[SUBMISSION COVERSHEET]

15th International Command and Control Research and Technology Symposium
"The Evolution of C2"

Title: **Temporal reasoning models of a targeting decision chain**

Topics: (6) Modelling and Simulation, (8) C2 assessment Metrics and Tools

Authors: Robert J. Houghton and Chris Baber

Corresponding author: Dr Robert J. Houghton

Organisation: Human Factors Integration Defence Technology Centre (HFIDTC), The University of Birmingham, UK.

Address: Electronic, Electrical & Computer Engineering,
The University of Birmingham,
Edgbaston, Birmingham,
B152TT, UK

Telephone: +44 (0) 121 414 7511
Fax: +44 (0) 121 414 4291

Email: R.J.Houghton@bham.ac.uk

URL: http://www.hfidtc.com

This work is supported by a grant from the Human Factors Integration Defence Technology Centre, part-funded by the Human Sciences Domain of the UK Ministry of Defence Scientific Research Programme.

Temporal reasoning models of a targeting decision chain

Abstract

Time is a crucial variable in military operations and rapid decisions based on current information leading to swift action are essential. However, there are many challenges to be faced in modelling the behaviour of socio-technical systems over time, principally because the complexity inherent in such systems can be difficult to express. In the present paper we discuss a candidate solution to this problem, the event calculus (EC), a form of temporal logic that affords the analyst considerable latitude of expression to create a detailed representation that can be formally tested.

Whilst the core model itself is useful as a formal specification of a system, it is most powerful when used as a basis for automated reasoning to predict, abduct and postdict within the specified constraints. Thus, we can test the system model against alternative descriptions of operational context to see how the proposed system performs and automatically generate plausible narratives that expose both opportunities for enhanced effectiveness and potentially dangerous error-modes. An example of the modelling and subsequent analysis of a targeting decision chain is provided and the prospect of repurposing the EC model as part of a situationally-aware process management tool to aid command staff is discussed.

[198 words]

1. Introduction

Time is a critical variable in military operations. Opportunities for effective actions can be fleeting in nature, especially in asymmetric warfare contexts where potential targets may fade away as quickly as they have appeared. From the literature concerning John Boyd's OODA loop (Observe-Orient-Decide-Act) (see Coram, 2002) the point is made that to navigate this process faster than an adversary can deliver significant advantages. A point in military operations where time is particularly of the essence is in the area of Time Sensitive Targeting (TST) where targets of opportunity are only briefly available. One of the key challenges of TST is for command staff to respond as quickly as possible to avoid losing a fleeting opportunity while at the same time maintaining adherence to rules concerning safety, legality and effect.

In analysing such socio-technical systems (that is, the combinations of social and technological systems), the consideration of time is frequently problematic for existing analytic and modelling approaches. The difficulties have been summed up thus:

"Task representation techniques are well equipped for modelling the sequential structure of single tasks, but are often inadequate for representing multiple concurrent or interleaved tasks and their durational properties [...] Queuing and scheduling models, on the other hand, support the analysis of multi-task

environments, but often treat tasks as atomic units with no representation of their sequential structure". (Hildebrandt & Rantanen, 2004, p. 705)

Our proposition is that a logic-based formalism (the Event Calculus or EC) can provide a candidate solution to this problem by representing both concurrency in task performance and scheduling of activities in a manner that treats specific tasks as having meaning in their embedded context. First, EC through its roots in logic can move towards formal validation of socio-technical systems; if the assumptions are correct and the modelling of those assumptions is correct we can demonstrate that the system is fit for purpose on logical grounds. We can also enumerate the circumstances in which it will and will not lock and terminate.

Second, and perhaps even more usefully, because we can use an EC model as material for reasoning, we can carry out the testing models against counterfactual narratives to see how the system functions. In addition, we can carry out automated temporal reasoning concerning *what could happen* in future given a specific state of affairs or *what might have happened* in the past given a specific outcome state. These forms of reasoning have relevance to the prediction (and possible remediation) of error, retrospective accident/fault analysis and the preparation of instructions/system documentation.

The structure of this paper is as follows; first we begin with a discussion of the Event Calculus (EC) itself and describe the key features of the approach. Second, we describe how task structure, staffing and information requirements can be treated within a model. Third, work to-date in modelling Time Sensitive Targeting is described. We conclude with an evaluation of the EC as a modelling technique and discuss the possibilities for repurposing the extant model as a decision support tool.

2. The Event Calculus

The event calculus (EC) is a logic-based formalism for reasoning about events and time (Kowalski & Sergot, 1986; Miller & Shanahan, 2002; see Shanahan, 1999 for an introduction). Although originally motivated by an interest in formalising the logical representation of time and applications in database updating and narrative understanding, this technique has been used since in myriad applications including commonsense reasoning for artificial intelligence (Mueller, 2006, 2009), making business systems more flexible (Yolum & Singh, 2004) and high-level vision for cognitive robotics (Shanahan, 2005). The three key elements of the event calculus are events, fluents and time. Fluents (after Newton) are time-varying properties of the world and at any given time point have a truth value, that is they are either true or false. Events occur at given points in time and alter the truth value of fluents. For example, the event of pressing an off switch will cause the fluent "device powered" to change from true (the device was powered up) to false (the device is not powered up). To unpack further, prior to the event a fluent had a given truth value and after the event it has a different truth value. Four basic EC predicates set out the formal relationship between fluents, events and time (Table 1).

Table 1. Discrete event calculus predicates

Predicate	***Explanation***
Happens(*e,t*)*	Event *e* happens at time *t*
HoldsAt(*f,t*)*	Fluent *f* is true at time *t*
Initiates(*e,f,t*)	If event *e* occurs at time *t*, fluent *f* will be true after time *t*
Terminates(*e,f,t*)	If event *e* occurs at time *t*, fluent *f* will be false after time *t*

** Happens(e,t) and HoldsAt(f,t) can both be negated; ¬Happens(e,t) event e does not occur at time t, ¬HoldsAt(f,t) fluent f does not hold true at time t. Negation (¬) is typically represented by an exclamation mark (!) in computer code.*

In terms of putting a model together, we combine a narrative of events (what events happened when), an axomisation of the domain of interest (what events do) together with the logic underlying the EC -- essentially a set of axioms that establish the mechanics of time and order themselves (see Shanahan, 1999) -- to produce a model that can be queried as to the truth value of fluents at given timepoints (what is true when). This is shown in Figure 1.

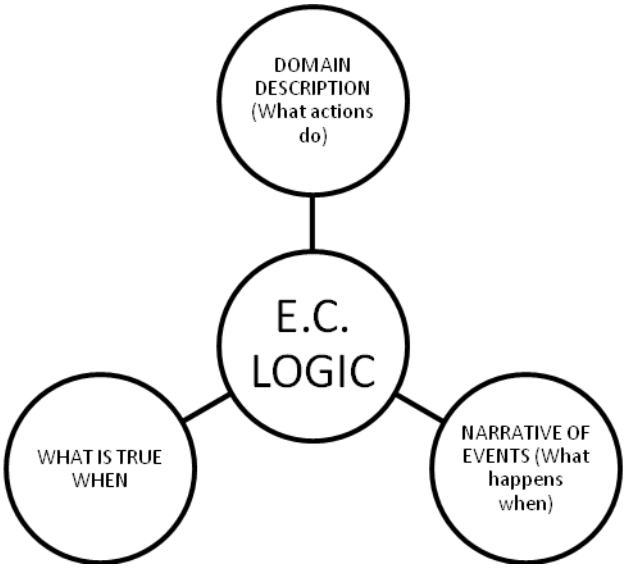

Figure 1. Elements of an Event Calculus model.

However, the relationship between these three sources of information can be rearranged if required (Figure 2); for example, if we know the state of fluents at different timepoints and what events do (to change fluents) we can infer what events might have taken place to make them so. Similarly, if we know what events occurred and how they changed the world, we could make inferences about effects those events have.

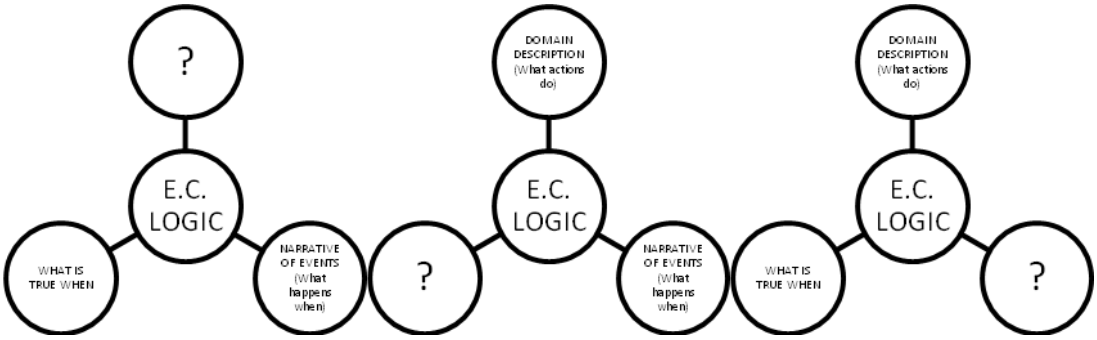

Figure 2. Possible re-arrangements of Event Calculus

Since the EC can be implemented using a computer these pieces of information take the form of a computer program. In the present paper we focus on a particular dialect of the 'Discrete Event Calculus' which represents time as integer timepoints that we can treat as arbitrary epochs primarily because of the ready availability of free software and tutorial/reference material (see Mueller, 2006).

2.1. Inertia, circumscription and the frame problem

One of the reasons EC is associated with problem-set described as commonsense reasoning in the Artificial Intelligence domain is that it embodies certain useful assumptions about causality in the world that are broadly understood to underlie the sort of implicit assumptions humans make but that are more difficult for AI. The so-called "commonsense law of inertia" holds that fluents retain their truth value until acted upon by an event that changes them (e.g., an appliance once switched off will not suddenly spring to back into life until switched on again). In specific cases this assumption can be relaxed if required, an example of this would be an object put in a box will change location in the world if the container it is within is moved ("release from inertia"). Circumscription holds that, unless we have reason to think otherwise, unrelated events such as opening a door do not cause an appliance to power up either. This assumption may at first glance seem trivial but is in fact very important in making representation of domains tractable; the alternative would be that we would have to exhaustively list for every event the non-effects of these actions however strange -- strange, because they militate against commonsense assumptions -- they might be (e.g., opening a door does not cause a CD to play). This difficulty is otherwise known as the Frame Problem.

2.2. Temporal reasoning

Given a start state and an ordered list of events, we can deduce what the possible end state(s) would be by deduction. Given an end state and a narrative of events we can also work out what the possible start state(s) were. Finally and perhaps most powerfully, given a start state and an end state we can also carry out a form of

reasoning termed *abduction*, that is, we can generate a plan or set of plans leading from the start state to the end state. We may also wish to use a partial narrative/set of observations concerning the state of fluents in the world and use the event calculus to generate ways of filling the gaps consistent with those facts. For example, if we know a door was closed and the room was empty at timepoint 1, and that someone was in the room at timepoint 5, then it can be deduced that some point in between, the door was opened to allow someone to enter. If the door was also locked, this would imply that had to be unlocked and imply it turn that the person who unlocked it had a key and so on. These forms of temporal reasoning are depicted in Figure 3.

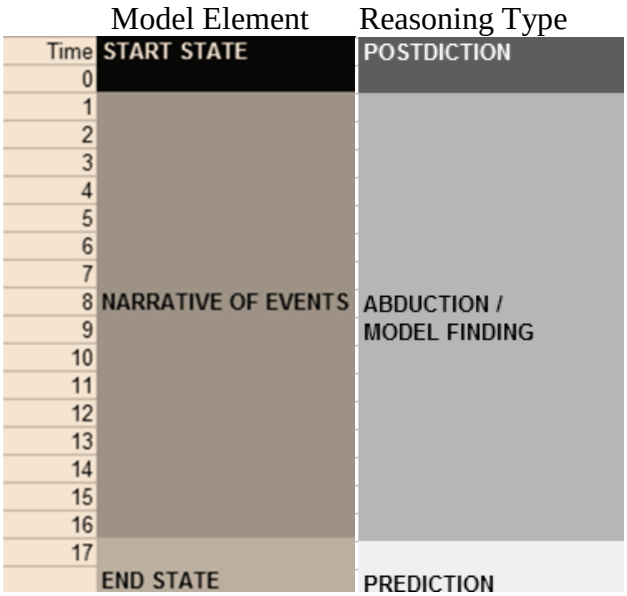

Figure 3. Forms of temporal reasoning in the Discrete Event Calculus

Therefore in terms of modelling and evaluating a system, we have two main approaches that we can employ. First, we can change the system (as it were) by altering its constituent axioms, and second we can change the narrative to test that system across different patterns of events. Indeed, by giving a narrative with a (semantically) negative ending (e.g., the fluent "target reached" is false) we can also generate narratives that expose latent and perhaps complex chains of events that can cause system failure. While this is arguably a reductionist way of representing events and tasks within a system, it does to some extent avoid the worst excesses associated with representing tasks as either hierarchical or ordered in a manner that is brittle in the face of changing demands or elaboration (e.g., for discussions see Suchman, 1987; Vicente 1999). Where events can be fluidly ordered or concurrent, this will be demonstrated, and if events do fall within a strict order, the reasons for this are directly demonstrable within the domain of interest itself (i.e., they result from genuine causal constraints and contingencies) rather than emerging as result of the analyst's need to impose order. Further, with reference to the meanings of the fluents changed by events, events can also be considered as meaningful and are given a specific context in events rather than being regarded as generic 'operations per unit time'.

3. Axioms for socio-technical systems

Sociotechnical systems theory is a concept originally attributable to the Tavistock Institute in London whose members, drawn largely from the social sciences, were concerned with how social systems relate to technological systems. A typical way of breaking this down is shown in Figure 4 (adapted from Bostrom & Heinen, 1977).

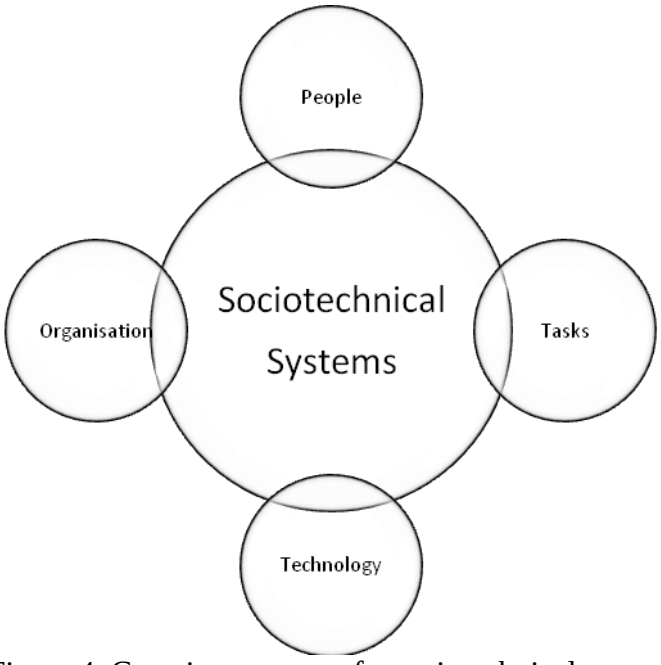

Figure 4. Constituent parts of a sociotechnical system

In constructing our model of military decision making as a sociotechnical system, it is necessary to axiomise the areas of interest; in essence this involves drawing up the rules that govern how our model functions. We considered three aspects; how tasks relate to each other in terms of order (task flow), staffing (people/organisation) and information requirements.

3.1. Task flow

Our approach to task-flow within our model was heavily influenced by the axiomisation suggested by Nihim Kesim Cicekli and Yakup Yildirim (Cicekli & Yildrim, 2000) and restated by Mueller in DEC form (Mueller, 2006). A set of axioms allow for activities to be in a state of being of either `Active` or `Completed` owing to Start and End activity events. Once an activity has been started, it is active, once an activity is ended it is completed, an activity that has been started no longer has the status of being completed, an activity that has been ended is no longer active;

```
Initiates(Start(activity),Active(activity),time).
Terminates(Start(activity),Completed(activity),time)
Initiates(End(activity),Completed(activity),time.
Terminates(End(activity),Active(activity),time).
```

The second line relating a Start event to the termination (i.e., rendering false a Completed fluent) can be removed in cases where we assert in the starting conditions that tasks begin "uncompleted" and we do not wish to examine situations where tasks may be performed more than once (i.e., repeated or reiterated). This is pragmatically appropriate but perhaps not rigorous in formal validation terms.

Further axioms using partially reified statements (i.e., that are specific to particular tasks here indicated by capital letters, not all tasks) allow the expression of particular forms of task flow:

In sequential action, one activity has be completed before another can start:

```
[agent,time] Happens(Start(agent,B),time)
->HoldsAt(Completed(A),time).
```

An AND-join requires that for an activity to start, two different activities must be completed:

```
[agent,time]Happens(Start(agent,E),time)
->HoldsAt(Completed(D),time) & HoldsAt(Completed(B),time).
```

XOR-split requires mutually exclusive conditions, task A can only have been completed if tasks C and B have not been completed or are active. Equally C can only have been completed if A and B have not been completed or active and task B can only have been completed if tasks A and C have not been completed or active.

```
[agent,time]Happens(Start(agent,A),time)
->!HoldsAt(Completed(B),time)& !HoldsAt(Active(B),time) &
!HoldsAt(Completed(C),time) & !HoldsAt(Active(C),time).
```

The XOR-join requires similarly exclusive conditions such that only one of a combination of tasks have been completed (A OR B OR C in this example):

```
[agent,time] Happens(Start(agent,D),time)-
>HoldsAt(Completed(A),time) | HoldsAt(Completed(B),time) |
HoldsAt(Completed(C),time).
```

There are various approaches that could be used to limit the duration of tasks, i.e., the period of time for which they are active. In the present case we limit each task to an active period of one epoch via a simple statement to assert that a task that was active in a previous epoch (n-1) cannot be active in the next epoch:

```
[activity,time] HoldsAt(Active(activity),time)
->!HoldsAt(Active(activity),time+1).
```

3.2. Staffing

Our present model of staffing is relatively rudimentary; we assert that one agent can perform one task at time and during that time it is true that the agent is busy. An activity can only start therefore if it has not been completed, is not active and the agent is not busy;

```
 [agent,activity,time] Happens(Start(agent,activity),time) ->
!HoldsAt(Completed(activity),time) &
!HoldsAt(Busy(agent),time)& !HoldsAt(Active(activity),time).
```

In order to further present agents either carrying out multiple tasks or multiple agents carrying out the same tasks we use the approach of stating that to do so would cause an inconsistency. Given the logical planner avoids inconsistency this is a way of articulating constraints, and perhaps a key area where EC description programming

appears somewhat counter-intuitive compared with conventional computer programming:

```
[agent,activity1,activity2,time]
Happens(Start(agent,activity1),time) &
Happens(Start(agent,activity2),time) ->
activity1=activity2.
[agent,activity1,activity2,time]
Happens(End(agent,activity1),time) &
Happens(End(agent,activity2),time) ->
activity1=activity2.
[agent1,agent2,activity,time]
Happens(Start(agent1,activity),time) &
Happens(Start(agent2,activity),time) ->
agent1=agent2.
[agent1,agent2,activity,time]
Happens(End(agent1,activity),time) &
Happens(End(agent2,activity),time) ->
agent1=agent2.
```

The relationship this gives between tasks, time and staffing can be seen in Figure 5 which depicts how fluents change over a six epoch period of time in a simple sequential A-B-C task model.

TIME	0	1	2	3	4	5	GOAL
EVENT	Start(A)	End(A)	Start(B)	End(B)	Start(C)	End(C)	no event
AGENT	Available	Busy	Available	Busy	Available	Busy	Available
TASK A		Active	Completed	Completed	Completed	Completed	Completed
TASK B				Active	Completed	Completed	Completed
TASK C						Active	Completed

Figure 5. Events and fluents over 6 epochs in a simple sequential task flow.

In time we hope to develop this staffing model more fully to better represent the use of specific resources (i.e., limit certain agents to specific tasks they are skilled at) and to model collaboration more satisfyingly than the present, which might at best be described as a "cooperation as task handover on completion" approach.

3.3. Information requirements

At the present time we have considered information flow only in a relatively simple manner pending further data collection. Information flow is often hard to track through socio-technical systems, particularly as "off grid" communication and habitual (but not standardised) patterns of informal communication may be common. In the light of this we have considered two simple approaches. The simplest way of representing information requirements is to simply assert that in the absence of information, certain tasks cannot occur (i.e., fluents representing the presence of particular types of information are false, representing that the information is not known). In the below example, we assert for example that tracking a targeting cannot occur without knowing the location and status of a target:

```
[agent,time] Happens(Start(agent,A),time)
->HoldsAt(HasInfo(Location),time) &
HoldsAt(HasInfo(Status),time).
```

However, in this formulation we are assuming there is only a single target and that "location" and "status" are single chunks of information across the whole system. As an alternative to this we may wish to have sets of information types associated with particular targets that certain tasks make true or false. For example, the act of target identification leads it becoming true that we know the location of any sort of target. Specific types of the sort called target may include HighValueTarget, Van, AmmoDump and so on.

```
Initiates(Targetidentification(target),KnowLocationTarget(target),tim
e).
```

This area of our work is at present somewhat undeveloped; in future work we hope to consider the situation where the information flow model can, in a sense, create a separate network to the stated taskflow network and to examine the interactions of the two. Some sketches for how this might work are given in the next section. We also hope to come to a more sophisticated view of how information is created and represented, possibly in-line with prior work carried out in our lab concerning the way in which various 'cognitive artifacts' allow the sharing, manipulation and re-representation of information (see McMaster, Baber and Houghton, 2006). This would also allow us to say something about the neglected "technology" quarter of the socio-technical system model in Figure 4.

4. Implementations

4.1. As task-flow only

The first implementation of a work-flow for planning and tasking was based around considering tasks as part of a largely linear workflow as shown in Figure 6. The model was inspired by the author's own observations of command teams at work although it should not be necessarily considered accurate but rather stands as a generic 'strawman' example to illustrate the approach. The purpose of the system is, from the point of target identification, to move through various forms of planning (threat assessment, coordination of airspace) and coordination/organisation (seeking approvals and clearances) until the point at which a UAV asset can be tasked and dispatched. Thus target identification can be regarded as the start state that initiates the process, and tasking assets can be regarded as the final endstate and goal.

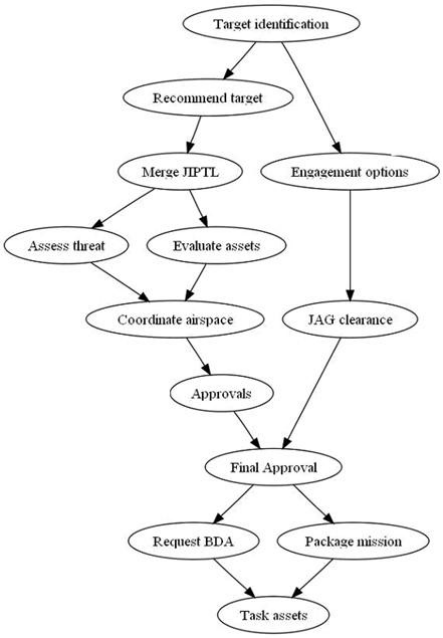

Figure 6. Simple model as task-flow

Once axiomised, a model is produced that can then be tested; given a fixed time frame of 25 epochs only a set of models were produced. Given this time limit, the left-hand thread (Recommend target → Final Approval) is constrained to only 'fit' if AssessThreat and Evaluate Assets are concurrent; meanwhile other models were possible but this only concerned when activities Engagement options and JAG clearance occurred; the limiting factor being JAG Clearance had to be completed in time to meet the synchronisation point of Final Approval. Counterfactual models, where for example Assess Threat took an epoch longer to complete than Evaluate Assets were rejected because the chain as a whole cannot complete in time; or when Final Approval was given without Engagement Options because it meant in turn JAG clearance could not happen) occurring were ruled out.

When a longer time frame was supplied (30 epochs) a wide range of plausible models were then possible, but clearly delays before synchronisation points propagated forward, thus in domain terms a delay in Coordinating Airspace pushed back the eventual Task Assets to at least a commensurate degree assuming no delays after that point.

This way of modelling a decision chain highlights some of advantages and disadvantages of managing organisations in this way. A signal advantage of the workflow approach is that, for example, we can be assured that assets are not tasked before JAG clearance has been given as a prior synchronisation AND-gate at the point of final approval requires that it has already happened. However, the issue of delays prior to synchronisation points propagating forward illustrates the disadvantages of working in this manner. Thus by choosing an axiomisation we can represent *a way of thinking about things*, in this case, the notion that decision chains should be largely linear and that through AND gating can be synchronised at given decision points.

4.2. As information flow only

An alternative way of considering (and thus axiomising) a decision chain is in terms of information requirements and the states prior decision reach. This is broadly the

notion that Network Enabled Capability/Network Centric Warfare will allow more distributed activities through the shared provision of information and great situational awareness not just of the situation in the field but also of what colleagues (perhaps geographically distributed across the battlefield or maybe even the world) are doing. With such information available for "pull down" from information systems, there is no longer the mandatory requirement to wait to be passed that information sequentially in the planning process. Equally, the products of various processing steps can be pushed back onto the net to be drawn down at the appropriate decision point. However, there is the risk that such approaches could lead to desynchronisation of the team as a whole and dangerous or unwanted events occurring. This poses a problem for analysis and management: how can we be sure given that activities could occur in myriad orders that we have designed a protocol that will be safe? We suggest one way is to model the protocol using the Event Calculus.

Another way of thinking about this is to consider a system protocol as a contract. As with written contracts, at various stages in the lifespan of contract certain things must be true for it to continue. In the event these things fail to be true, either the contract as a whole fails or certain tasks must be undertaken or re-undertaken in rectification. As well as a statement of what *must* happen, a contract will also describe various things that *must not* happen or else the contract to fail. Changes in these states either allow further actions that were anticipated to occur or trigger specific events in response. Note that owing to the temporal extension of the contract, states may switch back and forth between being true and false during its lifetime. A particular problem in temporally extended contracts is where events and states might interact in unforeseen ways; this may occur because the contract itself is badly structured (this stepping back from the analogy, the decision chain has potential flaws) or because events have not unfolded in the order originally anticipated.

In formalising a decision chain we used various fluents to represent this time not the status of given activities, but rather their outcomes; knowledge of the target's location, JAG approval and so on, has planning been completed and so on. For example the fluent `KnowLocationofTarget` can become true only after target identification:

```
Initiates(Targetidentification(target),KnowLocationTarget(target),tim
e).
```

(i.e., After target identification of a target, it becomes true that we know the location of a target).

In the case of some activities they may alter more than one fluent. For example, in the case of coordinate airspace we use a fluent to record that this has occurred (because it is necessary for packaging the mission) and also that airspace has been deconflicted (which is later necessary at the point of tasking assets):

```
Initiates(Coordinateairspace(target),
Airspacecoordinated(target),time).

Initiates(Coordinateairspace(target),
Deconfliction(target),time).
```

Similarly, other activities are possible only when multiple fluents are true; for example; it is only valid to give final approval (making the mission approved) when it is true that approval has been given and JAG clearance issued and we know the location of the target:

```
[target,time]Happens(Finalapproval(target),time) ->

HoldsAt(Approved(target),time) & HoldsAt(JAGcleared(target),time)

& HoldsAt(KnowLocationTarget(target),time).
```

It may seem curious that we have included the requirement to know the location of the target at such a late state as presumably we would have to know target location in advance of giving initial approval and JAG clearance. The point of its inclusion is that if we force a change in that fluent to false (i.e., we have lost the target), it is clear how the decision chain can shutdown or reiterate to a previous point depending on the circumstances. Furthermore by including these indirect dependencies we can ensure the same level of security that all that needs to be decided and known has been decided and known under more fluid orderings of events than a fixed task flow. Similarly, if we recast JAG approval as something that is either true or untrue at any point in the process (rather than making it a specific decision point in a linear chain) then this distributed protocol shows how it would be possible to withdraw approval at any point in the life of mission planning (as for example new information comes in) rather than at a single point. Such an approach would be possible given modern IT and furthermore would arguably be more appropriate to warfighting in highly dynamic circumstances that feature high levels of uncertainty. The point at stake here is that ground truth can change faster than the planning cycle can be completed and executed, it may be necessary to look at ways of reorganising planning.

An EC model can also 'trap' dangers inherent in a protocol. While in Figure 7 we see coordinating airspace is a procedural step dependent on the prior assessment of threat and assets, in our information requirements model we had not expressed this, nor had we made it a requirement for command approval; thus it was possible to find a path through the model that allowed a mission to be launched without airspace deconfliction having taken place. However, because the EC is elaboration tolerant, it was straight-forward to add a clause that airspace coordination should be in place prior to mission packaging.

On setting these definitions up such that abduction (plan finding) could take place we found a range of models that complete all the required activities in under 25 epochs. The general form of this class of models is shown in Figure 7. It is to be noted that while Figure 6 (workflow) was drawn and then formalised, this task flow was generated by the EC logic program itself and then drawn to permit comparison. It might also be considered slightly misleading in that in contrast to the workflow axiomisation, the event "Request BDA" is in point of fact as dependent upon "Target Identification" (which changes the `KnowLocationOfTarget` fluent from false to true) as it is on the preceeding "Engagement options" event (which changes the fluent `Missionplanned` from false to true).

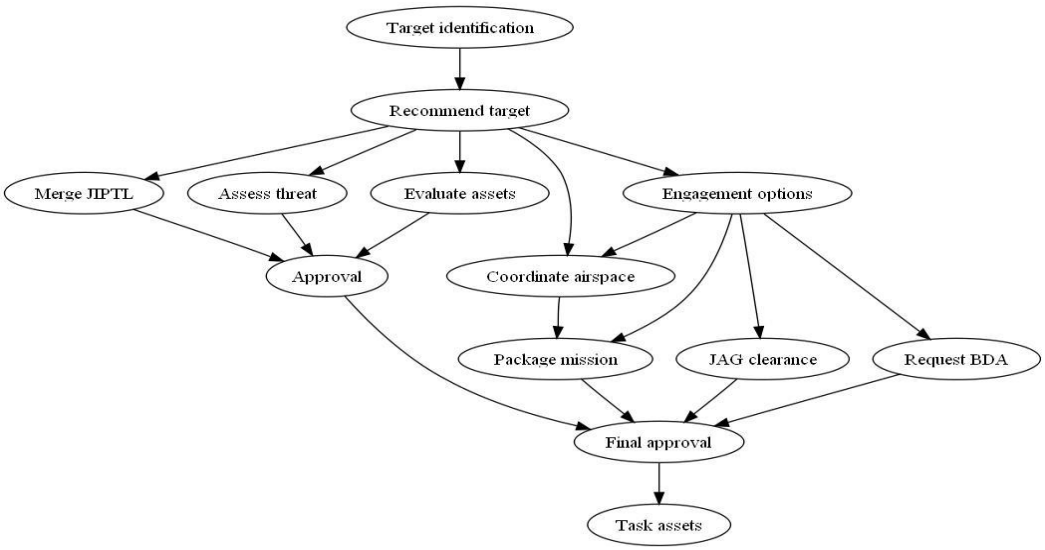

Figure 7. General form of information requirements model

4.3. Current work: Full implementation

Current work in our laboratory has involved collection of data from Subject Matter Experts and investigations of various scenarios involving the F2T2EA process to replace the "straw man" process described in previous examples with something better grounded. The Find, Fix, Track, Target, Engage, Assess process in generic form is shown below in Figure 8.

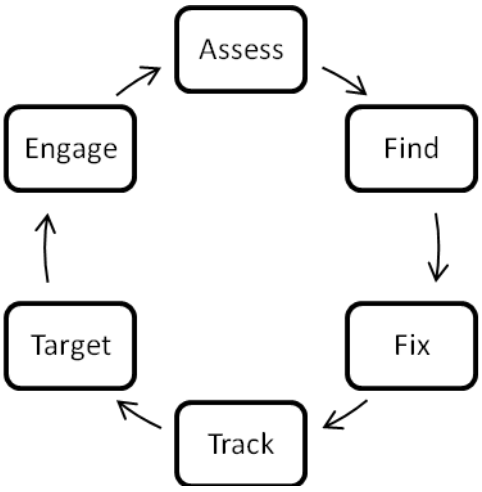

Figure 8. F2T2EA (**F**ind, **F**ix, **T**rack, **T**arget, **E**ngage, **A**ssess) process overview

So far we have considered in depth the Engage and Assess sections of the process, resulting in the task flow model in Figure 9.

Figure 9. Task flow model for Engage and Assess

Early investigations using this part of the model have concerned staffing (which in our present version of staffing essentially amounts to asking what degree of parallelism would be useful in carrying out taskflow). Running the model demonstrates that the high degree of linearity in the flow model creates little gains in time to completion by increasing parallelism as would be expected (Figure 10). However, with the addition of information requirements whose structure may not map directly on to task structure we believe it will be possible to identify discontinuities between the two and perhaps "hidden" synchronisation/pinch points.

Figure 10. Effects of staffing resource levels on time-to-completion

As an example of temporal reasoning possible with this model as-is, consider the following arbitrary question:

Knowing that without parallelism the process can terminate in 38 epochs, we ask the following question that has a far from obvious result: given that "confirming the target identity" (G) occurs late and thus by epoch 20 (instead of 12) "proceed to target decision" (M) is made by epoch 26 (later, but not as late relatively speaking) and from this point an extended deadline of 48 epochs, is it possible the process and complete and what would the potential narrative look like?

One possible solution is given in Figure 11.

Time	Agent1	Agent2	Agent3
0	A start		
1	A active		
2	A comp		B start
3			B active
4	C start		B comp
5	C active		
6	C comp		D start
7			D active
8	F start		D comp
9	F active		
10	F comp/G start		
11	G active		
12	G comp		I start
13			I active
14		E start	I comp
15		E active	
16		E comp	
17			
18			
19			
20			H start
21			H active
22			H comp
23	J start		
24	J active		M start
25	J comp		M active
26			M comp
27			
28			
29			
30	R start		
31	R active		
32	R comp	P start	
33		P active	
34	S start	P comp	
35	S active		Q start
36	S comp		Q active
37	T start		Q comp
38	T active		
39	T comp		U start
40			U active
41			U comp
42		V start	
43		V active	
44		V comp	
45	X start		
46	X active		
47	X comp		
48	MISSION ENDS	MISSION ENDS	MISSION ENDS

Figure 11. Possible timeline for mission completion in arbitrary scenario

4.3.1. Use as decision support and limitations

The above worked example demonstrates one way in which this framework could be used to produce a decision support tool that might help commanders and indeed a collaborating command team navigate the process itself. In current planning/execution activities it is common to colocate command staff and to charge a member of staff termed a "floorwalker" (who literally 'walks the floors' to observe how work is progressing) with maintaining the commander's situation awareness with regard to the execution of the planning process. The present framework, run on a reiterative basis with real-time data being timestamped and added to the list of known observations could serve to allow the ongoing simulation of future events. This might be of particular use in situations where via NEC/NCW teams are distributed and it may therefore be difficult to have visibility of how the work of planning is evolving answering questions such as "where are we up to?", "what next?" and "if the process is being held up, what is the root cause and can we deploy additional resource to improve this situation without slowing other ongoing work?". This event monitoring framework could also connect to other computer systems to help facilitate the planning process in a timely and complete manner.

However, by locating this work in a concrete setting (i.e., a decision support system) this brings to light several limitations of our present work that require discussion. The first issue is that we may believe the task flow model is open to challenge. For

example, Task E, "monitor target area" may not be a task of discrete duration, rather it could be argued that it will go on throughout the operation. There are four possible responses to this. First, this points to problem in our data collection and we need to return to SMEs for further clarification. That a model can provide a rebuttal to an analyst's misunderstandings is actually a vital service they can perform in and of itself. Second, it may be suggested we should use a different task model where the ongoing "active" status of Task E should persist throughout the operation. In our present formulation, each task is limited to a duration of one epoch, but clearly there will be cases in which this should be treated more intelligently. This is also particularly relevant to the consideration of staffing proper, as resources may be tied up on certain tasks for an extended period. A third approach would be to use the full EC (which can use continuous time), rather than the epoch-limited Discrete EC, to model the system. We have used the DEC in the present instance for the pragmatic reasons that the software and tutorial materials are readily available[1] and, for the purposes of this paper, because discussion and illustration of the DEC is somewhat simpler than the full EC. A fourth hybrid approach is also possible; retain the epochs of the Discrete EC but retrospectively apply and tally empirically derived typical unit times and some indication of variability (e.g., standard deviation) as is sometimes used in Petri net modelling.

A second limitation in the present work is the way we have treated decision points. In the present model we have simply considered making a decision as a task to be done. A variety of outcomes are in some cases possible (e.g., after the "Abort decision" point) that arise from making decisions, the output of our present model considers them all as equally likely and thus various output-narratives take different routes as if different decisions were made. This is required for situations where we are checking to see if undesired outcomes are possible (e.g., decide to abort mission, ordnance still deployed at a later time point). One caveat to this is that we have generally set the endstate condition as the completion of the mission (i.e., X "Mission complete") which means that some paths are never taken (e.g., task O which refers to divert). The reason these paths are never taken is itself through temporal inference, if the mission is completed following the dropping of ordnance, it cannot be the case it was previously diverted (at least within our model of it). In part this is because the model is incomplete and we have only so far looked at Engagement and Assessment; some of these alternate parts revert F2T2EA process back from engagement to the resumption of attempting to (re)Fix the target. In a full implementation these links would be included. An alternative approach to decision points might be to axiomise these decisions themselves. For example, some sort of probabilistic estimate of likelihood could be included. However, to make this approach satisfying it would probably be necessary to have these probabilities sourced from another program that runs such simulations that is queried by EC framework only at decision points and this input recoded into the state of a fluent. The EC framework itself deals best with matters of propositional logic and offers relatively little support for any sort of statistical data manipulation.

5. Evaluation

[1] Please see http://decreasoner.sourceforge.net/ and Mueller (2006), (2009) for further information.

Our evaluation and experiences of using this technique to-date can be summarised thus:

- The event calculus is well-suited to modelling how systems perform over a period of time.
- There is a learning curve involved in forming adequate and appropriate propositions to capture the key elements of systems; however the output of this process is arguably much easier to understand than, for example, mathematical outputs, as logical statements translate relatively easily into normal English.
- Furthermore, the rather involved process of constructing axioms has the useful side-effect that it forces the analyst to clarify their understanding of the system in a formal manner; arguably if we lack the information to construct axioms, we lack a detailed understanding of the system.
- As the work reported here is ongoing, we have yet to reach a satisfying level of analysis of staffing and information requirements. We hope to improve these aspects of the model iteratively by gradually increasing the complexity. Suggestions are given in sections 3.2, 3.3, 4.2 and 4.3.1 for how future work may progress.
- Constructing these models is time consuming. While not excessively so compared to comparable human factors/systems design and evaluation methods, for larger and more complex systems than those considered here it might be necessary to write a computer program that can itself produce EC logic programs rather than as in the present case, writing each line by hand. This approach has been used previously in applying the EC to natural language understanding that employed a parser to convert free text into logical form.
- Combinatorial explosion (and the consequent load this places on computational resources) may put a limit on the scale of usable models. The largest models described in this paper took up to 13 minutes to encode, if albeit less than a second to process to solution after that point. The largest single variable in increasing processing time in terms of the contents of our models is the presence of multiple agents. While further optimisation of satisfiability encoders may be possible, this is an active area of research and we may consider what we are currently using in conjunction with the DEC using to be state of the art. Thus, a better candidate for reducing processing time may be the brute-force approach of using greater hardware resources, the present models ran on a single-core 32-bit processor with 1 gig of memory, upgrade to a workstation (e.g., multi-processor, XXX gigs of memory) might improve matters.
- Functionally, the model described herein in 4.1 and 4.3 most closely approximates the Petri net approach. It differs from the Petri net model in that (1) the ordering of tasks is not necessarily fixed (i.e., it can under certain circumstances "rewire" itself to produce different patterns of flow, as per the implementation in 4.2) and (2) the range of expression possible in a logic-based formalism is arguably wider than that permitted within a Petri net approach. Whether a range of expression is an advantage or a vice probably depends on the rigour and accuracy of the implementation.

5.1. Process

In terms of how the EC and use cases may fit in and around other analysis techniques in Human Factors and Systems Engineers, we suggest the overarching process from elicitation to measurement in Figure 12. Thus we begin with an elicitation and representation phase in which information about an existing system or specifications for a future system are collected. At this point techniques like Hierarchical Task Analysis might take place which would also lead to the initial draft of an EC model. Then through model finding (abduction in the EC) various configurations of the system are examined on the basis of the elicited facts. At this stage it may well be necessary to go back to further elicitation in order to "sanity check" outputs and also to tighten up any aspects of the axiomisation that appear unclear or open to question.

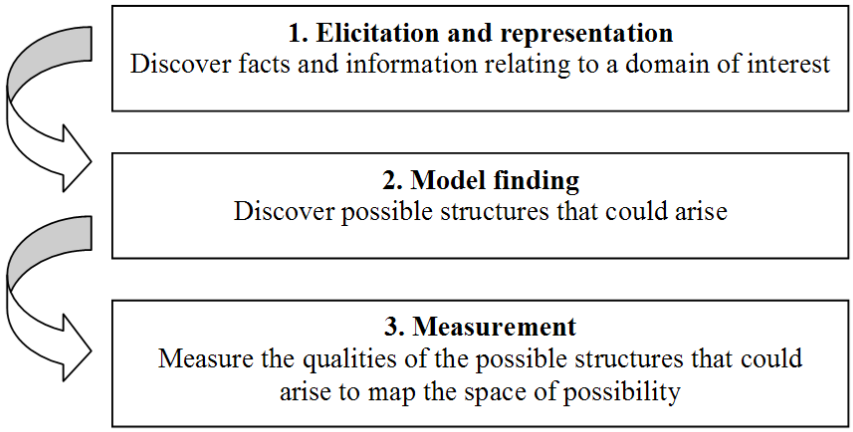

Figure 12. Possible process model for analytical prototyping.

Once models have been found, these can then be submitted to simulation and measurement (this may include Petri Nets and other workload and error analysis techniques). Event calculus could also be present at this stage to see whether the system is compatible with different (and perhaps exceptional) narratives of events.

References

Bostrom, R., Heinen, J. S., (1977), MIS Problems and Failures: A Socio-Technical Perspective, *MIS Quarterly, 1* (3), 17-32.

Coram, R. (2002). *Boyd: The fighter pilot who changed the art of war*. New York: Little, Brown and Company.

Cicekli, N. K. & Yildrim, Y. (2000). Formalizing workflows using the event calculus. In M. T. Ibrahim, J. Kung, and N. Revell (Eds.), *Database and Expert Systems (Lecture Notes in Computer Science), 1873*, 222-231.

Hildebrandt, M. & Rantanen, E. M. (2004). Time design. *Proceedings of The Human Factors and Ergonomics Society 48th Annual Meeting*. 703-707.

Kowalski, R. & Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing, 4*(1), 67-95.

McMaster, R., Baber, C. & Houghton, R. J. (2006). Investigating Distributed Cognition Processes during Effects-Based Operations. In *Proceedings of The Technical Cooperation Program Human Systems Integration Symposium*, May 4-5, Canberra, Australia.

Miller, R. & Shanahan, M. (2002). Some alternative formulations of the event calculus. In A.C. Kakas and F. Sadri (Eds.), *Computation and logic: Essays in Honour of Robert Kowalski, Part II, Lecture notes in Computer Science*. Berlin: Springer, 452-490.

Mueller, E. T. (2006). *Commonsense Reasoning*. New York, NY: Morgan Kaufman.

Mueller, E. T. (2009). Automating commonsense reasoning using the event calculus. *Communications of the ACM*, 52 (1), 113-117.

Shanahan, M. (1999). The event calculus explained. *Lecture Notes in Artificial Intelligence*, 1600, 409-430.

Shanahan, M. (2005). Perception as abduction: Turning sensor data in meaningful representation. *Cognitive Science*, 29, 103-134.

Suchman, L. (1987). *Plans and situated actions : The Problem of Human-Machine Communication*. New York: Cambridge University Press.

Vicente, K. J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer-based work.* Mahwah, NJ: Lawrence Erbaum Associates

Yolum, P. & Singh, M. P. (2004). Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Artificial Intelligence, 42*, 1-3, 227-253.