

Business Case for Behavior Engineering

**Geoff Dromey
Software Quality Institute
ARC Centre for Complex Systems**



Plato's Advice

*“The beginning
is the most important part
of the work”*

Applies very much to Software & Systems Engineering

What is Behavior Engineering (bE)

- **Is an integrated discipline that supports the systems and software engineering of large-scale, dependable software-intensive systems.**
- **Behavior Engineering's strength lies in the innovative way it squarely addresses the problems of scale, complexity, and imperfect knowledge associated with the large set of requirements needed to guide the development of challenging integrated systems.**

What is Behavior Engineering (bE)

- **Behavior Engineering employs a graphical Behavior Modeling Language (BML) that is used to construct complete behavioral, compositional and structural *integrated views* from natural language descriptions of a large set of requirements.**
- **Integration of requirements discovers inadequacies in a set of requirements, in the same way putting together the pieces of a jigsaw puzzle allows us to discover some pieces are missing and/or do not fit.**

In One Slide

- To manage risk – manage complexity
- To understand – integrate requirements
- To eradicate confusion – resolve vocabulary
- To reduce costs – detect defects very early
- To build the right system – preserve intention
- To be productive – work by construction
- To manage change – match change currency

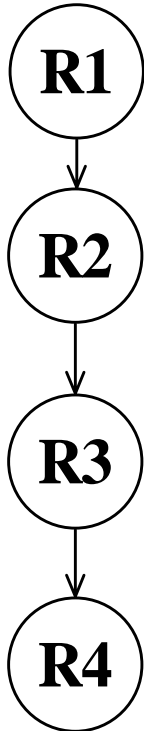
**To Manage Risk
- Manage Complexity**

The Problem of Complexity

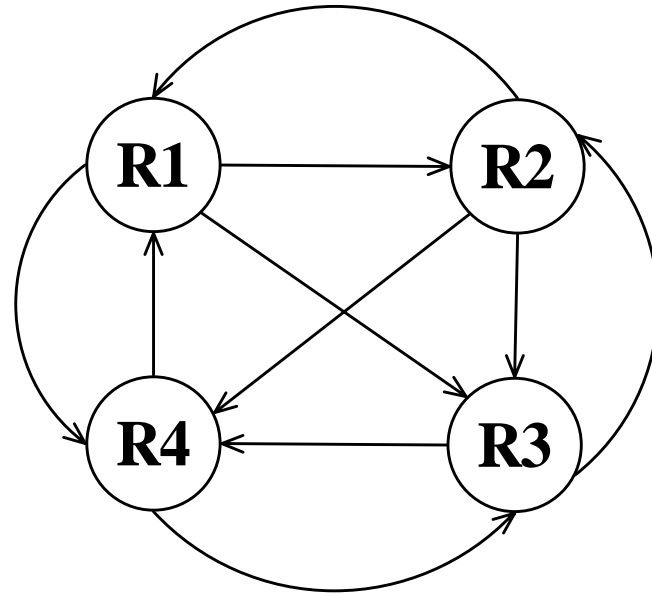
- **Existing modeling methods do not have a clear and effective means for managing requirements complexity**
- **Initial complexity arises because there are hundreds or thousands of requirements that need to be formalized**
- **Requirements interact – the more the interaction the greater the complexity**
- **Interaction → short-term memory overflow**

Requirements Interaction

LESS ← **Interaction** → **MORE**



LESS
Complexity



MORE
Complexity

Managing Complexity – bE Approach

- **Individual requirements provide the initial and the natural “decomposition” of a problem – accept it, exploit it**
- **Initially focus on and formalize the detail in each requirement one at a time – avoids risk of short-term memory overflow and uncovers “localized” defects**
- **Identify precondition for each requirement executing its behavior → interaction**

To Understand
- Integrate Requirements

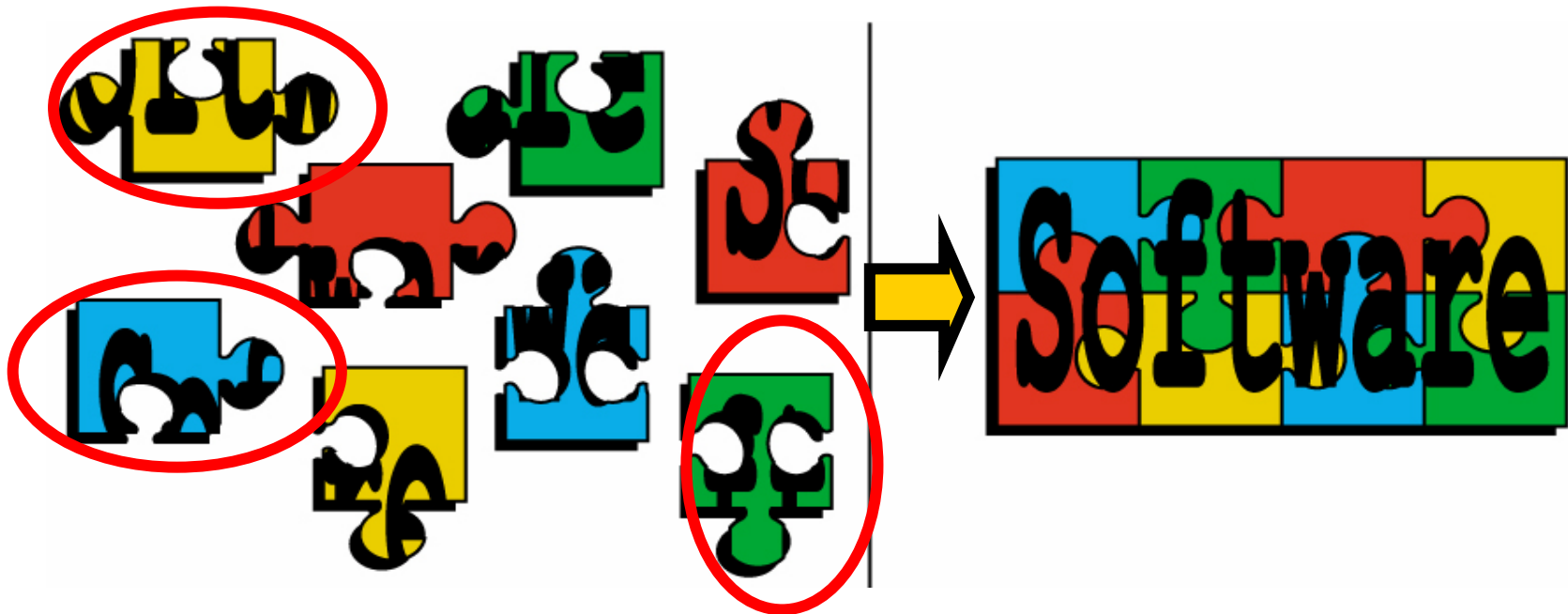
Increasing Understanding – bE

- **To understand something you must be able to see it as a “whole”**
- **A set of requirements imply a system that exhibits an integrated set of behaviors**
- **Existing methods – no simple way of seeing the ‘whole’, or formalizing and visualizing very large sets of requirements - therefore there is a barrier to increasing our understanding.**

Increasing Understanding – bE

- **Integrating the requirements for a system allows us to see the integrated behavior that they imply – we see the whole.**
- **Our method is the only one currently available that provides a way of seeing the “whole”, and formalizing and visualizing very large sets of requirements.**

Constructing an Integrated View



- Only see there are missing pieces when integrate pieces
- Only see some pieces don't fit when integrate pieces
- Order of placing is not important BUT position where placed is
- Information about “f” is spread across THREE pieces

Similar IDEAS apply with requirements

Integrated View - Advantages

- **Once requirements have been formalized and integrated we can perform a variety of operations on the integrated view:**
 - Simulation**
 - Model-checking**
 - Failure modes & effects analysis (FMEA)**
 - Refinement to specification/design**
 - Code generation**
 - Inspection – automatic and manual**
 - Conversion for use by other tools**

**To Eradicate Confusion
- Resolve Vocabulary**

Eradicate Confusion – bE Approach

- **For large systems – subsets of requirements written by different people**
- **Different people – different vocabularies and different understandings → confusion**
- **Complete system vocabulary – constructed one requirement at a time – integrates, localizes, related info. – see inconsistencies**
- **Integrated view exposes aliases and other vocabulary problems by juxtaposition – other methods – don't do systematically**

To Reduce Costs
- Detect Defects Early,
Systematically

Early Detection of Defects – bE

- **Requirements integration – single most important way of finding requirements problems – same idea as for jigsaw puzzles**
- **Whenever requirements don't integrate you have found a requirements defect**
- **The integrated view of requirements – puts each requirement in its execution context – makes many types of defects easier to see**
- **Other methods – lack potent, high-yield, early defect detection strategies**

Early Detection of Defects – Results

- **Behavior Engineering trials on a series of large projects with one large company consistently found 10 – 15% of requirements analyzed contained significant defects not found by their review processes.**
- **Company is a CMMi company with mature processes.**
- **Similar statistics on projects for other large companies and organizations**

Early Detection of Defects – Results

The following table contains statistics on recent projects where we have applied the method.

	Recent Project- RP		Last 3 Projects – (RP excluded)		
			Total	Ave	
Number of Pages Analysed:	101pages		265	88.33pages	
Number of Requirements Analysed :	920requirements		3142	1047.33requirements	
Major Defects Only	128defects		412	137.33defects	
Incompleteness	73	57.03%	260	86.67	63.11%
Inconsistency	3	2.34%	30	10.00	7.28%
Ambiguity	19	14.84%	93	31.00	22.57%
Redundancy	31	24.22%	13	4.33	3.16%
Inaccuracy	2	1.56%	16	5.33	3.88%
Number of Queries:	7queries		98	32.67queries	
Effort (Includes reporting, analysis, modeling)	94Person-hours		325	108.33Person-hours	

What the results show is that the Behavior Engineering method consistently finds 130 major defects per 1000 of requirements after normal reviews and correction have been carried out. In addition the integrated work products constructed to detect defects can subsequently be corrected and refined to create an executable design.

Early Detection of Defects – Benefits

- **Detection and correction of these defects early in the life-cycle costs one tenth what it would cost to find and fix the same defects at testing (Hughes Aircraft stats.)**
- **This translates to substantial savings given reworking of requirements defects accounts for as much as 40-50% of a software-intensive project's costs (Capers-Jones).**

**To Build the Right System
- Preserve Intention**

Building the Right System Right

- **Building a system out of its requirements maximizes the chances of building the right system – easy validation – direct traceability to original text requirements.**
- **Rigorous translation of each requirement that preserves the original vocabulary maximizes the chances of preserving original intention – and therefore building the right system.**
- **Existing methods weak on both these counts.**

**To Manage Change
- Match Change Currency**

To Manage Change – match currency

- **The prime currency of change is individual requirements.**
- **By formalizing and integrating individual requirements we achieve direct traceability to original requirements.**
- **Changes represented as formalized individual requirements are of the same currency – as with jigsaw puzzles pieces, changes may be implemented by traceably integrating the “changed” requirements.**

**To Be Productive
- Work Constructively**

To Be Productive – Work Constructive

- The work product(s) for every step are built out of the work-products of the previous step.
- The method supports people working in parallel then combining what they produce – constructive development. With tools “combining” is relatively cheap.
- Even defect finding is part of a constructive process – both requirements translation and integration are constructive.

To Be Productive – Work Constructive

- **Use of a collaborative editor combined with strict vocabulary control enforced by the composition tree allow people to translate requirements in parallel then combine their work.**
- **Example - If you have 1000 requirements to translate and 10 people you can give them 100 each to translate - then combine their work – and get the job completed in roughly a tenth of the time of one person doing the job.**

Conclusions

Behavior Engineering - bE

- **Gets complexity under control early**
- **Reduces project risk**
- **More effective early defect detection**
- **Better chance of building right system**
- **Decreased development schedule**
- **Less rework, less wastage, reduced costs**
- **Makes productive dealing with customers**
- **Easier to handle requirements change**
- **Easier to manage/deploy project team**

*“If you keep doing what you have
always done, you will keep getting
what you have always got”.*

– W. Edwards Deming.