

# *Tools for Thought*

Dr Rob Collins

© Rob Collins, 2013

<http://users.ox.ac.uk/~kell0956/>

## Tools to Enable ..

- Collaboration and Consensus building between diverse teams
- Documentation of Rationale and Explanation
- Alignment from Enterprise Strategy to Systems Design

# Proof!

To show:  
 $\mu.X.F(X) = F(\mu.X.F(X))$

Establish an ordering relationship:  
 $(A,S) \subseteq (B,T) = (A=B \wedge S \subseteq T)$

This is a partial ordering in the sense that:

L1:  $P \subseteq P$   
 L2:  $P \subseteq Q \wedge Q \subseteq P \Rightarrow P = Q$   
 L3:  $P \subseteq Q \wedge Q \subseteq R \Rightarrow P \subseteq R$

Define a chain in a partial ordering:  
 $\{P_0, P_1, P_2, \dots\}$  such that  $P_i \subseteq P_{i+1}$  for all  $i$

Define the limit (least upper bound) of each chain ...  
 $\sqcup P_i = ( \alpha P_0, \cup \text{traces}(P_i) )$

..Is a complete partial ordering iff

L4:  $\text{STOP}_A \subseteq P$  , provided  $\alpha P = A$   
 L5:  $P_i \subseteq \sqcup P_i$   
 L6:  $( \forall i \geq 0, P_i \subseteq Q ) \Rightarrow ( \sqcup P_i ) \subseteq Q$

“I would like to spend the first 45 minutes of this session, talking through this proof in some considerable detail  
 ....

That’s what I would like to do ... sadly, I don’t really understand a word of it. What I know about the Fundamental Theory of Recursion you could write on the back of a very small postage stamp and I don’t know a lemma from a lemming.

I do, however, find this sort of argument very beautiful and elegant. And, of course, it has a level of ‘certainty’ associated with it: If we accept the premise of a logical, formal argument, and if there is no error in the proof, then we are pretty much bound to accept the conclusion. That is the nature of formal arguments.

And certainly there are some really important applications of formal reasoning in our field of large scale Systems Engineering most notably in the formal methods applied in Safety Critical or Security Critical systems to first unambiguously specify a system and then to construct it through a process of successive refinement.

But I do have a couple of objections to this sort of formal, logical argument as it might be applied in our domain of interest. Firstly, it does require a level of sophistication in the notation and process of formal reasoning. This is not, after all, the kind of thing that I could put in front of many systems ‘end-users’ and hope that they can connect it with their personal, affective, rather more ‘woolly’ but desperately important requirement such as “I want the system to be safe”.

Secondly, it has been argued (most notably by Steven Toulmin in the 1950s) that most of the really useful and interesting arguments in the world are not of this formal, logic type. We cannot just ‘plug in the numbers’ and guarantee that the correct answer will fall out. Most of the useful, interesting and common arguments are ‘defeasible’. Our arguments are ones over which reasonable, intelligent and moral individuals may weigh all of the evidence and argument and still, in all conscience arrive at different conclusions.

And here I do find a resonance with large-scale Systems Engineering. Diverse groups of individuals, with skills from many different domains, with various priorities and personal experience may consider a system design and may reasonably disagree about almost any aspect of it: its basic necessity, its requirements, its architecture, its detailed design, its implementation and so on. And yet still we need to work together as diverse teams and finally arrive at a realised solution. And if we are smart and diligent then we should do that by taking on board the wisdom of the large crowd of people that contribute to a modern large-scale system development. We need to listen, integrate the knowledge, share it, resolve reasonable differences, reject some of it (for good reasons or bad) and finally come up with a solution.

It is this process of dialog, argument, sense-making, consensus building and knowledge representation with which this presentation is concerned.

(But for those who would rather stay with the comfortable, rationale, predictable arguments I would certainly, and with some enthusiasm refer you Prof. Sir Tony Hoare’s “Communicating Sequential Processes” - Fundamental theory of recursion. Page 94 – from which the (incomplete) proof, above, was extracted)

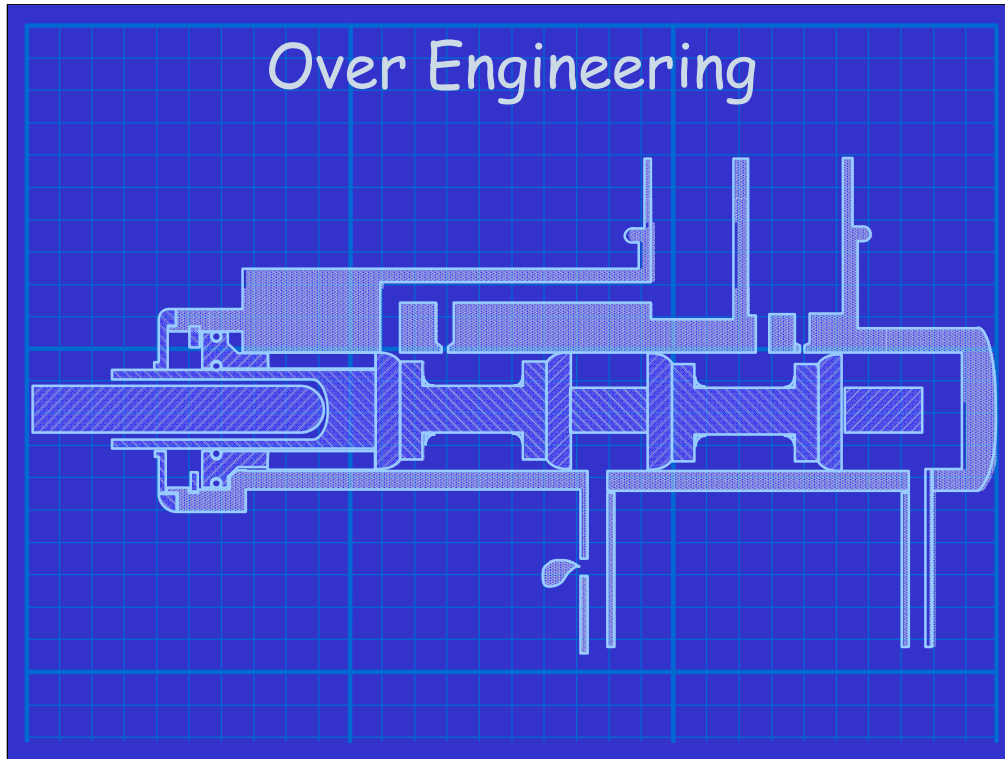


So ... since I am egregiously unable to share much wisdom in the domain of formal logic or mathematical proof I thought I would do better to share with you something from the world I actually inhabit. The world of real, tangible, 'Engineering'. This beautiful machine is my 1979\* VW Berlin Deluxe, Original Westfalia conversion VW 'Kombi'. As Jeremy Clarkson might say "79 brake horse-power of unadulterated motoring pleasure"

The thing is ... 35 year-old vehicles do tend to fall apart a bit. And this summer I had a problem with a leaky bit, which caused her to fail her MOT.

Before reading the notes on the next slide, have a look at the schematic and see if you recognize what it is .. or (heaven forefend) if you are not a VW bus enthusiast work out its function from its physical form....

\*Ignore the T registration, she was an import.



You will immediately recognise this as a Master Break Cylinder unit – which is common to many vehicles, not just VW camper's. Without wishing to sound too technical – the plunger on the left is the “foot-pushy-downey bit” and the two pipes from the bottom are the “hydraulic squirty out tubes” which go to the brakes (I think). The two reservoirs at the top are the “top them up with brake fluid or the MOT guy complains” thingeys.

Now, I found it rather difficult to source a new component to replace mine when it started to leak – partly on account of the heinous decision by VW in Brazil to stop manufacturing these vehicles this year! (Hence robbing us enthusiasts of a ready supply of shoddy, low-quality parts.)

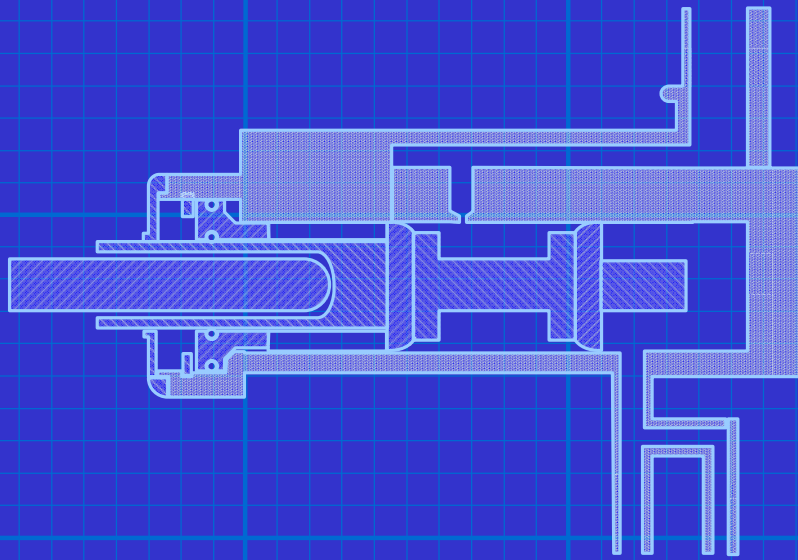
So, in looking at this component I could not help but think how rather over-engineered it was. It does seem rather ... complicated.

So, Lean Six-Sigma Black Belt\* that I am, I have had a go at ‘leaning’ out the design and producing a more simple, easier to manufacture and less costly alternative.

Perhaps you would care to do a quick design review of the schematic on the next slide and consider the excellence of my design?

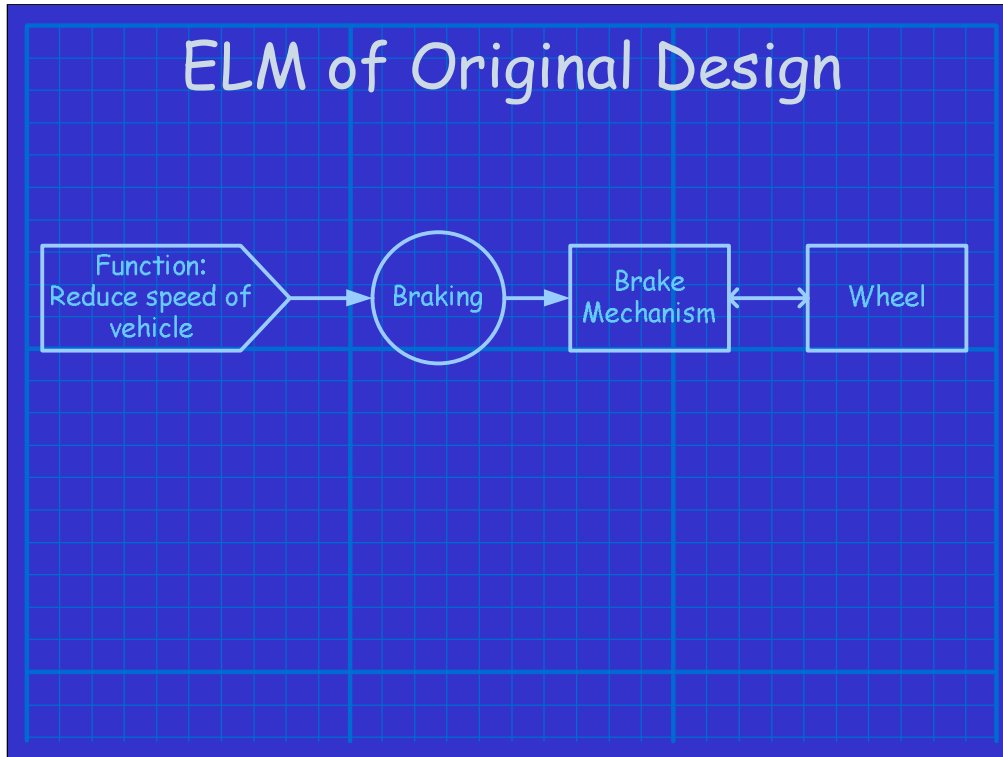
\* Though sadly, not ‘Trained Hydraulic Designer’

# Lean Engineering!



'Nuf said really!

A magnificent improvement by any standards!



But what about if those clever German VW engineers had read my 1997 paper on capturing design rationale (Collins, 1997)? Maybe they would have drawn up something like this slide.

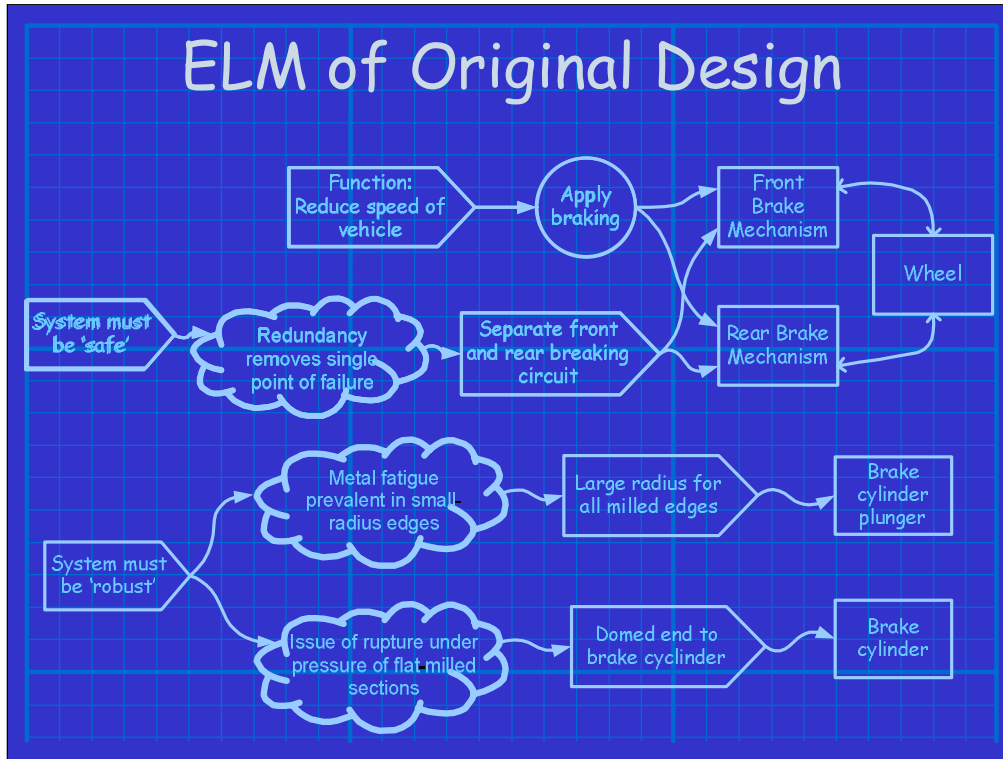
The notation is rather straight-forward: The 'arrow-box' represents a requirement. The circle represents a Process (verbs). The oblongs represent Objects (nouns).

There are two types of arrows. The first, with a single, solid head is a soft implication. You can read it as "leads to" or "suggests" or "traces to" \*. The other, double-header arrow is a (possibly labelled) relationship.

This diagram tells a story. "We have a requirement to reduce the speed of this vehicle. This leads to a process we call 'braking'. This in turn leads us to a mechanism to implement this 'braking' process. This mechanism is associated with the Wheels – although not in a causative sense: Wheels don't exist because of braking – but in this design they are associated with it.

But at some level this 'story' is obvious – and it does rather seem as if something is missing which would explain the relative complexity of the original design.

\*Let's not get hung up at the moment on formal semantics – because that is what we are trying to escape from a little bit.



These diagrams are “Essential Logic Models” (ELMs). In the top diagram there is a new symbol – a fluffy cloud, representing an idea or concept. This is crucial to the explanation. We can’t move directly from requirements to designed artefact without the special, expert knowledge that we as system designers bring to the party. Or, to put it another way, we need those rather important bits of knowledge that exists inside the head of the trained, experienced hydraulic engineer or safety engineer on the team. We need to be thinking about safety and hence redundancy.

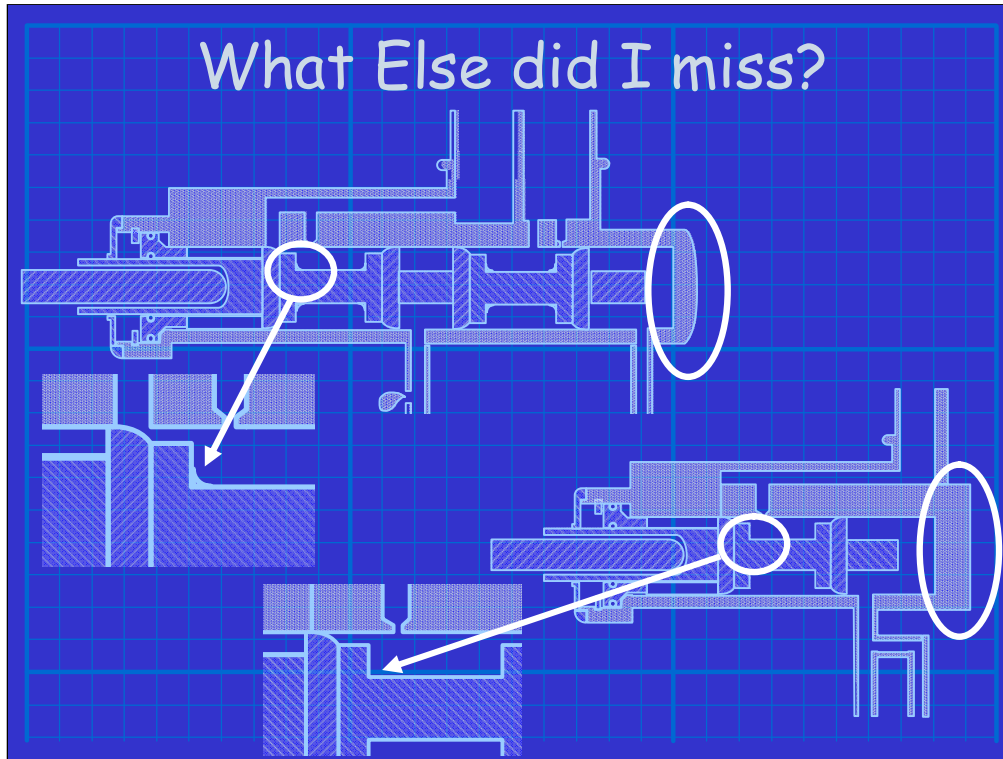
In my view the physical design of this object makes it function rather self-evident. It seems to me that this is often the case – function is rather ‘explicit’ in a realised design.

But this other, safety, aspect is rather less explicit. At least, it may be obvious to some ... but not to all.

If I have not made the case for the safety aspects of this design, then maybe considerations of the ‘robustness’ aspects make the case even better. Consider the bottom diagram and notice that there are some features of the first design that are not ‘immediately self-evident’ and are not replicated in my ‘improved’ design. The comparison on the next page makes this clearer ..

The original ELM paper is here: <http://users.ox.ac.uk/~kell0956/docs/elm.pdf>





So .. As well as not really understanding the safety aspects of this device, I also failed to understand something about its design for robustness.

And as a Systems Engineer this is actually not very surprising. Most of the Systems Engineers I meet are pretty smart people, and rather polymathic. But we can't really hope to be experts in all of the diverse disciplines that are required to construct a modern, large-scale system: Mechanics, hydraulics, electronics, logistic support, maintainability, safety, reliability, software, human factors, aerodynamics, electro-magnetic susceptibility, security ... and the list really does go on for a very long time.

As part of my Systems Engineering course at Oxford University I say that the key 'exam question' is: How do you get 2000 people to design a machine which is so devilishly complicated that nobody understands it?

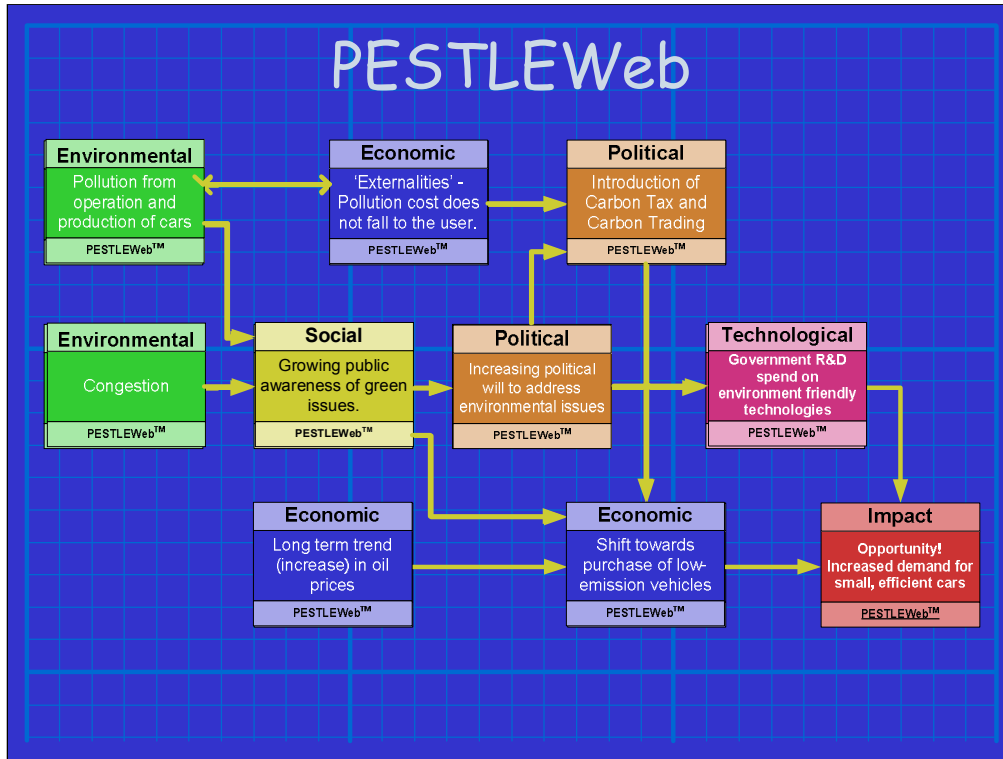
My conclusion is that in Systems Engineering we are engaging in a social activity which has to draw on a huge and diverse group of experts and then to synthesise this huge body of knowledge through increasingly elaborate series of constraints to the ultimately realised object.

And so my question is .. How do we engage in this dialog? How do we record it for posterity - not forgetting this that large system may be operational for 10-50 times longer than it took to build. People may still be making modifications to our systems when we are long gone.

So ... at this point you might reasonably say "Ok. I see that these type of 'story telling' diagrams might have some application in product design... But what about the level we are thinking about here, Enterprise Systems Engineering?" And my answer to that, is to refer you to the results of my MBA thesis, discussed on the next slide.

# Enterprise Environment

- Prerequisite of successful strategy ...
  - Develop a shared understanding of context
- Traditional PESTLE analysis ..
  - P = Political
  - E = Economic, etc. etc.
- We need more than a mnemonic to:
  - Gain deep understanding
  - Develop consensus in diverse teams
  - Consider structure and dynamics



PESTLEWeb is a tool for understanding and explaining the Enterprise Environment - whether this be for reasons of business strategy, or as context for a large scale systems development. As for the ELM, this diagram tells a story. It shows cause and effect relationships between elements in the model.

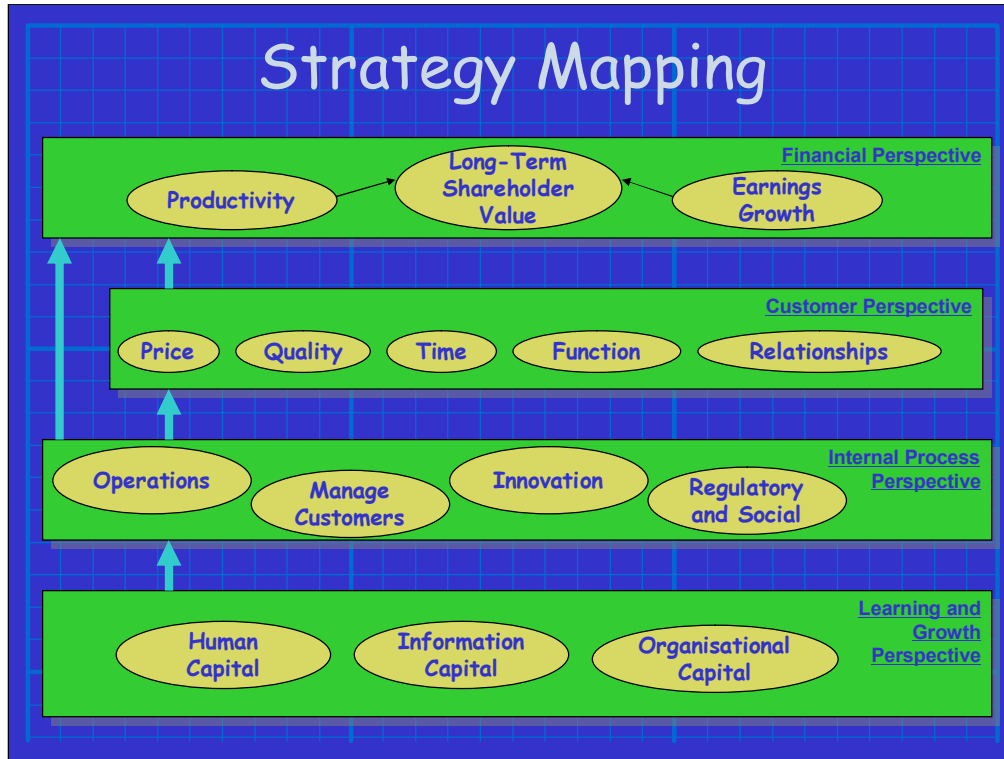
The objective here is 'sense-making' : to intellectually engage with a series of disparate 'facts' and try to weave them together into a coherent, and rationale structure. Simple put, we need to move from having some observations about the Enterprise environment to having a theory about it. We need a theory – because we are about to do an 'experiment' – and the experiment is a rather important one: it is the design and implementation of our Enterprise strategy.

This is certainly a defeasable argument – reasonable people might disagree about it. At least by documenting it in this way we can draw people into the dialog, and record and respect their ideas and opinions. These diagrams can represent multi-dimensional relationships which are rather difficult to express in narrative text.

There are a lot of reasons why such diagrams seem to work, and I would refer you to my MBA thesis for a full discussion of this:  
[http://users.ox.ac.uk/~kell0956/docs/pestleweb\\_thesis.pdf](http://users.ox.ac.uk/~kell0956/docs/pestleweb_thesis.pdf)

## PESTLEWeb Results

- In a controlled, experimental study, PESTLEWeb models were considered:
  - More persuasive
  - More engaging (interesting) and
  - More 'rational'
  - (Statistically significant result at 1% level)
- In a qualitative, observational study:  
Easy to teach, learn and use
- 6000+ downloads of tutorial slides on [slideshare.net](http://slideshare.net)



What about all the stuff that happens between the most abstract and ethereal ‘Enterprise Environment’ and the low-level product design? For some of that, we can turn to the Strategy Maps of Kaplan and Norton.

Part of my argument is that Enterprise Systems Engineering must necessarily be tightly bound with Enterprise strategy

- These large scale, high-value systems will in many cases be central to achieving the corporate strategic objective
- We therefore need alignment between enterprise strategy and Systems design

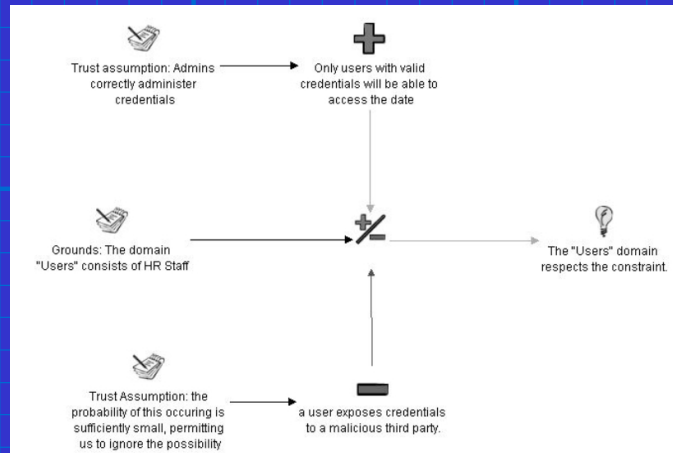
Kaplan and Norton propose their ‘Strategy Maps’ as tools for thinking about strategic alignment – getting us all marching in the same direction. My example here is generic (to aid explanation) but I would refer you to Kaplan and Norton’s excellent book on this topic within which they provide many examples of real strategy maps. Note:

- Strategy Maps are Visual Arguments
- They are intended to support collaboration, build consensus, make the implicit visible
- They represent an evolving conversation with the organisation..

Interestingly Kaplan and Norton do mention ‘systems’ many times in the book – but they are generally using the term in a more specific way that we do: Enterprise management systems : Computer databases, stock control, data-warehousing. I.e. IT systems. They locate ‘systems’ more specifically than we would – in the area of ‘Information Capital’, we would probably think of systems having an impact much more broadly in these maps:

- Customer Perspective: User Requirements, Externalities, Stakeholders, reducing costs, saving time, providing new functions....
- Internal Process Perspective: I would change this to “Process and Systems Perspective” – technology systems (lumps of hardware, human systems etc.)

# More examples of Visual Argumentation for Design



**Buckingham Shum's use of Compendium Argumentation tool in Software Engineering (Example argument about security)**

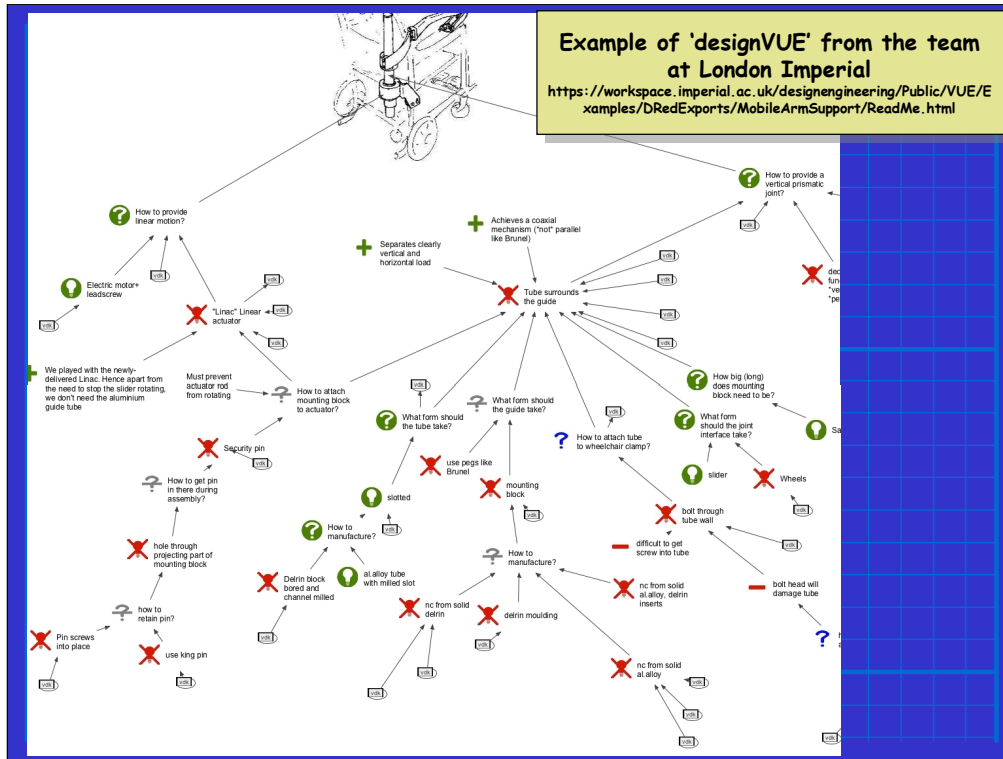
These next two slides are principally to illustrate that this subject of 'Defeasible Visual Argumentation' is neither obscure nor new. There has been excellent work done by the team at the Open University over many years and on the following page is an example from the team at London Imperial. I have included a list of references on the last slide of this deck which provide a route into this body of work – I definitely believe it is worth reviewing in context of Systems Engineering. The 'designVUE' tool from the team at London Imperial is available for download from their website.

I believe that there is really important work here. I would only say that I differ in my views in two areas from these other teams. Firstly, they seem to be focussing on the development of specific tools for argumentation. I actually believe we Systems Engineers already have tools that are perfectly capable of representing such arguments. It would actually be a mistake to let this information leak into another tool.

We need these stories / arguments to be embedded in, and integrated with our Systems Engineering Database. In fact, I believe we have here the mechanism for creating the rich traceability which has so often alluded us. We may consider the rather weak mechanism provided by sysML to embed requirements into models and attach them to classes, objects and properties etc. But sysML does not intrinsically provide the notation I am talking about here\*. Argumentation is the missing 'glue' that enables a coherent thread to be drawn from requirement to realised design.

The second area I disagree with the other teams is their area of focus. The Open University team seems to have worked a lot in Software Engineering and Education. The Imperial team seems to be focused on product design and particularly mechanical products. I believe that the work being done would be of more value applied in the area of Systems Engineering, and particularly Enterprise Systems Engineering. We are in much greater need of eliciting, recording and representing the dialog between diverse, expert individuals than will ever be the case for low-level product design.

\* That is, there are not first-class meta-model element within sysML for 'argument' etc. However, as I will show later, these elements are easily represented through the use of stereotypes

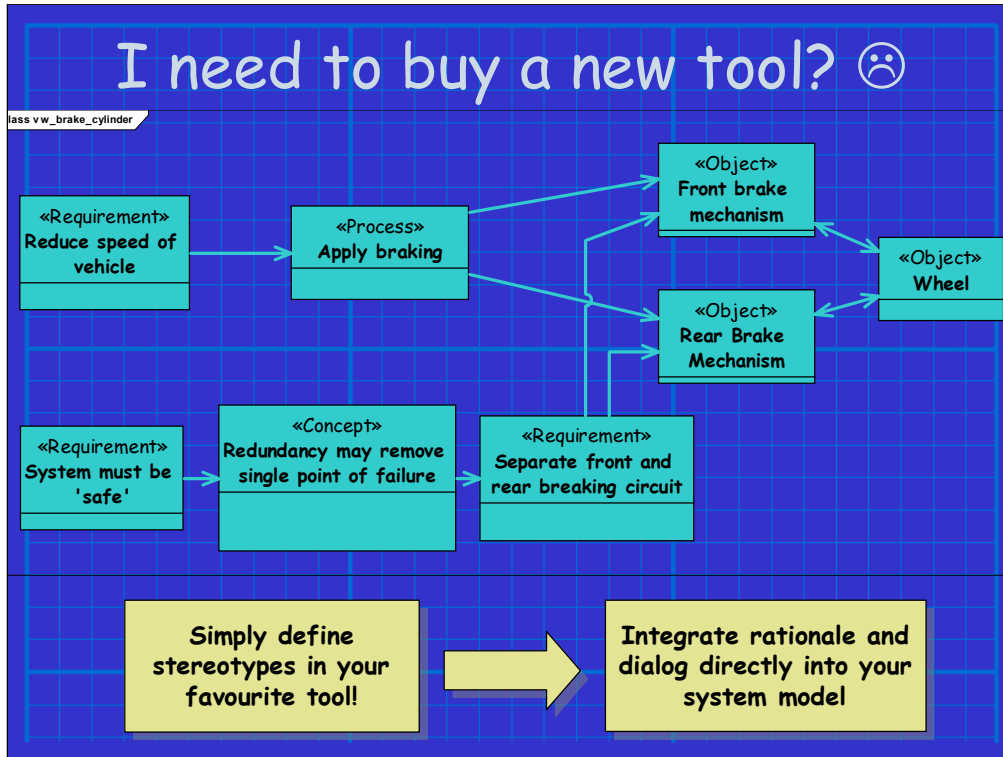


Note one feature of these diagrams. They do not need to only record the ideas that were adopted – they can also record the ideas that were rejected, along with an explanation of why that choice was made. Here is a source of real Enterprise Wisdom – deep knowledge and understanding about decision made in the process of design.

## What is VA doing for us?

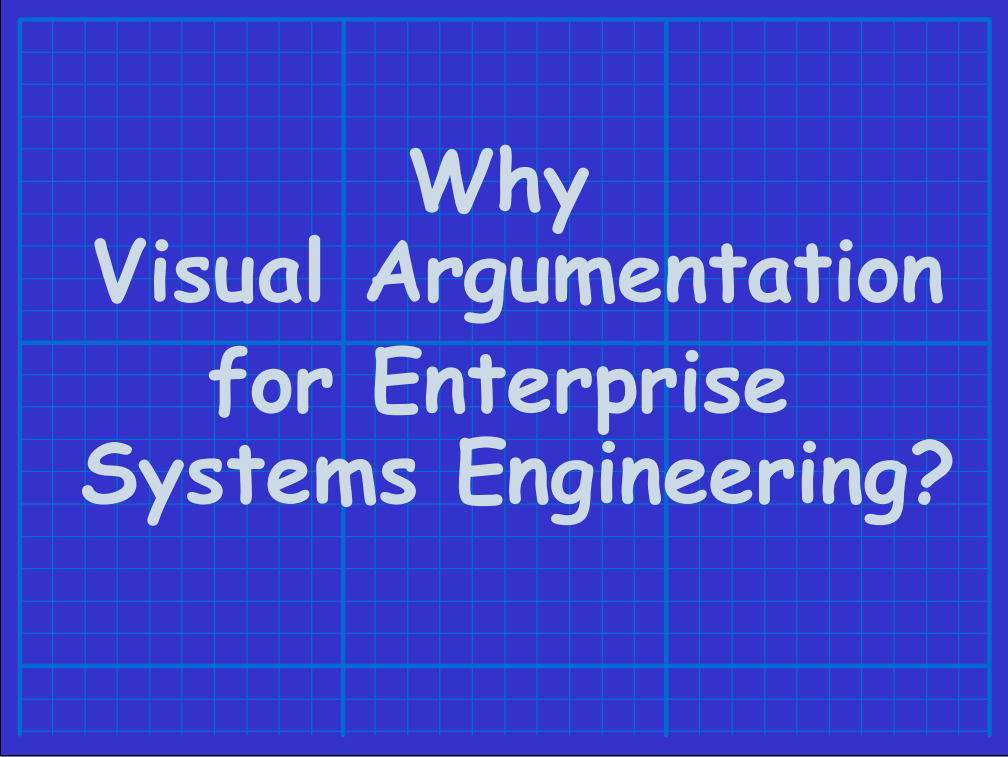
- Providing explanations
- Capturing deep knowledge and experience
- Mapping dialogues ...
  - Drawing on the wisdom of crowds
  - Building consensus
  - Respecting diverse viewpoints
- Representing rich traceability from abstract ideas and loosely defined concepts down to concrete design realisations





This example was drawn in a few minutes using the Enterprize Architect tool. It is a UML class-diagram in which I have created stereo-types for the various ELM types. Of course, almost any competent modelling tool is capable of extension in this way.

An interesting exercise for the reader would be to take some of their existing models and to see to what extent they could be useful improved by embedding this sort of visual argument. Does it help you with traceability? Capture of rationale? Explanation?



Why  
Visual Argumentation  
for Enterprise  
Systems Engineering?

## Wicked Problems

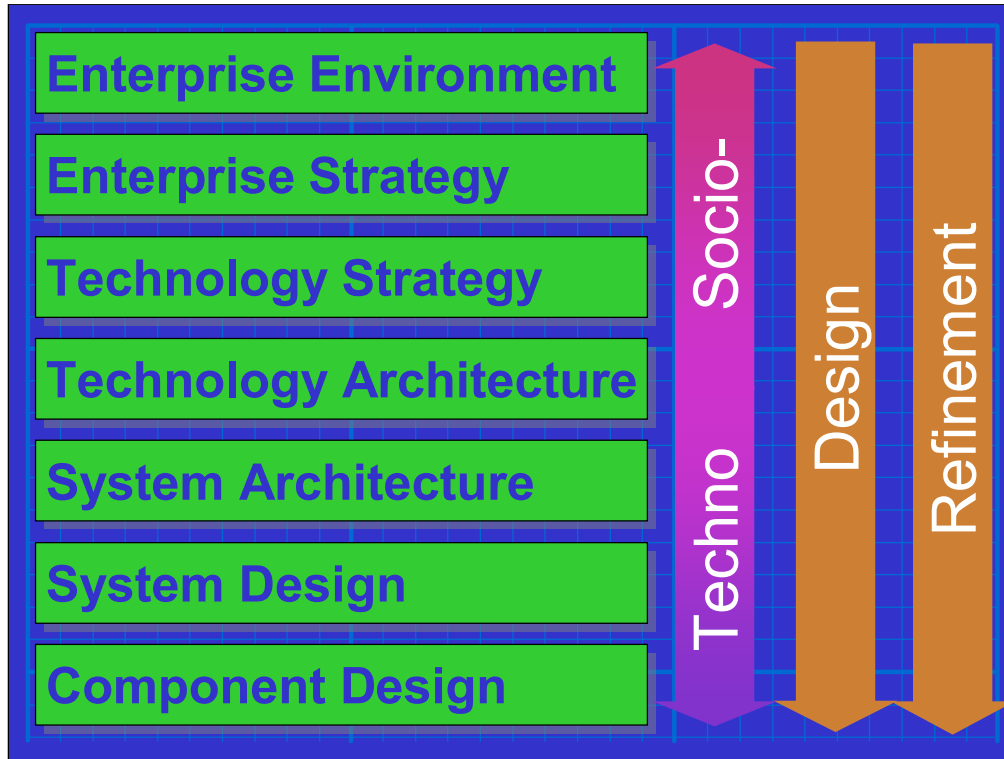
- Cannot be easily defined; stakeholders don't even agree on the problem
- Complex judgments about level of abstraction in which to define problem
- No clear stopping rules
- Often have a strong moral or political dimension - particularly for success
- Solutions are not 'right' / 'wrong' but 'better' worse
- Every solution is a 'one-shot' operation

[Horst Rittel](#) and [Melvin M. Webber](#) formally described the concept of wicked problems in a 1973

Conklin later generalized the concept of problem wickedness to areas other than planning and policy.

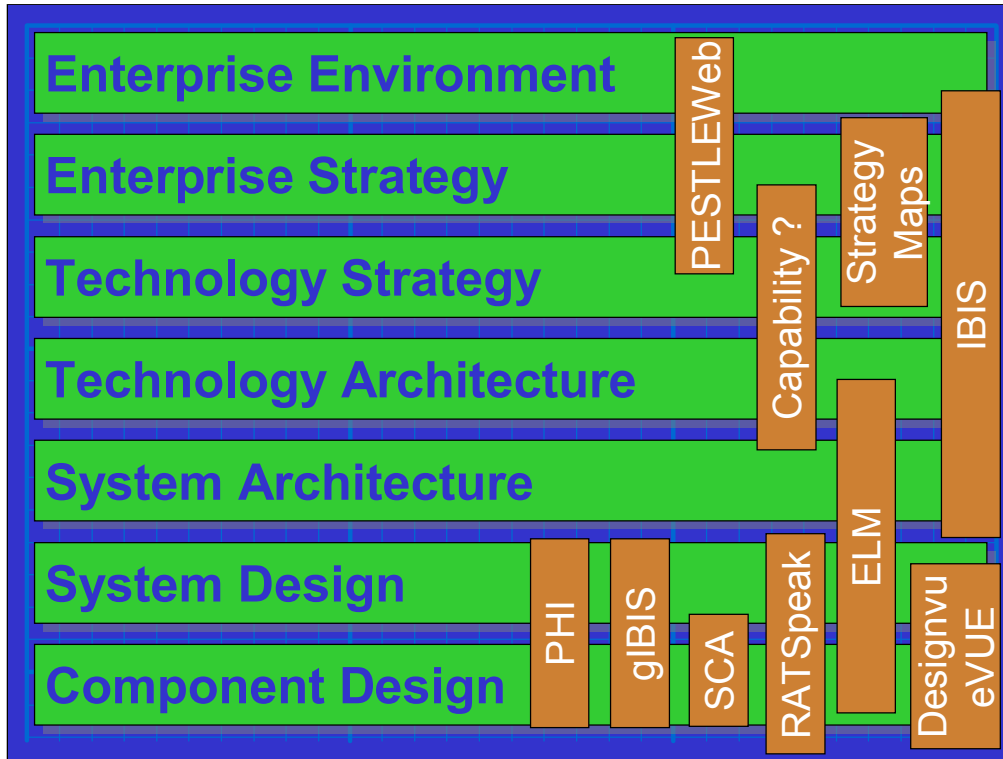
- "My recommendation for future design methodologies would be to emphasise investigations into the understanding of designing as an argumentative process"

Rittel, H. W.J. (1972) "Second Generation Design Methods" Interview in: Design Methods Group 5<sup>th</sup> Anniversary Report



My argument here is that Enterprise Systems Engineering operates across many layers of abstraction – from the business environment at the top, to low-level product design at the bottom. The design process must, as far as is practically possible, obey a refinement relation: Constraints introduced at the top level must be observed at the lower levels. If our Enterprise Environment is shouting for ‘Green Issues’, then we cannot implement System Designs that require components that are highly polluting, and so on.

In the presentation, there were some objections raised to my placement of “Socio-Techno” on this diagram. The objector’s arguments were essentially that “Socio-Techno” arguments pervade each level of this hierarchy, there is no bifurcated delineation between the least and most abstract. I acknowledge that argument, I can see it has validity. The point I was trying to express was that the focus shifts from the ‘Social’ to the ‘Technical’ as we move down this stack – and part of our role is to maintain coherence or ‘refinement’ between these two; Our technical solutions must be socially acceptable.



This diagram simply shows how the various argumentation languages span the levels of this hierarchy. I feel that there is a much of value that can be drawn on here by Systems Engineers. I also believe that there is much we can contribute. Some of the ground work has been done for us – but we have a specific and powerful need in our domain.

For reference, the ‘tools’ mentioned here are:

- Scenario-Claims Analysis (SCA) – Carrol and Rosson, 1996
- RATSpeak – Burge and Brown 2004
- Procedural Hierarchy of Issues (PHI) – McCall 1979b, McCall 1991
- gIBIS – Potts and Bruns 1988

## Visual Argumentation Works

- Refine from the very abstract to the very concrete
  - Enable coherence in design
- Draw on the knowledge and experience of very diverse groups
  - Do we know what we know? - Autoepistemic
- Develop consensus
- Engage in dialog
- Enable iteration ..
  - whilst reducing endless iteration

This was my bid for the prize for gratuitous use of the word “Autoepistemic”

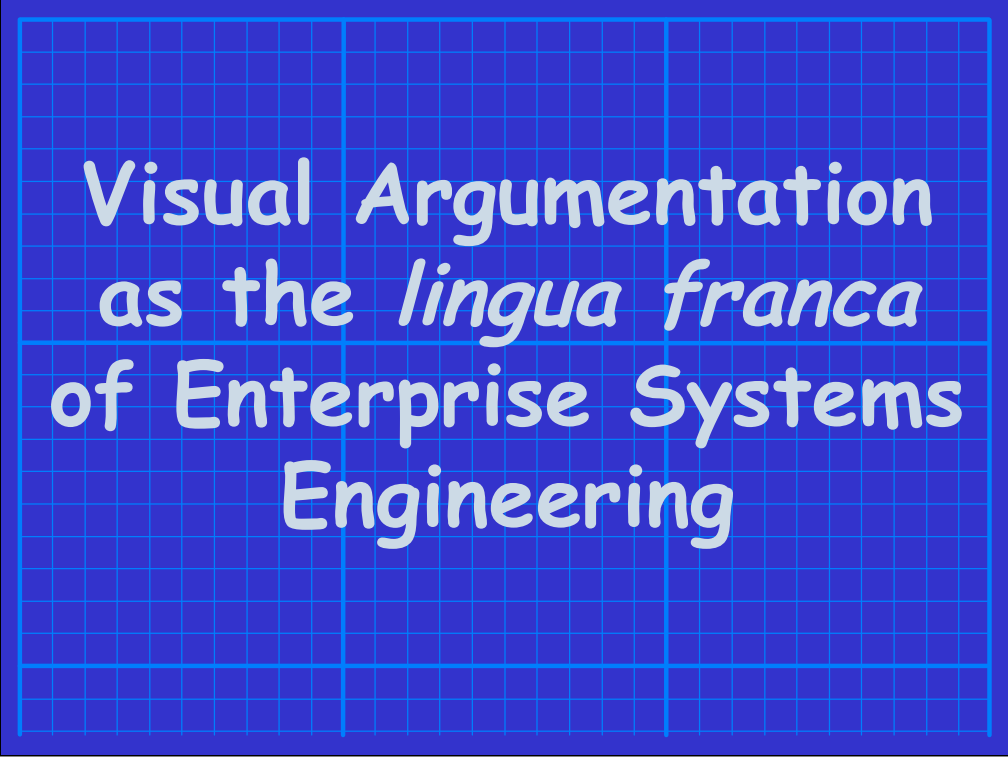
## The VA Thesis

- The case for Model Based Systems Engineering has been proven
  - 'Traditionally' that has meant fairly concrete representations of structure and behaviour
- But in the worlds of ESE and SoS we additionally need tools / models that operate at higher levels of abstraction
- We need tools for dialog, consensus, refinement, diversity of knowledge and belief and resolution...
- .. We need Visual Argumentation as the *lingua franca* of Enterprise Systems Engineering

**Thesis:** A proposition stated or put forward for consideration, especially one to be discussed and proved or to be maintained against objections:

I am hoping that the Systems Engineering community will engage with this thesis – but more importantly, and more straight-forwardly simply start to use the tools of visual argumentation to support their work.





# Visual Argumentation as the *lingua franca* of Enterprise Systems Engineering

A last word ... and teaser for next time....

In this presentation I have said a lot about 'representation' – drawing visual arguments and why that is a 'good thing'. I have not said very much about how I believe that all of this should inform practice. That is, what does this mean for how we do Systems Engineering. You might guess that my answer to this lies in the domain of discourse, dialog, argument and knowledge representation and may not be surprised in my view that our practice can usefully draw on techniques of community dialog include Action Learning....But that is for another time.

## References

- Baroni, P., Romano, M., Toni, F., Aurisicchio, M., Bertanza, G. (2013) "An Argumentation-Based Approach for Automatic Evaluation of Design Debates", Computational Logic in Multi-Agent Systems, Lecture Notes in Computer Science Volume 8143, 2013, pp 340-356
- Burge, J.E., Carroll, J.M., McCall, R. Mistrik, I. (2008) "Rationale-Based Software Engineering", Springer-Verlag : London
- Buckingham Shum, S.J., Selvin, A.M., Sierhuis, M., Conklin, J., Haley, C.B. and Nuseibeh, B. (2005) "Hypermedia Support for Argumentation-Based Rationale: 15 Years on from gIBIS and QOC", Technical Report KMI-05-18
- Collins, R.J. (2012) "Is there a Better Way to Model the Business Environment", MBA Report, Henley Business School. Available from: [http://users.ox.ac.uk/~kell0956/docs/pextreview\\_final.pdf](http://users.ox.ac.uk/~kell0956/docs/pextreview_final.pdf)
- Collins, R.J. (1997) "The Essential Logic Model : A Method for Documenting Design Rationale in Safety Critical Systems", ESREL '97, <http://users.ox.ac.uk/~kell0956/docs/elm.pdf>
- Kaplan, R.S. and Norton, D.P. (2004) "Strategy Maps: Converting Intangible Assets into Tangible Outcomes", Harvard Business School Press : Boston Massachusetts
- Kirschener, P.A., Buckingham Shum, S.J., Carr, C.S. (Eds) (2003) "Visualizing Argumentation: Software Tools for Collaborative and Education Sense-Making", Springer-Verlag : London
- Toulmin, S. (1958) "The Uses of Argument", Cambridge University Press : Cambridge
- Okada, A., Buckingham Shum, S., Sherborne, T. (Eds) (2008) "Knowledge Cartography: Software Tools and Mapping Techniques", Springer-Verlag, London.
- Zhu, L., and Gorton, I. (2007) "UML Profiles for Design Decisions and Non-Functional Requirements ", ICSEW '07: Proceedings of the 29th International Conference on Software Engineering Workshops, page 41. Washington, DC, USA, IEEE Computer Society