We Cannot Master Complexity with a Complicated Process

Colin Hood Systems Engineering Ltd
4 Summerhouse Road
Northampton NN3 6BJ
Great Britain

info@colinhood-se.com

Aims

- To show that many processes are becoming not fit-for-purpose in an increasingly connected environment.
- To discuss directions for improving our readiness to tackle today's challenges

Explanation of Terms

Here is a simple explanation, inspired by Dr. Martin Grundel:

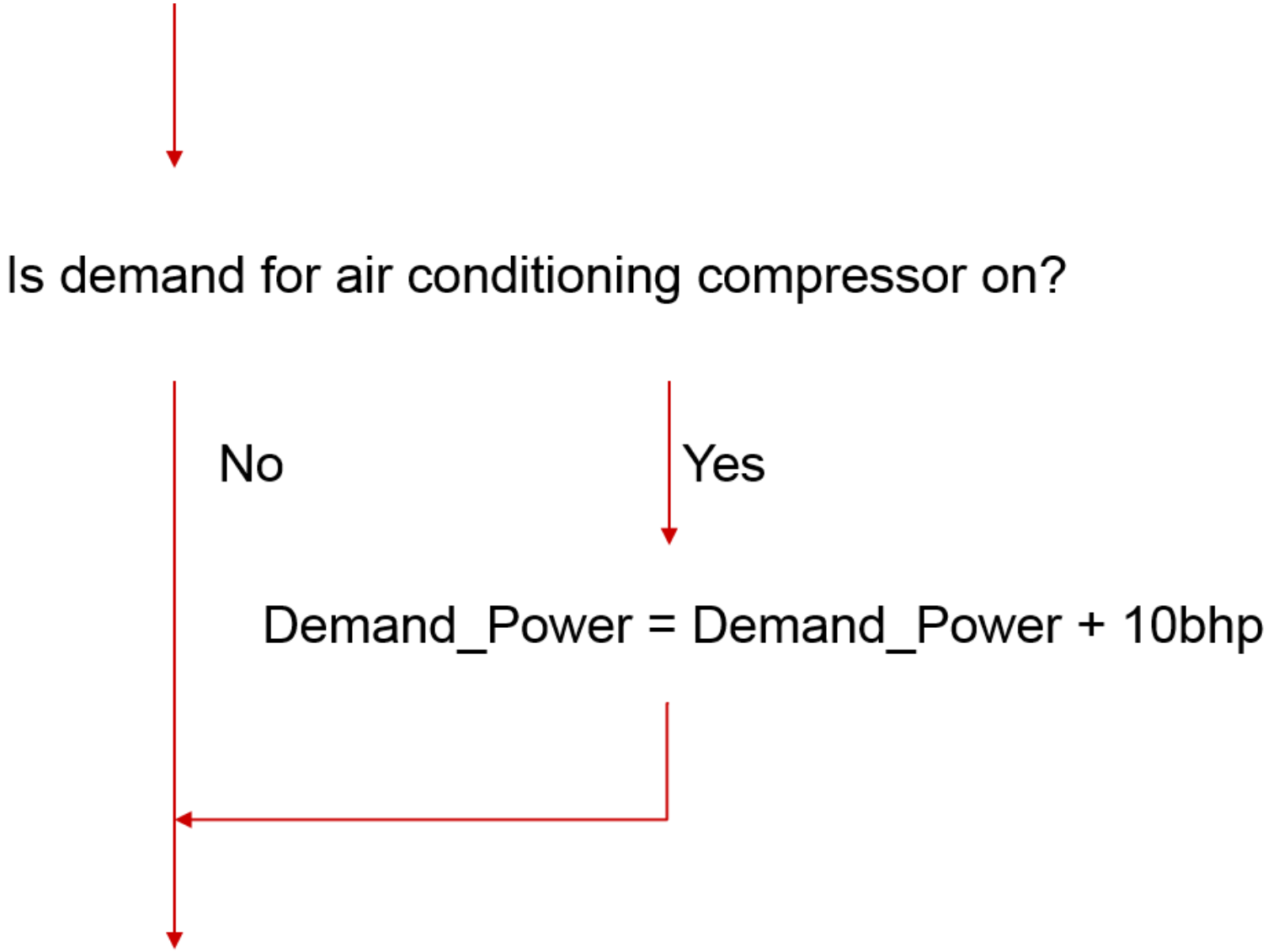
- Complex: We cannot know everything. E.g. predicting the trajectory of a leaf falling
- Complicated: The stimuli and reactions are known, even though there may be many

and, taken from “A Complexity Primer for Systems Engineers”

- No easy, agreed-upon definition
  We often call something complex when we cannot fully understand its structure or behavior: it is uncertain, unpredictable, complicated, or just plain difficult
  (see Sillitto (2009): Subjective Complexity)
- Hallmarks of complexity include interdependence, nonlinearities, adaptation and innovation.
- The opposite of “complex” is “decomposable”, not “simple” (Alex Ryan)

Examples

- Automatic Parking Brake
- Climate Control
- Reverse Thrust
- Education
- What we Need is More Process Documentation



Examples

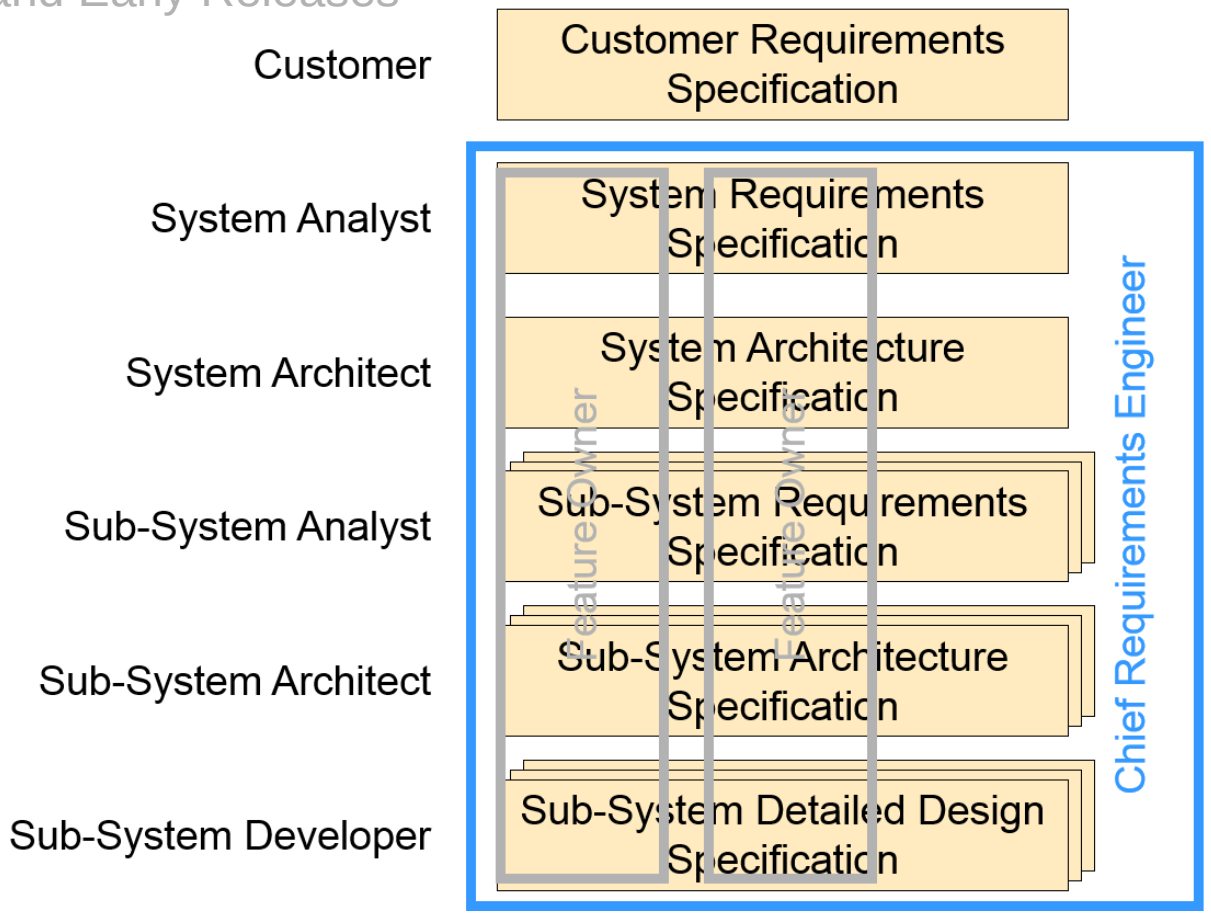
- Automatic Parking Brake
- Climate Control
- Reverse Thrust
- Education
- What we Need is More Process Documentation

What has helped deal with complexity

- Feature Owner and Chief Requirements Engineer
- Model Based Systems Engineering
- Prototypes and Early Releases
- Agile

What has helped deal with complexity

- Feature Owner and Chief Requirements Engineer
- Model Based Systems Engineering
- Prototypes and Early Releases
- Agile



Unless someone is responsible for satisfying customer requirements, no-one is responsible for satisfying customer requirements

What has helped deal with complexity

- Feature Owner and Chief Requirements Engineer
- Model Based Systems Engineering
- Prototypes and Early Releases
- Agile

What has helped deal with complexity

Colin Hood SYSTEMS ENGINEERING

- Feature Owner and Chief Requirements Engineer
- Model Based Systems Engineering
- Prototypes and Early Releases
- Agile
  - These values and principles are based upon experience of what works in practice.
  - In 2001 these values and principles were summarised by a group of experienced people with similar concerns
  - Agile helps teams to concentrate on working together.
  - Increasing collaboration and building a common understanding of how parts interact and how systems operate helps to improve systems engineering.

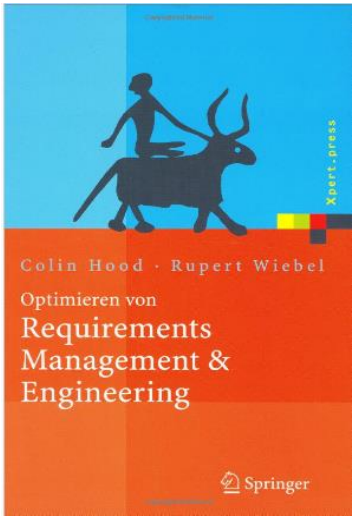
Principles behind the Agile Manifesto [JCJ92]	Feature Teams	Early Releases and Prototypes	User Stories	Purpose Interfaces and Constraints
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Yes	Yes		
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.		Yes		
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.		Yes		
4. Business people and developers must work together daily throughout the project.	Yes		Yes	
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Yes			Yes
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Yes			
7. Working software is the primary measure of progress.		Yes		
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	yes			
9. Continuous attention to technical excellence and good design enhances agility.	Yes			
10. Simplicity--the art of maximizing the amount of work not done--is essential.				Yes
11. The best architectures, requirements, and designs emerge from self-organizing teams.	Yes			
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Yes			

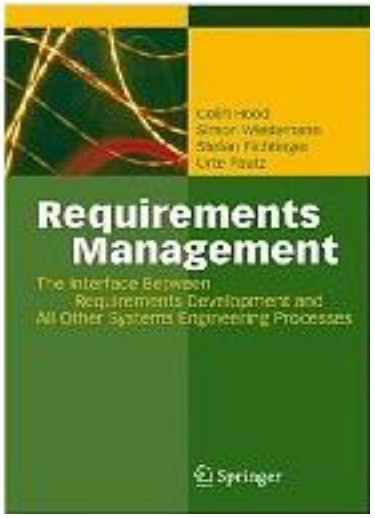
Conclusions

- It is essential that an holistic system view be taken
- Be responsible for system level success
- Harness the effects of interactions and interdependencies in a changeable environment

Systems Engineering since 1985

Colin Hood
SYSTEMS ENGINEERING





Colin Hood Systems Engineering Limited
Evans Business Centre
Regents Pavilion
4 Summerhouse Road
Northampton NN3 6BJ – UK

info@colinhood-se.com

Founding Board Member of IREB 2006
(International Requirements Engineering Board)

IREB International Requirements Engineering Board

INCOSE member since 1991

Bibliography

- [Bec18] Beck, K. et al: Manifesto for Agile Software Development. http://agilemanifesto.org/, last check August 2018
- [DeM78] DeMarco, T.: Structured Analysis and System Specification. Prentice Hall Inc., Englewood Cliffs, NJ, USA, 1978
- [Gil88] Gilb, T.: Principles of Software Engineering Management. Addison Wesley, Boston, MA, USA, 1988.
- [Hoo05] Hood, C.: Optimieren von Requirements Management & Engineering – Mit dem HOOD Capability Model. Springer, Berlin, 2005.
- [Hoo16] Hood, C.: Zurück in die Zukunft: Effizienzsteigerung des Anforderungsmanagements durch agile Methoden. In: OBJEKTspektrum, edition 02/2016.
- [Hoo86] Hood, C.: Structured Development Training Courses. 1986.
- [INC18] INCOSE: A Complexity Primer for Systems Engineers – White Paper. https://connect.incose.org/WorkingGroups/ComplexSystems/, San Diego, July 2016.
- [JCJ92] Jacobson, I.; Christerson, M.; Jonsson, P.: Object-Oriented Software Engineering - A Use Case Driven Approach, Addison-Wesley, Boston, MA, USA, 1992.
- [McC16] McChrystal, S.: Team of Teams: New Rules of Engagement for a Complex World. Penguin Books, London, 2016.
- [Roy70] Royce, W.: Managing The Development of Large Software Systems. http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf, last check August 2018
- [SM88] Shlaer, S.; Mellor, S.: Object-Oriented Systems Analysis: Modeling the World in Data. Yourdon Press Computing Series, Prentice Hall Inc., Englewood Cliffs, NJ, USA, 17978
- [Ste98] Stevens, S. et al: Systems Engineering: Coping with Complexity. Prentice Hall, Upper Saddle River, NJ, USA, 1998.
- [Tin12] Tincq, B.: From Henry Ford to Joe Justice – WikiSpeed – Manufactoring in the Age of Open Collaboration. http://ouishare.net/2012/10/wikispeed-agile-manufacturing/, last check August 2018
- [YC78] Yourdon, E.; Constantine, L.: Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. Ashgate Publishing Limited, Farnham, UK, 1978.