



Lean Software Factories and Digital Transformation

Yves Caseau

Group CIO, Michelin

NATF (National Academy of Technologies of France)



<https://twitter.com/ycaseau>

<http://informationssystemsbiology.blogspot.com/>

JFTL - Beffroi de Montrouge
September 7th, 2021 - v1.0

Outline

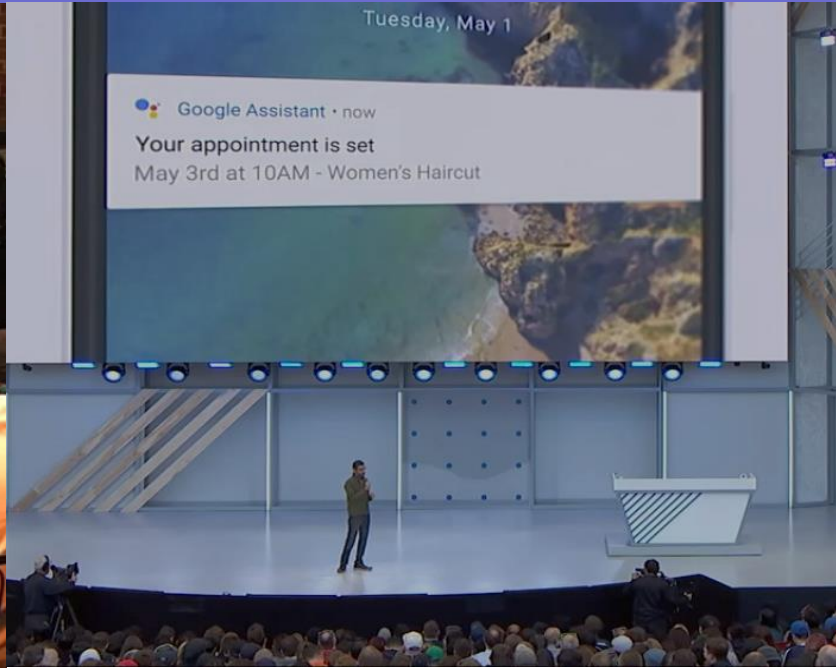
- Part 1 : Lean Digital Transformation
From customer to code, from code to customer
- Part 2: Software Factories
Automation & Excellence to Deliver Change and Quality
- Part 3: Software Craftmanship and Tests
Lean Discipline and Love for the Code

Digital Transformation



Markets are conversations

- In the world of content abundance, to grab attention, you must listen
- Sharing a conversation with the customer requires a content strategy



Digital Homeostasis

- Digital change is pushed from outside (customers and technology)
- Leverage vibrant service ecosystems to deliver the best experience.



Continuous Discovery

- Innovation is grown, not designed
- Minimum Viable Product : How to collect feedback as early as possible, but not earlier 😊

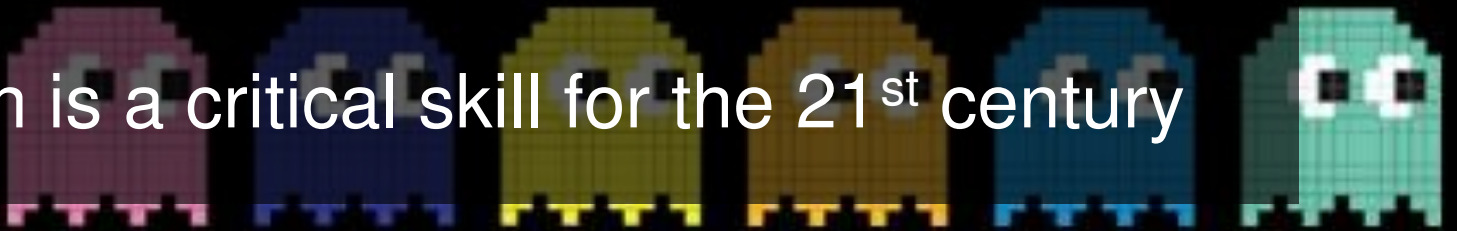
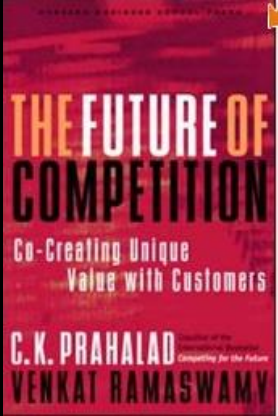
“Software is eating the world”

- “The customer is the architect of his experience” ...
Customers pick their (software) environment (B2C & B2B)

- User Experience Design is a critical skill for the 21st century

- The acceleration of change requires new & iterative ways

- ... iterative methods (e.g., Agile) produce technical debt



Exponential Information Systems

- ⦿ Customer-centric & outside-in thinking: constantly adapt to its environment
- ⦿ Leverage the best of exponential technologies (AI, ML, ...)
- ⦿ Architected for constant refresh
- ⦿ Antifragile : Organic growth from successive experiments

EXPONENTIAL
ORGANIZATIONS

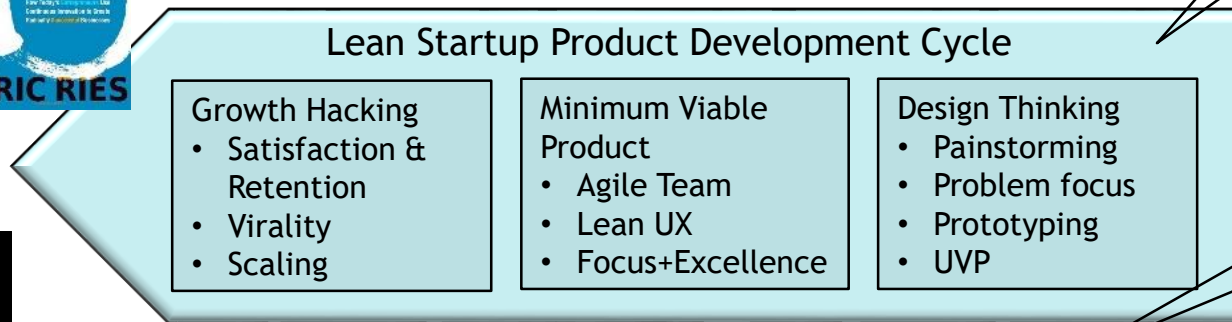
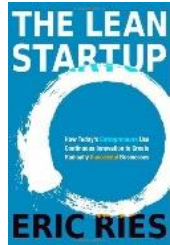
Why new organizations are ten times
better, faster, and cheaper than yours
(and what to do about it)

SALIM ISMAIL

WITH MICHAEL S. MALONE & YURI VAN GEEST
FOREWORD BY ANDREW W. LO, TERRY H. CHURCHMAN

© 2013 MIT PRESS

From Customer to Code, From Code to Customer



Embedded cycles, not linear processes

Cross-functional teams from two processes



Customer

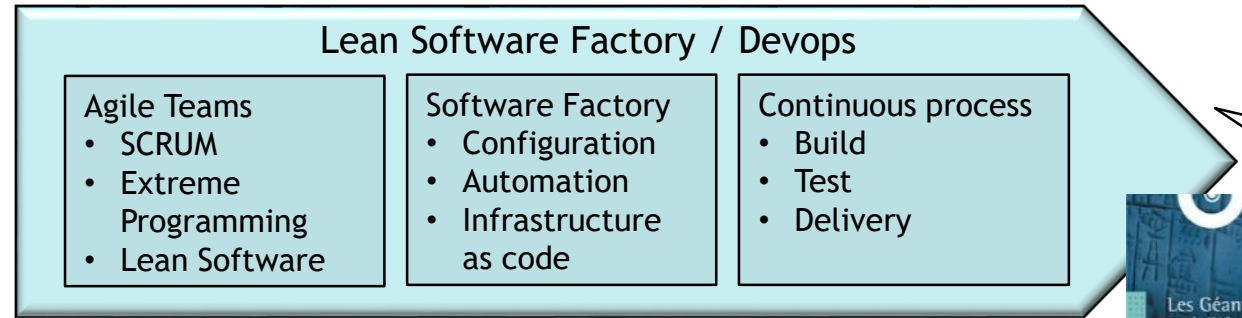
```

// Size propagation rule
(R1) :- rule( c.value = v =>
  store(c.column_counts,v,0),
  store(c.square_counts,v,0),
  for c2 in ( 1..9 )
  ( if (v != v2 & c.possible(v2)) noLarger(v2),
    for c2 in (c.line_cells but c2.forbidden(v),
    for c2 in (c.column_cells but c2.forbidden(v),
    for c2 in (c.square_cells but c2.forbidden(v)))

// If a count = 5, the only possible value is corner
(R2) :- rule( c.count = y & y = 1 =>
  c.value = someq in ( 1..9 ) | c.possible(q))

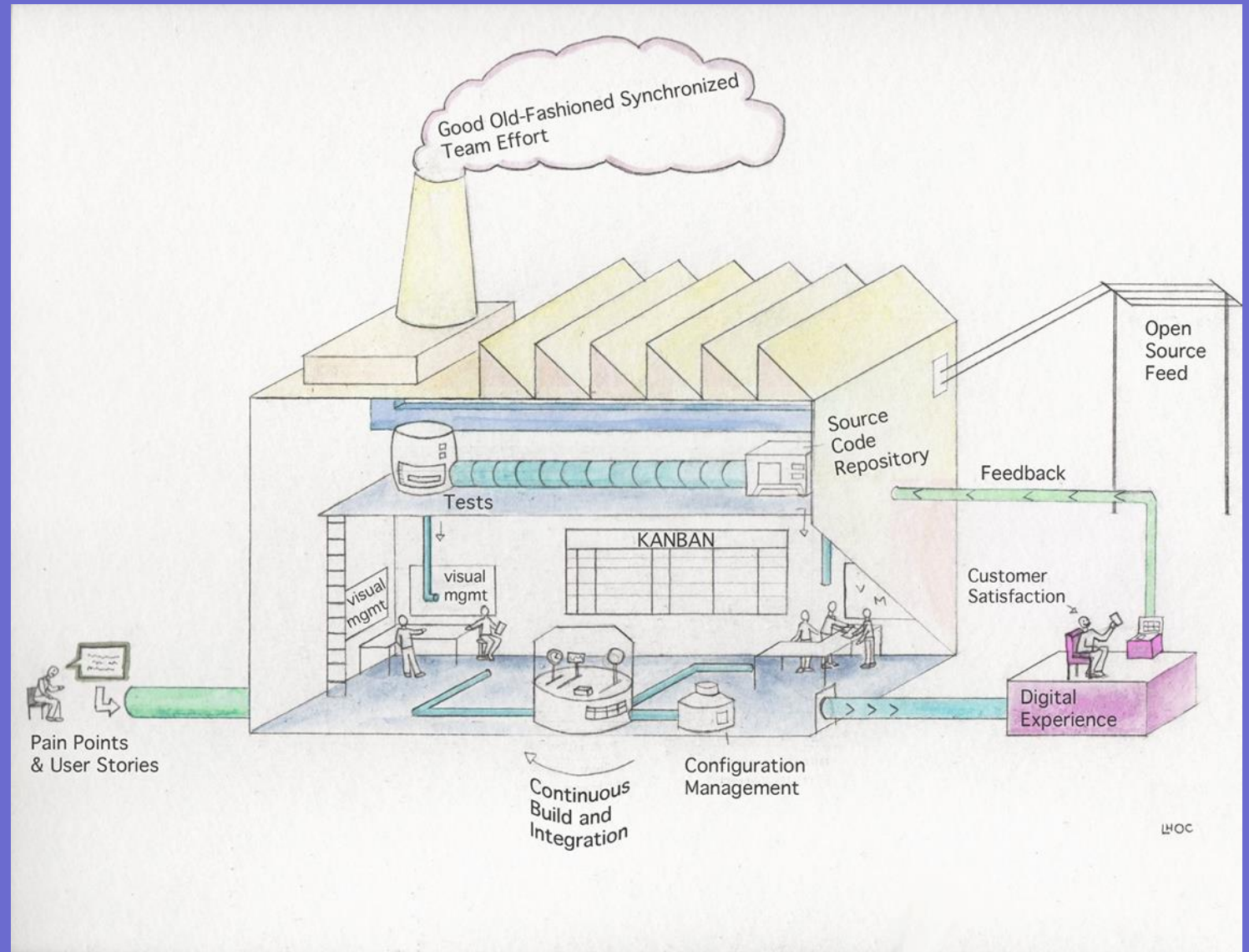
// If a value v is possible only in one cell, it is certain
(R3) :- rule( updateCount(c,v) & c.counts[v] = 1
=> when c = someq in (c.cells |
  c.value = 0 & c.possible(v)) in
  c.value = v
  else contradiction))
    
```

(software) code

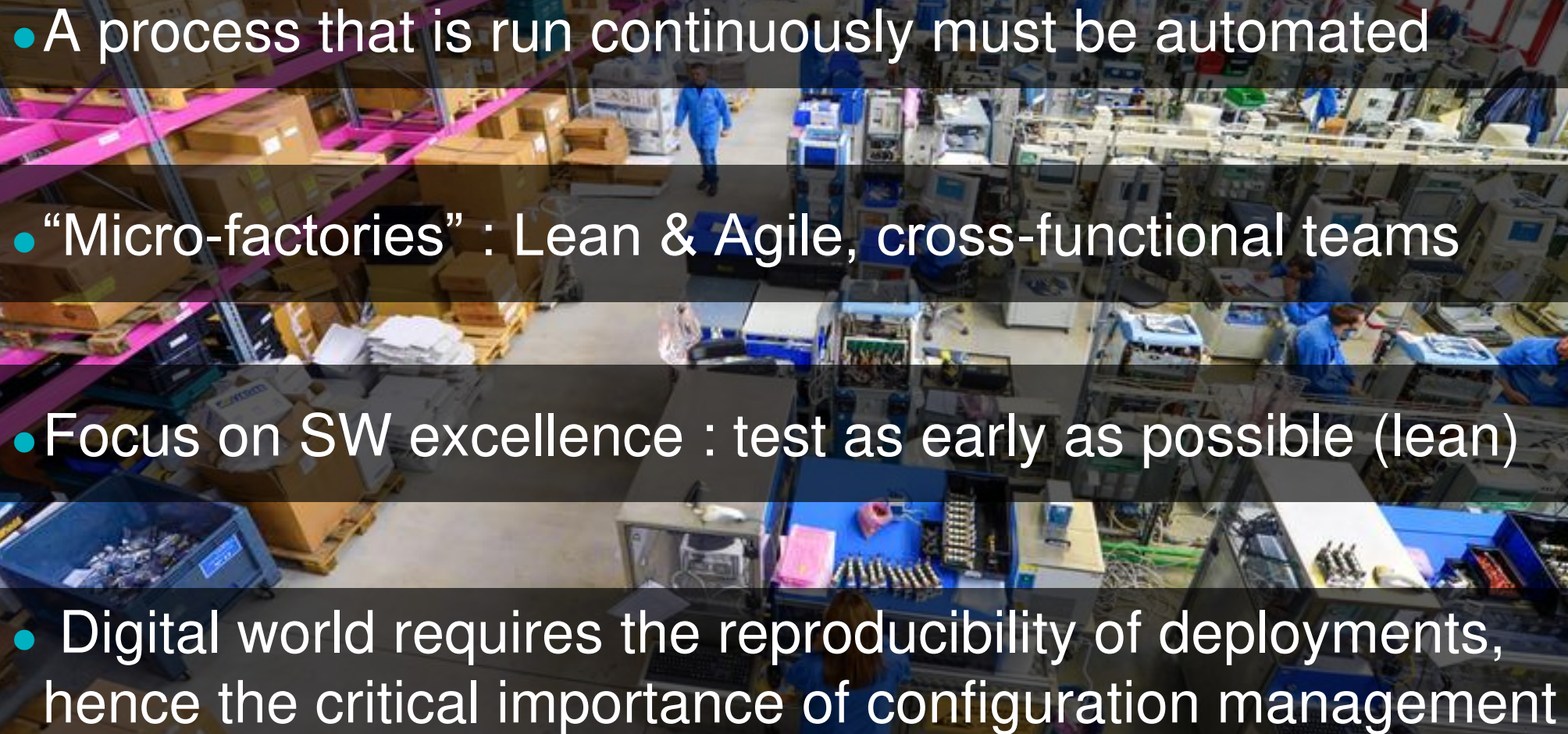


Common artefacts: UVP, User Stories and Product

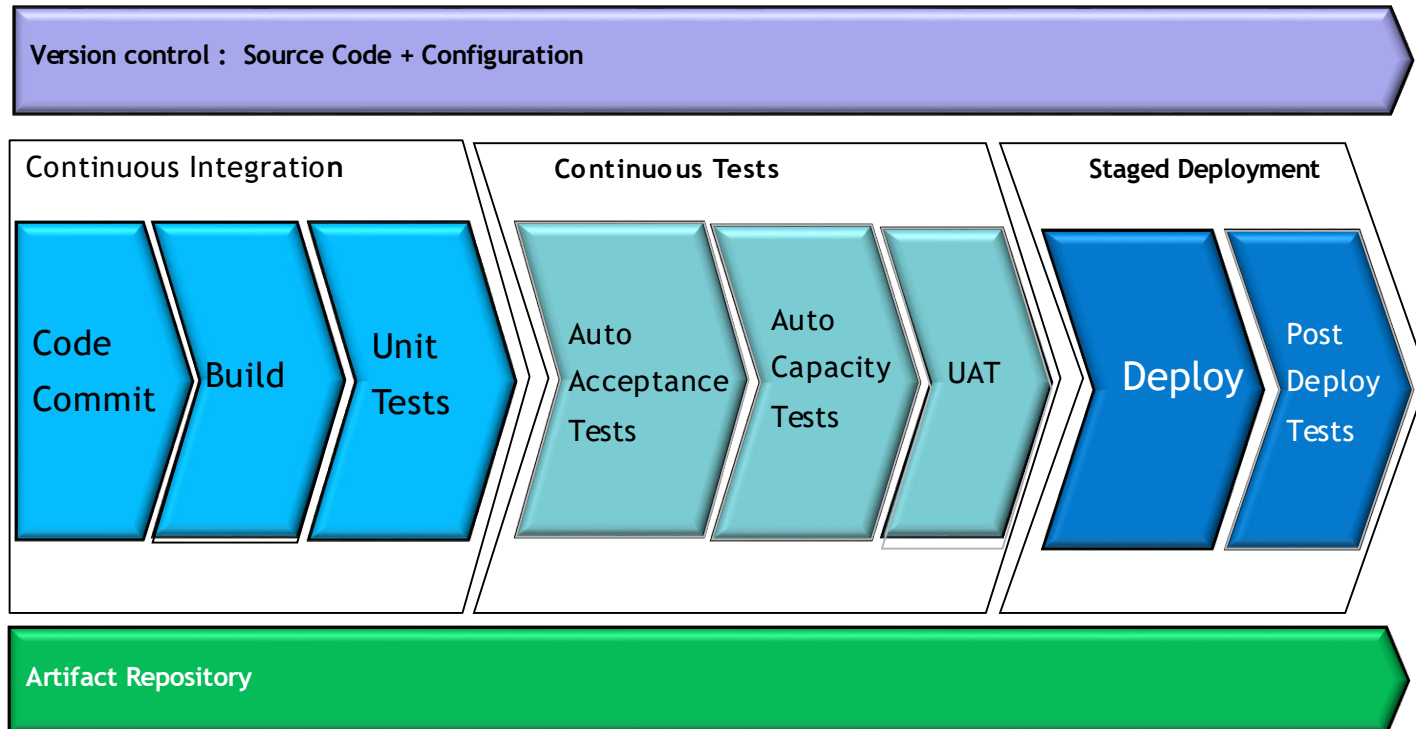
Part II Software Factories



“Factories”: Focus on Discipline & Automation

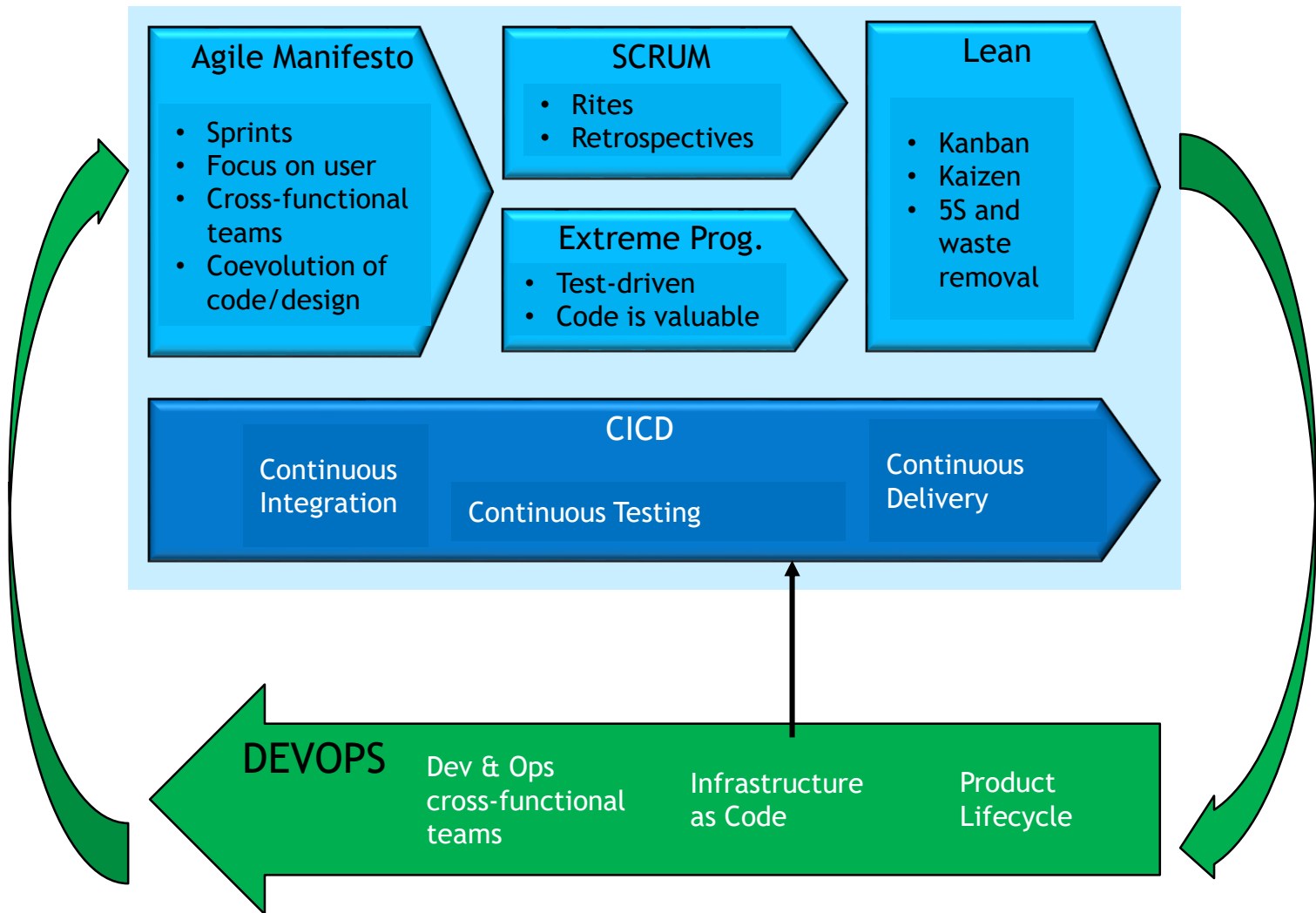
- A process that is run continuously must be automated
 - “Micro-factories” : Lean & Agile, cross-functional teams
 - Focus on SW excellence : test as early as possible (lean)
 - Digital world requires the reproducibility of deployments, hence the critical importance of configuration management
- 
- A high-angle, wide shot of a large industrial factory floor. The space is filled with various pieces of machinery, workstations, and equipment. Several workers in blue uniforms are visible, engaged in tasks. On the left side, there are tall shelving units with numerous cardboard boxes stacked on them. The floor is a light-colored concrete, and the overall atmosphere is one of a busy, organized manufacturing environment.

CICD Pipeline



- **Test as early as possible**
Do not let the integration debt grow
- **Automation** to reduce takt time and to improve quality (CICD & Testing) –
“If the pipeline is nor working well, run it more frequently and highlight the difficulties” (Jezz Humble)
- Short steps and automated rollbacks, from build to delivery
- **Automation requires industrial configuration management – Tools matters !**

Lean Software Factories



- **Lean & Agile:** short-term delivery of small value increments, long-term iterative learning
- **Lean thinking :** continuous management of technical debt to develop « situation potential » (tomorrow's agility)
- **Lean practices :** right on the first time, continuous learning through kaizen
- **Product mode :** short-term delivery of small value increments, long-term iterative learning

The Role of Lean Architecture

- ⦿ Architects on-demand
- ⦿ System engineering coach and champion
- ⦿ Refactoring champion - promotes TD cleaning in backlog
- ⦿ API interface and “integration grammar” owner

Part III
**Software
Craftmanship**



Software Craftmanship – Embedded Agility

- “Show & Share” : Develop and value software excellence through peer reviews
- Make Code Reviews more pleasurable and more efficient: Coding standards and Pair-programming
- “Love your code” : code elegance as a support for business agility (code that one likes to modify)
- Elegant code is reusable code : open-source culture & communities (e.g., Stack Overflow)

Testing Lifecycle



Build

- Unit tests
- Some acceptance tests, performance tests
- Test automation is the name of the game (long list of benefits 😊)



Delivery

- User Acceptance Tests
- Some global system integration tests
- Stress capacity tests, reversibility tests



Post Deployment

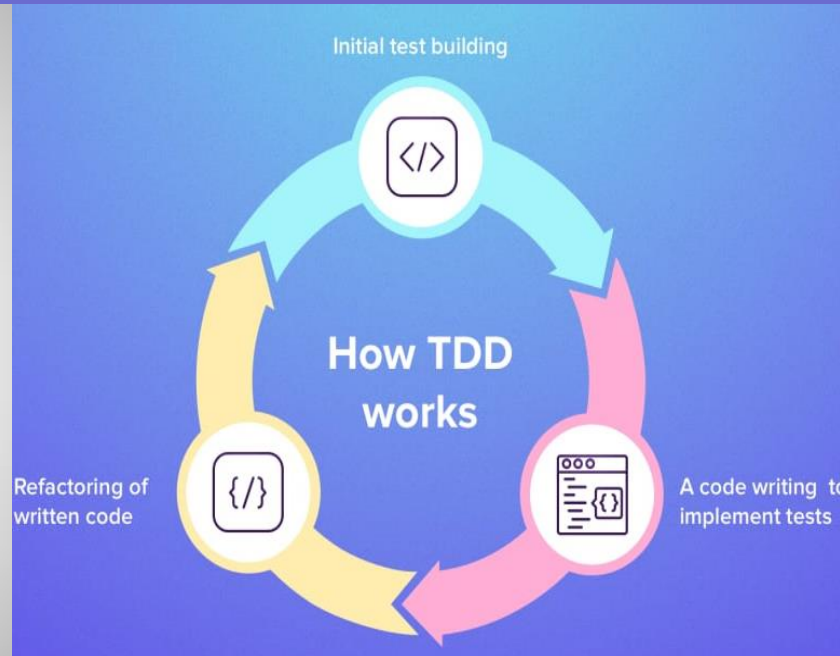
- Error Analytics : detect errors before users
- AI4Ops : Predictive Maintenance Smart monitoring
- Product Discovery (e.g. A/B testing) – continuous UI testing

Lean Software Standards



Five S

- **Sort:** reduce code base, eliminate dead code, refactor
- **Systematize:** organize, code standards
- **Shine:** pass quality tests, clean up, code reviews, improve test coverage rates
- **Standardize :** continuously improve the teams' standard
- **Sustain :** Make software excellence a habit



Test-Driven Development

- Write Unit Tests first
- Relate tests to user stories
- Treat unit tests with same care as source code



Visual Management

- Voice of Customer
Always on display for developers
- System Engineering :
“See how the system works”
- Kanban : limit WIP

Digital Era Testing Practices

```
// finds a cell with a min count (heuristic)
```

```
findPivot(g:Grid) : any
```

```
-> let minv := 10, cmin := unknown in
```

```
(for c in g.cells |>
```

```
  (if (c.value = 0 & c.count < minv)
```

• Test generation and randomization
One of the untold benefits of process models & term algebra

```
    (minv := c.count, cmin := c),
```

```
  cmin)
```

```
// solves a sudoku : branch on possible
```

```
// value of a cell is 0, then branch on possible
```

```
// branch on possible
```

```
solve(g:Grid) : boolean
```

```
-> when (g != null & g != null)
```

```
  exists(v in (1 .. 9) |
```

```
    (if c.possible[v]
```

```
      branch((c.value := v
```

```
        solve(g)))
```

```
  else true
```

```
// first propagation rule
```

```
r1() :: rule( c.value := v =>
```

```
  (store(c.line.counts,v,0),
```

```
    store(c.column.counts,v,0),
```

```
    store(c.square.counts,v,0),
```

```
    (if (v != v2 & c.possible[v2]) noLonger(c,v2),
```

```
      for c2 in (c.line.cells but c) forbid(c2,v),
```

```
      for c2 in (c.column.cells but c) forbid(c2,v),
```

```
      for c2 in (c.square.cells but c) forbid(c2,v))))
```

```
// if c.count = 1, the only possible value is certain
```

```
r2() :: rule( c.count = 1 =>
```

```
  c.value := some(y in (1 .. 9) | c.possible[y]))
```

```
// if a value is possible only in one cell, it is certain
```

```
r3() :: rule( updateCount(cs,v) & cs.count[v] <= 1
```

```
  & cs.count[v] == 1 & cs.cells |
```

```
  (c in cs.cells |
```

```
    c.value = 0 & c.possible[v]) in
```

```
  c.value := v
```

```
  else contradiction!())
```

• Capacity Planning models and performance testing

AI4ITOps : train ML from performance logs

Conclusion

- *Software is eating the world ...* the expected level of experience excellence has changed in the digital era
- Lean Software Factories : combining Lean & Agile with DevOps and Product approach
- “*From customer to code, from code to customer*” a *lean* mindset and culture, based on tools, discipline and software craftsmanship

