28th Annual INCOSE international symposium

Washington, DC, USA

July 7 - 12, 2018

Requirements Management or Assurance Management?

Keep the Data Integrated!

www.incose.org/symp2018

Authors

Richard Beasley joined Rolls-Royce in 1986 with a Physics Degree from Bristol University, and an MSc in Gas Turbine Engineering from Cranfield University. After working on Integration Aerodynamics, Safety, Reliability and Life Cycle Engineering, he became the Global Chief of Systems Engineering and in 2011 was made a Rolls-Royce Associate Fellow in Systems Engineering. He was part of the BKCASE SEBoK author team, and is the Immediate Past-President of the UK INCOSE Chapter. He is a Chartered Engineer, Fellow of the Royal Aeronautical Society, INCOSE ESEP, and a Visiting Fellow to the Systems Centre at Bristol University.

Iain Cardow joined Rolls-Royce in 1998 as a Development Engineer after doing a Graduate Traineeship with James Howden, an industrial Fan and Heater manufacturer. He has a degree in Aeronautical Engineering from the University of Glasgow. The majority of his career within Rolls-Royce has been in the Development Engineer area of Defence products, looking after the verification and testing of a range of engine types, before moving to be a Project Systems Engineer working across Defence and Civil products. He is a Chartered Engineer with the Royal Aerospace Society, and a trained CEng assessor for the Society. He is a member of INCOSE UK, and represents Rolls-Royce on the INCOSE UK Corporate Advisory Board.

Andrew Pickard joined Rolls-Royce in 1977 after completing a Ph.D. at Cambridge University in Fatigue and Fracture of Metals and Alloys. He is a Rolls-Royce Associate Fellow in System Engineering, a Fellow of the Institute of Materials, Minerals and Mining, a Chartered Engineer and a member of SAE International and of INCOSE. He is Chair of the SAE Aerospace Council, represents Rolls-Royce on the INCOSE Corporate Advisory Board and is Chief of Staff for INCOSE.

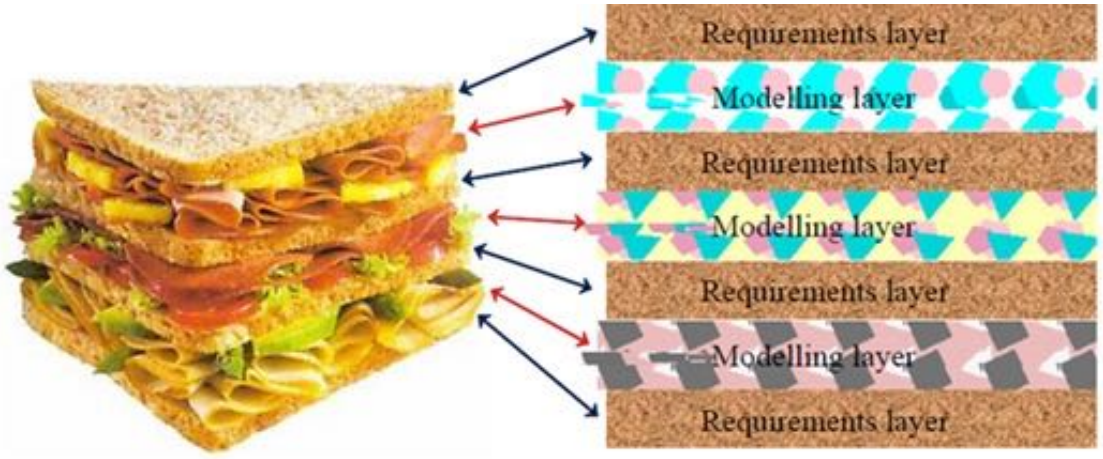
Contents

- Introduction
- Configuration vs. Information
- What is REDV?
- Processes that create/control REDV information
- Communicating between system levels
- Complexity and coupling
- Link to Verification
- The Assurance V
- Conclusions

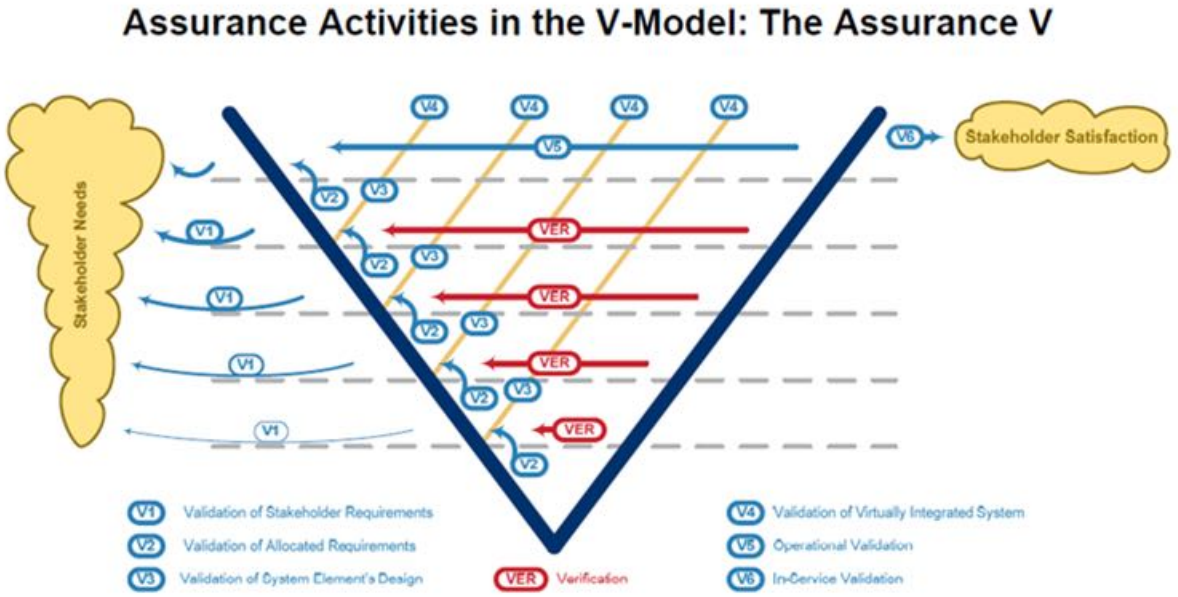
Introduction

- What is the Problem?
  - Requirements, Requirements, Requirements!
  - What about the rest of the related information produced during the Product Development/Systems Engineering activities?
- Why is this related information important?
  - Evidence models confirm that the proposed solution definition is capable of meeting the requirements
  - Definition allows the system to be broken down into a set of interacting sub-systems together with the associated needs for the behavior of those sub-systems
  - Verification confirms when the system has been instantiated that it meets requirements at all levels in the system hierarchy

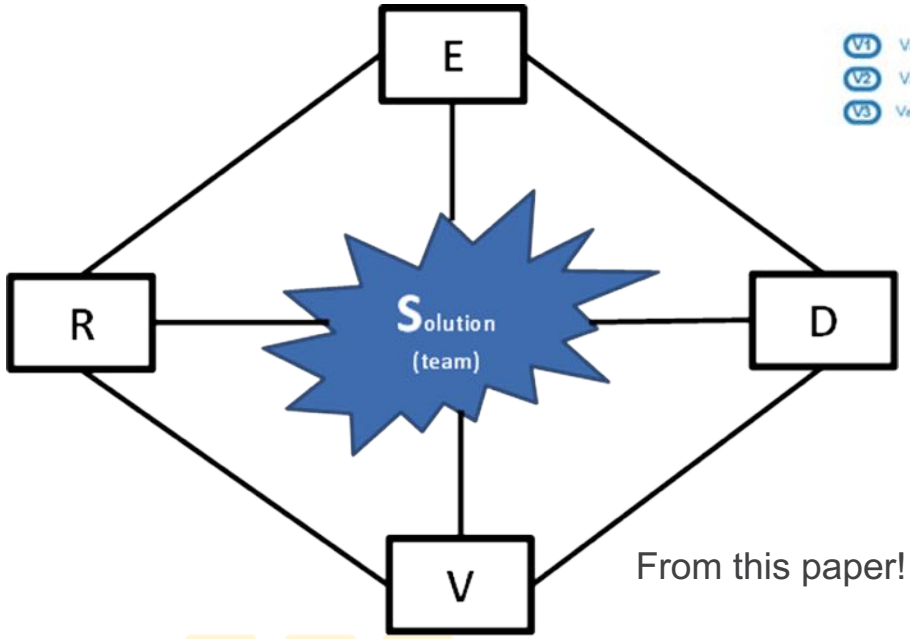
Dramatis Personae



From Dick and Chard 2004



From Scheithauer and Forsberg 2013

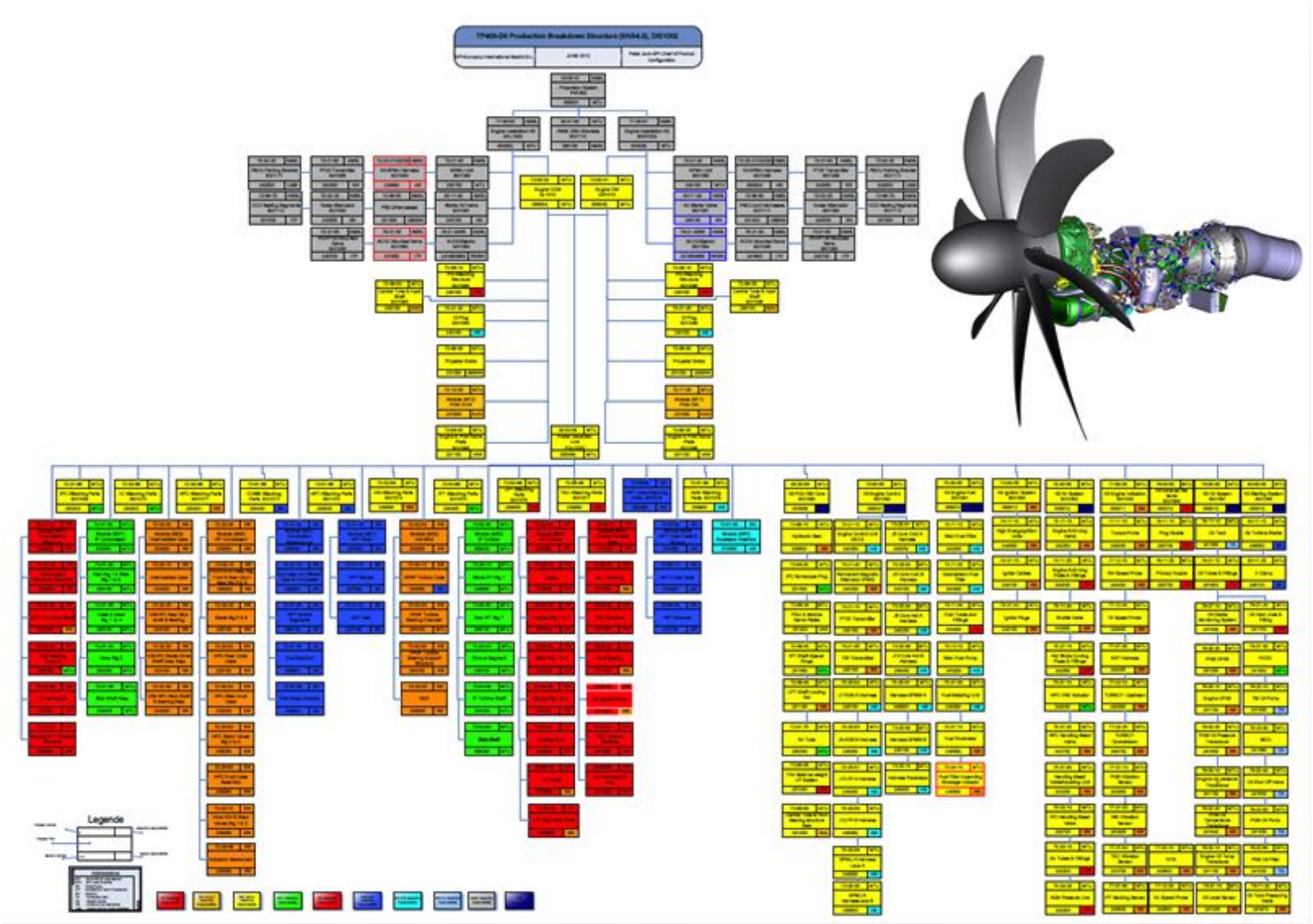


From this paper!

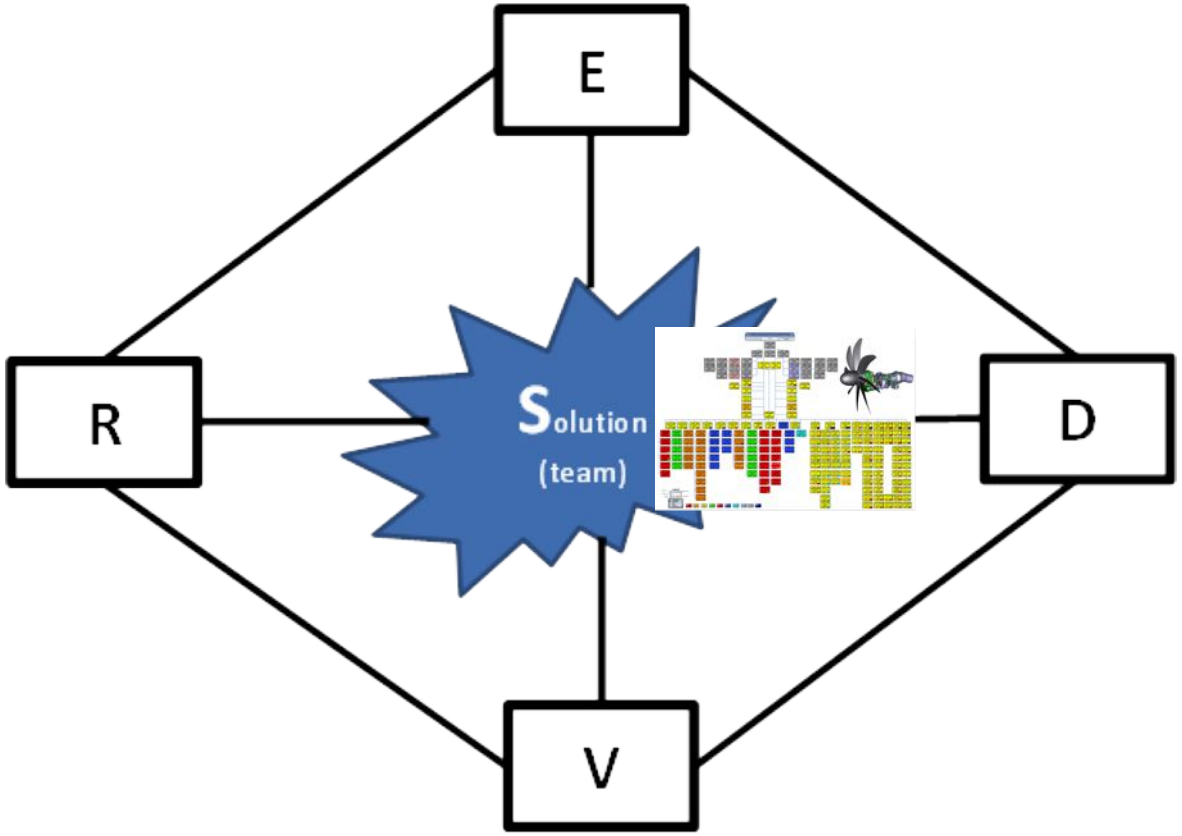
Configuration vs. Information

- Configuration Management:
  - Output of the Engineering Process
  - Definition of the Product, together with how to make, test, operate, inspect, maintain, repair and dispose of it.
  - Aimed at ensuring that every instantiation of the system behaves the same as the version that was verified against the system requirements
  - Very strict change control; change is “bad”
- Information Management
  - Change is inevitable, especially early in the product development lifecycle, where uncertainty is high

Illustrative Product Configuration Structure



Proposed Information Structure



R = Requirements

E = Evidence

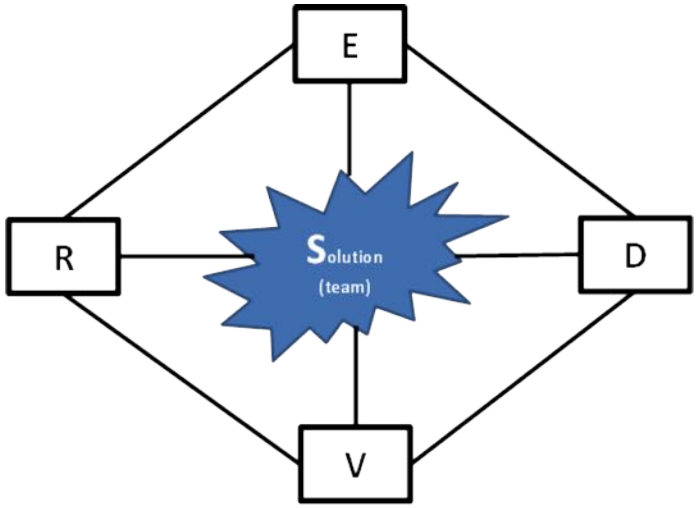
D = Definition

V = Verification

Standards and Processes

- What level of Configuration Management should be applied to Requirements, Evidence, Definition and Verification information?
  - Early in the system development lifecycle, there will be significant uncertainties in stakeholder needs and resulting derived requirements
  - When should the information be placed under formal configuration control?
  - What level of configuration control?
- Example – Aviation Safety Critical Systems with embedded Software:
  - Standards are RTCA DO-178C, SAE ARP 4754A
  - Both require assessment of Development Assurance Level (DAL)
  - CC1 and SC1 – control at item level
  - CC2 and SC2 – control at document level
- We have two linked processes in Rolls-Royce to manage the Configuration and REDV information

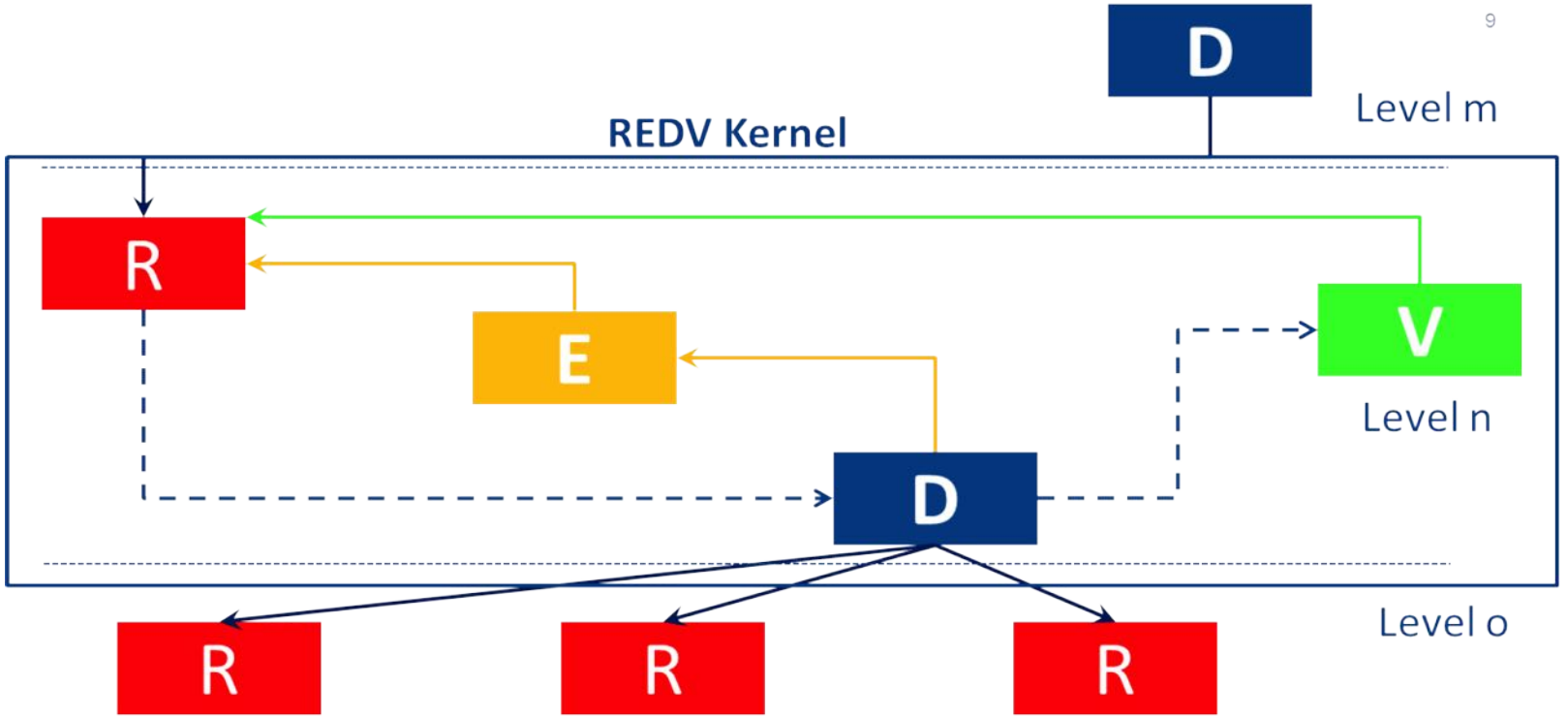
What is REDV?



The information types are:

- **R is Requirements** – the definition of the constraints and requirements for the System of interest.
- **E is Evidence** – the evidence that the solution proposed is expected to meet the requirements. This will include a gap analysis of attributes, comparing the requirement and evidence information.
- **D is Definition** – specifically the definition of derived needs that must be met (from another system / system element) in order for the solution to be realized. This includes the definition of the sub-elements that make up the system of interest, and the derived needs to flow down to these sub-elements below.
- **V is Verification** – specifically the plan and then the record of the execution of the plan, the verification information produced (and source) and status of whether each requirement is verified as met or not.

The REDV Kernel – Connecting the Information



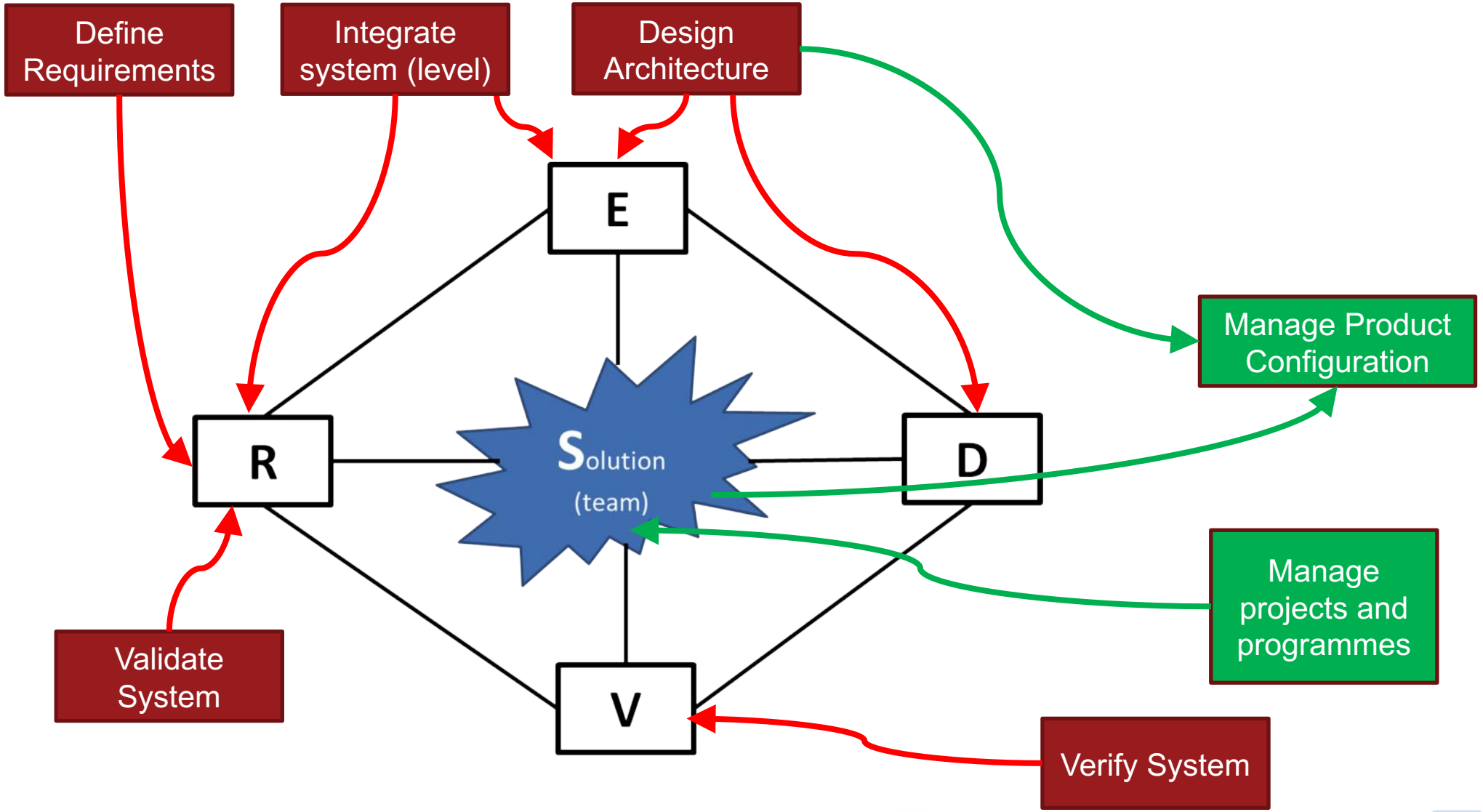
The structure connects:

1)R-E-D-V data for a layer in the solution

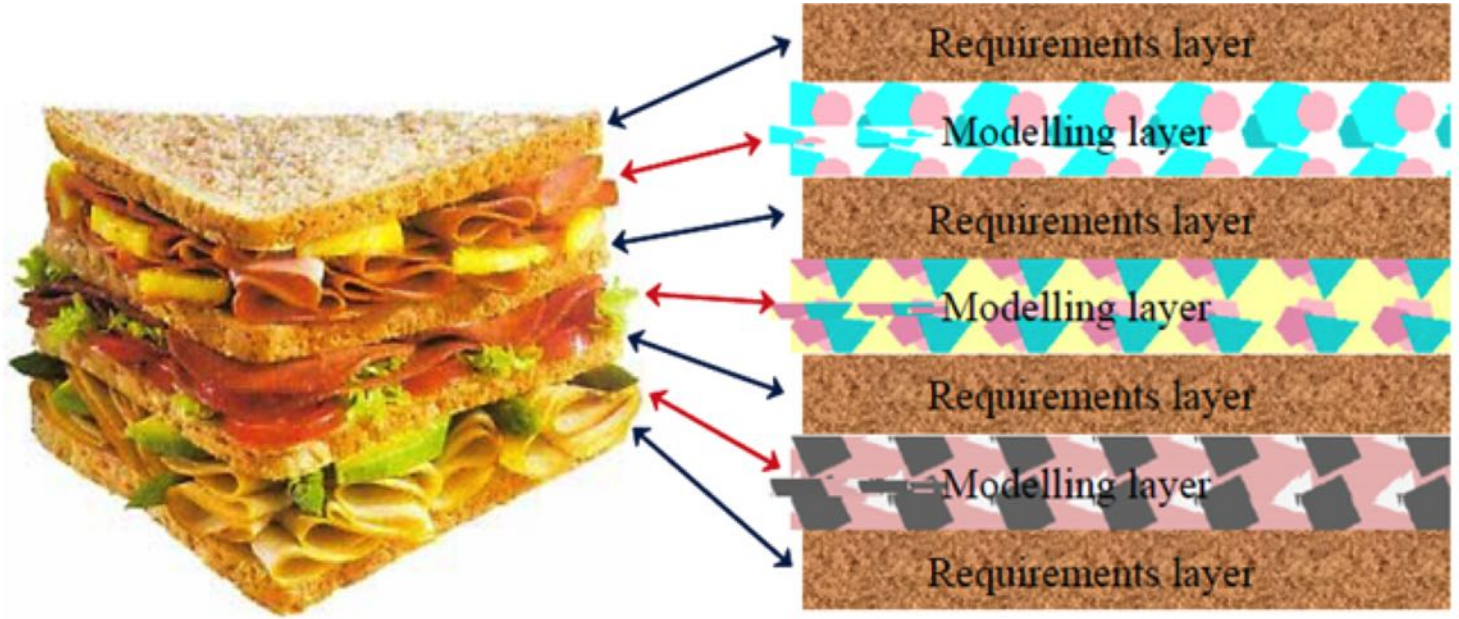
2)The layers, clearly and consistently

KEY:-
Drives
Design Rationale Evidence
Verifies
Requirement Flow

Process Landscape



The Requirements Sandwich

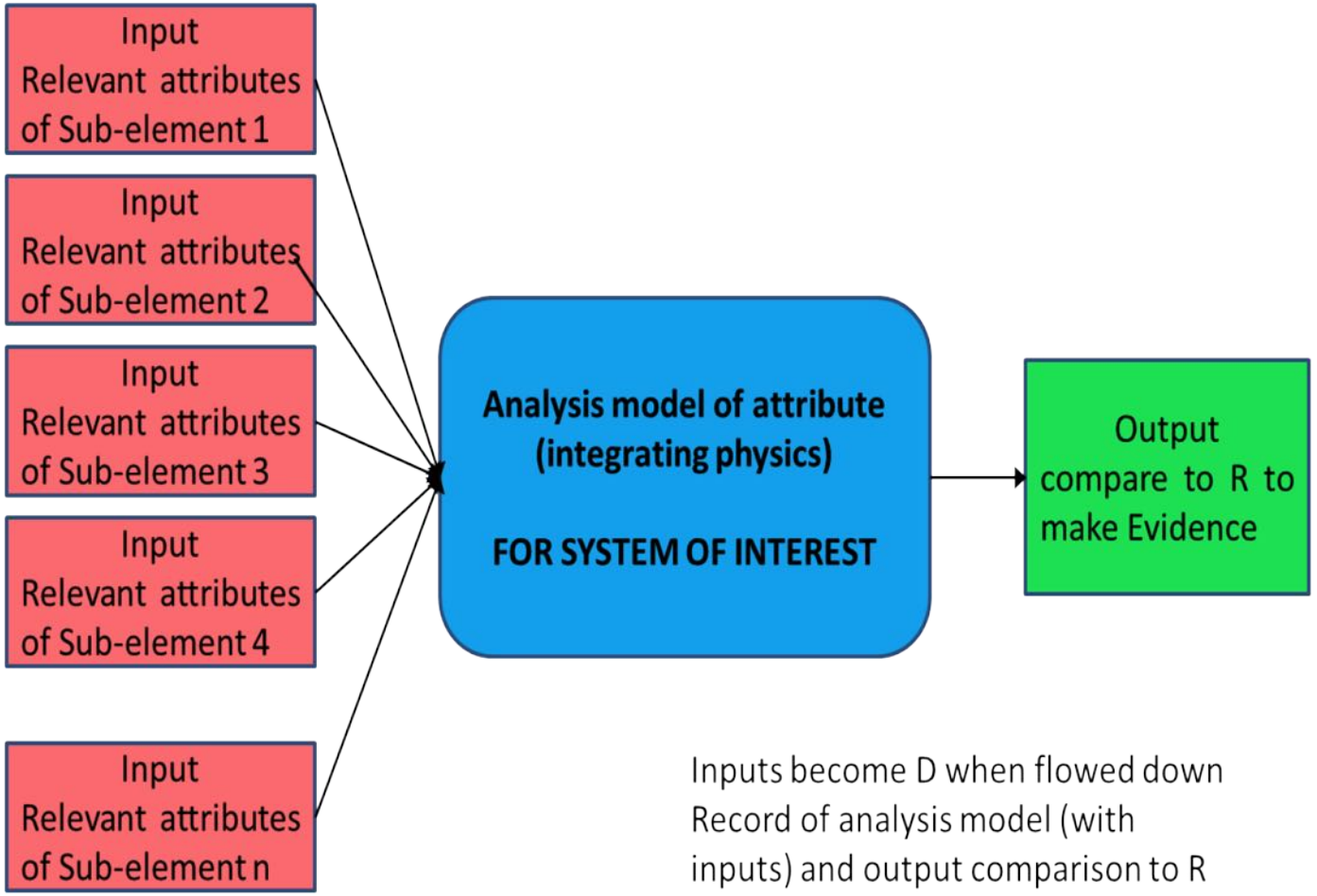


The requirements for lower level elements are derived from the solution model of the system of interest.
The left hand side of the V is a dog-leg.
(From Dick and Chard 2004)

Solutions are Stakeholders!

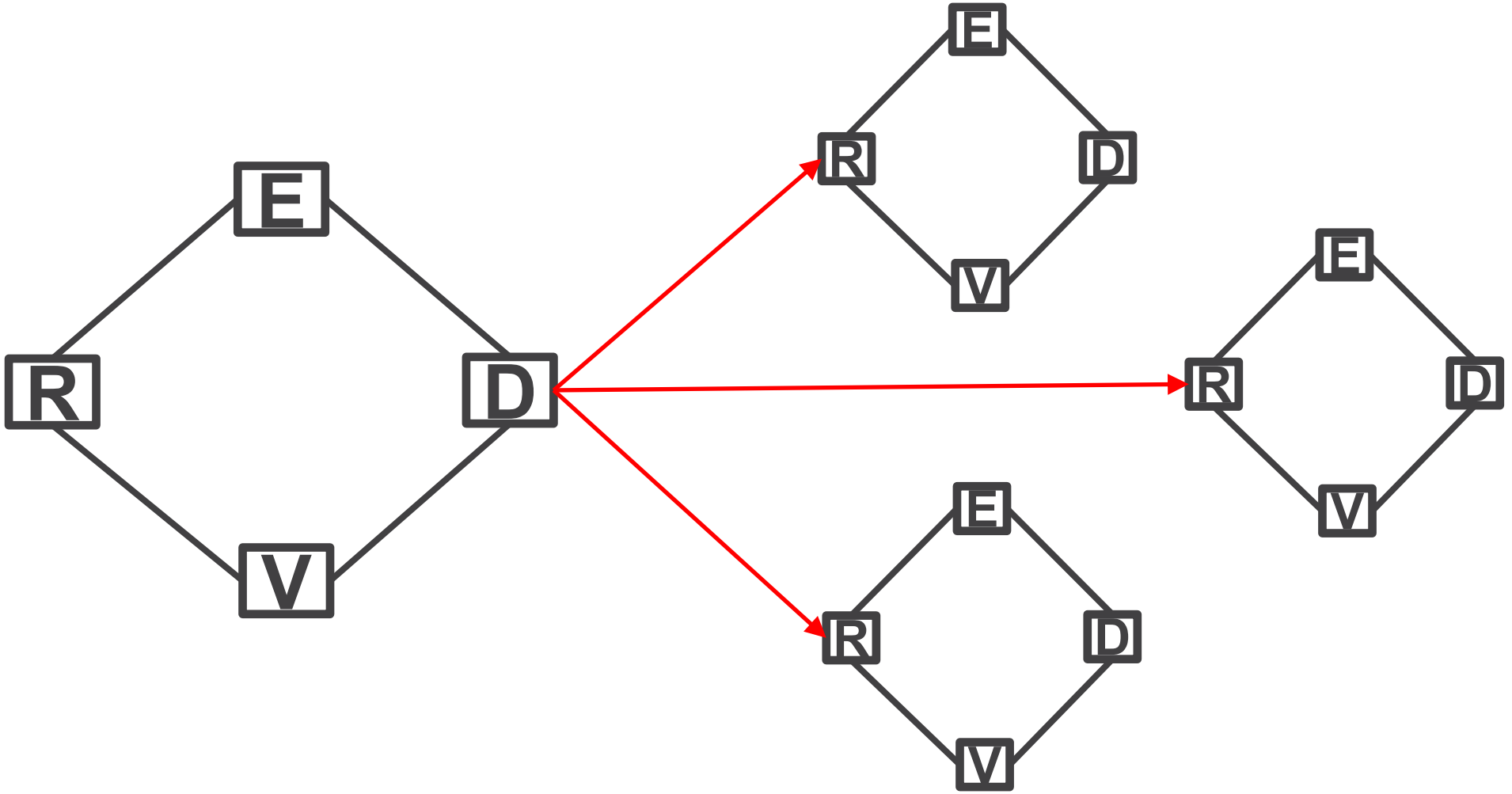
- Stakeholders express their needs for the behavior of the Solution System
- These needs will often be poorly structured, un-prioritized and conflicting
- Requirements Analysis is the task of understanding these needs, de-conflicting them, prioritizing them and structuring the agreed needs in the form of “good requirements”
- This applies at all levels in the system hierarchy
- The Definition of each level of system creates the architecture of sub-systems that when integrated become this system, and places derived needs on these sub-systems
- Requirements Analysis is performed at all levels of the hierarchy to understand needs that come from many sources, including the system level above
- Ultimately the solutions at all levels in the system hierarchy are stakeholders in their sub-systems via these derived needs

Generic Model of an Attribute



Inputs become D when flowed down
Record of analysis model (with inputs) and output comparison to R becomes E

Connecting REDV Kernels



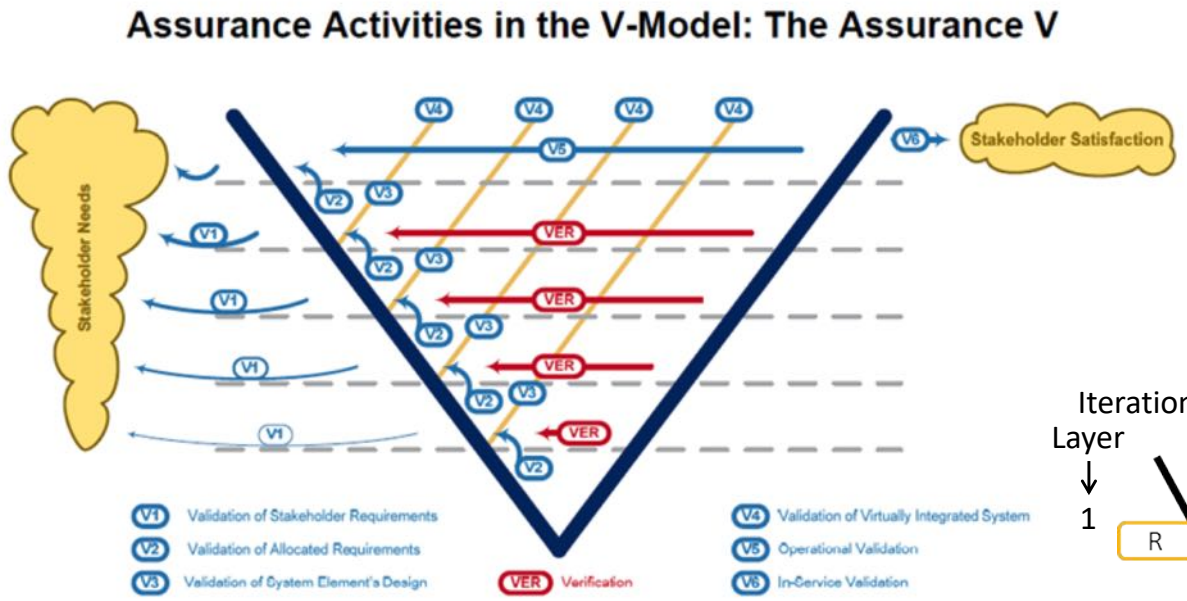
Complexity and Coupling

- Architectural guidance suggests that partitioning of a system into sub-systems should minimize coupling between these sub-systems
- Unfortunately, Gas Turbine engines are inherently highly coupled, so this strategy will not work
- Consider the cooling air system within a Gas Turbine:
  - Source is in the Compression sub-system
  - The air is used to cool the Turbine sub-system
  - Pressure losses are influenced by both of these systems, the air path and the seals between different cavities in the engine
  - The turbine blades and vanes rely on a film of cooling air that is ejected from holes in the components
  - This doesn't work very well if the film cooling holes suck instead of blow!
- In integration, the requirements from the coupling at the level below need to be integrated into the D of the system of interest, so that this system is informed and can control the implied interfaces.
- This explains why architectural advice is to have loosely coupled sub-systems!

Linking to Verification

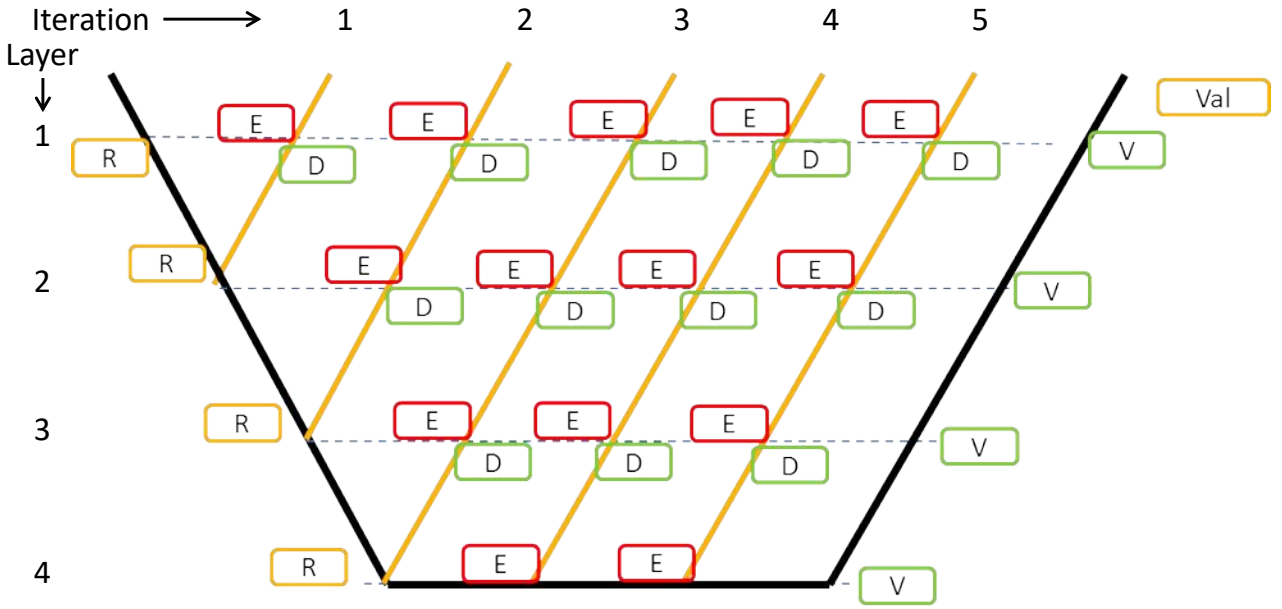
- One of the characteristics of a good requirement is that it should be verifiable
- Therefore verification activities start the moment requirements are being created!
- This includes consideration of how the requirement will be verified, and the construction of a verification plan
- Requirements may be verified by analysis as well as by testing
- So what is the difference between “Evidence” and “Verification Evidence”?
- Validation!
  - The initial models used in “Evidence” will include assumptions, for instance about boundary conditions and environment in which an element of the system is to perform
    - Think of the Cooling Air System in a Gas Turbine!
  - For gas turbine engines, the Development Engine Test program will include testing of heavily instrumented engines to validate these boundary conditions and environments
  - These tests validate the Evidence models and allow them to be used as “Verification Evidence” models

The Assurance V

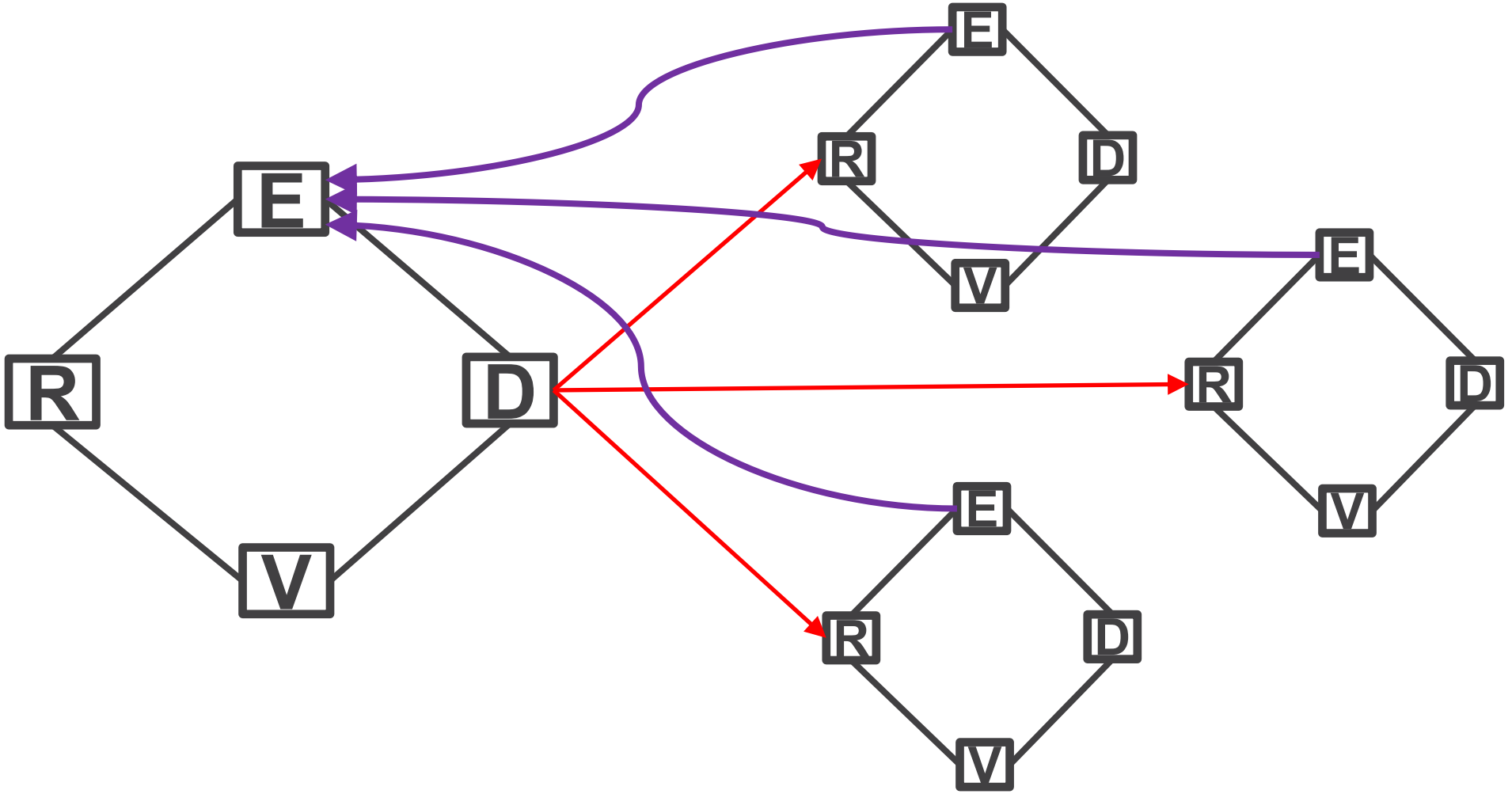


From Scheithauer and Forsberg 2013

Note: The REDV model fits exactly onto this, only we have widened the base to allow us to show design iterations at every level



Looking for Emergence



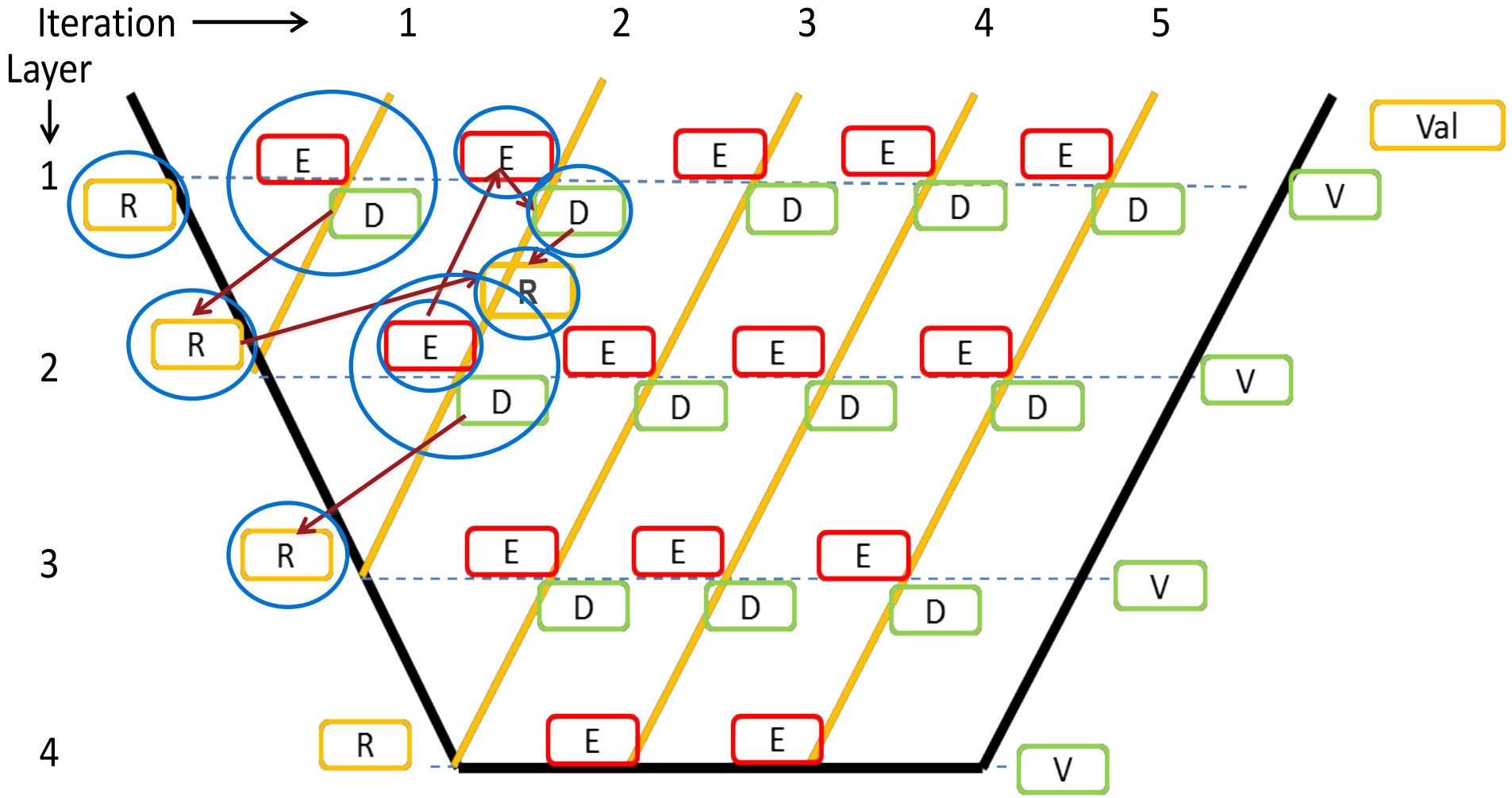
Iterations

Scenarios considered in the paper:

1. All sub-elements meet their requirements and when integrated the models confirm that the whole system still meets requirements.
2. There is some emergence, so despite all Evidence meeting Requirements the updated (more detailed) system level model no longer shows compliance.
3. One of the sub-elements doesn't meet its requirements (all others do), but its current design integrated acceptably into the whole engine model.
4. One of the sub-elements doesn't meet its requirements – and others do or have exceeded and the integration shows that the sub-elements together meet the Requirements of the System of Interest, as defined in a new Evidence model. Therefore the Definition and sub-element Requirements need to be updated.
5. One (or more) of the sub-elements does not meet its requirements. After integration the model shows non-compliance to the Requirements of the System of Interest, but after review it is considered that the derived requirements are achievable by the sub-elements and they should "try harder" (or again) in the next iteration with unchanged requirements
6. One (or more) of the sub-elements does not meet its requirements. After integration the model shows non-compliance to the Requirements of the System of Interest. However, review shows that (with the more detailed design analysis) it is not feasible for the sub-elements to do "better" and so it is concluded that the Requirements of the System of Interest are not achievable. Thus the system team needs to go back and review / negotiate requirements with the stakeholders, recording the agreement in updated SoI Requirements

- The REDV information matures as work progresses, therefore integration, re-evaluation and update (iteration) is vital and drives the way REDV information must be managed.
- Continual checking of E against R gives progressive assurance that we have a good solution

Using the REDV and Assurance Models



Conclusions

- The Requirements, Evidence, Definition and Verification (REDV) information for every element at every level in a system should be managed in a joined up information, or assurance, database – one source of the truth.
- REDV information management should be re-named “Assurance Information Management”
- Don’t focus on Requirements Management – all elements in Assurance Management are important
- Assurance Management and Emergent Properties – the need for repeated Iteration and Integration
- Configuration Management of the Solution and of the Assurance information are similar but subtly different
- Reviewer comment:
  - “The paper proposed a approach with deployment of systems engineering, but it proposes something that seems to be commonplace throughout the industry”.
- Do you agree, or is this new to you?

28th Annual INCOSE international symposium

Washington, DC, USA

July 7 - 12, 2018

www.incose.org/symp2018