SUPPORTING ORGANISATIONAL DECISION MAKING IN PRESENCE OF UNCERTAINTY

Vinay Kulkarni[(a)], Souvik Barat[(b)], Tony Clark[(c)], Balbir Barn[(d)]

[(a),(b)] Tata Consultancy Services Research, India
[(c)] Sheffield Hallam University, UK
[(d)] Middlesex University London, UK
[(a)]vinay.vkulkarni@tcs.com, [(b)]souvik.barat@tcs.com, [(c)]T.Clark@shu.ac.uk
[(d)]B.Barn@mdx.ac.uk

ABSTRACT
Effective organizational decision-making often requires deep understanding of various aspects of an organisation such as goals, structure, business-as-usual operational processes etc. The large size of the organisation, its socio-technical characteristics, and fast business dynamics make this a challenging endeavor. Current industry practice relies on human experts thus making organisational decision-making time-, effort- and intellectually-intensive. This paper proposes a simulatable language capable of specifying the relevant aspects of enterprise in a machine-processable manner so as to support simulation-driven decision-making in presence of uncertainty. A possible implementation of the language is outlined. Validation of the proposed approach using a real-life example is discussed.

Keywords: Decision making; organisational decision making; uncertainty.

1. INTRODUCTION

Modern enterprises need to respond to a variety of change drivers in order to stay competitive in a rapidly changing business context. The cost of erroneous decisions is often prohibitively high and there may not be an opportunity for subsequent diversion (Daft, R., 2012). Minimizing such undesired consequences calls for a-priori judicious evaluation of the available courses of action as regards their influence on the desired objective. The decision-makers are thus expected to understand, analyze and correlate existing information about various aspects of enterprise such as its goals, its structure, business-as-usual operational processes, change drivers and their influences on overall organisation *etc*. Scale and complexity, inherent socio-technical features (McDermott et al, 2013), and multiple stakeholders with possibly conflicting goals all contribute to the complexity of organisational decision-making. Moreover, increasing demands for agility and certainty make this endeavor even more challenging.

A modern enterprise is a large and complex system and the sheer volume of information makes manual analysis ineffective as well as inefficient. Moreover, with the required information pieces typically strewn across multiple sources such as documents, spreadsheets, pictures, logs etc., these pieces need to be stitched together to form a coherent, consistent and integrated view.

Modern enterprises operate in increasingly dynamic environment that must be kept up-to-date at an increasingly rapid rate. The current inability to stitch together an integrated, complete, and timely view makes manual analysis further untenable. The inability to address all these factors to the desired level of sophistication is the principal contributing reason for the current state of organisational decision-making[1].

A pragmatic approach to organisational decision-making therefore seems to hinge on the availability of: (i) the information required for decision-making in a machine-processable form, (ii) suitable machinery for effective processing of this information, and (iii) a method to enable repetitive use of this machinery at the hands of knowledgeable users. Moreover, the form as well as the machinery needs to be capable of respectively representing and processing the inherent uncertainty.

A variety of Enterprise Modeling (EM) languages exist that provide information-capture and analysis support across a wide spectrum of sophistication. Majority of these languages can be traced to Zachman framework (Zachman, J., 1987) advocating that capture of the *why*, *what*, *how*, *who*, *when* and *where* aspects leads to the necessary and sufficient information for addressing a given problem. Thus it can be argued that complete specification of enterprise is possible using Zachman framework, however, there exists no support for automated analysis as the information is captured typically in the form of texts and pictures. It can be observed of the existing EM languages that: the languages capable of specifying all the relevant aspects of enterprise for organisational decision-making lack support for automated analysis (Zachman 1987, Krogstie 2008, Jonkers et al 2004), and the languages capable of automated analysis can cater to specifying only a subset of the aspects required for decision-making (Yu et al 2006, Meadows and Wright 2008, White 2004).

Co-simulation using a relevant set of EM languages can be a pragmatic solution. For instance, as shown in Fig 1, i* (Yu et al, 2006) to specify the why aspect, Stock-n-flow (Meadows and Wright, 2008) to specify the what aspect, and BPMN (White, S., 2004) to specify the how aspect can be used collectively to come up with the

[1]http://www.valueteam.biz/why-72-percent-of-all-business-transformation-projects-fail

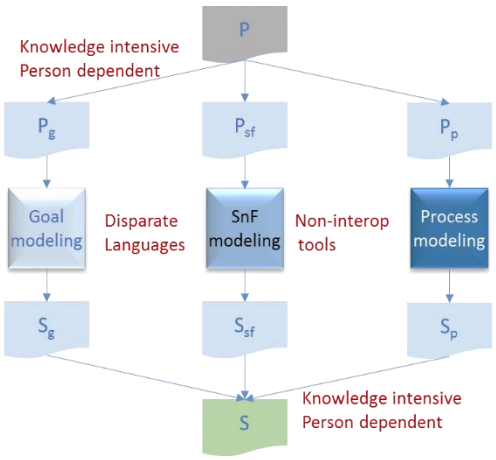

Fig 1. Co-simulation based approach for decision-making

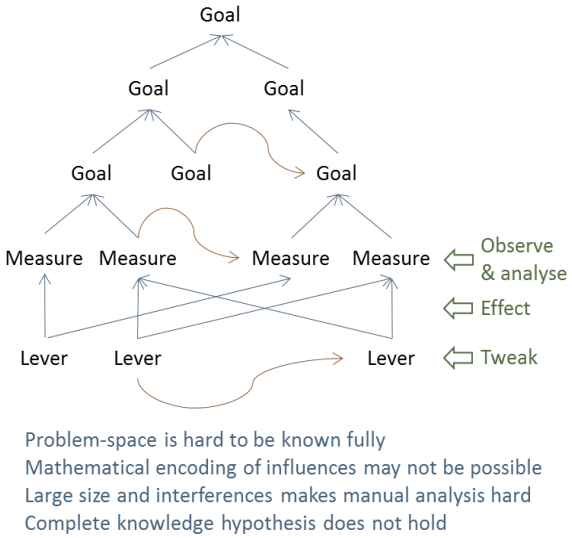

Fig 2. Desired approach to decision-making

necessary and sufficient specification which is amenable for analysis albeit in parts. However, human expertise is still needed to stitch together the results of part analyses into a consistent whole. This is an intellectually intensive activity further exacerbated due to paradigmatically diverse nature of the three languages and non-interoperable tools[2,3,4].

Moreover, increasingly fast business dynamics makes it very difficult to have the complete information required for decision-making available and/or with full certainty. This puts new demands on enterprise specification as well as processing that are not well supported by existing EM languages and associated processing machinery. This paper proposes an approach to support organisational decision-making in presence of uncertainty. Contributions of this paper are twofold. First, it proposes a language capable of specifying the necessary and sufficient aspects of enterprise for organisational decision-making. Second, it describes how the language and its simulation engine can support uncertainty.

This paper is organized as follows: Section 2 describes requirements of a desired approach. Section 3 presents available support for organisational decision-making in the form of EM languages and tools. The proposed language is presented in Section 4 along with a discussion on support for uncertainty. Section 5 presents validation of the proposed approach using a representative case-study from real life. An evaluation with respect to the current state of art and practice is also provided. Section 6 provides a summary and a brief outline of future work.

[2]http://www.cs.toronto.edu/km/ome/

[3]http://www.iseesystems.com/Softwares/Business/ithinkSoftware.aspx

[4]http://www.bizagi.com/en/products/bpm-suite/modeler

2. DESIRED APPROACH

An approach based on the principles of "separation of concerns" and "divide and conquer" seems required for organisational decision-making. It should be possible to decompose the overall goal into sub-goals, sub-sub-goals *etc*., to the desired level of granularity. It should be possible to identify a set of variables (*i.e.*, Measures) that need to be observed in order to determine whether the finest-level goal is met – a goal once met is a measure. It should be possible to identify a set of variables (*i.e.*, Levers) that influence a given Measure and be able to specify the influence in a machine-processable manner. It should be possible to make explicit the dependencies between levers, between measures and between goals.

The goal-measure-lever graph structure of Fig 2 captures the understanding of problem domain in a manner that is amenable to automation. Decision-making is then a bottom-up walk of this graph structure provided it is possible: (i) to compute values for the measures based on the values of levers, (ii) to evaluate whether a goal is met based on the values of measures, and (iii) to honour lever-to-lever, measure-to-measure and goal-to-goal dependencies in the bottom-up walk terminating with evaluation of the overall goal.

Therefore, organisational decision-making can be viewed as a human-guided exploration of the design space as shown by the decision loop of Fig 3. The ability to specify the influence of a lever on a set of measures is the key. Based principally on the Zachman framework (Zachman, J., 1987) it can be argued that an organisation can be understood well by knowing *what* an organisation is, *why* it is so, and *how* it operates involving the key stakeholders (the *who*) thereby constituting the necessary and sufficient information for decision-making. Two primary requirements emerge: (i) the ability to capture the *why*, *what*, *how* and *who* aspects in a formal manner and (ii) the ability to perform what-if analyses of the formal specification.

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

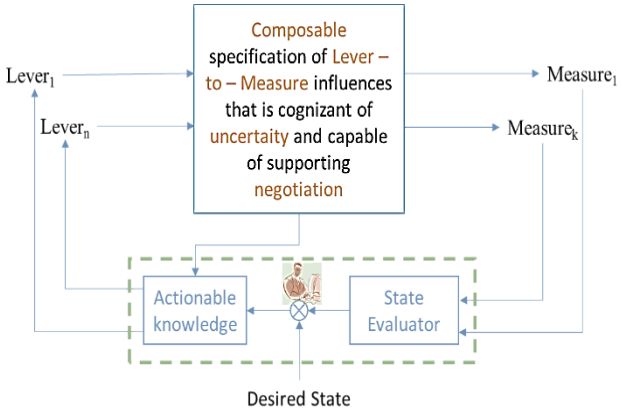

Fig 3. The decision loop

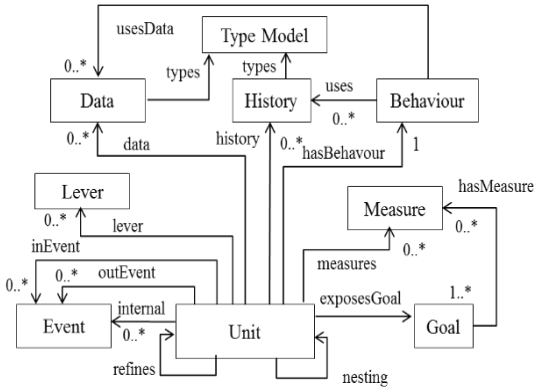

Fig 4. Conceptual Model

In addition, the specification for decision-making should capture the key characteristics of organisation such as: *reactive* (*i.e.*, an organisation reacts to the events taking place in its environment), *adaptable* (*i.e.*, an organisation responds to changes taking place in its environment by transforming itself), *modular* (*i.e.*, an organisation as a whole is a sum of its parts each can be viewed as an organisation), *autonomous* (*i.e.*, an organisation is capable of determining its own course of action so as to achieve its stated goals), *intentional* (*i.e.*, an organisation works towards achieving its stated objectives), and *uncertain* (*i.e.*, an organisation exhibits probabilistic behaviour).

The socio-technical nature of enterprise and the inability to have complete understanding of the problem space means it is difficult to specify lever-to-measure influence in pure mathematical terms. In other words, a closed-form solution to the decision-making problem is unattainable. That leaves an open-form solution using, say, simulation as the only pragmatic recourse. However, simulation is known to deliver only in situations where mechanistic world view holds (Barjis, J., 2008). Modern enterprises, on the other hand, are socio-technical systems that rely on automation systems as well as human agents for their operation. Also, modern enterprise is a system of systems (Ackoff, R., 1971) whose behaviour is too complex to be fully known a-priori. Instead, the overall system behaviour emerges from the interactions between its various agents whose behaviour is simple enough to be fully known a-priori and hence specifiable. The specification language and its simulation engine should cater to these additional requirements as well.

3. STATE OF ART AND PRACTICE

Table 1 summarizes an evaluation of existing EM languages as regards their adequacy to support organisational decision-making in presence of uncertainty. The evaluation criterion comprises of ability to support: (i) specification of aspect views, unit views, socio-technical characteristics, and multiple perspectives, and (ii) processing of these views to support data-driven analysis. As can be seen, the languages capable of specifying all relevant aspects are

Table 1 Current state of EM languages

Modelling Language	To support aspect views				To support unit views		To support socio-technical characteristics					To support data-driven analysis		
	Why	What	How	Who	Modularity	Composability	Reactive	Autonomous	Intentional	Adaptability	Uncertainty	Machine Processability	Quantitative Analysis	Qualitative analysis
Zachman	S	S	S	S	S	N	N	N	S	N	N	N	N	N
Archimate	S	S	S	S	S	I	S	I	S	N	N	N	N	N
EEML	I	S	S	S	N	N	N	N	S	N	N	N	N	N
i*	S	N	N	I	S	S	N	N	S	N	N	S_{Why}	S_{Why}	S_{Why}
SnF	N	S	I	I	I	N	S	S	N	N	N	S_{What}	N	S_{What}
BPMN	N	I	I	S	S_{How}	S_{How}	S	I	N	N	N	S_{How}	S_{How}	S_{How}
S=Supported		Sx =Supported for Aspect X					I = Inadequate Support					N=No Support		

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

found wanting in terms of analysis capabilities e.g., Archimate (Jonkers et al, 2004), EEML (Krogstie, J., 2008), and UEML (Vernadat, F., 2002). On the other hand, the languages capable of sophisticated analysis can cater to specification of only a subset of relevant aspects. For instance, i* (Yu et al, 2006) can specify the why aspect only, BPMN (White, S., 2004) can specify the how aspect only, and Stock-n-flow (Meadows and Wright, 2008) can specify the What aspect only. Therefore, a language capable of supporting all the desired characteristics of organisation specification needs to be designed and the relevant analysis machinery needs to be developed.

4. ENTERPRISE SPECIFICATION LANGUAGE

4.1. Conceptual Model

Looking outside-in, an organisation raises and responds to a set of events as it goes about achieving its stated goals. It consists of several autonomous units, organised into dynamically changing hierarchical groups, operating concurrently, and managing goals that affect their behaviour. We describe structure and behaviour of an organisation using a small set of concepts and their relationships as depicted in Fig 4.

Organisation is a Unit that comprises a set of Units and strives to accomplish its stated Goal. It does so by responding to Events taking place in its environment (InEvents), processing them, and by interacting with other external Units in terms of Events raised/responded (OutEvents). A Unit may choose not to expose all events to the external world (InternalEvents). A declarative specification of event processing logic constitutes the behaviour of a Unit. Thus, looking outside-in, a Unit is a Goal-directed agent that receives events (InEvents), processes them, and raises events (OutEvents) to be processed by other Units. Also, Unit is a parameterized entity whose structure and behaviour can be altered through Levers. A Unit interacts with other Units in a-priori well-defined Role-playing manner. TypeModel provides a type system for structural as well as behavioural aspects of a Unit.

Unit, Event, Data, History and unit nesting together specify the *what* aspect, Goal specifies the *why* aspect, Behaviour specifies the *how* aspect, and Unit, as an individual stakeholder, specifies the *who* aspect of an organisation. Event helps to capture the *reactive* nature of Unit. Also, Unit is *adaptable* as it can construct and reconstruct its structure; *modular* as it encapsulates the structure and behaviour of an organisation; *intentional* as it has its own goals; and *compositional* as it can be an assembly of Units.

The Unit abstraction draws from a set of existing concepts. Modularization and reflective unit hierarchy are taken from fractal component models (Bruneton et al, 2006). Goal-directed autonomous behaviour can be traced to agent behaviour (Bonabeau, E., 2002). Defining states in terms of a type model is borrowed

```
type ::= Var                              type variable
       | Act { export dec* Mes* }         actor type
       | (type*) → type                   λ-type
       | tt                               term type
       | Int | Bool | Str                 constant type
       | Void                             undefined
       | [ type ]                         lists
       | Fun(Name*) type                  parametric type
       | ∀(Name*) type                    polymorphic type
       | rec Name . type                  recursive type

tt   ::= Name(type*)                      term type

exp  ::= var                              variables
       | num | bool | str                 constants
       | self                             active actor
       | null                             undefined
       | new name[type*] (exp*)           create actor
       | become name[type*] (exp*)        change behaviour
       | exp op exp                       binary exp
       | not exp                          negation
       | λ(dec*)::type exp                λ-abstraction
       | let bind* in exp                 local bindings
       | letrec bind* in exp              local recursion
       | case exp* arm*                   pattern matching
       | for pat in exp { exp }           looping
       | { exp* }                         block
       | if exp then exp else exp         conditional
       | [ exp* ]                         list
       | []                               empty list
       | exp(exp*)                        application
       | Name(exp*)                       term
       | exp ← exp                        message
       | name := exp                      update
       | exp . name                       name reference
       | probably(exp)::type exp exp      uncertainty
       | exp[type*]                       type limitation

bind ::= dec = exp                        value binding
       | name(pat*)::type=exp when exp λ-binding
       | type Name[Name*] = type          type declaration
       | data Name[Name*] = tt*           algebraic type
       | act name(dec*)::type {           behaviour def
            export name*                  interface
            bind*                         locals
            → exp                         initial action
            arm*                          behaviour
         }

dec  ::= name[Name*] :: type              declaration

arm  ::= pat* → exp when exp              guarded exp

pat  ::= dec                              binding
       | dec = pattern                    naming
       | num | bool | str                 const pattern
       | pat : pat                        cons pair
       | [ pat* ]                         list
       | [][type]                         empty list
       | Name[type*](pat*)                term pattern
```

Fig 5. ESL syntax

from UML[5]. An event driven architecture (Michelson, B., 2006) supports flexible interactions between components, and the concept of intentional modelling (Yu et al, 2006) is adopted to enable specification of component goals.

4.2. Implementation

We have provided an implementation of the conceptual model in the form of Enterprise Simulation Language (ESL). ESL is an extension of an existing event-driven lan guage LEAP (Clark and Barn, 2013) with concepts borrowed from actor model of computation (Agha, G., 1985), multi-agent systems (Van Harmelen et al, 2008), goals (Yu et al, 2006) and linear temporal logic (Pnueli, A., 1977). ESL and its associated development and run-time environment is a language that has been designed to support our thesis that simulation and emergent behaviour can be used to support organisational

[5] http://www.omg.org/spec/UML/

decision-making. ESL together with a supporting toolset is currently in development[6].

The syntax of ESL is shown in Fig 5. It is statically typed and includes parametric polymorphism, algebraic types and recursive types. An ESL program is a collection of mutually recursive bindings. Behaviour types Act {...} are the equivalent of component interfaces and behaviours act {...} are equivalent to component definitions. A behaviour b is instantiated to produce an actor using new b in the same way that class definitions are instantiated in Java. Once created, an actor starts executing a new thread of control that handles messages that are sent asynchronously between actors. Pattern matching is used in arms that occur in case-expressions and message handling rules. Uncertainty is supported by probably(p) x y that evaluates x in p% of cases, otherwise it evaluates y. Functions differ from actors because they are invoked synchronously.

A minimal ESL application defines a single behaviour called main, for example:

```
1 type Main = Act{ Time (Int) };
2 act main :: Main {
3 Time (100) ! stopAll ();
4 Time (n:: Int) ! {}
5 }
```

An ESL application is driven by time messages. The listing defines a behaviour type (line 1) for any actor that can process a message of the form Time(n) where n is an integer. In this case, the main behaviour defines two message handling rules. When an actor processes a message it tries each of the rules in turn and fires the first rule that matches. The rule on line 3 matches at time 100 and calls the system function stopAll() which will halt the application. Otherwise, nothing happens (line 4).

4.3. Supporting Uncertainty

4.3.1. Decision space

Decision space can be broadly classified into four categories namely Known Knowns (KK), Known Unknowns (KU), Unknown Knowns (UK), and Unknown Unknowns (UU) as shown in Fig 6.

In the context of decision making, KK space denotes full certainty about what are the possible next states as well as availability of information to decide which of them to be the next state. KU space denotes full certainty about what are the possible next states but uncertainty as regards determining the next state – instead a probability distribution is considered available. UK space denotes partial information i.e. only a subset of possible next states are known and there could well be uncertainty as regards which would be the next state. UU space denotes lack of information about

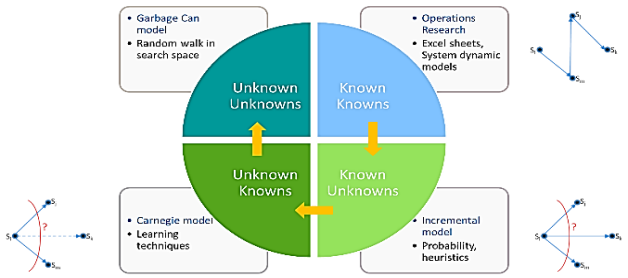

Fig 6. Decision spaces and kinds of uncertainty

the possible next states as well as uncertainty about which would be the next state.

This paper addresses decision making for KU space only.

4.3.2. Specification of uncertainty

Organisation is a set of interacting Units. A Unit interacts with other Units by sending messages.

Unit is a 5-tuple < I, S, B, St, Q > where,

I denotes immutable identity of the Unit,

S denotes its structure in terms of Unit and Unit relations shown in Fig 4,

B denotes behavior of the Unit in terms of event-handling i.e., a set of $< e_i, eh_i >$ tuples where 'e' corresponds to an event (or a message) and 'eh' corresponds to the corresponding event handling logic

St denotes state of the Unit in terms of its Data and History i.e., a set of $< var_i, val_i >$ tuples where 'var' corresponds to an attribute (of the Unit) and 'val' corresponds to its value

Q denotes a FIFO queue wherein messages received by the Unit are arranged in temporal order

The notion of Uncertainy impacts Unit specification in the following manner,

Unit.State takes the form $var_i = val_i @ p_i \mid val_j @ (1 - p_i)$ where 'p' is the probability. This can be extended to 'n' values with the constraint that sum of the probabilities equals 1.

Unit.Behaviour: $< e_i, eh_i >$ association takes the form $< e_i, eh_i @ p_i \mid eh_i @ (1 - p_i)$ where 'p' is the probability. This can be extended to 'n' event-handlers with the constraint that sum of the probabilities equals 1.

Unit.Queue gets impacted with "Send m_i to U_i" taking the form "Send m_i to $U_i @ p_i \mid U_j @ p_j$ where 'p' is the probability. This can be extended to 'n' values with the constraint that sum of the probabilities equals 1.

On similar lines, Becomes takes the form "Becomes $U_i @ p_i \mid U_j @ (1 - p_i)$ where 'p' is the probability. This can be extended to 'n' values with the constraint that sum of the probabilities equals 1.

Unit.Identity has no impact

Unit.History has no impact

[6] The current version of ESL is available at https://github.com/TonyClark/ESL

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

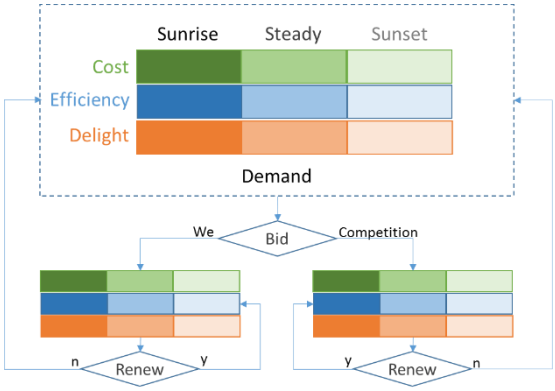

Fig. 7. Business Process Outsourcing

The above mentioned concept of Uncertainty is supported in ESL by the construct probably(p) x y.

4.3.3. Simulation

Language outlined in the previous section helps specify behavior of a system i.e. the input-output transfer function of Fig 3. Given such specification, an initial setup, and a sequence of events, it is possible to observe the impact of perturbation of *levers* onto the *measures* using simulation. In the interest of space, we do not describe simulation of the language outlined in the previous section, instead, we focus on the impact of uncertainty on simulation.

As discussed earlier, introduction of uncertainty impacts handling of an event by a Unit, sending of a message to a Unit, and assigning of a value to a variable of a Unit. Each of these, in fully certain situation, have exactly one course of action available. Presence of uncertainty introduces multiplicity of actions along with probability for each. In other words, a single value is replaced by a probability distribution.

Let's assume that uncertainty manifests at 'n' places in system specification. As a result, there are 'n' probability distributions – one for each place. Identifying a value from the corresponding probability distribution for every single place where uncertainty has manifest leads to fully certain specification that can be simulated without any change (to the simulator) whatsoever. However, the resultant output can be thought of as a true representation of the system behavior as a whole only if the 'n' probability distributions are fully covered. This identifies two additional demands on simulation: *how many simulation runs are required to fully cover the 'n' distribution functions*? *which value to pick from the probability distribution for a given simulation run*? These questions can be readily answered by making use of established results from probability and statistics (Dellino and Meloni, 2015).

Let's assume that the 'n' probability distributions necessitate 'k' simulation runs. This leads to 'k' sets of output variables each describing just a snapshot of the analysis. Thus emerges the question: *how to present the 'k' output datasets to the user so that meaningful inference can be drawn*? This question too can be readily answered by making use of established results from probability and statistics. For instance, the 'k' datasets can be collapsed into a single dataset describing value for an output variable, its confidence level, and confidence interval.

Thus, impact of introduction of uncertainty in system specification can be fully and effectively addressed by making use of established results from probability and statistics while leaving the simulation engine largely untouched.

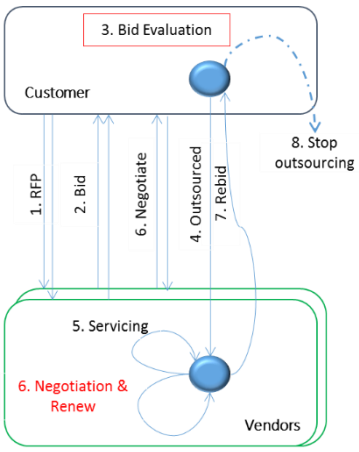

Fig. 8. Interactions and transitions

5. VALIDATION

5.1. Illustrative Example

Consider business process outsourcing (BPO) space. Customers outsource business processes for a variety of reasons such as reducing cost (C), increasing efficiency (E), bringing about a major transformation (D) *etc*. The outsourced processes can be classified into three buckets based on maturity of BPO for the specific process and the vertical. For instance, Transcript Entry process of Healthcare vertical was one of the first to take to BPO and has derived almost all potential benefits accruable from outsourcing (i.e. Sunset or SS). On the other hand, IT Infrastructure Management process being a late adopter of BPO has a large unrealized potential to be tapped (i.e. Sunrise or SR). And there are processes such as Help Desk, Account Opening, Monthly Alerts etc., that fall somewhere in between the two extremes as regards benefits accrued from BPO (i.e. Steady or ST). Thus, BPO demand space can be viewed as a 3 x 3 matrix of Fig 7. A Customer invites bids from the vendors for a specific BPO project or takes help of an external agent to identify a vendor. Typically, factors such as *Quadrant* (i.e. ranking as per independent agency such as analysts), *FTE Count Range* (i.e. min-max count of full time employees to be deployed on the outsourced process), *Billing Rate Range* (i.e. min-max range for per hour rate of full time employee), *Market Influence* (i.e. perception of the market as regards delivery certainty with acceptable quality) etc., influence who wins the bid. Other soft issues such as familiarity with the processes being

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

	Quadrant	Billing Rate ($/Hr)		FTE Quotation	Negotiation Levers		Market Influence	Delivery Excellence
		Min	Max	Deviation from Standard	FTE Deviation (%)	Billing Rate deviation (%)		[Probability of Excellent,Probability of Good, Probability of Normal, Probability of Below Normal]
Efficiency	Leader	8	12	8	2	5	Excellent	[60, 40,10, 0]
Effectiveness	Visionary	16	24	6	0	5	Good	[50, 40,10, 0]
Delight	Contender	120	140	4	5	0	Normal	[20, 60,20, 0]

Fig 9.a Characteristics of 'We' Unit

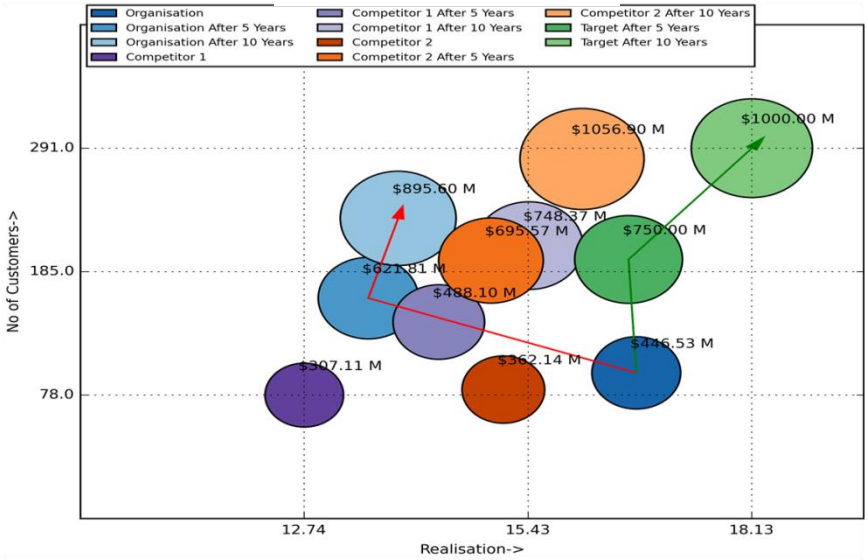

Fig 9.b Comparative Progression

outsourced, rapport with the vendor *etc*., also play a part in selection of the vendor. It is common observation that BPO contracts come up for renewal after few years. Customer may renew the contract with the existing vendor on modified terms (typically advantageous to the customer) or may opt for rebidding. Factors influencing the renewal decision are reduction offered in *FTE Count*, *Billing Rate*, number and degree of escalations, perception the external agent has as regards ability to meet the project requirements *etc*. Contracts that fail to get renewed become candidates for later bidding. Fig 7 shows a high level schematic of BPO space with events of interest therein detailed out in the form of a state transition diagram of Fig 8.

The demand space exhibits temporal dynamism. For instance, new processes emerge as candidates for outsourcing and some of the existing processes no longer need to be outsourced as, say, technology advance eliminates the need for human intervention in the process thus making it straight-through. Thus BPO space can be viewed as an event-driven system where events have a certain frequency and are stochastic in nature. The frequency and stochastic characteristics typically vary from process to process. While operating in this uncertain space, a BPO vendor needs to make decisions of the following kind: *Will continuation with the current strategy keep me viable 'n' years hence? What alternative strategies are available? How effective will a given strategy be? By when a given strategy will start showing positive impact? Will I be growing at the expense of competition or vice versa?* And so on.

Answers to the above questions are essentially linked to the evaluation of portfolio basket i.e., 3 x 3 matrix of Fig 7, of the organisation in terms of revenue accruable and expense. Therefore, ability to predict portfolio basket of the organisation and its competitors after a given time period becomes critical to support data-driven decision making.

5.2. Simulation setup and results

In the interest of space, we only address the question: *Will continuation with the current strategy keep me viable 'n' years hence with respect to the competition*?

We model Demand as a Unit comprising of set of SR, ST and SS Units each comprising of set of C, E and D Units. Increase (or decrease) in SR demand is modelled as increase (or decrease) in SR.C, SR.E and SR.D Units at a pre-defined frequency and probability – and on similar lines for ST and SS processes.

We and Competition are modelled as Units each comprising of a set of SR, ST and SS Units such that each member of the set comprises of a set of C, E and D Units. Fig 9.a shows sample characteristics of 'We' Unit. Values of its attributes such as *Quadrant*, *Billing Rate*, *FTE count*, *Market Influence* and *Delivery Excellence* contribute towards its ability to win a bid. As can be seen from Fig 9.a, 'We' Unit is best equipped to win BPO contracts aimed at cost reduction. Value of *Delivery Excellence* attribute is a probability distribution. For instance, 'We' Unit is confident of delivering 'Excellent' quality on 60% of 'C' kind of BPO projects won. The values for 'Good', 'Normal' and 'Below Normal' quality for this kind of BPO projects are 30%, 10% and 0% respectively. The Unit is

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

	Quadrant	Billing Rate ($/Hr)		FTE Quotation	Negotiation Levers after term completion		Market Influence	Delivery Exc
		Min	Max	Less FTE (in %) Quotation in Bid	FTE Deviation (%)	Billing Rate deviation (%)		[Probability of Excellent,Proba Probability of Normal, Probab
Efficiency	Leader	12	14	10	5	1	Excellent	[90, 10,0, 0]
Effectiveness	Visionary	25	30	10	5	1	Good	[80, 10,10, 0]
Delight	Contender	110	120	10	5	1	Excellent	[70, 30,0, 0]

Fig 10.a Modified characteristics of 'We' Unit

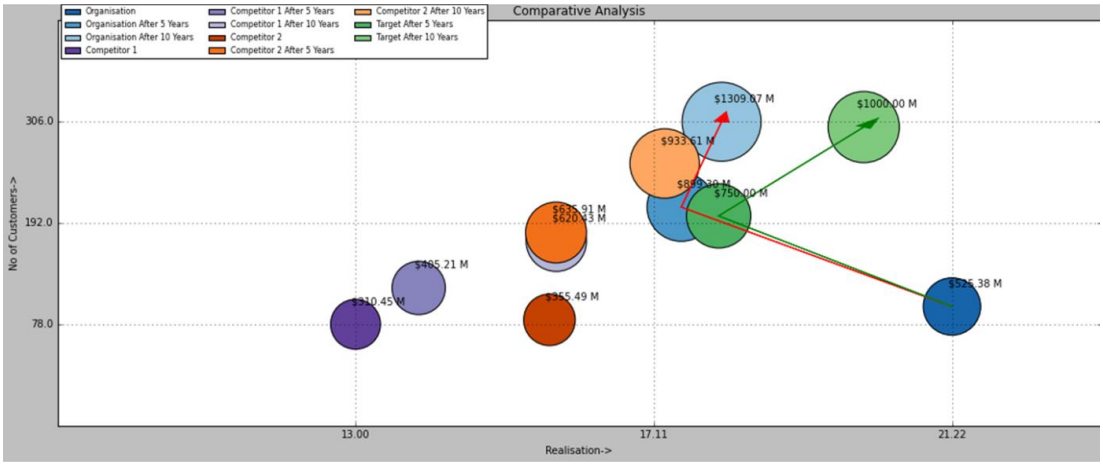

Fig 10.b Comparative Progression

equipped with two negotiation levers namely, what's the percent reduction possible in number of full time employees billed against the outsourced process, and what's the percent reduction possible in per hour billing rate for full time employee. The variables *Billing Rate* and *FTE Count* are both ranges and a value is picked at random from the specified range.

The 'Competition' Unit is modelled on the same lines as 'We' Unit. We can model multiple 'Competition' Units each having different characteristics.

Simulator raises RFP events at random. Each RFP event is characterized by the kind of process being outsourced (i.e. SR or ST or SS), the objective for outsourcing (i.e. C or E or D), size of the process in terms of FTE count, and the desired billing rate. All vendors respond to the RFP event by picking suitable values for their characteristics at random. Bid evaluation function is a weighted aggregate of the various elements of RFP response and a random valu e to capture effect of inherent uncertainty. The vendor with the lowest bid wins the project which gets executed as defined by the characteristics of the particular vendor. In other words, we acknowledge that it won't be possible to know everything about the domain and we can't be fully certain of the domain knowledge acquired so far.

The decision to renew existing contract is modelled on similar lines but with a different set of characteristic attributes influencing the decision. Here too, we take cognizance of incomplete and uncertain knowledge about the problem domain by introducing a random variable in the evaluation function.

This completes the simulation setup and the system is allowed to run for 10 years. Results of the simulation run are shown in Fig 9.b. As can be seen, our current revenue is 446.54 MUSD from 90 customers with a realization of nearly 17 USD per hour per person. Corresponding numbers for competitor#1 and competitor#2 respectively are < 307.11, 78, 12.74 > and < 362.14, 80, 15 >. In short, at present we are doing much better than competition.

We are set a goal to deliver < 750, 200, 17 > after 5 years and < 1000, 290, 18 > after 10 years. As can be seen, by continuing to operate the same way we will be delivering < 621.81, 160, 13.5 > after 5 years and < 895.6, 215, 14 > after 10 years thus missing both the targets by a considerable margin. More importantly, competitor#2 will be overtaking us after 5 years and both the competitors will be significantly ahead of us after 10 years.

Clearly, we cannot afford to continue with our current way of operation. Further detailed analysis, involving model elements not described in this paper for want of space, shows that much of our current revenue is from sunset processes outsourced for cost reasons. Over time this market is going to shrink considerably with demand for steady as well as sunrise processes (for objectives other than pure cost reduction) increasing significantly. We need to bring about a change in our characteristics so as to be able to win more bids in this demand situation. Fig 10.a shows the modified characteristics of 'We' Unit leading to the improved performance as shown in Fig 10.b. We are able to beat both revenue and customer targets while failing to meet the realization target narrowly.

5.3. Evaluation

For the kind of decision making problem illustrated in this paper, industry practice relies extensively on excel sheets. Such an approach typically represents the influence of lever onto measures in terms of static algebraic equations. However, value of a lever and

influence of a lever onto a set of measures can vary over time. This behaviour cannot be captured using excel sheet. There is no support for encoding stochastic behaviour either.

Systems dynamics models are also used for this kind of decision-making wherein the system is specified in terms of stocks, flows of stocks, and equations over system variables that control the flows. Value of a stock or a flow or a variable can be a discrete number or a range or a distribution. The quantitative nature of systems dynamics models and sophisticated simulation support enables decision making through what-if scenario playing. It is possible for a stock or an individual variable to have a value that is a probability distribution, however, structure of the stock-n-flow model must remain unchanged. Thus, systems dynamics modeling provides only a partial support for specifying and processing the inherent uncertainty within a system. Moreover, it is best suited for an aggregated and generalized view of a system where individual details get eliminated through averaging, and sequences of events are grouped as continuous flows. This generalized approach and ignorance of individual characteristics that significantly influence the system over time often leads to a model that is somewhat removed from reality. Though not designed to specify specialized behaviour, it can be done using systems dynamics modeling. But this is an effort-intensive endeavour, and more importantly leads to model size explosion. For example, modeling of 4 competitors each having special characteristics leads to roughly 4 times increase in the size of systems dynamics model.

The proposed language enables modelling of a system as a set of units each listening/responding/raising events of interest and interacting with other units by sending messages. A unit encapsulates state (i.e. a set of attributes), trace (i.e. events it has responded to and raised till now) and behavior (i.e. encoding of lever-to-measure influence). Thus, the language subsumes systems dynamics model. As the language supports 'time' concept, value of a variable and relationships between variables can change with respect to time. Consider the example of determining the impact of track record on winnability of organisation where the value of track-record variable changes over time thus affecting winnability. Since a process is an individual actor, simulator can determine the impact of successful contract completion, renewal with/without negotiation etc., for that specific process – systems dynamics model falls short here. A trace of events serves as a memory that can be queried to establish more complex relationships between levers. For example, successful completion of contract leads to improved track record as well as better rapport with the customer thus improving winnability of future outsourcing bids everything else remaining the same. Thus, the language provides primitives for creating models that closely mimic reality.

Thus, it can be said that the proposed language subsumes Excel sheets as well as systems dynamics model. Excel sheets provide no support for specifying or processing uncertainty. Systems dynamics model provides only partial support for uncertainty as it is possible for variables to have value as a probability distribution (as opposed to a discrete value or a range) but the flows remain fully deterministic. The proposed language provides for the flows to be probabilistic too. Therefore, proposed approach presents a pragmatic solution for supporting decision-making in presence of uncertainty using simulation. The solution addresses uncertainty by using established results from probability and statistics while leaving the simulator implementation largely unchanged. The proposed approach is validated using a representative example from real world.

6. SUMMARY AND FUTURE WORK

Effective decision-making is a challenge all modern enterprises face. It requires deep understanding of aspects such as organisational goals, structure, operational processes etc. Large size, siloed structure, and increasingly fast business dynamics means the available information is either incomplete or uncertain or both. Inability to handle this inherent uncertainty is a present lacuna in current industry practice of organisational decision-making. We began by outlining an approach based on the principles of "separation of concerns" and "divide and conquer" that enables creation of goal-measure-lever graph to capture understanding of the problem domain at a higher level of abstraction. We outlined a language for specifying the graph structure in a manner that is amenable to what-if scenario playing. Decision-making thus is a bottom-up walk of this graph structure i.e., human-guided simulation-aided exploration of solution space. We then discussed where and how uncertainty affects the graph structure and proposed extensions to the language so as to be able to externalize the uncertainty at specification level. We showed how well-established results from statistics and probability can be used to implement the manifest uncertainty without changing the core simulation engine of the language. We presented a validation of the approach using a representative example from real life.

The approach has been illustrated with a substantive example from Business Process Outsourcing domain. We have shown the example can be modeled and simulated leading to the ability to influence the strategically selected measures. However, we recognise that the current implementation of ESL is not sufficiently high-level for direct adoption by decision-makers. Our immediate next step is to develop higher level abstractions to support the core ESL concepts in a business-facing manner. In doing so, we will adopt language processing and model transformation technology to enable support for defining domain specific languages geared for specific problems. We note that decision-making is more a satisfaction problem rather than an optimisation problem. Consequently, we will draw upon game theory and

computational economics to consider extending our proposed solution to impart this characteristic.

REFERENCES

Ackoff, R., 1971. Towards a system of systems concepts. Management science, 17(11), pp.661-671.

Agha, G., 1985. Actors: A model of concurrent computation in distributed systems. No. AI-TR-844. Massachusetts Institute Of Technology Cambridge Artificial Intelligence Lab.

Barjis, J., 2008. Enterprise, organization, modeling, simulation: putting pieces together. Proceeding of EOMAS.

Bonabeau, E., 2002. Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences, 99(suppl 3), pp.7280-7287.

Bruneton, E., Coupaye, T., Leclercq, M., Quéma, V. and Stefani, J.B., 2006. The fractal component model and its support in java. Software: Practice and Experience, 36(11-12), pp.1257-1284.

Clark, T. and Barn, B., 2013. Goal Driven Architecture Development using LEAP. Enterprise Modelling and Information Systems Architectures 8(1): 40-61 (2013).

Daft, R., 2012. Organization theory and design. Nelson Education.

Dellino, Gabriella, and Carlo Meloni. Uncertainty management in simulation-optimization of complex systems. Springer, 2015. Evans W.A., 1994. Approaches to intelligent information retrieval. Information Processing and Management, 7 (2), 147–168.

Jonkers, H., Lankhorst, M., Van Buuren, R., Hoppenbrouwers, S., Bonsangue, M. and Van Der Torre, L., 2004. Concepts for modeling enterprise architectures. International Journal of Cooperative Information Systems, 13(03), pp.257-287.

Krogstie, J., 2008. Using EEML for Combined Goal and Process Oriented Modeling: A Case Study,[w:] T. Halpin, J. Krogstie, E. Proper. In Proceedings of EMMSAD'08. Thirteenth International Workshop on Exploring Modeling Methods for Systems Analysis and Design.

McDermott, T., Rouse, W., Goodman, S. and Loper, M., 2013. Multi-level modeling of complex socio-technical systems. Procedia Computer Science, 16, pp.1132-1141.

Meadows, D. and Wright, D., 2008. Thinking in systems: A primer. Chelsea green publishing.

Michelson, B., 2006. Event-driven architecture overview. Patricia Seybold Group, 2.

Pnueli, A., 1977, October. The temporal logic of programs. In Foundations of Computer Science, 1977., 18th Annual Symposium on (pp. 46-57). IEEE.

Van Harmelen, F., Lifschitz, V. and Porter, B. eds., 2008. Handbook of knowledge representation (Vol. 1). Elsevier.

Vernadat, F., 2002. UEML: towards a unified enterprise modelling language. International Journal of Production Research, 40(17), pp.4309-4321.

White, S, 2004. Introduction to BPMN. IBM Cooperation, 2(0).

Yu, E., Strohmaier, M. and Deng, X., 2006, October. Exploring intentional modeling and analysis for enterprise architecture. In Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International (pp. 32-32). IEEE.

Zachman, J.A., 1987. A framework for information systems architecture. IBM systems journal, 26(3), pp.276-292.

AUTHORS BIOGRAPHY

Vinay Kulkarni is Chief Scientist at Tata Consultancy Services Research (TCSR). His research interests include model-driven software engineering, self-adaptive systems, and enterprise modeling. His work in model-driven software engineering has led to a toolset that has been used to deliver several large business-critical systems over the past 20 years. Much of this work has found a way into OMG standards. Vinay also serves as Visiting Professor at Middlesex University London.

Souvik Barat is a Senior Scientist at Tata Consultancy Services Research (TCSR). His research interests include model-driven software engineering, and enterprise modeling.

Tony Clark is Professor of Software Engineering at Sheffield Hallam University in the UK. His academic research on meta-modelling led to the development of a tool called XModeler that has been used in a number of commercial applications including the development of tool support for a new Enterprise Architecture modeling language.

Balbir Barn is Professor of Software Engineering at Middlesex University London with extensive experience in industrial Software Engineering including the design and implementation of the IEF.