Designing for Innovation: Systems Engineering as a Key to Optimizing Business Value in Product Development

Enrico Mancin

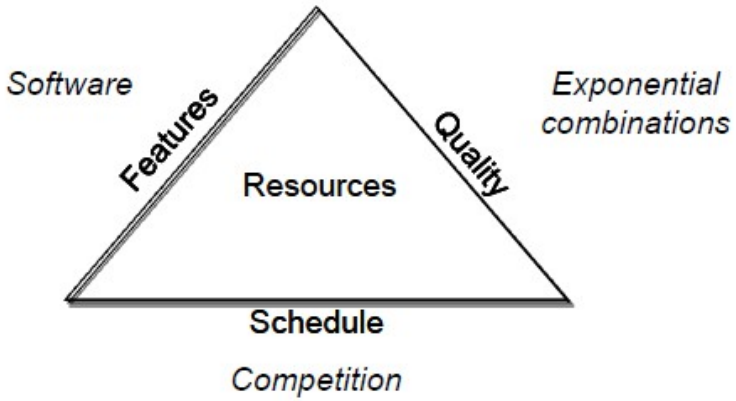
Tiger Team Systems Europe, Rational Software

Product Development is the Heart of the business

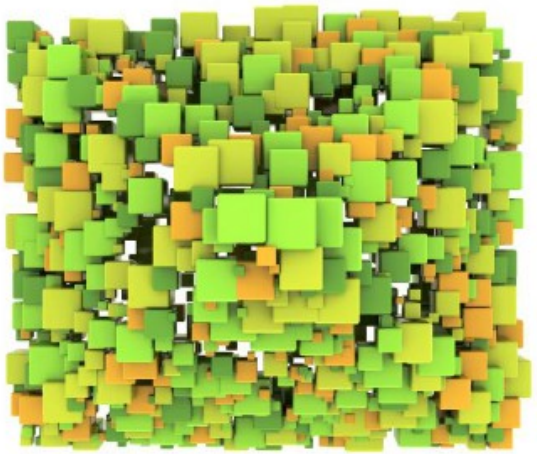
- Create or expand markets > revenue
  - Driven by delighting customers through
    - Features
    - Quality
    - Schedule

- Development Efficiency > profitability
  - Cannot reduce costs infinitely

Competing objectives requires a new way of doing things



"How do I increase the number of features, while improving quality, all in a shorter amount of time?



The human mind cannot wrap around all the combinations of touchpoints across engineering disciplines

Silo'ed organizations hinder collaboration and reuse

Rain Sensing Wiper: Example of a Design Failure

Individual Systems Worked, But Failed When Integrated

- Windshield provided by local supplier
  - Incompatible with the operation range of the sensor
  - No captured requirement for proper system calibration (i.e., verifying sensor and windshield compatibility)
  - ***Cars were sent to customers with non-functioning wiper systems***

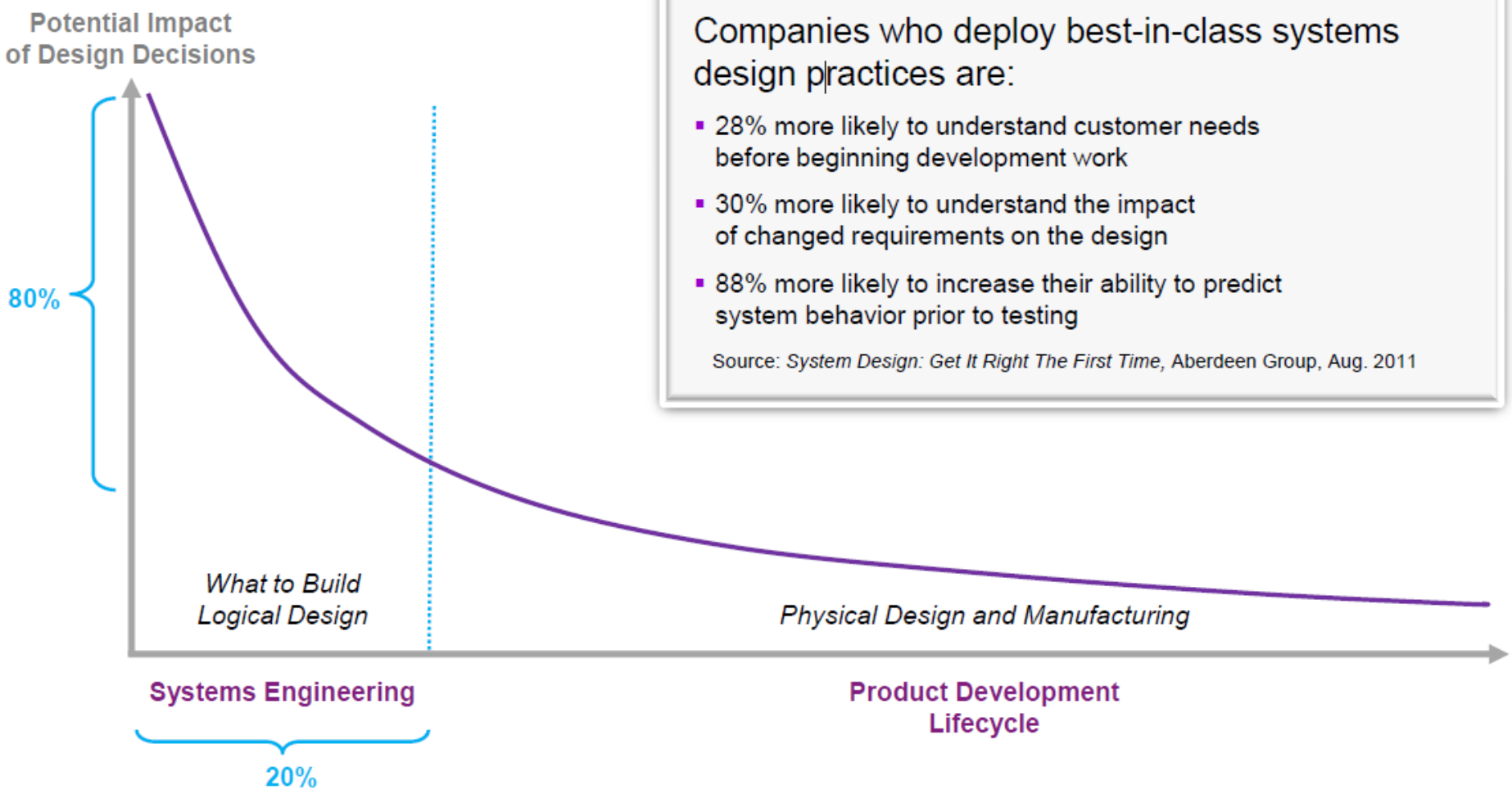
- Initial diagnostics designated software as culprit for malfunction
  - Mechanics couldn't test software behavior
  - Other components (electronic control unit, sensor, and windshield) functioned normally when tested independently
  - ***Failure was not of individual components, but in the interaction at a system level***

Let's step back... what do we really need to do?

- Build what the customer wants – no more, no less
- Test what you build – no more, no less
- Reduce the risk due to your inherent lack of knowledge – before you design the product
- Make sure everyone agrees and works from the same picture

Applying the 80-20 rule to the product development lifecycle

Systems engineering *can have the biggest impact on overall success*



Companies who deploy best-in-class systems design practices are:

- 28% more likely to understand customer needs before beginning development work
- 30% more likely to understand the impact of changed requirements on the design
- 88% more likely to increase their ability to predict system behavior prior to testing

Source: *System Design: Get It Right The First Time,* Aberdeen Group, Aug. 2011

Systems engineering is key to combating complexity, but what is it?

INCOSE (International Council on Systems Engineering) :

Systems engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs

Why systems engineering? *Systems engineering has the breadth and depth to address program-wide concerns*

•Many systems development programs trace difficulties to lack of systems engineering

• Key program issues are at the heart of systems engineering practice
- Requirements engineering
- Architecture Design/
  - >System-level modeling
  - >Interfaces
  - >Integration
- Verification, Validation and Quality
- Collaboration
  - >Configuration Management
  - >Change management
  - >Product lines

Best in class companies use systems engineering:
- ⊡85% of products launch on time
- ⊡9 out of 10 products meet revenue goals
- ⊡21% decrease in development time

Source: System Design: Get It Right The First Time, Aberdeen Group, Aug. 2011

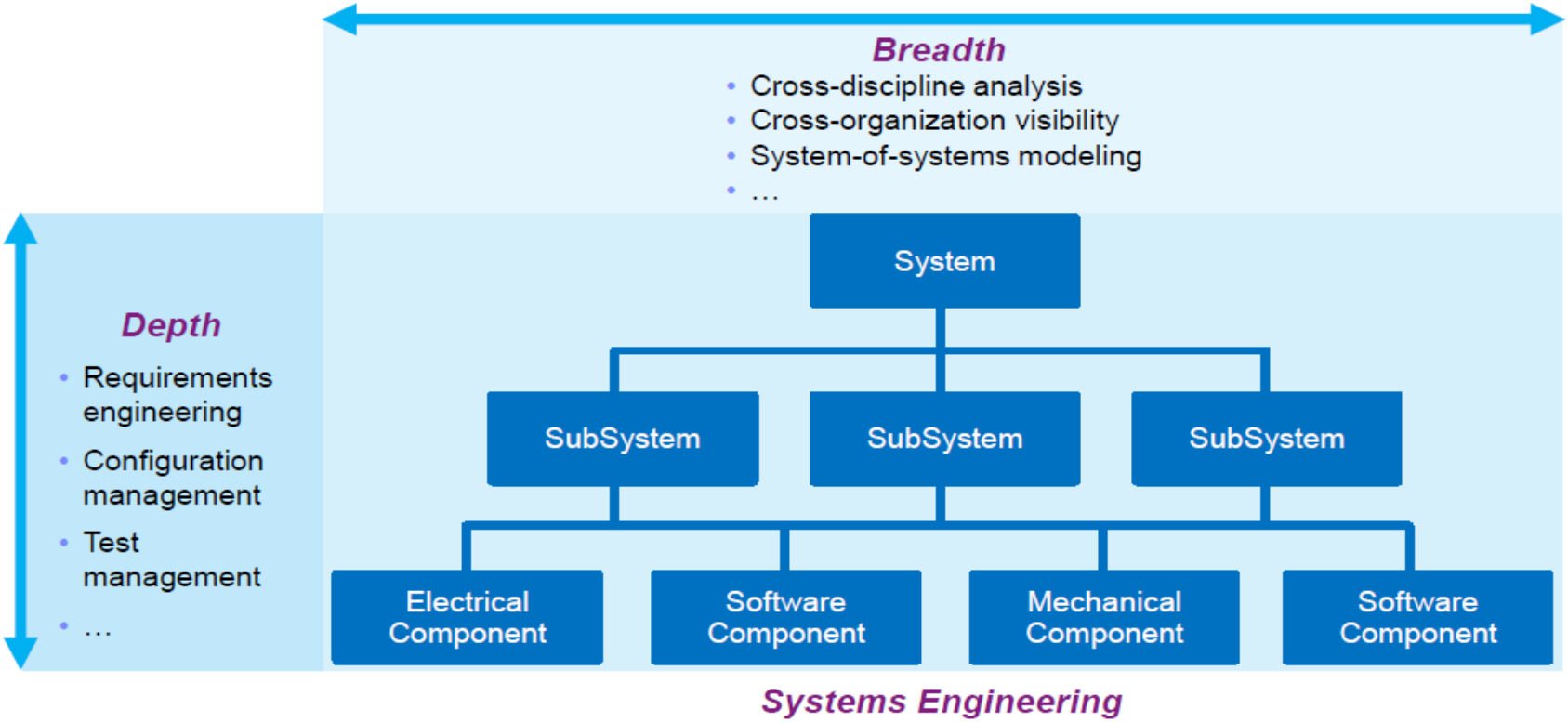
The breadth and depth of systems engineering

Breadth of systems engineering

- Systems engineering is a profession, a role, even a job title, for those who work exclusively at a system-of-systems level.

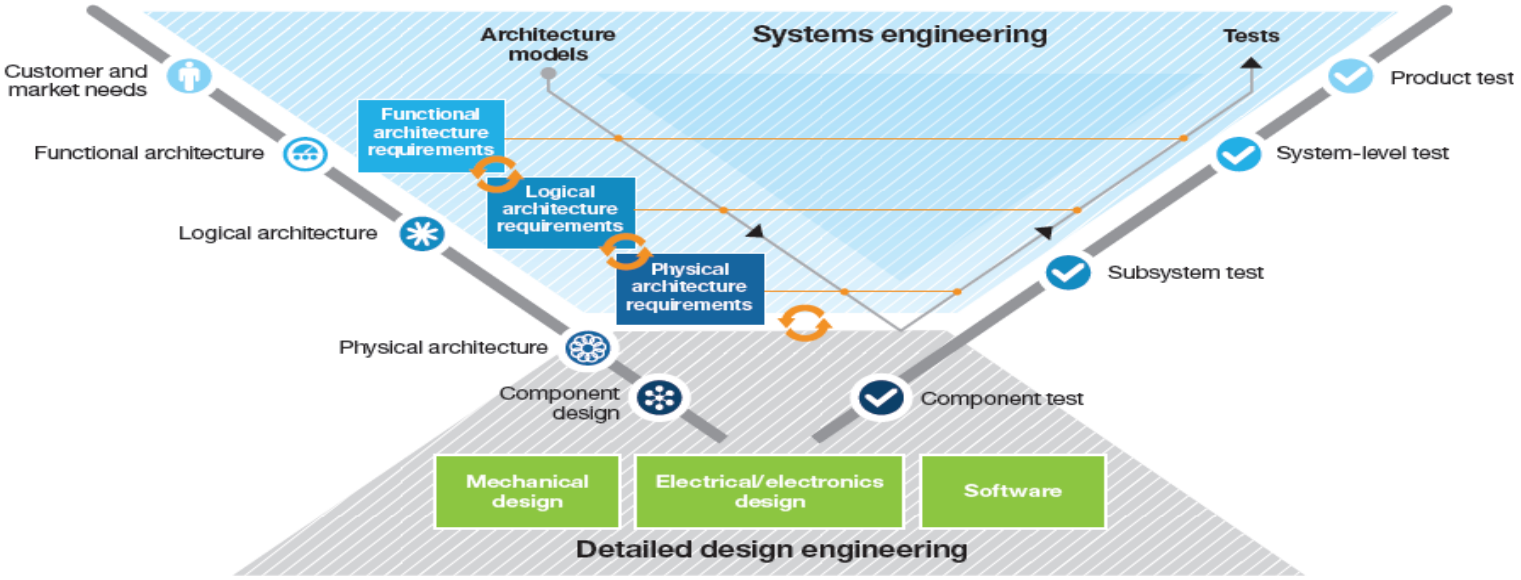
Depth of systems engineering

- Systems engineering also names a set of methods, skills and techniques that can be applied both at a system-of-systems level and within specific engineering disciplines.

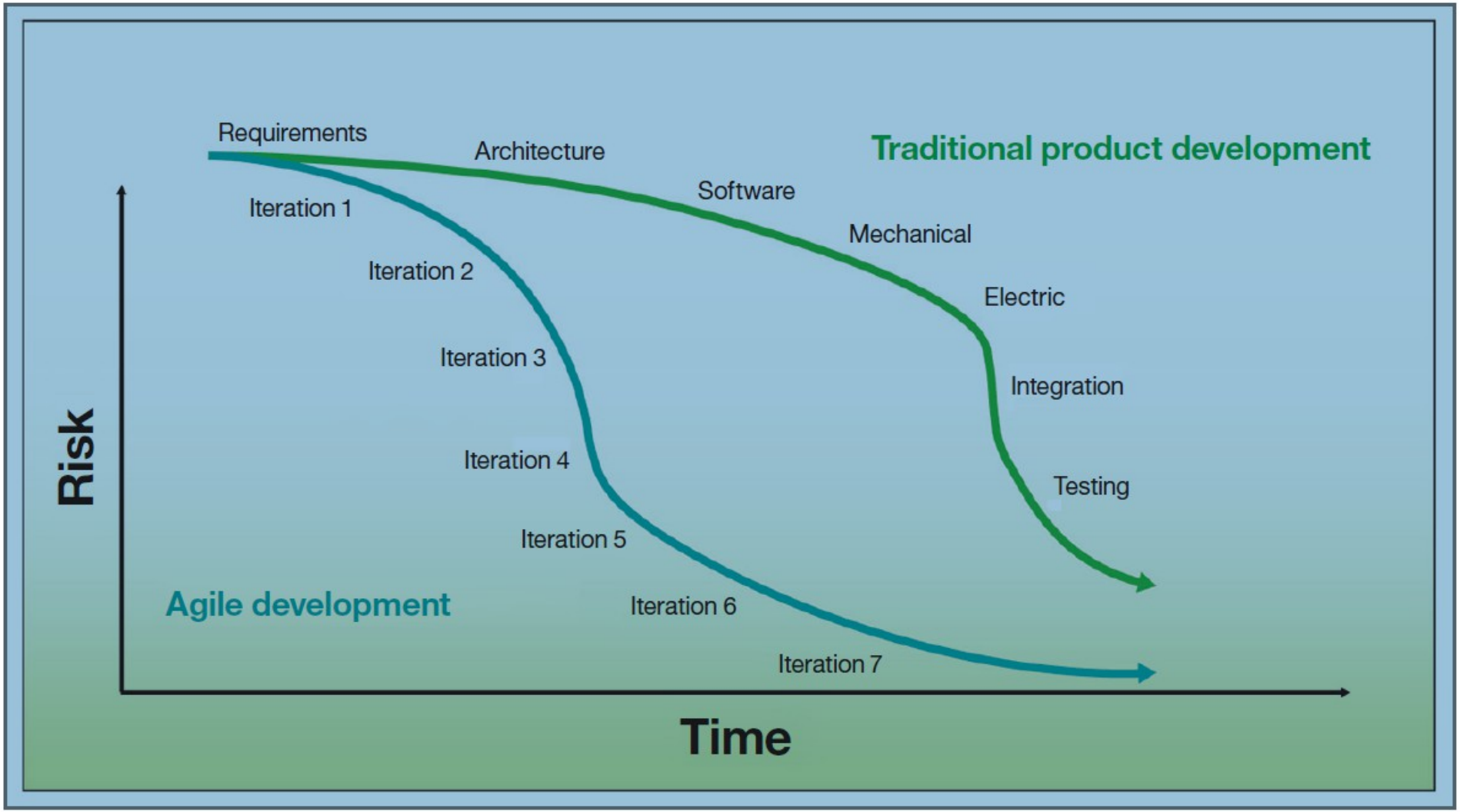


Key Components of Systems Engineering

- Requirements Management
  - Build what the customer wants – no more, no less
- Verification, Validation and Quality Management
  - Test what you build – no more, no less
- Architecture Design
  - Reduce the risk due to your inherent lack of knowledge – before you design the product
- Collaboration
  - Make sure everyone agrees and works from the same picture

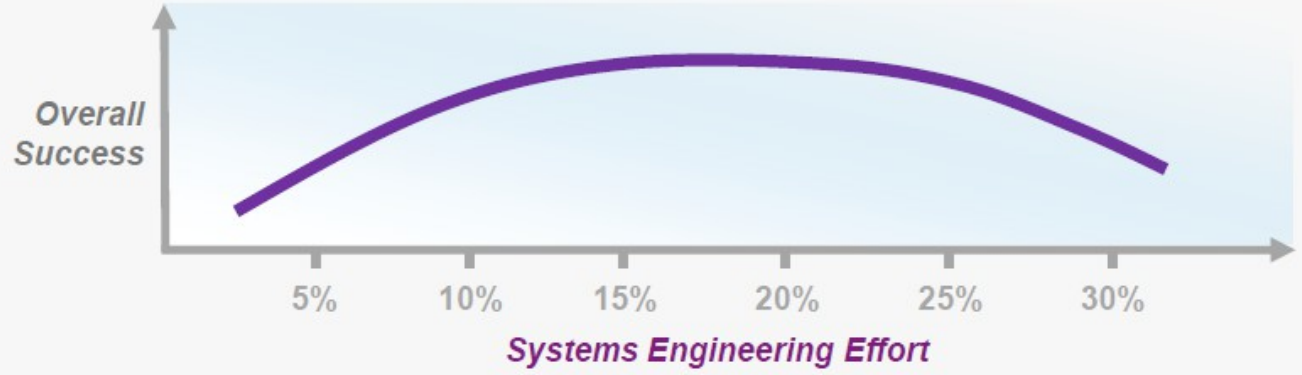


Agile product development is not an oxymoron



Applying the right amount of systems engineering is critical to program success

- Too little systems engineering
  - Systems engineering seen as an afterthought, documentation, process-if-we-have-time
- Too much systems engineering
  - Applying “space shuttle” processes to ordinary projects
  - Forcing development into “system-of-systems” paradigms
- Main systems engineering objective: Common understanding



Source: Honour, Eric (2010), *Systems Engineering Return on Investment*, University of South Australia, p. 9

Motivating change

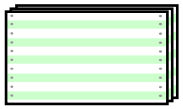
Building the business case
Prioritizing change initiatives

Enrico Mancin

Tiger Team Systems Europe, Rational Software

Rational. software

Historical vs. Modern Perspective on Productivity



- As an example, in software development Software Lines Of Code (SLOC) was easy to measure so we do:
  - There is a large pool of SLOC based productivity statistics
  - These statistics are still the most complete set of data existing
- Function Points & Use Case Points were developed to solve shortcomings, however:
  - In software, different programmers code differently, and not all code add value
  - Functions or Use Cases can have different "sizes"
- Experience indicates that utility of these measures is limited:
  - Intra-team, it can be used (*carefully*!) to calibrate productivity within the team
  - Across organizations & platforms, comparisons break down

- Many engineering organizations now have sophisticated dashboards
  - Richer measurement set
  - Much less human intervention

- Productivity is not just about asset volume – Productivity is about *business value produced*
  - Challenge: determining a consistent meaning for business value that is useful to the entire organization
  - Challenge: making the measures meaningful, transparent, real-time, and focused on outcome
- **Conclusion**: a productivity strategy should include
  - Transparency at all levels in an organization
  - Consistent and useful leverage of a carefully crafted chain of related measures focused on development throughput
  - Using modern development technologies that include measurement facilities to foster continuing relevance, accuracy, and real-time results

Transparency to address business value

CEO/CFO

CIO/CTO

Project Manager

Developer

Value chain

Business management
Earnings per share
Profit

Portfolio management
Portfolio complexity analysis
Total Cost of Ownership
Investment analytics
Investment prioritization

Operational effectiveness
Time to market
Predictability
Cost

Project/release
Quality metrics
Product stability
Test coverage, etc

Core dev loop
Sprint cycles
Burn rate, etc

SCOPE

Individual ***Team & Organization*** ***Business***

Transparency to address business value

to the key constituencies of the business

CEO/CFO

CIO/CTO

Project Manager

Developer

Value chain

Business management
Earnings per share
Profit

Portfolio management
Portfolio complexity analysis
…l Cost of Ownership
…ment analytics
…ent prioritization

Ope… effec…
Time …
Predicta…
Cost

Pr…

Core dev lo…
Sprint cycles
Burn rate, etc

Customers

Employees

Owners

SCOPE

Individual

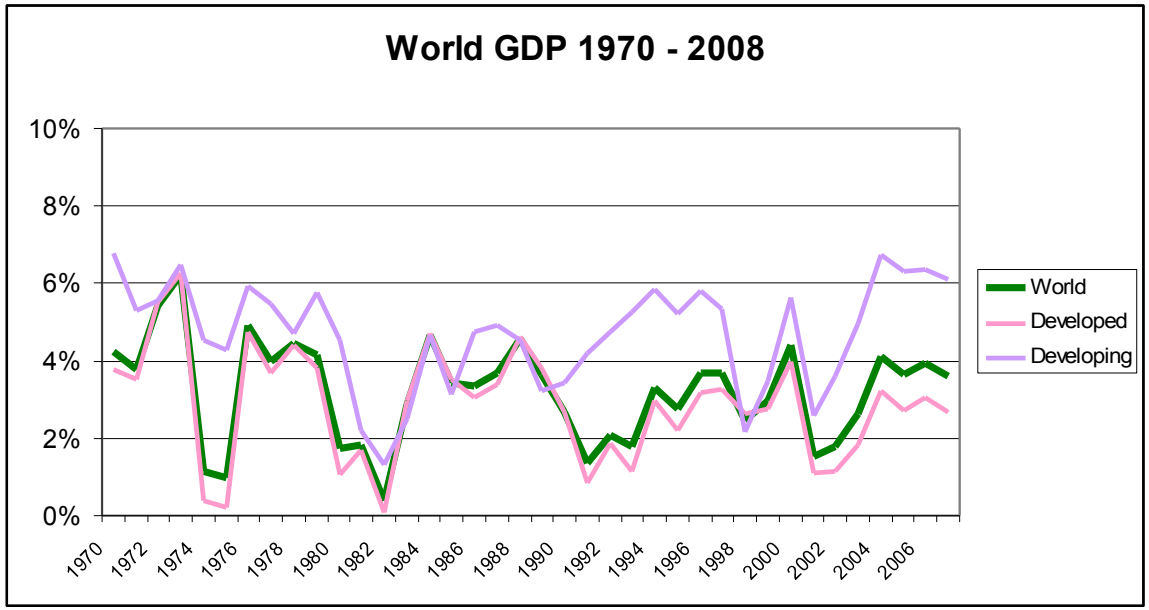
Team & Organization

Business

What we have learned: how help improve organizational value by ~3-5% per year

- Organization value = A View of Productivity
  - Productivity = Cost reduction
  - Productivity = Increased capacity

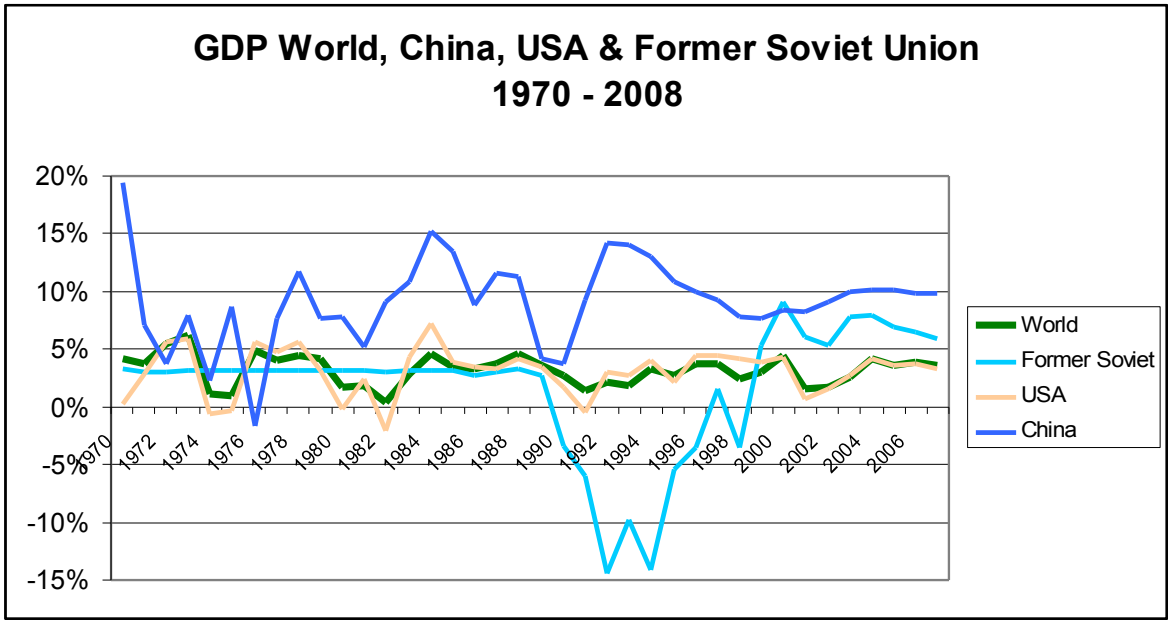
- A mature approach to organizational improvement
  - The organizational productivity increases are well documented
  - Sustained improvement is incredibly difficult
  - Many disruptive forces: Global economic downturns, M&A, new business models…



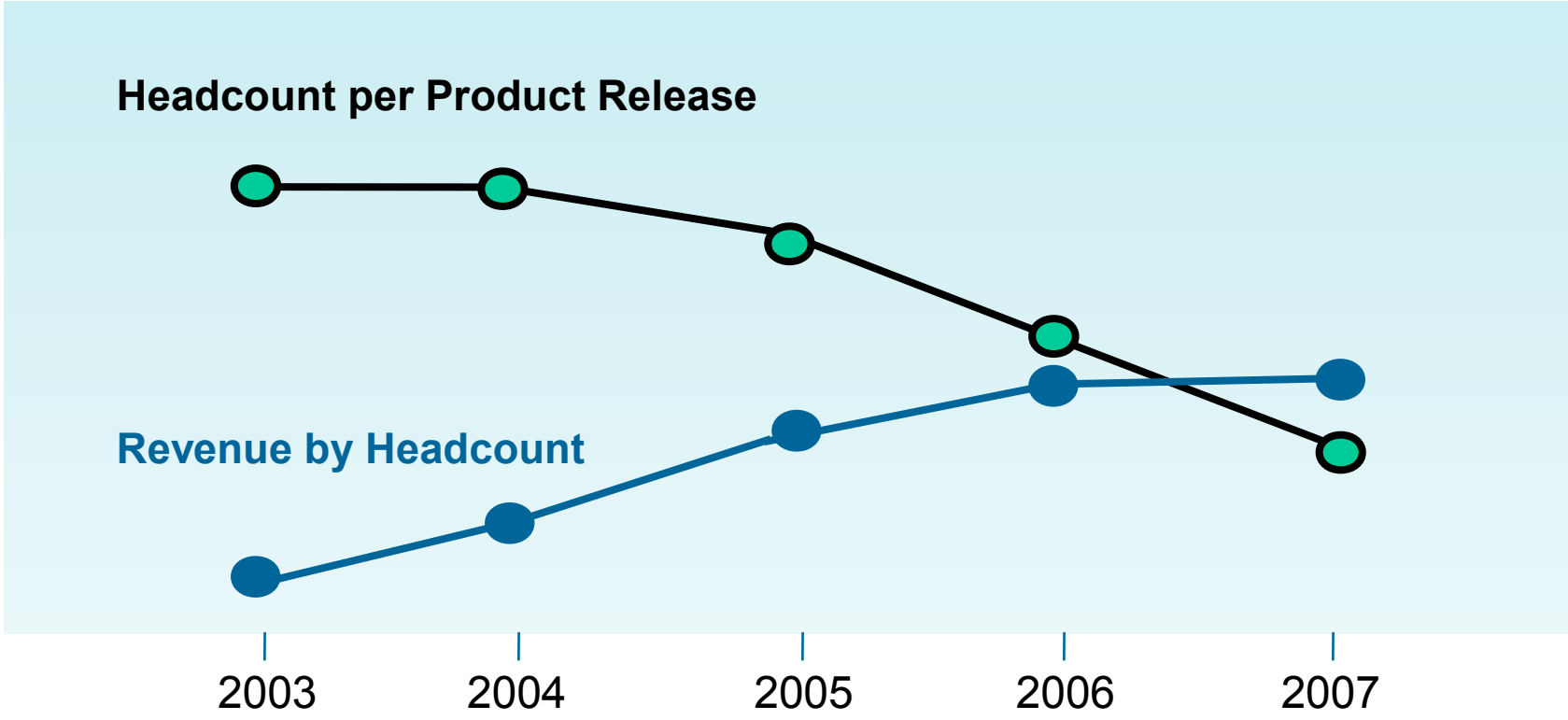
What we have learned: how help improve organizational value by ~3-5% per year

- No one has sustained ~10% annual improvement for more than 5 consecutive years in the world in the last 40 years
  - Suitable as comparison for immature organizations

- The 5% Rule
  - Mature organizations: Sustained annual productivity gains of 5%
  - Emerging organizations: Target to outperform the rest of the market by 5% over a sustained period



Example: Improving productivity in IBM SWG through improved agility

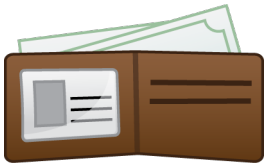


You Can Only Do So Many Things at Once

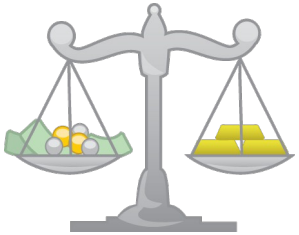
- If you change everything at once, it is likely that some things will go right and some things will go wrong
- The bad can make the good things look bad
- Thus each change needs to be isolated from the others
- There are two ways to do this:
  - One improvement across the entire organization
  - Different discreet projects get different discreet improvements
- These experiments take time and serialize the improvement thus lengthening the time to get the total improvement

Thoughts on Transforming an Organization - Financial Perspective

- A way of looking at the practices outlined is as a portfolio of investments with varying risks and returns



- The challenge is to find the efficient frontier that balances risk and returns



Thoughts on the Ordering of the Practices

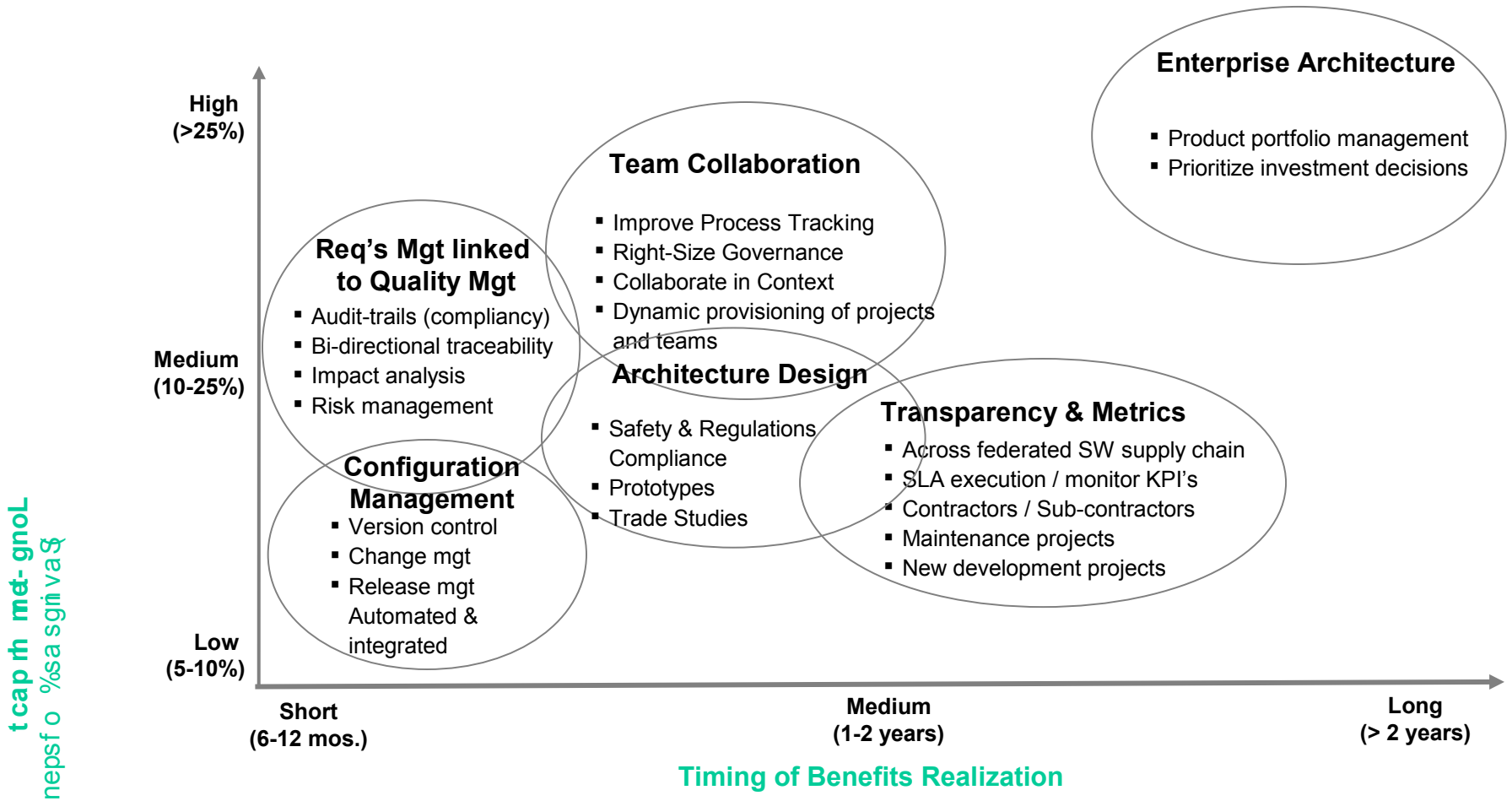
- **The earlier in the process that one uses the practice, the higher the risk and the return**. The risk is higher because even if an organization has superb alignment with business initiative, requirements management and requirements development practices, design and code practices – errors can still occur and if the defect removal practices are faulty, then the strength early in the SDLC will be lost.
  - This manifests itself by projects going from green to red late in the lifecycle
  - Or through low customer satisfaction because of poor quality
  - In addition the results of the improvements typically take time because the customer does not see an impact much later
- The return is higher because scrap and rework is removed early in the lifecycle
  - If an error occurs early and then is carried forward, all the development and test effort is wasted as well
- **For practices in focus late in the development lifecycle, risk and returns are lowered**:
  - Scrap and rework discovered after a few steps are executed thus costing money
  - Risk is lowered because there are fewer steps left and the results manifest themselves quickly
  - One does not have to wait 8x for test and deployment improvements to reach the customer because the results of the improvements can show in 1 to 2x.
- Thus working late in the lifecycle is low risk with a visible return

Example – risk vs. impact discussion

- Requirements
  - Addressing requirements is high-risk as early in the cycle. Does have relatively high impact to productivity .
  - Medium size mature project: chance of success ~3%
- Architecture Design
  - Works well when cost of failure is high and the cost of testing is high
    - Such as DO 178 Level 1 or Level 2; wire transfer room; medical devices
    - Analysis does not work well when it is lower risk to prototype, code and deploy such as small-scale agile software development
    - Productivity improvement in high regulation or high cost of failure is around 5 to 10% over 2 years
- Configuration Management
  - Mitigates risk and improves productivity by avoiding scrap and rework caused by using incorrect versions
  - Medium size mature chance of success ~90%
- Transparency and Metrics
  - Metrics foundational practice to be able to verify impact of change
  - Impact of metrics itself is low
  - Medium size mature project: chance of success ~80%
- Verification, Validation and Quality Management
  - Typically lower risk effort and more reliable impact
  - Medium size mature project: chance of success ~70%

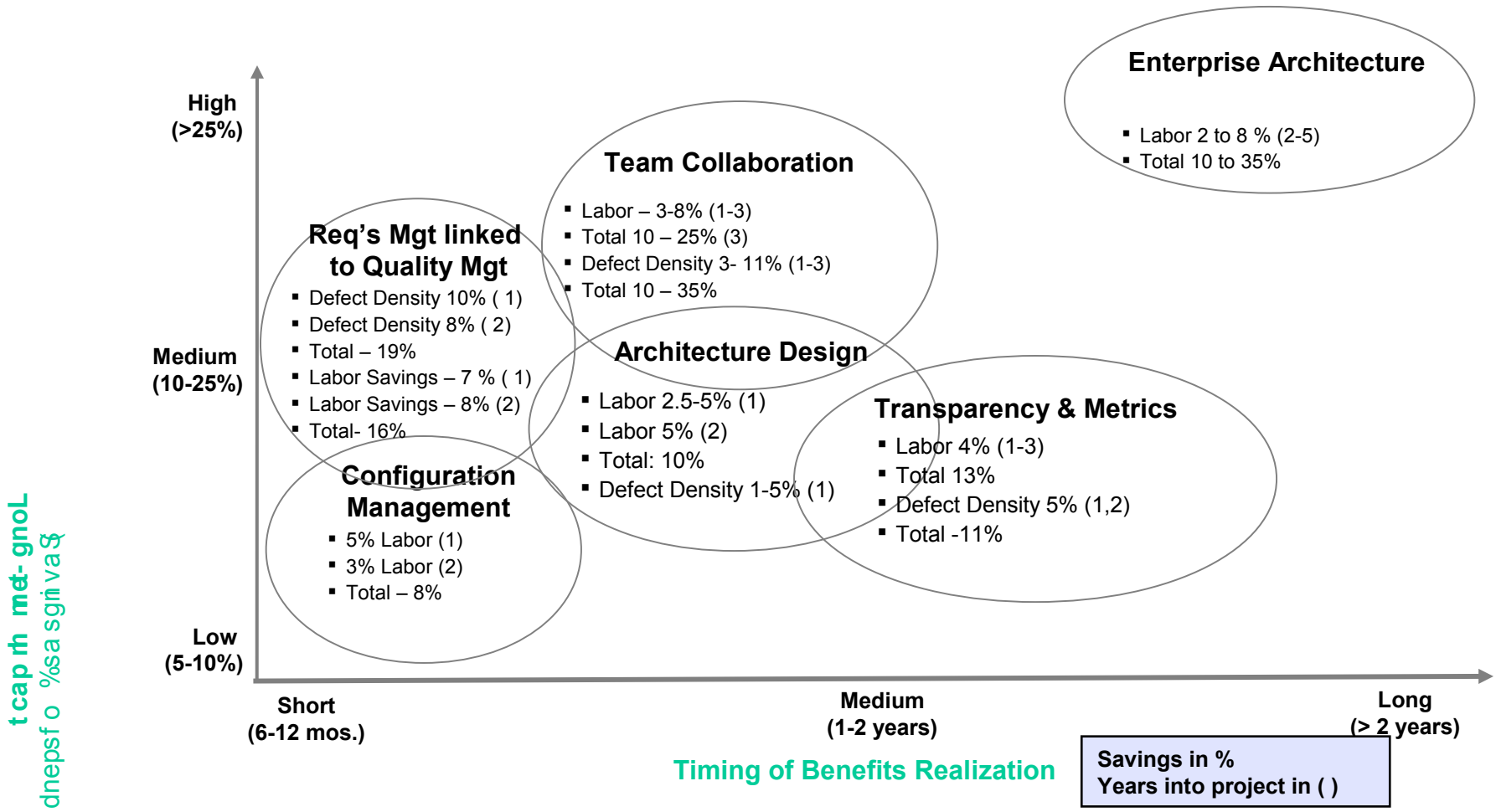
An Illustration of Impact and Timing

The Bigger Picture



An Illustration of Impact and Timing

The Numbers



Do you need help?: SE and Business Value Assessment