l e a n

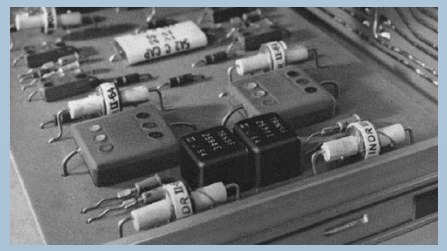
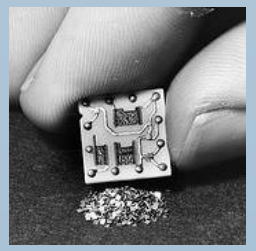
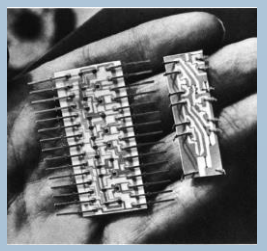
software development

Abstraction and Federation

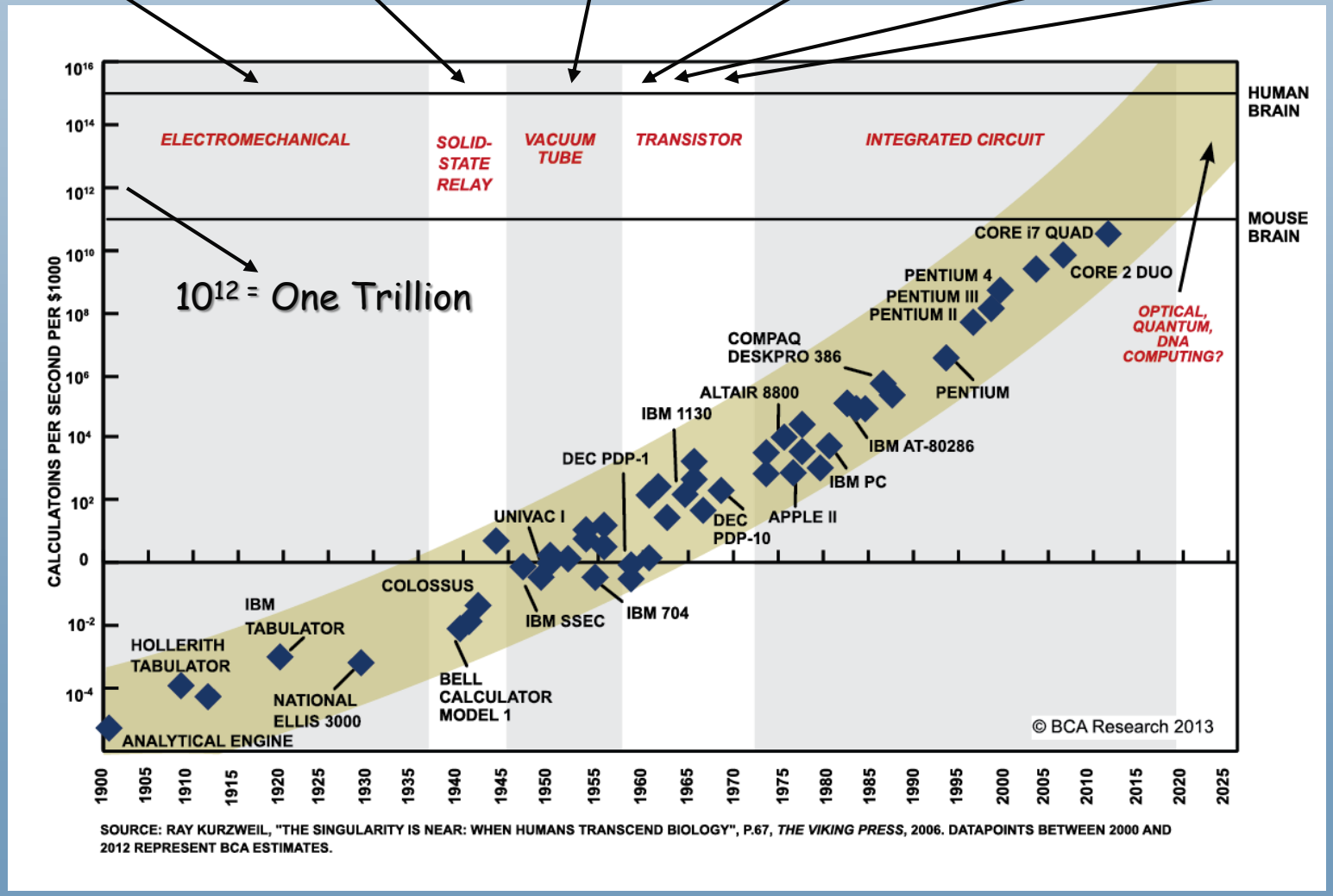
From Micro Chips to Microservices

mary@poppendieck.com Mary Poppendieck www.poppendieck.com



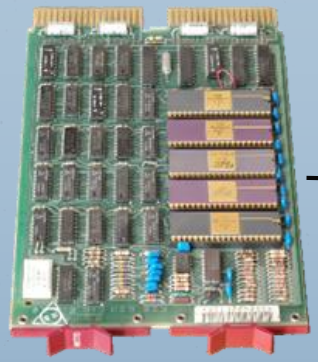
1910 1943 1950 1960 1964 1967



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Hardware Scales by Miniaturization and Abstraction

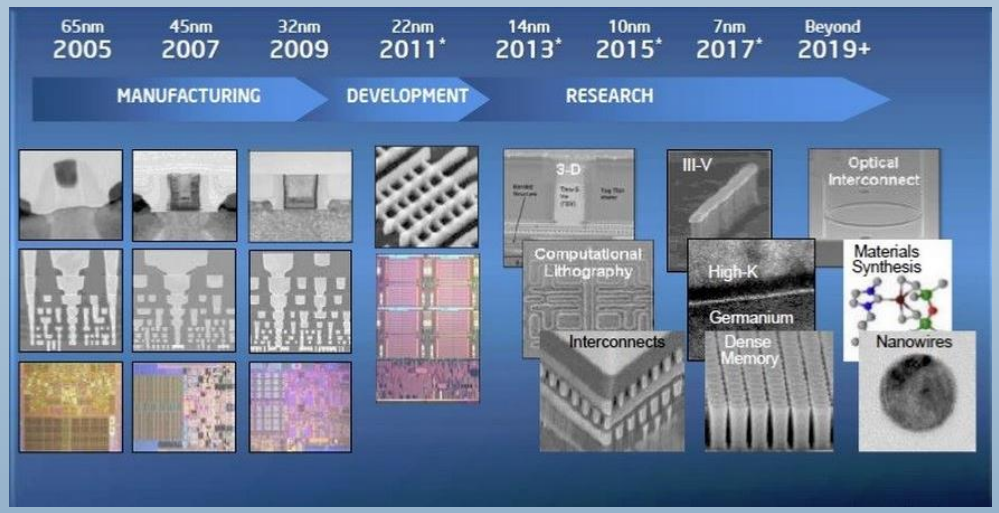
Miniaturization



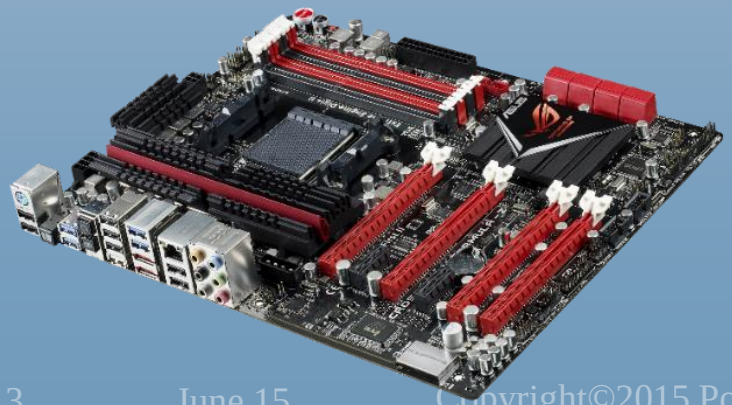
40 years

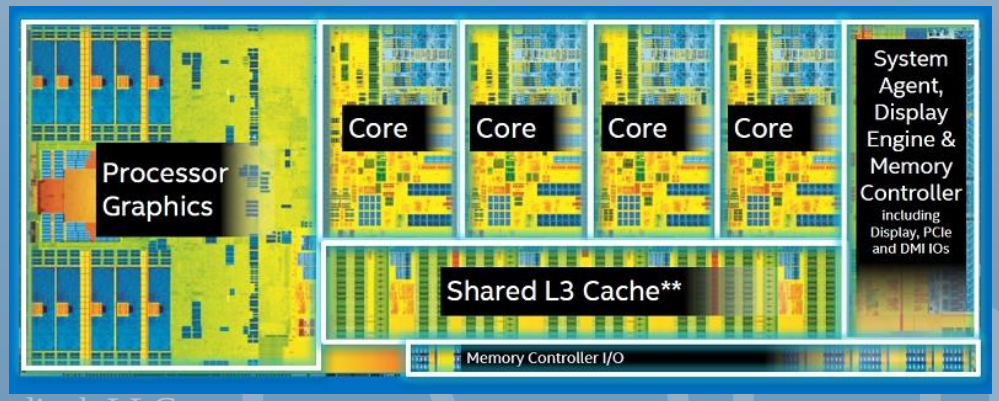
1975 LSI 11

2015 Core™ M



Abstraction





Does Software Scale through Abstraction?

1975 ⟶ 2015

Simple Fortran IV program

Multiple data card input

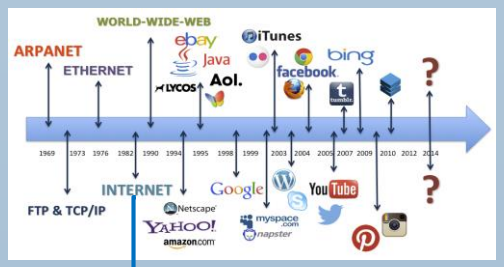
This program has two input checks: one for a blank card to indicate end-of-data, and the other for a zero

Either condition causes a message to be printed.

```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT, ONE BLANK CARD FOR END-OF-DATA
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPAY ERROR MESSAGE ON OUTPUT
  501 FORMAT(3I5)
  601 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,12HSQUARE UNITS)
  602 FORMAT(10HNORMAL END)
  603 FORMAT(23HINPUT ERROR, ZERO VALUE)
      INTEGER A,B,C
   10 READ(5,501) A,B,C
      IF(A.EQ.0 .AND. B.EQ.0 .AND. C.EQ.0) GO TO 50
      IF(A.EQ.0 .OR.  B.EQ.0 .OR.  C.EQ.0) GO TO 90
      S = (A + B + C) / 2.0
      AREA = SQRT( S * (S - A) * (S - B) * (S - C))
      WRITE(6,601) A,B,C,AREA
      GO TO 10
   50 WRITE(6,602)
      STOP
   90 WRITE(6,603)
      STOP
      END
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
    <!-- TemplateBeginEditable name="doctitle" -->
    <title>Poppendieck.LLC</title>
    <!-- TemplateEndEditable -->
    <meta name="description" content="Designed and developed by Codify Design Studio - codifydesign.com" />
    <link rel="stylesheet" type="text/css" href="stylesheet.css" />
    <link rel="SHORTCUT ICON" href="http://www.poppendieck.com/images/poptop.JPG" />
    <!-- TemplateBeginEditable name="head" -->
    <!-- TemplateEndEditable -->
    <script src="SpryAssets/SpryAccordion.js" type="text/javascript"></script>
    <script src="SpryAssets/SpryMenuBar.js" type="text/javascript"></script>
    <script src="SpryAssets/SpryCollapsiblePanel.js" type="text/javascript"></script>
    <script src="SpryAssets/SpryTabbedPanels.js" type="text/javascript"></script>
    <script type="text/javascript" src="DC_PictureCalendar/public.js"></script>
    <script>initPictureCalendar("call",1,2,"ENG",{},75,75);</script>
    <style>
    #callContainer .yui-calendar td.calcell {width:75px;height:75px;}
    </style>
    <link href="SpryAssets/SpryAccordion.css" rel="stylesheet" type="text/css" />
    <link href="SpryAssets/SpryTabbedPanels.css" rel="stylesheet" type="text/css" />
    <link href="SpryAssets/SpryMenuBarHorizontal.css" rel="stylesheet" type="text/css" />
    </head>

    <body>

      <div class="container">
        <div class="bannerArea">
            <a href="http://www.poppendieck.com">
            <div class="toplogo">
            </div></a>
          <ul id="MenuBar1" class="MenuBarHorizontal">
           <li><a class="MenuBarItemSubmenu" href="#">Essays</a>
             <ul>
                        <li><a href="http://www.leaneassays.com/">Latest Essay</a></li>
             <li><a href="http://www.leaneassays.com/2004/08/team-compensation.html">Compensation</a></li>
             <li><a href="http://www.leaneassays.com/2011/07/how-cadence-determines-process.html">Cadence</a></li>
             <li><a href="http://www.leaneassays.com/2010/12/product-owner-problem.html">Roles</a></li>
             <li><a href="http://www.leaneassays.com/2011/02/before-there-was-management.html">Peer Cultures</a></li>

             </ul>
            <li><a class="MenuBarItemSubmenu" href="#">Workshops</a>
             <ul>

            <li><a href="http://www.poppendieck.com/workshop1.html">Lean Software Development Workshop</a></li>
            <li><a href="http://www.poppendieck.com/workshop.html">Lean Mindset Workshop</a></li>
            </ul>
            <li><a class="MenuBarItemSubmenu" href="#">Media</a>
             <ul>
                 <li><a href="http://www.youtube.com/watch?v=ypEMdjslEOI"><strong>Video:</strong>  The Role of Leadership in Software Development</a></li>
                 <li><a href="http://www.youtube.com/watch?v=ZSdsjpCcpFw"><strong>Video:</strong>  Competing On The Basis Of Speed</a></li>
                 <li><a href="http://www.infoq.com/presentations/tyranny-of-plan"><strong>Video:</strong> The Tyranny of "The Plan"</a></li>
                 <li><a href="https://twitter.com/mpoppendieck"><strong>Twitter</a></strong></li>
              </ul>
            </li>
            <li><a href="http://www.poppendieck.com/reference.htm">Bookshelf</a> </li>
            <li><a href="http://www.poppendieck.com/people.htm">About Us</a></li>
          </ul>
          </div>
```

1983

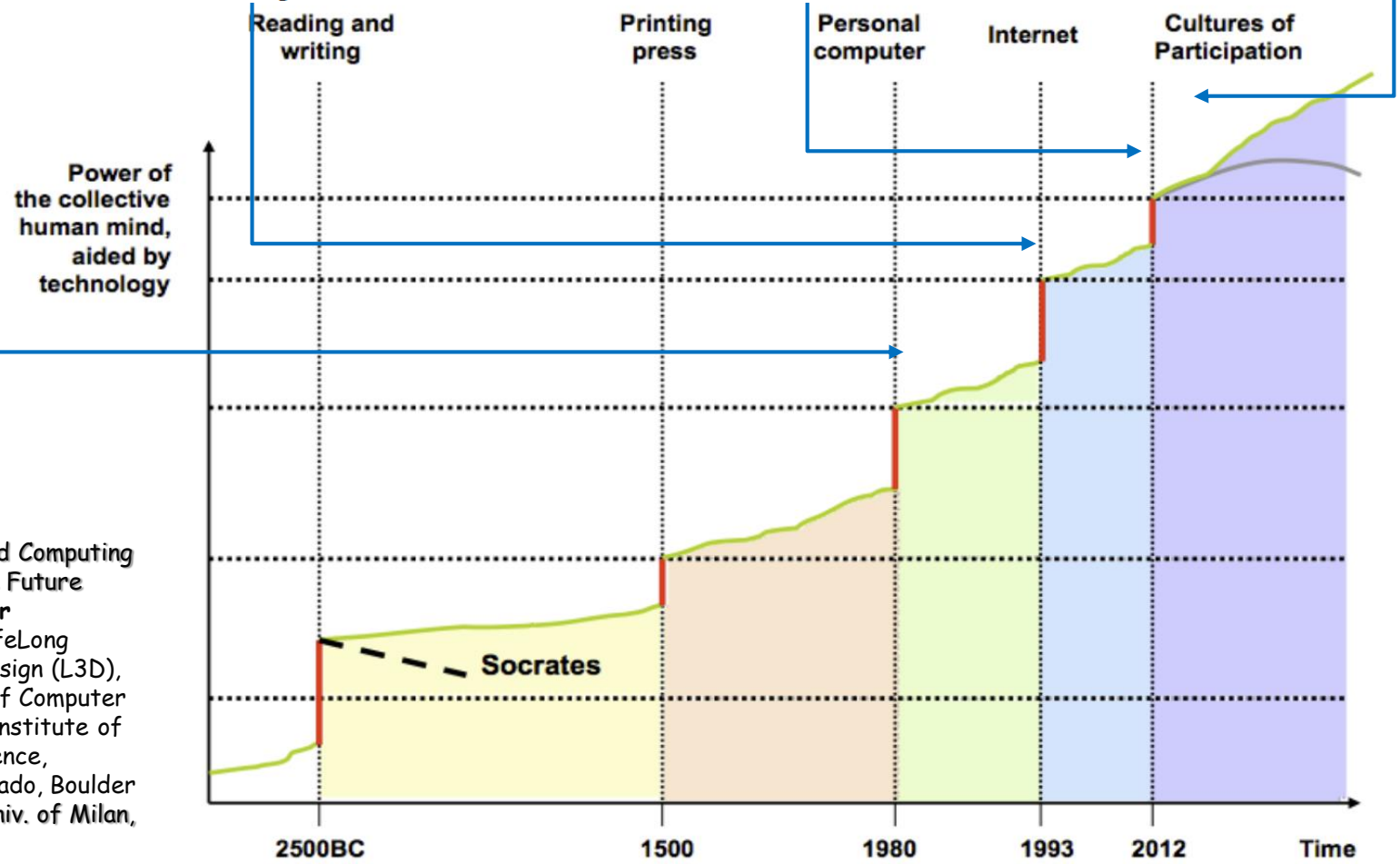


1993

2007

2015 +

Beyond the Unaided, Individual Human Mind



Human-Centered Computing Themes for the Future

Gerhard Fischer
Center for LifeLong Learning & Design (L3D), Department of Computer Science and Institute of Cognitive Science, Univ. of Colorado, Boulder

Presented at Univ. of Milan, February 2012

Software Scales by Federation and Wide Participation

Federation

Minicomputer Embedded

- ✓ Local equipment control (Isolation increases reliability)

PC Servers

- ✓ Retail software packages (Independent install and run)

Internet Services

- ✓ Websites (Classic federated architecture)

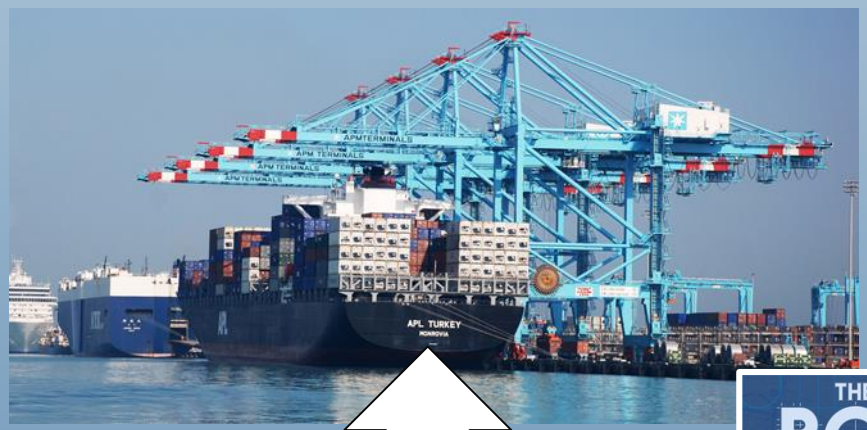
Smartphone Wearables

- ✓ Apps (Interact through the cloud)

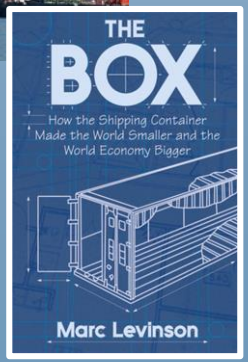
Wide Participation

Friction: Large System Disease

"*Everything in war is simple, but the simplest thing is difficult. The difficulties accumulate and end by producing a kind of* ***friction*** *that is inconceivable unless one has experienced war.*" --Carl von Clausewitz

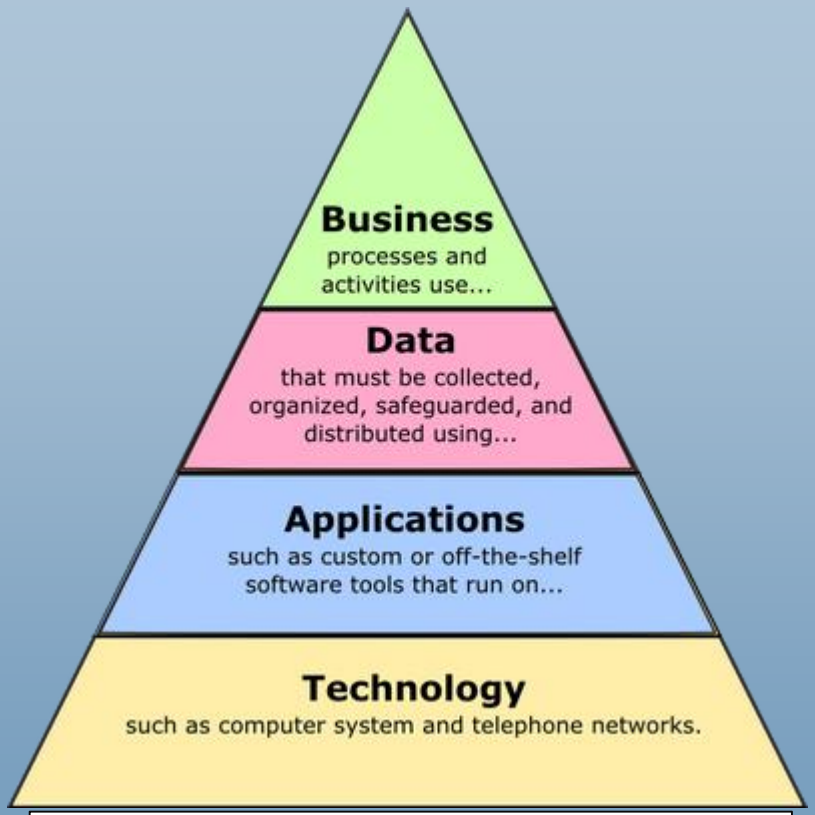


High Friction *Low Friction*



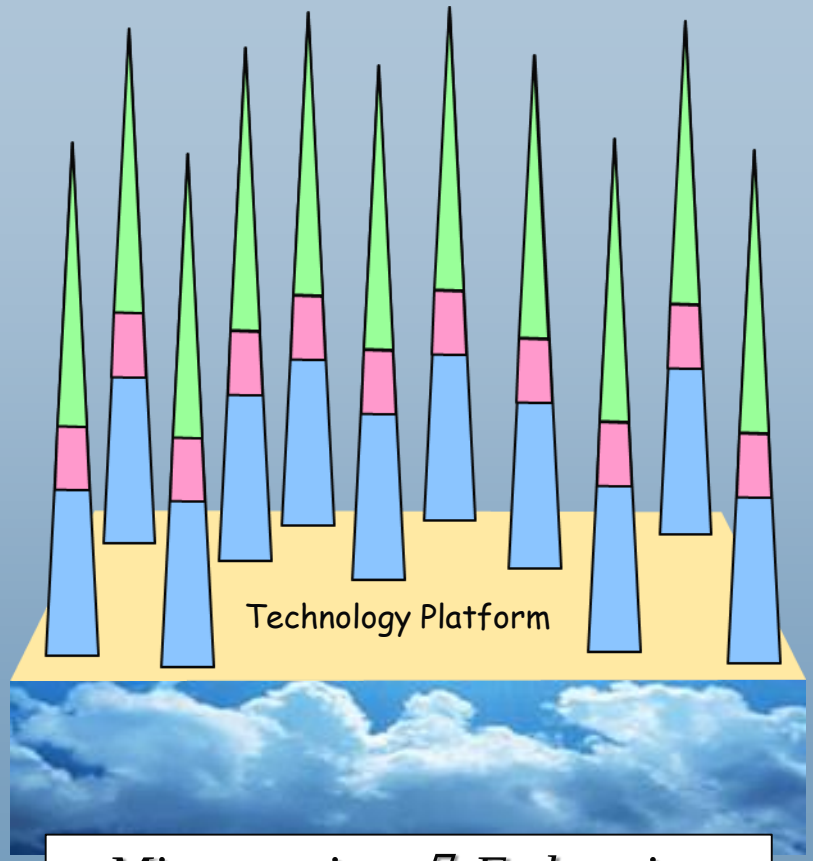
The Problem with Databases

Monolithic Architecture



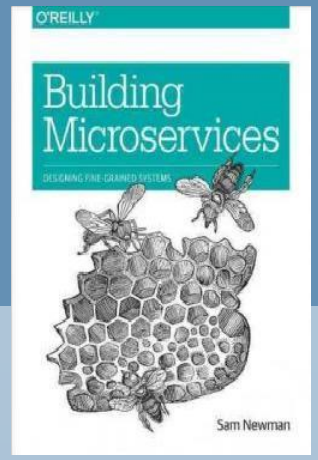
Database → Deep Dependencies → High Friction

Microservice Architecture



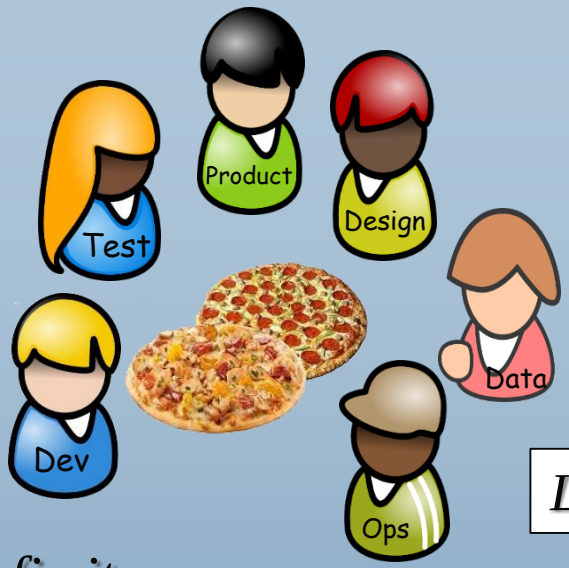
Microservices → Federation → Low Friction

Reduce Friction: Monoliths ➔ Microservices



What is a Microservice?

1. **Small service:**
   Does one thing well.
   Independently Deployable.
2. **Small team:**
   End-to-end Responsibility.
   You build it – you monitor it – you fix it.
3. **Practices:**
   No Central Databases.
   Extensive Automation and Monitoring
   Double Mock Contract Testing*
   Smart Versioning Services.
   Canary Releasing.



Low Friction

amazon.com

Spotify NETFLIX

GILT realestate.com.au

*Beth Skurrie

*https://github.com/realestate-com-au/pact

BUT

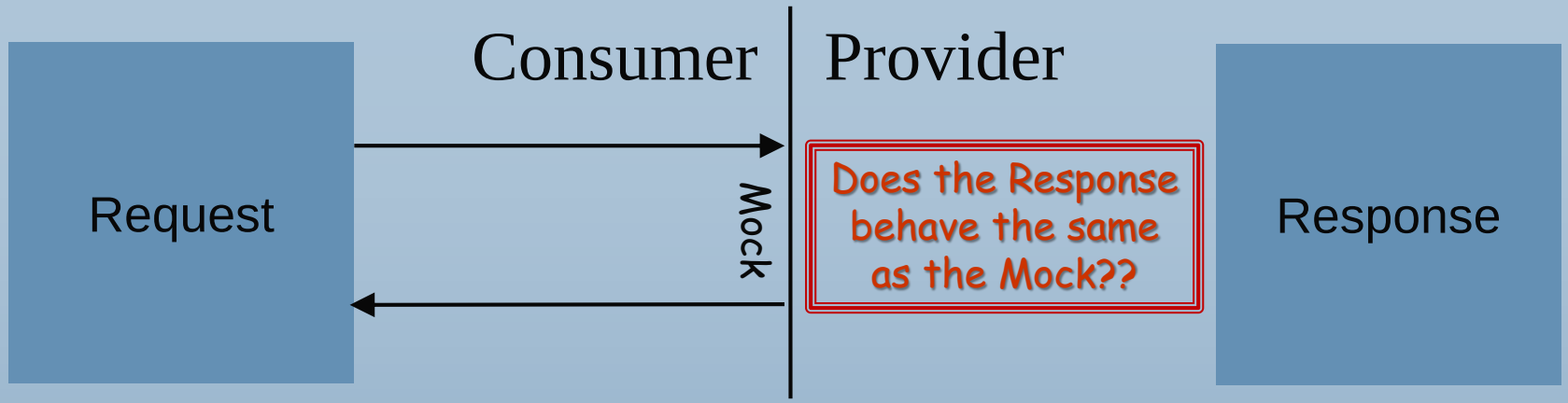
Microservices ↑ Risk

High Risk

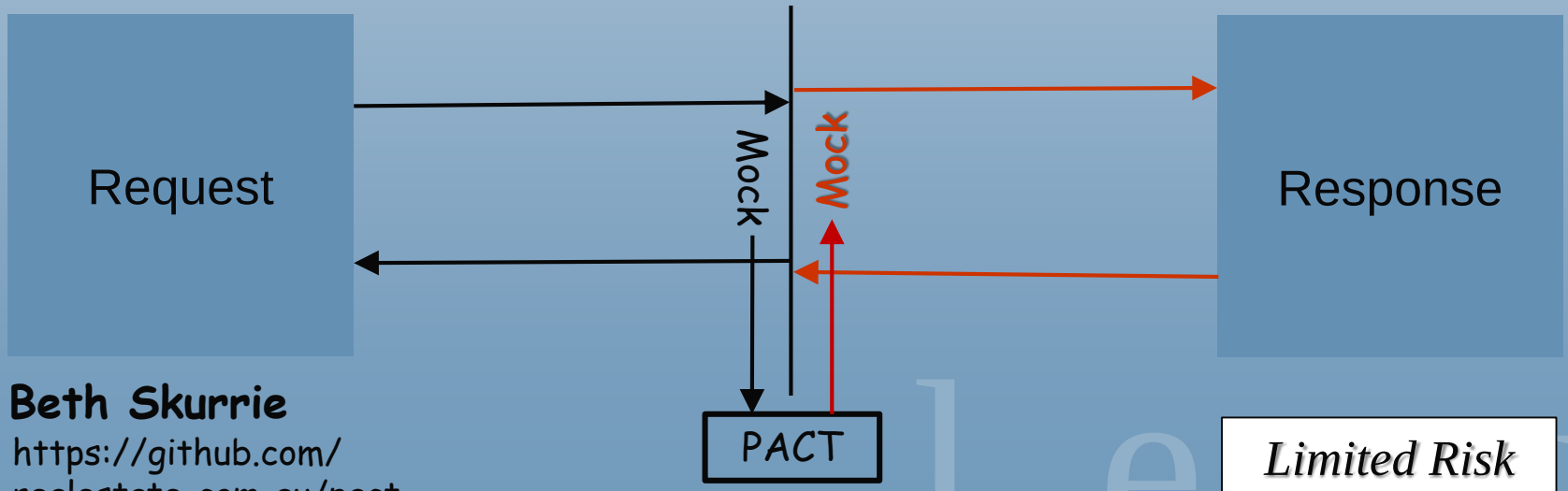
Avoiding Dependency Hell

1. ***Is it Right for the Domain?***
   a. Very high volumes seem to require microservices
2. ***Do You Understand the Domain?***
   a. Get the bounded contexts right before you start!
   b. Refactoring across services is difficult
3. ***Maintain Strict Discipline***
   a. Interaction restricted to hardened interfaces
   b. Service teams maintain ***situational awareness*** of their service, its consumers, and its providers.

Limit Risk with Situational Awareness



http://martinfowler.com/articles/consumerDrivenContracts.html

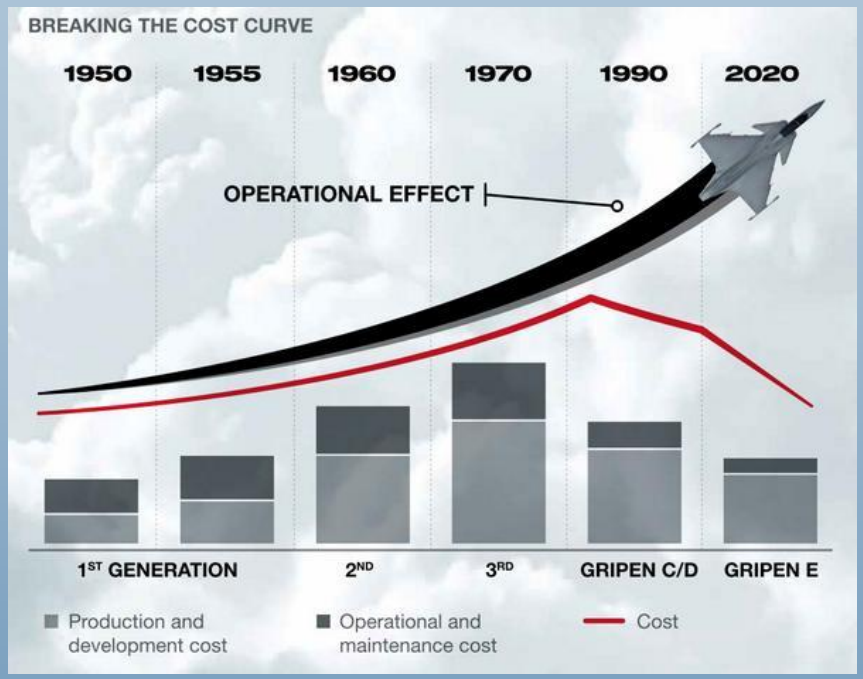


Beth Skurrie
https://github.com/realestate-com-au/pact

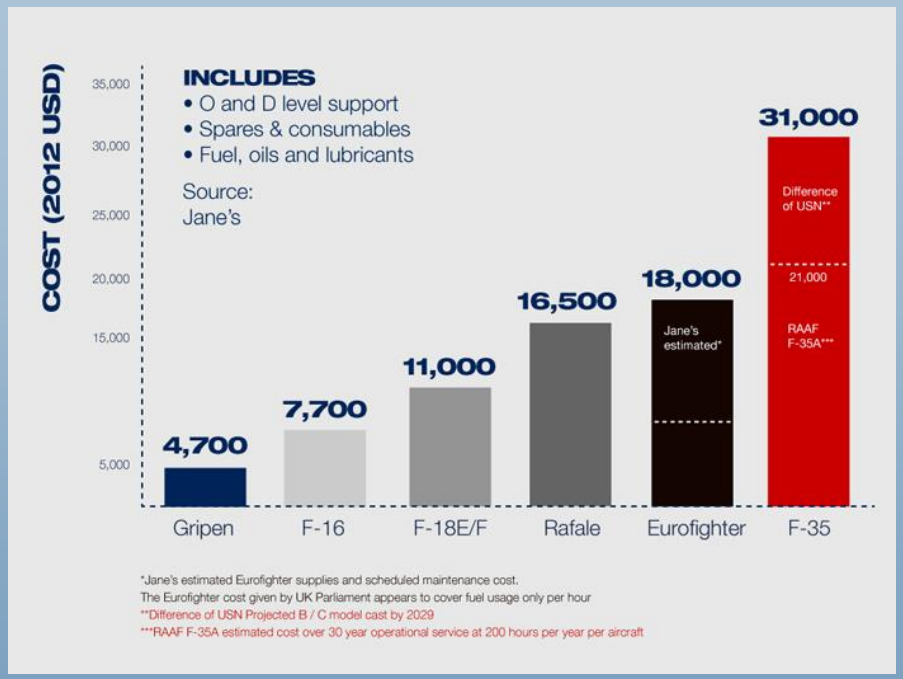
Limited Risk

System of Systems

Cost to Develop & Operate



Compared to Competitors



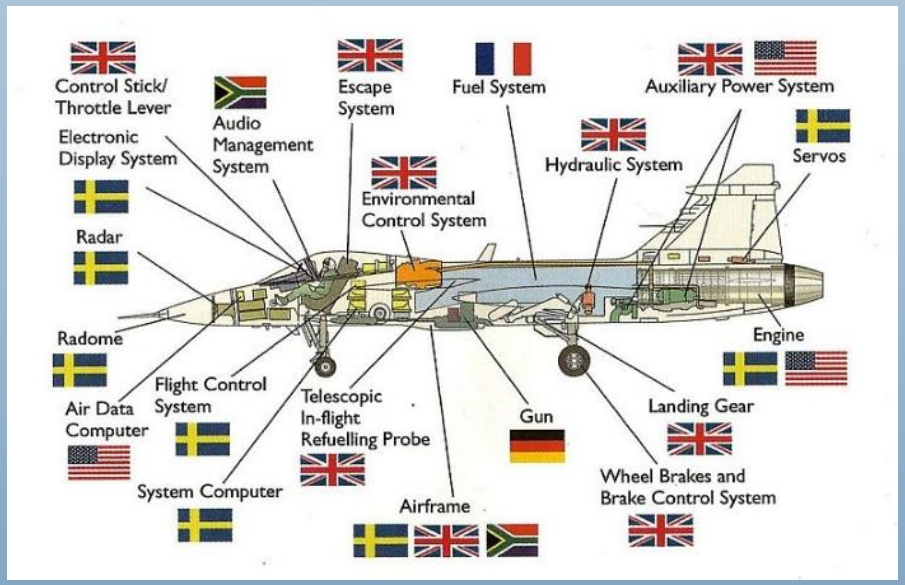
Design goals: 1) low cost, 2) high reliability, 3) highly maneuverability

Limited Risk

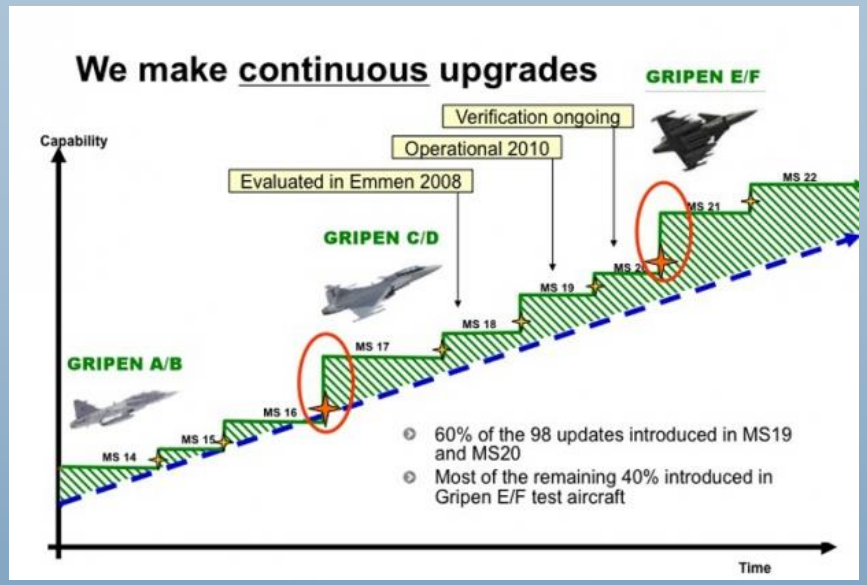
Low Friction

The Gripen

Integrated Federated Architecture



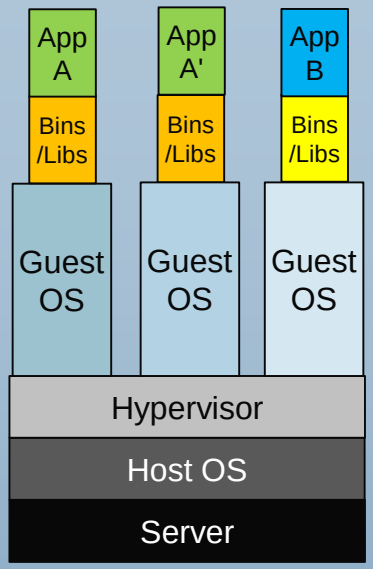
Rapid Development



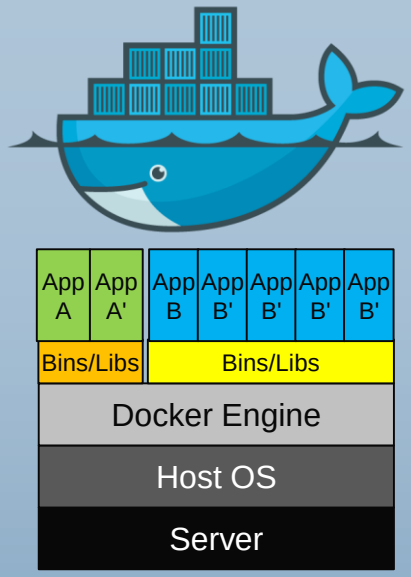
Limited Risk

Lower Friction

Strategies for Monoliths: Pack Dependent Code into Containers



Virtualization



Containers

Why Containers?

Portability
- ✓ Build once run anywhere

Consistency
- ✓ Configuration managed by containers at runtime

Isolation
- ✓ Hardened containers protect the contents

Easy of Use
- ✓ 1 sec start up
- ✓ 15 min learning curve

Utilization
- ✓ Many more apps/server

Low Friction

Limited Risk

Packing Containers

1. Start with an important module.
2. Find its [major] dependencies.
3. Package related code into a container.
4. Repeat, refactoring to keep containers small.

.

Strategies for Monoliths: One Thing We Know for Sure

For Complex Systems

This does not work

Smash!

Strategies for Monoliths: One Thing We Know for Sure

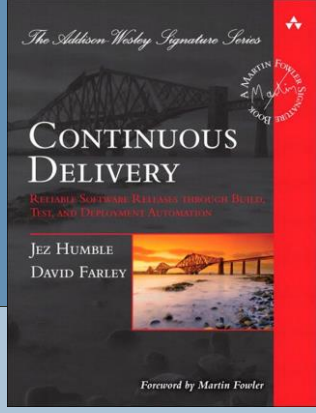
For Complex Systems

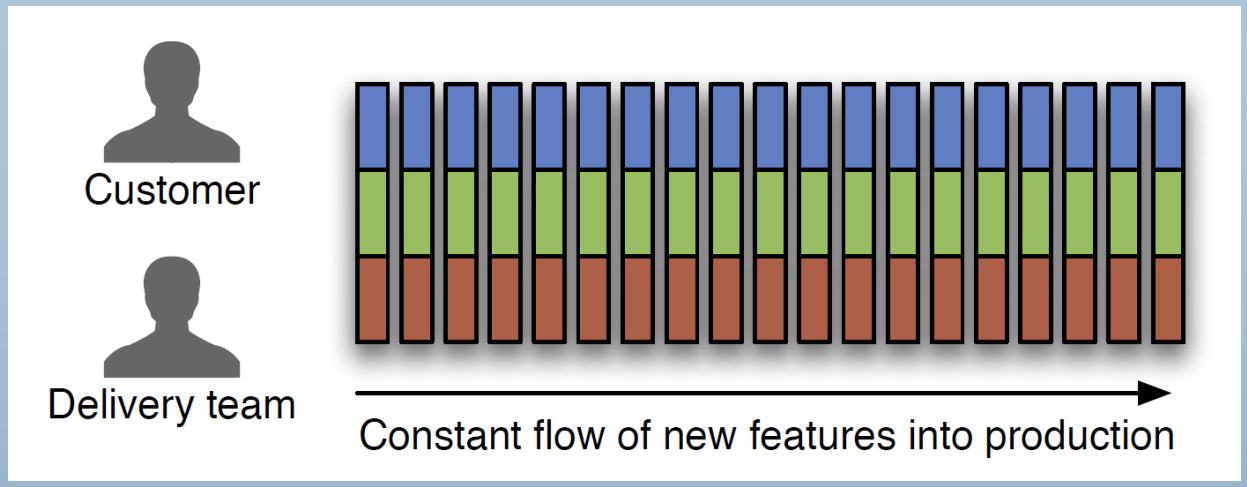
This works

Poke. Poke.

l e a n

Strategies for Monoliths: Use Continuous Delivery





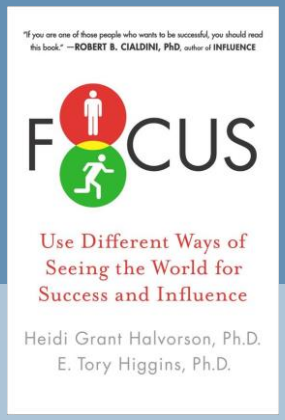
Low Friction

Limited Risk

- Acceptance test driven development process
- Tight collaboration between business and delivery teams
- Cross-functional teams include QA and operations
- Automated build, testing, db migration, and deployment
- Incremental development *on mainline* with continuous integration
- Software always production ready
- Releases tied to business needs, not operational constraints

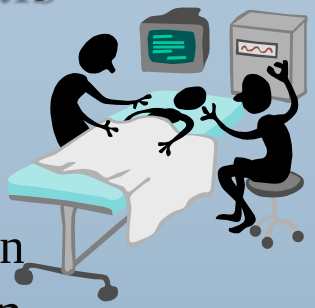
Credit: Jez Humble

Dev and Ops are Different



Regulatory Focus

Safety-Focused Goals (Prevention Focus)



- ✓ Prevent Failure
  - × Is it safe?
  - × Find the safest option
- ✓ Duty and Obligation
  - × Setbacks => redoubled efforts
  - × Praise => more relaxed efforts
- ✓ Rewards
  - × Attention is for bad behavior
  - × Nothing going wrong (no loss) is the ideal reward

Aspirational Goals (Promotion Focus)

- ✓ Create gains
  - × Let's do it!
  - × Explore all the options
- ✓ Aspirational Goals
  - × Praise => redoubled efforts
  - × Setbacks => discouragement
- ✓ Rewards
  - × Attention is for good behavior
  - × Gains are the ideal reward

Limit Risk

Lower Friction

One Goal
Shared Responsibility

Who is Responsible?

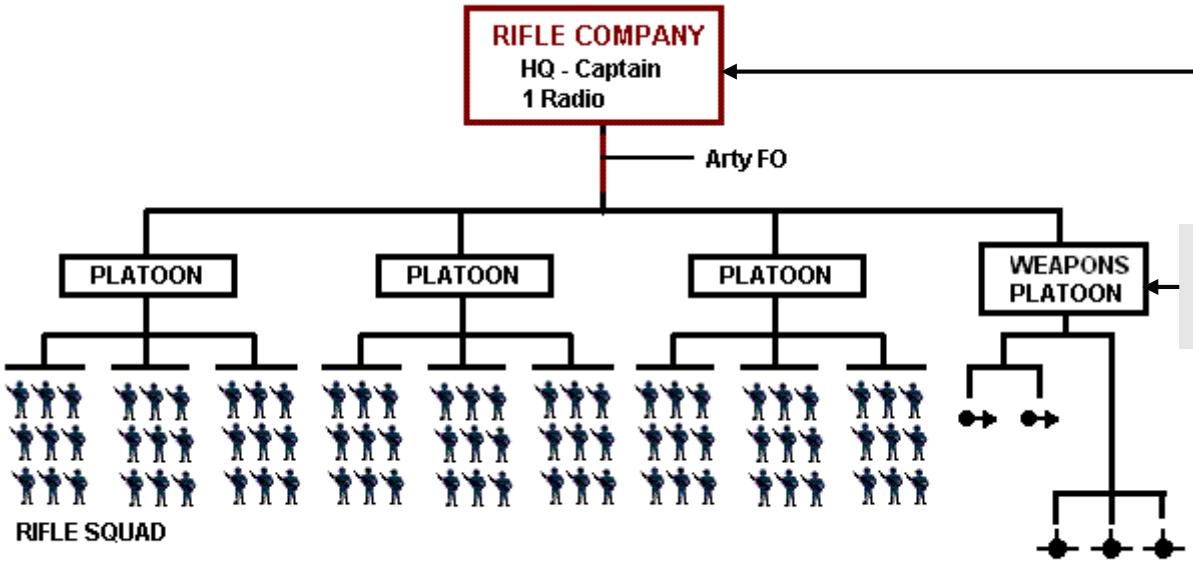
"The Business"
"The Product Guys"
"The Engineers"
"Operations"

Not Me

"We Work Together"
"Nobody Succeeds Unless Everyone Succeeds"

All of Us

The Military Model



Understand **Command Intent**
Two Levels Up

Maintain **Situational Awareness**
One Level Up

Command Intent:

A concise expression of the purpose of the campaign, the desired results, and the expected team progress toward achieving the desired end state.

1. Collaborative Planning
2. Situational awareness of the progress of other squads/platoons
3. Adapt to make sure the company reaches the end state

*Maneuverability**

The ability to gain, shed, or redirect momentum really fast. *Low Friction*

The ability to capitalize on agility with flawless execution. *Limited Risk*

The most maneuverable competitor wins.

* Thanks to Mike Nygard!
http://www.michaelnygard.com/blog/2015/04/manueverability/

l e a n

software development

Thank You!

For a copy of these slides, e-mail mary@poppendieck.com