CEO's talk : a and b are obvious, c expresses that model Ts are more efficient than model Xs. Obviously, the procurement rates and the number of money units used by the process have to be controlled in time to represent the « *progressive switching* » of procurement from model Xs to model Ts but these signals are not drawn on the PM, they should be added at a later stage when building the full KPG.

¹⁵ We need also to know if the primitive concepts of table 1 are sufficient. In fact, to tackle more complex cases, I have had to complement them with a few other elements (array elements, time-varying connectors, conditional stocks, conditional switches....) However, until now, I have developed more than 100 PMs in various fields and the set of primitives needed has remained quite small and does not seem to expand anymore. It is thus probably close to be generic.

¹⁶ In order to cope with space limitations, this example is just a caricature but it is sufficient to make the argument cogent.

Coming now to production, we represent it by a single process which has a significant delay (indicated by D) and thus some “work in process” represented by the internal store [WIP]. It uses various inputs : a $\text{\$}$ -stock of money, the various entities aggregated in [resources], and trained people from the b-junction. Two points needs mentioning. First the repartitor R_1 expresses the fact that we may use either Ts or Xs as tools for production. The repartition coefficient expresses the proportion in which these two stocks are used and may be modified by a policy signal (non illustrated). Secondly, the flow of manpower from [Trained people] is represented by an interrupted bond. This is a shortcut notation for «an enzyme stock» i.e. a resource which is both an input and an output of a process with equal packing coefficients. It participates in the production and is a factor in the computation of its process rate but is not used by it. Remark that the stock [WIP] immobilises some trained people for a while. The model Ts produced go either in a stock [model Ts for sales] or in the stock [model Ts for production]. The proportion affected to each of these two stocks is defined by the repartitor R which is also modulated in time by a policy signal (non illustrated).

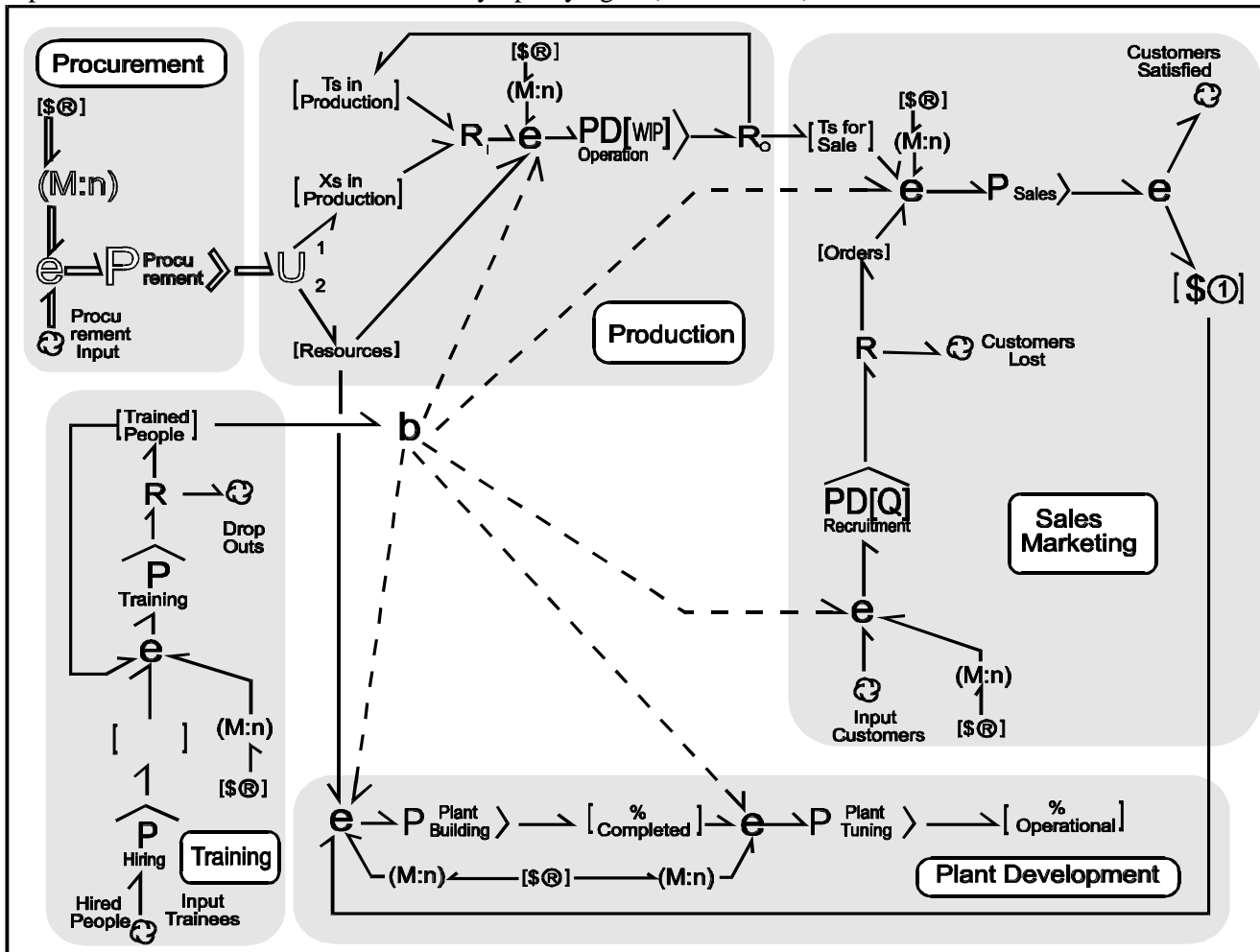
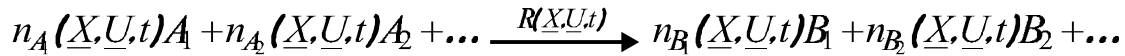


Fig.8 : an example of PM of intermediate complexity.

The production sector just described gives us the GM-structures needed to translate the sentences marked d, e and f in the CEO's talk by adding signals (non illustrated in Fig.8) to modify its various coefficients. A signal from Plant Development should be added at a later stage to allow a progressive increase in capacity of production and a signal from the same origin should modify the packing coefficient of money used in production to express a freeing of funds. The other sectors may be described similarly. The sales and Marketing sector shows a customer recruitment process needing trained people (sales force) and obviously money and producing orders and loss of customers. The orders are combined with the Ts to produce money spent everywhere in various proportions. The training sector hires and trains people to distribute them to various processes through the b-junction. Finally, the plant development sector computes the operational capacity reached. As said before, we should now add signals controlling all these processes and creating many feedback loops. Remark nevertheless that, in contrast with a SF-diagram the PM which describes just the basic «Think Physics» infra-structure of the model shows already many loops, a fact decreasing the complexity of the signal network which should now be added.

6. KINETIC PROCESS GRAPHS : ADDING CONTROL SIGNALS TO A PM

To define the GI-network, we have thus to represent graphically the computations of the rates and modifier coefficients. As shown in Fig.6-A, they are in general, functions of several kinds of variables: contents of stocks, external inputs and auxiliary variables. Indeed, let us consider again the general process equation :



\underline{X} is the vector of stocks on which the process depends. \underline{U} is a vector of input signals and t is time. The A_i are the donor stocks and the B_i the acceptor stocks. The $n_{A,i}$ are the donor modifier coefficients and the $n_{B,i}$ are the acceptor modifier coefficients. Finally, $R(\dots)$ is an operator defining the process rate. We can partition \underline{X} in two subvectors :

$$\underline{X} = (\underline{X}_c, \underline{X}_d)$$

where \underline{X}_c (close stocks) contains the donors A_i and acceptors B_i appearing in R and \underline{X}_d (distant stocks) contains stocks appearing in R but which are not donors or acceptors of R . In chemistry, R is given by an algebraic formula and $n_{A,i}$ and $n_{B,i}$ are constant. However, in our case, they may be much more complex, including for instance their own state variables, time delays, functionals, expert systems, fuzzy rules, neural networks or even human interactions. We call « *influences on R* » the variables appearing as arguments in R . We may distinguish two classes of influences according to their level of proximity to the flow:

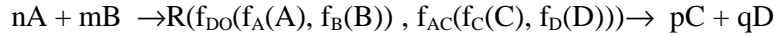
- “*The class of close influences*”: the dependencies on \underline{X}_c i.e. on the donors or acceptors of the process. In many models, a large, often major, part of the influences belongs to that class. Most processes depend indeed on their donors. For instance, if they are not available, the process cannot take place and often their level of availability influences the rate in subtle ways. Moreover, quite often, at least in controlled processes, the level of acceptors is also a controlling factor through laws like «The more products I have, the less I want to produce them » making R an inverse function of its acceptors. We have already given an example of close influence (see box C in Fig. 7-C)
- “*The class of distant or indirect influences* » which contains all the other variables appearing in R :
 - Distant stocks \underline{X}_d : stocks are the most important variables in a model. It is therefore obvious that many finely tuned processes must depend from stocks which, being not direct donors or acceptors , are more distant in the graph but nevertheless influence its rate through some control mechanism. An example is the final product of a production chain , the availability of which should influence all the steps by controlling the rate of the chain of processes.
 - Input signals, i.e. time functions $U_1(t), \dots, U_m(t)$ representing influence dependencies (different from sources or sinks) from the non-modelled environment.
 - The time variable t itself (which may always be assimilated to an input signal $y=t$).
 - Auxiliary variables defined from the above close and distant variables and used either in the computations (e.g. the monthly average of a flow of orders controlling a production rate¹⁷) or as intermediates in the computations.

Some models have only a few distant influences. However, in many socio-economic models, they are encountered much more frequently creating thus the well known dizzying feeling that everything depends on everything. Moreover, some distant influences may act on many processes (e.g. in a socio-economic model, an auxiliary block computing an estimate of the economic conjuncture would influence most processes).

It is the explicit representation of all these influences by signal arrows which creates the maze characteristic of large SD-models. We need thus to find different ways of representing them. Our first effort must bear on close influences which are, by far, the most frequently encountered.

¹⁷ Algebraic loops and incompatibilities may result from defining flows as functions of other flows. In general, we cannot exclude these possibilities. However, frequently, a flow-dependency occurs through an intermediate operation like averaging and uses only the past values of the flow in a given horizon window.

To represent close influences as parsimoniously as possible, we will adapt to KPGs a notion of effort-causality imported from the theory of bond graphs and which in our case will become an « *influence causality* ». To introduce it, let us consider a simple process :



in which the modifier coefficients are constant and the rate depends on all the donors and acceptors. We have represented the rate function in a general manner emphasising its various levels of dependencies. A first function f_{DO} expresses the interaction between the influences of its donor stores. These influences themselves are expressed individually by the functions f_A and f_B . The same kind of representation is used for the acceptors. A global function f_{AC} expresses their interaction and each of their participation to f_{AC} comes through individual functions f_C and f_D .

Fig.9-A represents a possible notation for the KPG of this process. Its PM is reproduced from Fig.7-A. It is transformed into a full KPG by adding a GI-network (full arrows and circles in grey) with four operations. First it measures the values of the stocks and sends them to auxiliary blocks computing f_A, f_B, f_C, f_D . Then these blocks compute the four individual influences. The third operation is to send these influences to the blocks f_{DO} and f_{AC} which compute their interactions. Finally, the results of these two blocks are sent to P> This representation is equivalent to the SD representation of a tap dependent on four stores. For a single process, this representation is acceptable but, if we have many processes in a network, each dependent on its donors and acceptors, the resulting graph will soon become cluttered with signal arrows.

The panel B of Fig.9-B is equivalent to the panel A but we redraw the information links in parallel with the bonds. We also place the individual functions f_A, f_B, f_C, f_D near the modifiers and the blocks f_{DO} and f_{AC} near the e-junctions. Apparently not much is gained. However, let us focus our attention on the parts indicated by grey filled circles. Each circle encloses an element (respectively a stock, a modifier and an e-junction) coupled to an operation on influences (respectively a measurement of influence in the stock, a function f_A of one argument A near A and a function f_{DO} of two arguments $f_A(A)$ and $f_B(B)$ near the e-junction. Let us remark also that each bond is now in parallel with a signal link going either in the same direction than the bond in the donor side or in the reverse direction on the acceptor side.

We may then redefine our basic elements like in panel C by putting, in an element itself, the operation on an influence placed close to it in the grey circles of panel B. With these new definitions, each element has two functions, one on a GM-quantity and one on an influence. The functions on GM-quantities have been defined before (table 1) and do not change. We will now describe the functions on influences.

- A stock (panel C-a) is now denoted $[Q, I_Q]$. Like before, it stores a GM-quantity Q but now, in addition, it also generates an influence $I_Q(Q)$. Often we have just $I_Q=Q$ and we may drop the indication of I_Q getting back the former notation $[Q]$. Remark that Q being a GM entity is conserved : if it flows through a bond, its value changes necessarily. On the other hand, I_Q is a signal and thus sending its value somewhere else in a signal link does not modify it.
- A modifier (panel C-b) is now denoted by $(M: n, f)$. Like before, it transforms a GM-flow by multiplying it by its coefficient n but in addition, it receives an influence I, transforms it by the function $f(I)$ and send the result outside. We need to distinguish two cases (panel 9-C-b). A modifier in the donor side of a process has a GM-flow and an influence running in the same direction (left side). In a modifier placed in the acceptor side of a process, the directions of the GM-flow and the influence are opposed.
- The e-junctions are now denoted by (e, g) (panel 9-C-c). Their operation on GM-flows is unchanged but now, they also receive two influences I_1 and I_2 , operate on them by the function $g(I_1, I_2)$ and send the result outside. We see in panel B that again, we need to distinguish two cases: an e-junction in the donor side in which signals and GM-flows run in the same direction and one for the acceptor side in which they run in opposed directions.

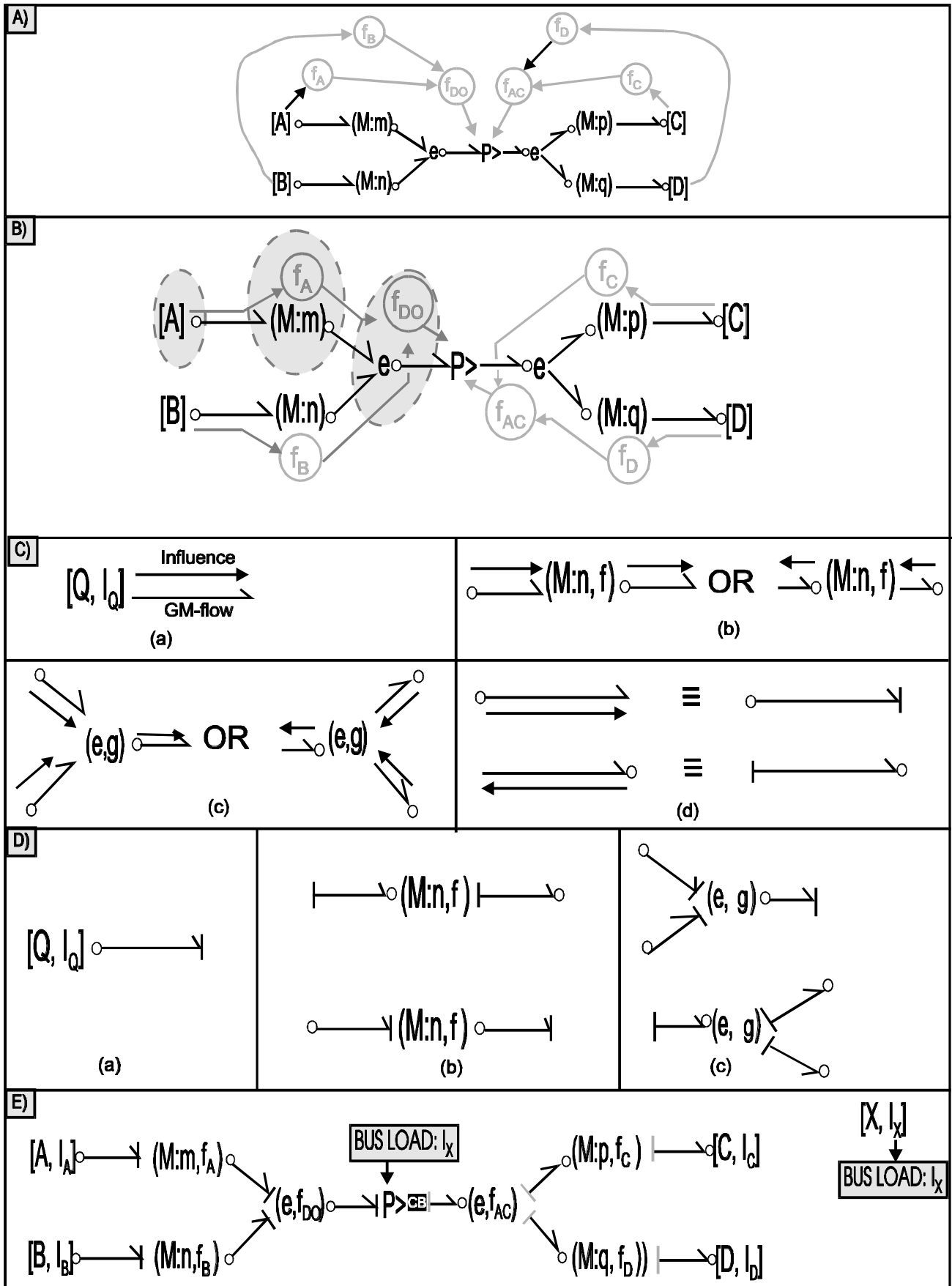


Fig.9: Introduction of influence causality strokes to eliminate the signal arrows denoting close influences.

With these new definitions, each bond joining two elements runs in parallel with an influence link which may have the same direction than the bond or the reverse direction. The two possible cases are represented in Fig.9-C-d (left side). Instead of considering explicitly the pair bond-link, we choose the notation indicated on the right side of this panel. This notation is imported from bond graphs. The bond is drawn with its flow causal marker (the circle) but instead of drawing the influence link, we indicate just its orientation by putting a small bar, called an « *influence stroke* », perpendicular to the bond on the side pointed by the full arrow of the influence link. Remark that, in each case, the flow causality marker (little circle) and the influence causality (the stroke) are opposed. Like our other elements, a bond has thus now two functions : it transports a GM-flow with a value determined by the element connected to its side without circle (where the stroke is) and imposed to the side with a circle¹⁸. On the other hand, it also transmits an influence from the side without stroke (where the circle is) to the side with stroke.

Using this new notation for a pair (bond-influence link), we may redraw the elements of panel C (a,b,c). This is done in panel D which shows the store (sub-panel a), the two cases of the modifier (sub-panel b) and the two cases of the e-junction (sub-panel c).

Finally, using the notations of panel D, we may redraw the complete process given in panel B. The result is given in panel E which is the full KPG of this process. Its GM-network is now completed by a GI-network which is represented unobtrusively but unambiguously by the causal markers and the symbols of the elements. Following these causal markers it is indeed possible to see the complete path of the influences. For instance, starting from [A] and [B], we see two strokes on their output bonds indicating that influences $I_A=A$ and $I_B=B$ arrive at the modifiers (M: m, f_A) and (M: n, f_B). These modifiers generate the influences $f_A(I_A)$ and $f_B(I_B)$ which are then sent by the two strokes seen in their output bond to the e-junction (e, F_{DO}). This junction generates the influence $f_{DO}(f_A(A), f_B(B))$ and sends it to the process P> by its output stroke. A similar reasoning shows that the acceptor side generates the influence $f_{AC}(f_C(C), f_D(D))$. The process receives thus the appropriate signals on its two causal strokes. The influence from the donor side is called « *forward influence I_F* » and the influence from the acceptor side is called the « *backward influence I_B* »¹⁹.

It happens frequently that a process does not depend on a specific close influence. For instance, the process discussed here may be independent of C or of both C and D. In such cases, we still draw all the causal strokes to check their eventual correctness (to test the validity of the connections made in the model) but the causal strokes manipulating influences which are not needed are drawn in grey and we indicate that they are not used by putting a little black square called an « *influence blocker IB* » which we place just near the unused influence closer to the process. For instance, if the process discussed here does not depend on C, we will place a blocker before the arrival on the junction (e, f_{AC}) of the influence $f_C(I_C)$ (case non illustrated in the figure). If it does not depend on both C and D, we may block directly the backward influence I_B . This is done by putting the influence blocker at the output of P>. When we see a blocker, we know that the influence computations leading to the blocked influence do not need to be executed.

It remains to describe the process P> itself. This is very simple : it just receives the influence signals I_F and I_B and computes its rate as a function $R=R_P(I_F, I_B)$.

¹⁸ We remind the reader that a GM-flow may be positive or negative and thus may flow in the direction of the semi-arrow or opposite to it independently on its flow causality (the circle) which indicates just the side which receives the information on the flow value which it has to provide or to receive.

¹⁹ These names are again inspired by a chemical analogy. Indeed, let us choose the following definitions :

(1) For $X=A,B,C,D$ we impose $f_X=\ln(X)$.

(2) For any I_1 and I_2 , $f_{DO}(I_1, I_2) = f_{AC}(I_1, I_2) = I_1 + I_2$

(3) $R = k_F \exp(f_{DO}) - k_R \exp(f_{AC})$. (k_F and k_R the forward and reverse kinetic constant ; k_R is 0 if the reaction is unidirectional)

Then the computations described by the KPG of Fig.9-D are close to the classical thermodynamic view of a chemical reaction controlled by the mass action law. We just need to add formulas for entropy and temperature to get a coherent thermodynamic model but then the result, with some changes of notations becomes a bond graph and is no longer a KPG. In that case the signals f_{DO} and f_{AC} are respectively called forward and backward affinities. This establishes a close filiation link between our version of SD described by KPGs and physical lumped system modelling.

This completes the description of our treatment of close influences. We have dealt only with four elements : stocks, modifiers, e-junctions and unidirectional rates. We do not have the space here to present them completely but transformations similar to those of panels C and D may be done for all the other elements of table I (constant stock, bi-directional process, process with delay, source, sink, balance junction and repartitor). It is then possible to specify the part of any GI-network dealing with close influences entirely by influence strokes. All the arrows indicating close influences are thus eliminated and replaced by strokes. This is obviously much less visually cumbersome than the SD notation and the topology (pattern of connection) of the PM is fully preserved. The addition of the GI-network of close influences preserves thus the two qualities of a PM : its parsimony and its intuitiveness.

The use of causal strokes presents an additional advantage. Indeed, on each bond, the flow marker and the stroke must be placed on opposite sides. Since we have seen that flow causal markers have specific patterns for each element, it follows that causal strokes also have specific patterns (i.e. laws of stroke placement) which are easily deduced for each element by taking the dual of the flow connection patterns given in table 1. These patterns of connection may be checked algorithmically. The software implementing KPGs is thus able to detect wrong patterns, i.e. connection mistakes. It may then issue warnings prompting the modeller to correct his KPG. This self-checking feature of KPGs speeds up model development appreciably.

To complete the specification of the GI-network, we still need to consider the representation of the distant influences acting on three kinds of elements : (1) processes, (2) packing coefficients of modifiers and (3) variables of the functions which we have now inserted to process close influences in stocks, modifiers and e-junctions. Obviously, if a model contains many distant influences and if we want to represent each of them graphically, we will probably end up again in the complexity trap. Indeed, the only way to represent these influences is to draw a network of auxiliary nodes and signal links like in SD. KPGs use several means to maintain this complexity to manageable levels:

- A BUS of signals has been introduced. Signals may be posted on it by associating the element defining them and an « *Entry point* » in the BUS. A BUS signal may be used as an influence in any element accepting a distant influence by placing an « *Exit point* » from the bus near the influenced element. This is illustrated in Fig.9-D where we have represented a stock [X] which is neither a donor nor an acceptor of the process represented. Its influence is placed in the BUS entry point and used in the process by downloading it from a BUS exit point. The availability of this BUS makes possible to represent explicitly by signal links the influences central to the problem studied while keeping all the other in the BUS. The repartition of the signals which are explicitly shown may be changed explicitly before a simulation in order to focus on different aspects of a model.

- The implementation of the GM and GI-networks are completely modular and hierarchical. Any part of a KPG may be defined as a sub-model which may be connected to other elements or sub-models through contact points called « ports ». It exists two kinds of ports : GM-ports which accept only a a bond and GI-ports to which we must connect an influence link. A sub-model may have a fixed but arbitrary number of ports. Graphically, a sub-model is encapsulated in a single box with an icon and a name chosen by the modeller. A sub-model may contain any other sub-models. A KPG may thus be defined hierarchically and the number of hierarchical levels is practically unlimited.

- Sub-models may be defined either directly as KPGs or by using other methods : block diagrams, bond graphs or even directly in an object-oriented language specific to the implementation software 20-SIM. This allows the representation of sophisticated influence computations like neuro-fuzzy controllers, decision maps or sets of rules in a very clear way.

- Finally, I have extended the KPG method to accept array variables in order to deal efficiently with disaggregated models (e.g. people disaggregated in various spatial areas or in various age classes, products differentiated by their quality level...). Fig.10 illustrates the graphical array vectorial notation.

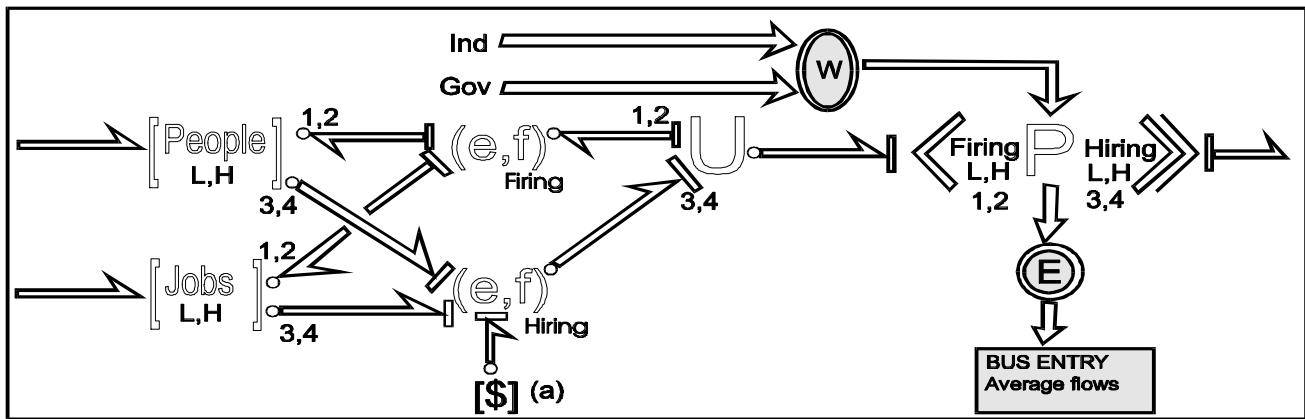


Fig.10 : Array notation for disaggregated KPGs

Array elements are indicated by symbols drawn with hollow letters and lines. The figure shows a simplified version of a small part of a model built for the Cabinet of the Belgian Vice Prime Minister in order to evaluate new proposals of employment policies. The part of the model illustrated here distinguishes two classes of people²⁰ according to their level of qualification (L=low and H=high). The stock of people contains thus two stocks, one for each class L or H. These two stocks form the component of an array of dimension 2. The stock of jobs is similarly indexed. The process is bi-directional with a left side indicating firing of people and the right side indicating hiring. The hiring direction corresponds to positive flows (double symbol >>). This process computes thus an array of four individual rates : $Firing_L$, $Firing_H$, $Hiring_L$ and $Hiring_H$ which we denote by the numbers 1, 2, 3, 4. The bonds transport arrays of flows of various dimensions. The bonds connected to the process transport its four flows (some are positive and others negative) but a new element called a « *U-junction* » may combine or separate these arrays. The U junction shown here receives the requirement (flow causal marker) to provide an array of four flows from the process and separates it into two separate requirements : a first array of two components (1,2) is sent to the junction $(e,f)_{Firing}$ and a second vector of two components (3,4) is required from the junction $(e,f)_{Hiring}$. Remark that the hiring reaction also uses money which is not an array but a scalar component.

Array elements may also be used for GI networks as illustrated by the upper part of this figure which shows two arrays of signals coming from the parts of the model generating the factors from industry (I) and government (G) used in policy determination. These factors are combined by the block W weighting conjunctures, decisions and policies. The block W implements a standard fuzzy controller and controls the four rates of the processes. Taken together, the array notations for GM and GI-networks provide a representation of disaggregated models keeping all the qualities of our previous scalar representations. It would be highly difficult to build, modify or use complex, highly disaggregated models without them.

7. CURRENT IMPLEMENTATION, DEVELOPMENT, RESEARCH AND APPLICATIONS

Graphically KPGs are closer to bond graphs than to the SF-diagrams of System Dynamics. Despite the facts that any SF-diagram may be transformed into a KPG and that any KPG may be de-constructed to provide an equivalent SF-diagram, implementation of KPGs in a SD software would most probably need many basic modifications to the software. On the other hand, if a BG software is built in an object-oriented way and allows the user to re-define the code and the graphical icons of each element and to add new elements, it is possible to adapt it quite simply to KPGs by changing all the basic laws of bond graphs and by replacing the bond graphs icons by our KPG icons. This needs only some modifications to the code of the methods in the library giving the class library of the elements. The bond graph package 20-SIM fullfils our conditions. I have thus used it for developing a complete implementation of the version of the KPG method which has just been described. In this way, KPGs have a full upward compatibility with BGs, a fact which brings much unity to the field of lumped modelling since it unifies SD and BGs, the two methods which are currently most frequently used in this field across the whole spectrum of disciplines.

²⁰ The disaggregation used in the real model is obviously bigger and based on sex, age classes, qualification and work expertise.

The library KPG-LIB provides a complete implementation of KPGs (see trademarks before). KPG-LIB contains our basic and advanced elements, frequently used sub-models, advanced sub-models and examples of applications. Elements supporting time-varying connections, self-organisation and dynamic reconfiguration of a model are also included. Finally, a library providing a hierarchical decomposition of multi-variable fuzzy rules was developed to provide a generic implementation of GI-networks from interviews of domain specialists giving us sets of fuzzy qualitative rules. As explained before, the KPG implementation uses the capability of 20-SIM to the full : we benefit indeed from its modular and hierarchical structure and from the possibility it offers to mix sub-models of different types (code, block diagrams, bond graphs, and now KPGs).

We may mix sampled and continuous models and even study much less classical types of systems. For instance, in a study of bacterial evolution, one of my PhD students is currently integrating a genetic algorithm with 20-SIM. In the model, bacteria are described by a gene which may mutate. Each gene existing at a given moment defines a stock storing the bacteria having that gene. The processes are the typical processes of bacterial life : growth, poisoning, infection by phages... Some population die and other are generated by mutation. In response to Natural Selection and to co-evolution simulated in the genetic algorithm, the number of stocks in our graphs may change from a few to several hundreds The KPG structure is thus constantly adapted. This system has two levels of dynamics : even when the number of stocks is constant, its variables show interesting dynamics like chaos or hyper-chaos but, in addition, the number of stocks and processes changes frequently. The topology of the graph is thus itself a dynamic object of a higher order controlled by the genetic and extinction rules. We call these rules a « meta-dynamics »²¹. Since computation of such high dimensional systems would be very slow, we are now busy defining KPG elements implementing finite automata and the whole model becomes then similar to a cellular automaton with modifiable connections.

Fig.11 illustrates our implementation by showing screen dumps of a small model developed for the previously mentioned study of the employment market in Belgium.

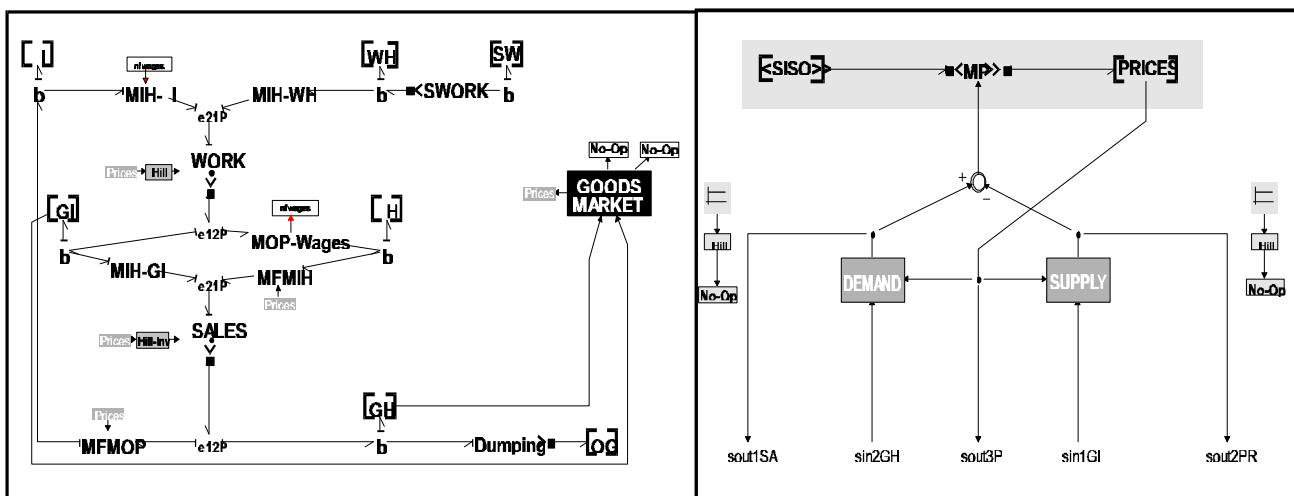


Fig.11 : Screen Dumps showing an example of KPG model implemented in 20-SIM

The left panel shows the model. We will not describe it in detail. We just want to pinpoint a few points of interest. First remark the BUS defining and using signals. Even in a small model like this one, it provides a very much needed simplicity. Secondly, remark that exception made of the few signals transported by the BUS, this model does not contain any distant influences. All the influences are close and transmitted by the causal strokes. The structure of the PM is thus clearly shown and easy to follow. We can use this model to

²¹ By providing adaptive SD models which change not only the values of their parameters but also their topology, this study links KPGs and thus SD to the field of complex adaptive systems as studied in the Santa Fe Institute or at the Ackhoff Center in Philadelphia. Indeed, it would probably be possible to define SD models of evolving industrial or economical eco-systems.

discuss with specialists directly in front of the computer during a simulation. This would be much more difficult if the screen was filled with a maze of signals. Thirdly, the block called « goods » market is a sub-model, if we click on it, we can see its structure which is shown on the right panel. It contains a KPG (upper part) and various sub-models which in turn are defined by block diagrams and code.

To end up this brief presentation of KPGs, let us describe their current status of development.

The basic 20-SIM implementation is complete but the library of elements is still under development. All the elements described here are implemented but the library of sub-models needs to be completed and extended in a major way. The current version of the library of fuzzy and neuro-fuzzy blocks is minimal and can only be used to express simple rules and policies. It is used more and more in policy studies and a major effort is needed to extend its scope and implement it in a more generic and flexible way.

Another important direction for the R/D effort on KPGs is the development of a generic framework for the GI-networks for business and organisation dynamics and more generally for managed systems. Using Morecroft's ideas on policy structuring and the "information-decision-action" framework it should be possible to define generic purposive managerial agents which could then be made to act in co-operation and/or competition in a mix of decentralised organisation coupled to hierarchical structure. Each agent would control some processes and interact with other agents in being controlled by the others or controlling them. They could also share the control of some processes. Adaptation of such agents to specific cases might provide a way to model managed systems by injecting a bit of « theory of organisations » in SD.

Finally, more research is needed on adaptive networks with meta-dynamics and on systems with time-varying connections which we might use for instance to study re-engineering or re-organisation in companies. Although the use of genetic algorithms to evolve optimal policies is tempting, it is still beyond what is possible. Indeed, the simulation of the invention of an innovative policy involves not only parametric adaptations but also and mainly the invention of new decision rules and the modification of the structure of a system. Designing new rules is now possible by genetic programming which evolves syntactic structures and not bit strings. However, new policies need new algorithms to build new observables and these observables may be quite sophisticated. Designing them is probably at the border of what genetic programming can achieve in its current state. Designing new structures by adding or deleting stocks, modifiers and processes is still more difficult. This necessitates the addition to a model of new stocks and processes with a well-defined interpretation or semantics. Finding automatically this semantics is currently impossible but, as demonstrated by our work on bacterial cultures, applying genetic programming to a pre-defined (eventually huge) library of stocks with pre-specified semantic interpretations is possible and could provide a good test-bed for studies on re-engineering or design of new managed systems.

Coming now to practical matters, the main applications of KPGs pursued at IDEA.SIM LTD have been the following: human resource management for the EU communities (DGXIII); effect of JIT on human resources in retail for the KSRC store chain (London); surgery cancellations for the Guy's and St Thomas Trust (London), sustainable re-development in the Luberon area for the Departmental Council of Vaucluse, France; prospective study of the effect of new policies in the employment market for the Belgian vice-Prime Minister and coupling between evolution and population dynamics in multi-bacterial populations with the Institut Pasteur de Lille. In addition, many smaller models have been built in various fields from biochemistry to business dynamics. This is still a small sample of applications but they show that KPGs are generic enough to support the development of intuitive and parsimonious models for the whole range of applications usually encountered in SD.

Since 1995, various versions of KPGs have been taught to a total of about 200 students in three EU-supported MSc degrees (Bioengineering in Lyon and London, Medical Informatics in Athens, Engineering in Lille). Although the present paper is our first attempt to contact the SD community, KPGs have been presented at Keynote Lectures at the European Congress of simulation EUROSIM – Helsinki, 1999, at the 6th World Congress on Bond Graphs ICBGM 1997 and at the American Congress of the Society for Computer Simulation, Phoenix, 1997.

8. CONCLUSIONS

- In recent addresses to the SD Society, two of its presidents wrote that, to expand and be better accepted, as a mainstream activity, SD had, among other priorities : (1) to develop research activities focusing on our methodological tools for problem definition and conceptualisation [27], (2) to establish bridges with other disciplines and more mainstream modelling communities while retaining our identity [28]. I would like to suggest that the KPGs method may provide a step forward in each of these directions.

- Indeed, by grounding its models on a deeper look at the « Physics » of a SD application, i.e. at its GM-flows and transformations, it provides a strong methodological foundation for problem definition and provides a conceptualisation tool focusing on physical, structural insight.

- Secondly, by bridging almost seamlessly with classical SD on one hand and with bond graphs and object-oriented modular, hierarchical modelling methods like MODELICA [29] on the other hand, it establishes a link with engineering disciplines and creates a common language for SD people and for model users in more technical fields while retaining entirely the SD identity and unique point of view. KPGs should thus favour co-operation between these communities. For instance, people in marketing using a KPG to study the potential growth of their market might find that they are not as far as it may seem from a team in concurrent engineering using bond graphs for the prototyping of a new car. This common basis should provide one of the pre-conditions for improving communication in a company, promoting thus its development as a truly integrated learning organisation.

- KPGs alleviate the mental abyss and visual complexity problems seen by Richmond as plaguing our current SD practice [14]. They include the SF-diagrams of SD as particular cases and are thus conceptually compatible with them. They may form a potential basis for a new generation of SD tools. Their current implementation in 20-SIM allows a fully modular and hierarchical development of large scale models which benefit from the numerical power and speed of a modern engineering-oriented simulation software. This is very good for model development and test. However, to develop management flight simulators with user-friendly interfaces and multi-media education, our implementation does not even come close to the current SD packages. Integration of KPGs in such packages would thus be highly desirable. Moreover, let us recall that KPGs do not replace but only extend current methods when our mental models are characterised by general processes. This is obviously not always the case and if a model has many qualitative soft levels and a small GM-part or even no GM part at all, KPGs will not be more adapted to its modelling than SF-diagrams. However, our implementation of modularity and hierarchy and the availability of fuzzy and rule-based policy blocks might still be interesting.

- The price to pay for the greater expressiveness and parsimony of the KPG models is that they use a few more « basic concepts » than SF-diagrams (see table 1 and the section on causal strokes). This may be a problem. Indeed, if we consider that, for some users, even SF-diagrams are too abstract, we will not be astonished if they initially consider KPGs as a game for theoreticians. However, our experience shows that the basic KPG-syntax may be learned in about a day. After this time, a non-mathematically minded person may start developing small models. Obviously, practice makes perfect and it takes much more time to become an expert in KPG development. For a team considering the development of a large model for an in-depth study, this is certainly reasonable. For superficial use of SD, this investment in time may be too big for some customers. In that case, I focus nearly exclusively on the process maps. Almost invariably, the discussion of these maps with customers reveals many points of interest and, as mentioned before, is often easier than a discussion centred on the equivalent SF-diagram.

- An area for development is the strengthening of the links of KPGs with the modelling methods used in other fields. Due to their close similarity, BG and KPG models may be coupled much more easily than SD and BG models. It becomes thus possible to develop hybrid models having a SD part and a more physically-based BG part. For instance a continuous or hybrid model of a chemical plant might be physically based and integrated with a model of its resource management built with a KPG. Similarly, it is desirable to couple for instance a detailed model of a manufacturing plant using discrete event-based methods like DEVS to a KPG model of the parts of the company surrounding the plant.

- Revisiting SD models with the reaction metaphor modifies our perspective considerably and often results in new hypotheses. For instance the analogies mentioned previously between biochemical control and production management are probably worthwhile to pursue. This suggest that the metaphor of the cell as « a chemical factory » should be considered seriously and studied as a source of inspiration for investigating the possibility of bio-mimicking management procedures to control the efficiency of the internal workings in a company or organisation. To this inside « *metabolic view of a company* » corresponds also a view concerned more with the outside of a company : its stake holders and the environment in which it works and develop. Some modern management scientists speak of a « *business ecosystem* » [30] and I suggest that the analogy between ecosystems and socio-economical or industrial networks should also be taken seriously by SD modellers. Adaptive KPGs with meta-dynamics might just be the tool for this kind of study.

Let us end up by insisting on the fact that until now, KPGs are only useful to model GM-networks. The interpretation of their basic metaphor in cognitive terms as a rule or semantic map points toward similar researches which should now be done for GI-networks for which other cognitive models might also be useful (cognitive or semantic maps [23]). Research linking mental models with cognitive sciences and AI is indeed very much needed in SD. Such links forms the basis of my current work on a generic structure for management agents (see section 7) incorporating adaptation, purposive behaviour and neuro-fuzzy rules.

BIBLIOGRAPHY

- [1] Kamopp D.C., Margolis D.L. and Rosenberg R.C. : System Dynamics, A unified Approach, Wiley, NY, 1990
- [2] LeFèvre J. : An elementary bond graph approach to structured biological modelling, in « Advanced Simulation in Medicine, » D.F. Möller ed. ; SpringerVerlag, N.Y., pp. 10-40, 1990
- [3] LeFèvre J. : Keynote address to EUROSIM 2000, Helsinki, pp.1-9, 2000
- [4] LeFèvre J.: European SCS congress on simulation, Marseilles, pp.122-130, 2001
- [5] LeFèvre J. : Keynote address to the 1999 Int. Conf. on bond graphs, Las Vegas, pp.1-4, 1999
- [6] Forrester J.W.: Industrial Dynamics, MIT Press, Cambridge, Mass. 1961.
- [7] Forrester J.W. : Principle of Systems, Wright Allen Press, Cambridge Mass., 1968
- [8] Forrester J.W.: World Dynamics, Wright Allen Press, Cambridge Mass., 1971
- [9] Forrester J.W.: Collected Papers, Wright Allen Press, Cambridge Mass., 1975
- [10] Sterman J.D. : Preface of reprint of [6], Syst. Dyn. Rev., 2, 158-170, 1986
- [11] Forrester J.W. : Syst. Dyn. Rev., 10, 1994 (draft in Forrester web site)
- [12] Richardson G.P. : Sys. Dyn. Rev.,
- [13] Roberts N., Andersen D. et al. : Computer simulation, a SD approach, Addison Wesley, Reading, Mass.,1983
- [14] Richmond B. : Syst. Dyn. Rev., 10, 135-157, 1994
- [15] Coyle R.G. : System Dynamics Modelling, Chapman & Hall, London, 1996
- [16] Bellinger G. : <http://www.outsights.com/systems/sts/stsf.htm> ,2000
- [17] Sterman J.D. : Business Dynamics, Mc Graw Hill, Boston, 2000
- [18] Forrester J.W. : in « The systemic basis of policy making in the 1990s », De Greene K.B. Ed., 1991
- [19] Stella « Getting Started Manual », HPS, 1997
- [20] Hannon B. and Ruth M. : Modeling biological systems, Springer, NY, 1997
- [21] Andersen D.F. and Sturis J. : Sys. Dyn. Rev ;, 4, 218-245, 1988
- [22] LeFèvre J. & Barreto J. : J. of Franklin Inst., 319, 201-215, 1985
- [23] Russell S. & Norvig P. : Artificial intelligence, Prentice Hall, NJ, USA, 1995
- [24] Anderson J.R., The architecture of cognition, Harvard, Cambridge, Mass., 1983.
- [25] Stella technical documentation, HPS, 1997
- [26] Kosko B. : Neural networks and fuzzy systems, Prentice Hall, N.J., USA, 1992
- [27] Mashayekhi A.N. : Presidential address, System Dyn. Newsletter, December 2001
- [28] Vennix J. : Presidential address, System Dyn. Newsletter, November 2000
- [29] Borutzky W. : SIMPRA, 7, pp. 439-462, 1999
- [30] de Geus A. : The living Company, Nicholas Brealey Publ., London, 1997