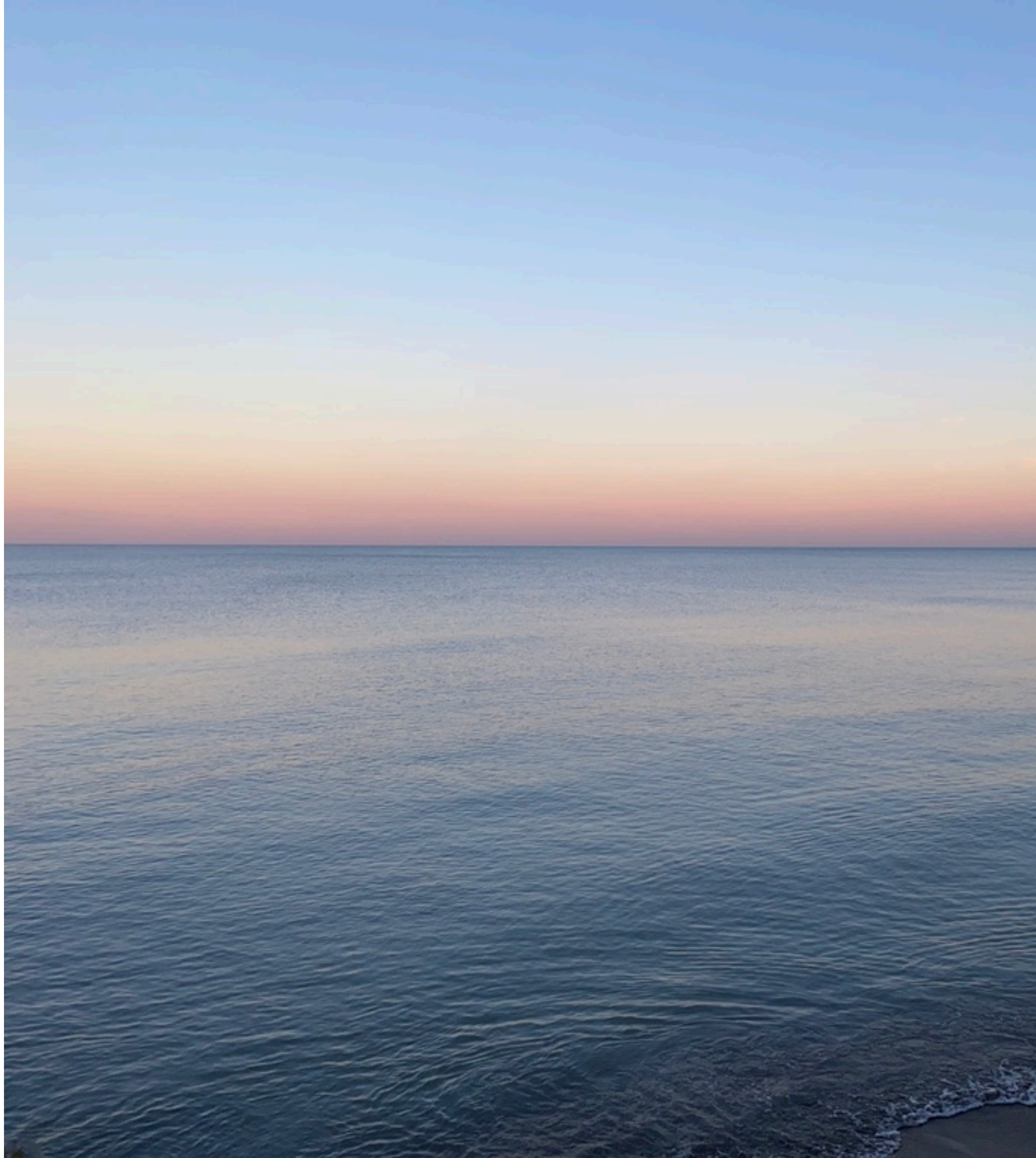
# Wardley Doctrine Patterns

Germán Flores 2021-02-12

# Agenda

* Doctrine patterns
* Team growth initiative
* Facilitation and blockers



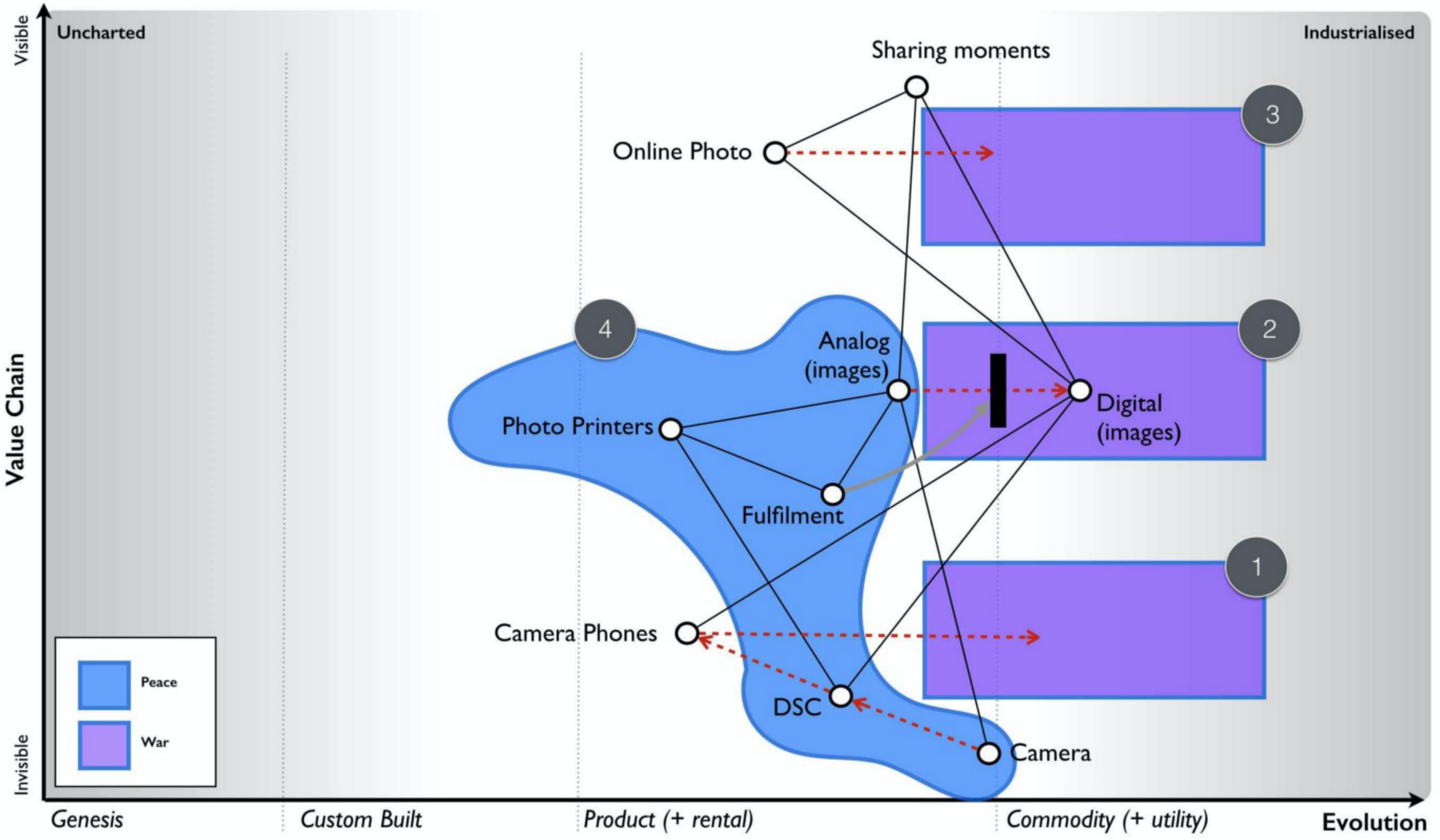
# Wardley Doctrine Patterns

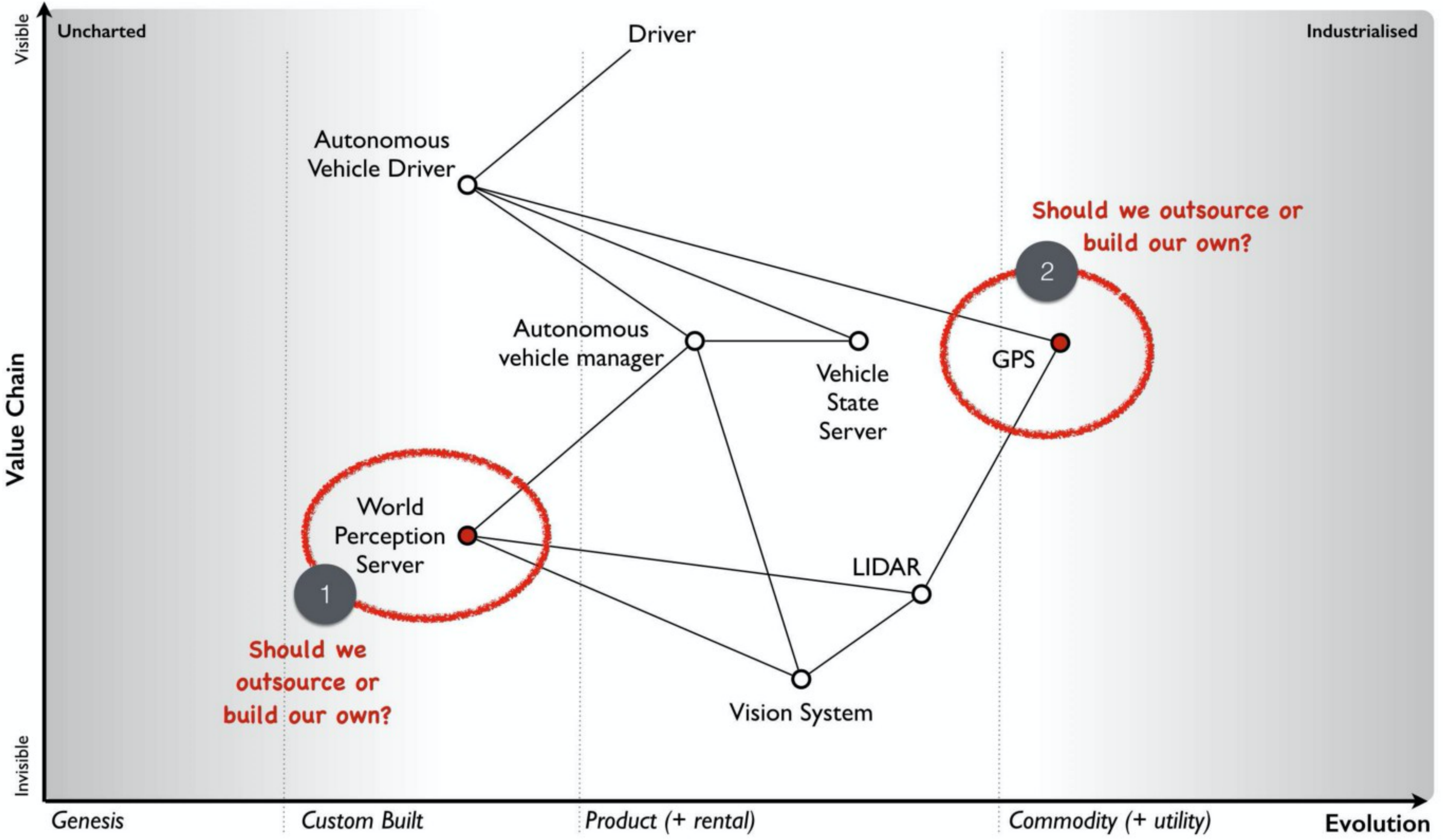
“Crossing the river by feeling the stones.”

Deng Xiaoping

# Doctrine Patterns

- Doctrine patterns come from Wardley Mapping by Simon Wardley
- A planning tool


Visible
Uncharted
Industrialised
Sharing moments
Online Photo
3
4
Analog
(images)
2
Digital
(images)
Photo Printers
Fulfilment
1
Camera Phones
DSC
Camera
Peace
War
Invisible
Value Chain
Genesis
Custom Built
Product (+ rental)
Commodity (+ utility)
Evolution


Visible
Uncharted
Driver
Industrialised
Autonomous
Vehicle Driver
Should we outsource or
build our own?
2
Autonomous
vehicle manager
Vehicle
State
Server
GPS
Value Chain
World
Perception
Server
1
LIDAR
Should we
outsource or
build our own?
Vision System
Invisible
Genesis
Custom Built
Product (+ rental)
Commodity (+ utility)
Evolution

“The map is not the territory.”

# Doctrine Patterns

- Doctrine, a list of universally useful **principles**.
- Patterns can be used as a checklist to **evaluate** an organization
- Each doctrine is a skill or habit, so the real goal is to **train** the organization to repeatedly use each skill.
- **Change management** instrument

| TECH GIANT | **Wardley's Doctrine** (universally useful patterns that a user can apply regardless of context) | | | |
|---|---|---|---|---|
| *Communication* | Be transparent *(a bias towards open)* | Focus on high situational awareness *(understand what is being considered)* | Use a common language *(necessary for collaboration)* | Challenge assumptions *(speak up and question)* |
| *Development* | Know your users *(e.g. customers, shareholders, regulators, staff)* | Focus on user needs | Think fast, inexpensive, restrained and elegant (FIRE, formerly FIST) | Remove bias and duplication |
| | Use appropriate methods *(e.g. agile vs lean vs six sigma)* | Focus on the outcome not a contract *(e.g. worth based development)* | Be pragmatic *(it doesn't matter if the cat is black or white as long as it catches mice)* | Use standards where appropriate |
| | Use appropriate tools *(e.g. mapping, financial models)* | | | |
| *Operation* | Manage inertia *(e.g. existing practice, political capital, previous investment)* | Optimise flow *(remove bottlenecks)* | Think small *(as in know the details)* | Effectiveness over efficiency |
| | Do better with less *(continual improvement)* | Set exceptional standards *(great is just not good enough)* | Manage failure | |
| *Structure* | Provide purpose, mastery & autonomy | Think small *(as in teams, "two pizza")* | Distribute power and decision making | Think aptitude and attitude |
| | Design for constant evolution | There is no one culture *(e.g. pioneers, settlers and town planners)* | Seek the best | |
| *Learning* | Use a systematic mechanism of learning *(a bias towards data)* | A bias towards action *(learn by playing the game)* | A bias towards the new *(be curious, take appropriate risks)* | Listen to your ecosystems *(acts as future sensing engines)* |
| *Leading* | Be the owner *(take responsibility)* | Move fast *(an imperfect plan executed today is better than a perfect plan executed tomorrow)* | Think big *(inspire others, provide direction)* | Strategy is iterative not linear *(fast reactive cycles)* |
| | Strategy is complex *(there will be uncertainty)* | Commit to the direction, be adaptive along the path *(crossing the river by feeling the stones)* | There is no core *(everything is transient)* | Be humble *(listen, be selfless, have fortitude)* |
| | Exploit the landscape | | | |

Good
Neutral / unknown
Weak
Warning

**BANKING GIANT**

**Wardley's Doctrine** (universally useful patterns that a user can apply regardless of context)

| | | | | |
|---|---|---|---|---|
| *Communication* | Be transparent *(a bias towards open)* | Focus on high situational awareness *(understand what is being considered)* | Use a common language *(necessary for collaboration)* | Challenge assumptions *(speak up and question)* |
| *Development* | Know your users *(e.g. customers, shareholders, regulators, staff)* | Focus on user needs | Think fast, inexpensive, restrained and elegant (FIRE, formerly FIST) | Remove bias and duplication |
| | Use appropriate methods *(e.g. agile vs lean vs six sigma)* | Focus on the outcome not a contract *(e.g. worth based development)* | Be pragmatic *(it doesn't matter if the cat is black or white as long as it catches mice)* | Use standards where appropriate |
| | Use appropriate tools *(e.g. mapping, financial models)* | | | |
| *Operation* | Manage inertia *(e.g. existing practice, political capital, previous investment)* | Optimise flow *(remove bottlenecks)* | Think small *(as in know the details)* | Effectiveness over efficiency |
| | Do better with less *(continual improvement)* | Set exceptional standards *(great is just not good enough)* | Manage failure | |
| *Structure* | Provide purpose, mastery & autonomy | Think small *(as in teams, "two pizza")* | Distribute power and decision making | Think aptitude and attitude |
| | Design for constant evolution | There is no one culture *(e.g. pioneers, settlers and town planners)* | Seek the best | |
| *Learning* | Use a systematic mechanism of learning *(a bias towards data)* | A bias towards action *(learn by playing the game)* | A bias towards the new *(be curious, take appropriate risks)* | Listen to your ecosystems *(acts as future sensing engines)* |
| *Leading* | Be the owner *(take responsibility)* | Move fast *(an imperfect plan executed today is better than a perfect plan executed tomorrow)* | Think big *(inspire others, provide direction)* | Strategy is iterative not linear *(fast reactive cycles)* |
| | Strategy is complex *(there will be uncertainty)* | Commit to the direction, be adaptive along the path *(crossing the river by feeling the stones)* | There is no core *(everything is transient)* | Be humble *(listen, be selfless, have fortitude)* |
| | Exploit the landscape | | | |

Good

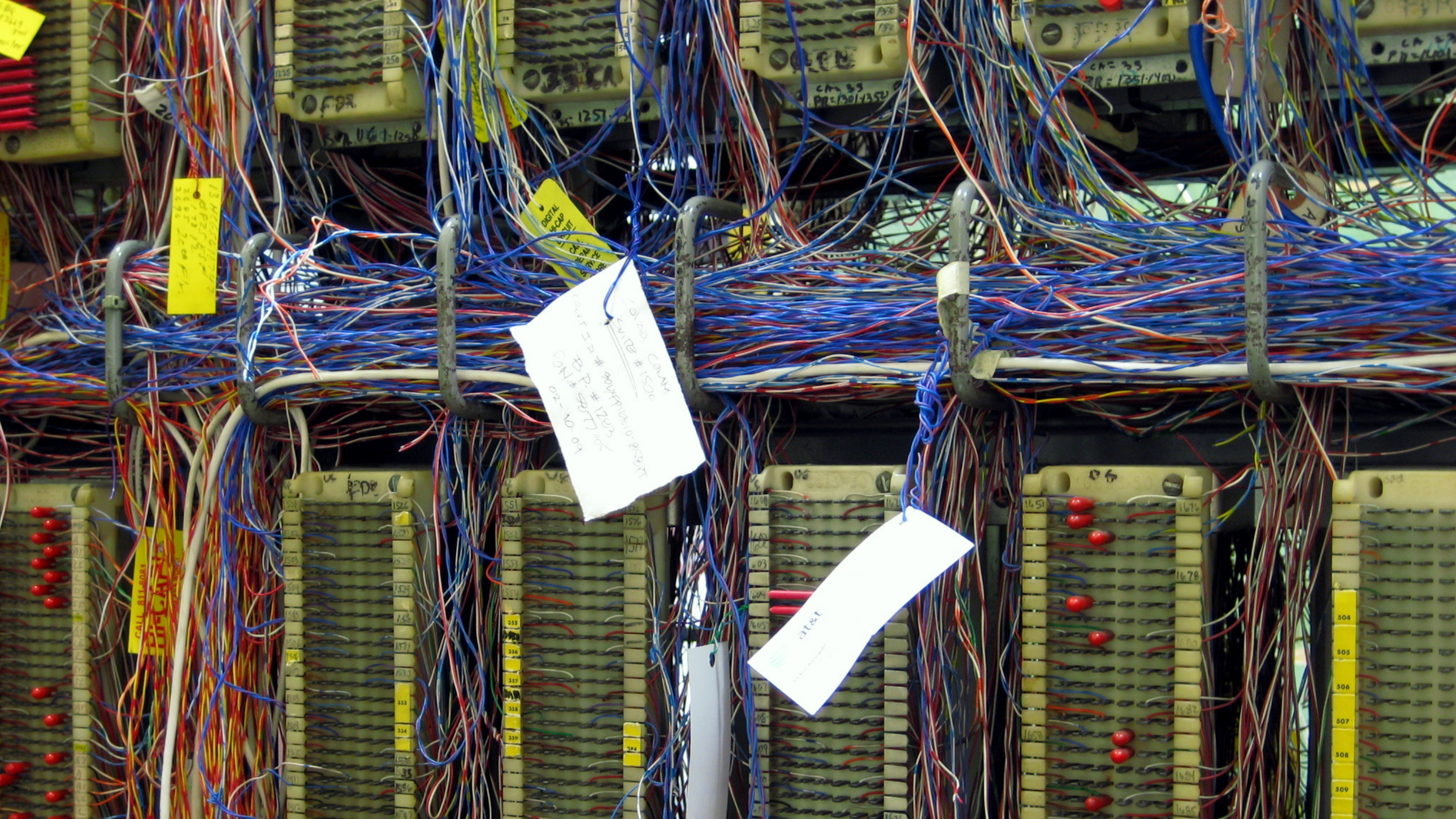
Neutral / unknown

Weak

Warning

| | **Wardley's Doctrine** (universally useful patterns that a user can apply regardless of context) | | | | | |
|---|---|---|---|---|---|---|
| | *Communication* | *Development* | *Operation* | *Learning* | *Leading* | *Structure* |
| *IV* | | | | Listen to your ecosystem | Exploit the landscape | Design for constant evolution |
| | | | | | There is no core | No single culture |
| *III* | | | Optimise flow | *Bias towards the new* | Commit to the direction | Provide purpose, mastery & autonomy |
| | | | Do better with less | | Be the owner | |
| | | | | | Inspire others | Seek the best |
| | | | Set exceptional standards | | Embrace uncertainty | |
| | | | | | Be humble | |
| *II* | | Focus on the outcome | Manage inertia | Bias towards action | Move fast | Think small teams |
| | | Think fast, inexpensive, restrained and elegant | | | | |
| | | Use appropriate tools | Manage failure | | Strategy is iterative | Distribute power and decision making |
| | | Be pragmatic | | | | |
| | | Use standards | Effectiveness over efficiency | | | Think aptitude and attitude |
| | A bias towards open | | | | | |
| *Phase I* | Common Language | Know your users | Know the details | Bias towards data | *STEVE PURKIS VARIATION | |
| | Challenge Assumptions | Focus on user needs | | | | |
| | Understand what is being considered | Remove bias and duplication | | | | |
| | | Use appropriate methods | | | | |

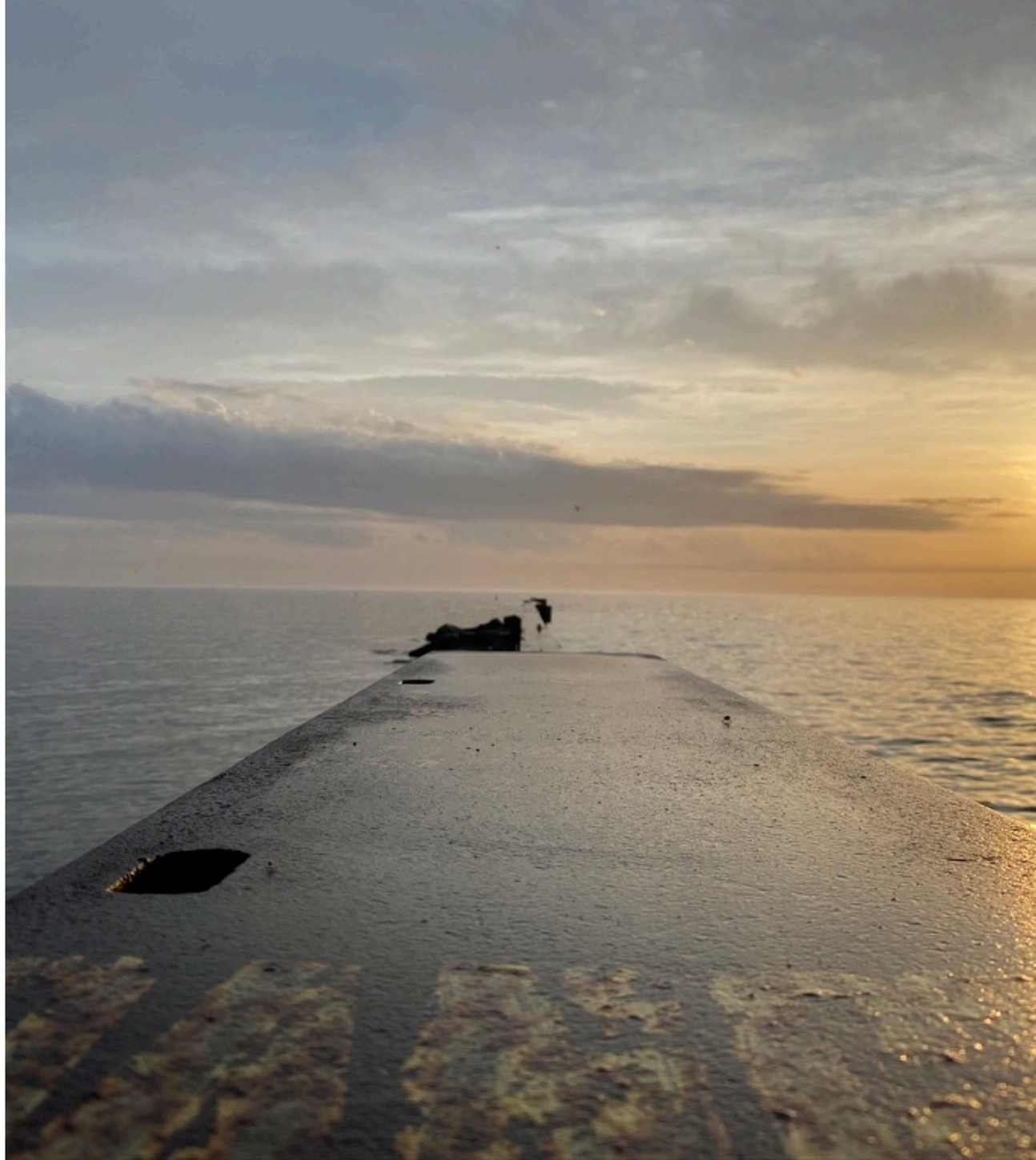
# Team Growth Initiative

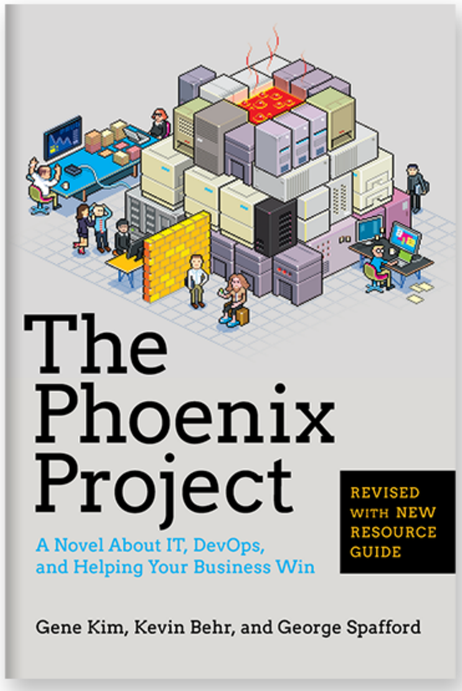

DIGITAL
HI-CAP
CALL R11-0001
at&t

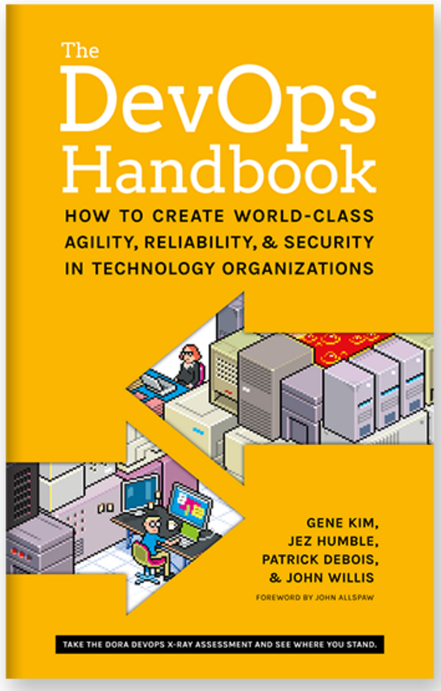
YELLOW
YELLOW
YELLOW
YELLOW
YELLOW
ORANGE
ORANGE
ORANGE
ORANGE
ORANGE
ORANGE
ORANGE

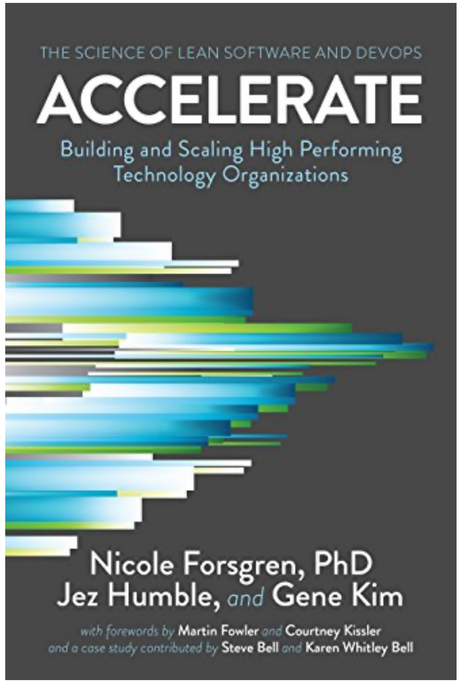
# Background

- Clarity over the **problem** we are trying to solve.
- **Concerted** approach to facilitate **growth** in our team.
- Set of activities and **metrics** to enable and facilitate growth.



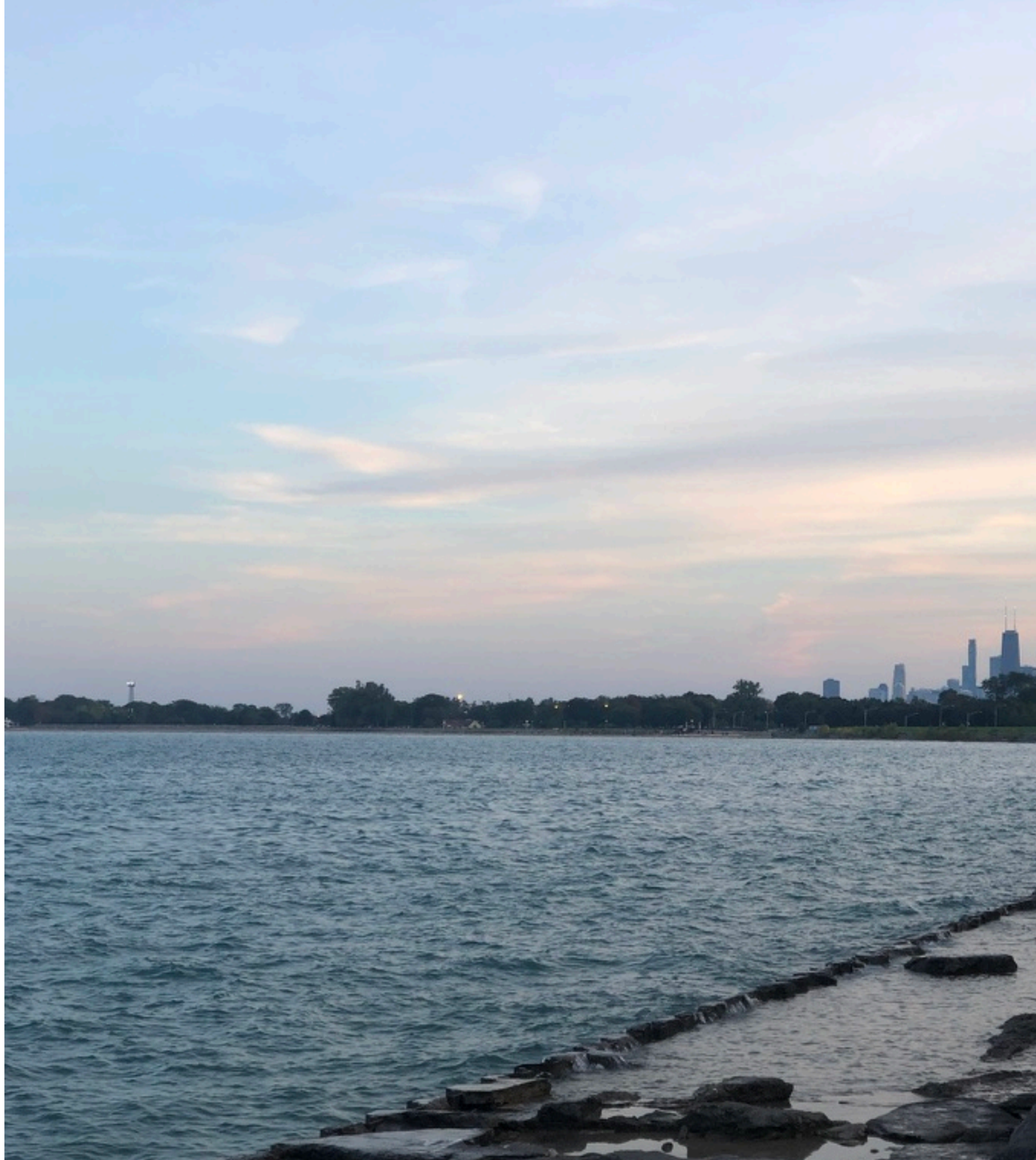

The Phoenix Project
REVISED WITH NEW RESOURCE GUIDE
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford


The DevOps Handbook
HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS
GENE KIM, JEZ HUMBLE, PATRICK DEBOIS, & JOHN WILLIS
FOREWORD BY JOHN ALLSPAW
TAKE THE DORA DEVOPS X-RAY ASSESSMENT AND SEE WHERE YOU STAND.


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

# Measures

- Lead time (LT)
  - Story to Production and work start to deployment
- Deployment rate (DR)
  - How frequently we deploy
- Deployment failure rate (FR)
- Time to repair (MTTR)



| | Technical Competence | Communication | Product | Culture | Systems/Capabilities | Process | Curiosity | Ownership | Managers |
|---|---|---|---|---|---|---|---|---|---|
| Phase 0 (current) | Technical Leadership from Management | SCQA's (situation, complication, questions, answers) | Technical competence from platform Product Owners | Open communication from technology management | Average of 7 deploys per week | Project management | Articles about new technologies are shared | Technical debt payments | |
| | Test coverage in code is encouraged | Incident Management | | | Lead time is between 1week and 1 month | Incident Management | | Swarming to solve problems | |
| | Automated testing in parts of platform | A3's | | | | Emerging | | | |
| | Change failure rate is x-y% | | | | | Scrum | | | |
| | Leadership from Tech Leads | | | | | Scope negotiation | | | |
| Phase I | Everyone in the technology department has read the DevOps Handbook. Reading the book is managed via a goal | Technical Diagrams | Align priority to goals and targets | Management promotes organizational learning | We understand rate of system failures and bugs per team | We use velocity to forecast capacity and release completion | New ideas and ways of working are shared | Developers are adding stories to the backlog to improve the system | |
| | Technical Leads start to work on aligning their current skills in relation to the developer competencies matrix | Communication happens in open channels | Organize release candidate list and ranking with management | Focus on quality, protect teams to ensure quality | Lead time is between 1-20 days (request to deploy) | Release process is understood by management, product, and technology | | | Developer competency matrix exists, managers understand it |
| | Technical Leads understand how to use a competency matrix based on experience and coaching from their managers | | A platform roadmap exists | | We have maximized efficiencies in the old platform (to make it least costly to make changes) | | | | |
| Phase II | Use Incidents to drive change and improvements | Challenge assumptions | Product department has read the Accelerate book. The book reading is managed via a goal. | | Focus on building capability not maturity or competence | Provide teams with slack for improvement and innovation | New technologies are introduced, evaluated, and adopted | Escalate cross-functional and systemic problems | Actively monitor and visualize performance to goals/targets |
| | Developer competencies are rolled-out to the department | Documentation is consistently maintained | Focus on quality, protect teams to ensure quality | | Average 15 deploys per week, from \trunk based development, new platform, test automation | Optimize flow (look for bottlenecks) | Coach team members and support team learning | | Coach managers, have your own coach |
| | Technical Leads can introduce the developer competency matrix to their direct reports and effectively work on growing them | | Break demand into small elements (MVP's) and release regularly and often | | Lead time is between 1-14 days (request to deploy) | Release process is consistently leveraged to deliver | | | Leverage the developer competency matrix to grow people |
| | Technical skills gap in the department is understood | | | | | | | | Do performance management |
| Phase III | As a department we are consistently using the developer competency matrix to grow people and manage individual performance | Default to open communications | Encourage teams with time for improvement and innovation | A bias towards open | | We understand our lead time (request to deployed) | New ideas are shared | Eliminate unnecessary controls, invest instead in process quality and team autonomy and capability | |
| | Apply disciplined problem solving to complex systemic issues to identify strategic improvement themes and targets (strategy deployment), apply learning to update standard work | | | Best ideas win | | Budget for and allocate time for creativity | New ways of working are introduced | | |
| | Technical Leads are viewed as leaders and can work across the organization to move and idea to a shipped state | | | Developers continuously try to improve the approach to software development and deployment | | | Willingness to experiment with new ways of thinking and working | | |
| Phase IV | Design for constant evolution | | | | Deploys on demand | We have a process to get work done and everybody understands it | Bias towards the new | | |

| | Technical Competence | Communication | Product | Culture | Systems/Capabilities | Process | Curiosity | Ownership | Managers |
|---|---|---|---|---|---|---|---|---|---|
| **Phase 0 (current)** | | | | | | | | | |
| | Technical Leadership from Management | SCQA's (situation, complication, questions, answers) | Technical competence from platform Product Owners | Open communication from technology management | Average of 7 deploys per week | Project management | Articles about new technologies are shared | Technical debt payments | |
| | Test coverage in code is encouraged | Incident Management | | | Lead time is between 1week and 1 month | Incident Management | | Swarming to solve problems | |
| | Automated testing in parts of platform | A3's | | | | **Emerging** | | | |
| | Change failure rate is x-y% | | | | | Scrum | | | |
| | Leadership from Tech Leads | | | | | Scope negotiation | | | |

| | Technical Competence | Communication | Product | Culture | Systems/Capabilities | Process | Curiosity | Ownership | Managers |
|---|---|---|---|---|---|---|---|---|---|
| **Phase I** | Everyone in the technology department has read the DevOps Handbook. Reading the book is managed via a goal | Technical Diagrams | Align priority to goals and targets | Management promotes organizational learning | We understand rate of system failures and bugs per team | We use velocity to forecast capacity and release completion | New ideas and ways of working are shared | Developers are adding stories to the backlog to improve the system | |
| | Technical Leads start to work on aligning their current skills in relation to the developer competencies matrix | Communication happens in open channels | Organize release candidate list and ranking with management | Focus on quality, protect teams to ensure quality | Lead time is between 1-20 days (request to deploy) | Release process is understood by management, product, and technology | | | Developer competency matrix exists, managers understand it |
| | Technical Leads understand how to use a competency matrix based on experience and coaching from their managers | | A platform roadmap exists | | We have maximized efficiencies in the old platform (to make it least costly to make changes) | | | | |

| | Technical Competence | Communication | Product | Culture | Systems/Capabilities | Process | Curiosity | Ownership | Managers |
|---|---|---|---|---|---|---|---|---|---|
| **Phase II** | Use Incidents to drive change and improvements | Challenge assumptions | Product department has read the Accelerate book. The book reading is managed via a goal. | | Focus on building capability not maturity or competence | Provide teams with slack for improvement and innovation | New technologies are introduced, evaluated, and adopted | Escalate cross-functional and systemic problems | Actively monitor and visualize performance to goals/targets |
| | Developer competencies are rolled-out to the department | Documentation is consistently maintained | Focus on quality, protect teams to ensure quality | | Average 15 deploys per week, from \trunk based development, new platform, test automation | Optimize flow (look for bottlenecks) | Coach team members and support team learning | | Coach managers, have your own coach |
| | Technical Leads can introduce the developer competency matrix to their direct reports and effectively work on growing them | | Break demand into small elements (MVP's) and release regularly and often | | Lead time is between 1-14 days (request to deploy) | Release process is consistently leveraged to deliver | | | Leverage the developer competency matrix to grow people |
| | Technical skills gap in the department is understood | | | | | | | | Do performance management |

| | Technical Competence | Communication | Product | Culture | Systems/Capabilities | Process | Curiosity | Ownership | Managers |
|---|---|---|---|---|---|---|---|---|---|
| **Phase III** | As a department we are consistently using the developer competency matrix to grow people and manage individual performance | Default to open communications | Encourage teams with time for improvement and innovation | A bias towards open | | We understand our lead time (request to deployed) | New ideas are shared | Eliminate unnecessary controls, invest instead in process quality and team autonomy and capability | |
| | Apply disciplined problem solving to complex systemic issues to identify strategic improvement themes and targets (strategy deployment), apply learning to update standard work | | | Best ideas win | | Budget for and allocate time for creativity | New ways of working are introduced | | |
| | Technical Leads are viewed as leaders and can work across the organization to move and idea to a shipped state | | | Developers continuously try to improve the approach to software development and deployment | | | Willingness to experiment with new ways of thinking and working | | |
| Phase IV | Design for constant evolution | | | | Deploys on demand | We have a process to get work done and everybody understands it | Bias towards the new | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Facilitation & Blockers

# Conditions

- Leadership onboard
- Managers onboard
- Make it measurable

# Facilitation

- Align
  - Top > Down
  - Down > Up
- Teach
- Enable them to own it
- Go/No-Go or not yet

Questions,
your opinions and experiences.

# Resources & Credits

* https://medium.com/wardleymaps/on-being-lost-2ef5f05eb1ec
* https://wardleypedia.org/mediawiki/index.php/Doctrine_Patterns
* http://doctrine.wardleymaps.com
* https://list.wardleymaps.com
* https://justin.stach.uk/doctrine_grid_tool/
* https://twitter.com/spurkis/status/1187730682589659136/photo/1
* https://itrevolution.com/devops-books/
* https://swardley.medium.com/what-culture-is-right-for-you-ba892f1f3bc5
* https://miro.medium.com/max/2000/1*eKRNJKYmO2gMim6QlW2dEQ.jpeg
* https://flic.kr/p/4rUkRs
* https://flic.kr/p/6fNxm3
* https://twitter.com/swardley/status/1357636351521357826/photo/1

# Wardley Doctrine Patterns

| | Technical Competence | Communication | Product | Culture | Systems/Capabilities | Process | Curiosity | Ownership | Managers |
|---|---|---|---|---|---|---|---|---|---|
| Phase 0 (current) | Technical Leadership from Management | SCQA's (situation, complication, questions, answers) | Technical competence from platform Product Owners | Open communication from technology management | Average of 7 deploys per week | Project management | Articles about new technologies are shared | Technical debt payments | |
| | Test coverage in code is encouraged | Incident Management | | | Lead time is between 1week and 1 month | Incident Management | | Swarming to solve problems | |
| | Automated testing in parts of platform | A3's | | | | **Emerging** | | | |
| | Change failure rate is x-y% | | | | | Scrum | | | |
| | Leadership from Tech Leads | | | | | Scope negotiation | | | |
| Phase I | Everyone in the technology department has read the DevOps Handbook. Reading the book is managed via a goal | Technical Diagrams | Align priority to goals and targets | Management promotes organizational learning | We understand rate of system failures and bugs per team | We use velocity to forecast capacity and release completion | New ideas and ways of working are shared | Developers are adding stories to the backlog to improve the system | |
| | Technical Leads start to work on aligning their current skills in relation to the developer competencies matrix | Communication happens in open channels | Organize release candidate list and ranking with management | Focus on quality, protect teams to ensure quality | Lead time is between 1-20 days (request to deploy) | Release process is understood by management, product, and technology | | | Developer competency matrix exists, managers understand it |
| | Technical Leads understand how to use a competency matrix based on experience and coaching from their managers | | A platform roadmap exists | | We have maximized efficiencies in the old platform (to make it least costly to make changes) | | | | |
| Phase II | Use Incidents to drive change and improvements | Challenge assumptions | Product department has read the Accelerate book. The book reading is managed via a goal. | | Focus on building capability not maturity or competence | Provide teams with slack for improvement and innovation | New technologies are introduced, evaluated, and adopted | Escalate cross-functional and systemic problems | Actively monitor and visualize performance to goals/targets |
| | Developer competencies are rolled-out to the department | Documentation is consistently maintained | Focus on quality, protect teams to ensure quality | | Average 15 deploys per week, from \trunk based development, new platform, test automation | Optimize flow (look for bottlenecks) | Coach team members and support team learning | | Coach managers, have your own coach |
| | Technical Leads can introduce the developer competency matrix to their direct reports and effectively work on growing them | | Break demand into small elements (MVP's) and release regularly and often | | Lead time is between 1-14 days (request to deploy) | Release process is consistently leveraged to deliver | | | Leverage the developer competency matrix to grow people |
| | Technical skills gap in the department is understood | | | | | | | | Do performance management |
| Phase III | As a department we are consistently using the developer competency matrix to grow people and manage individual performance | Default to open communications | Encourage teams with time for improvement and innovation | A bias towards open | | We understand our lead time (request to deployed) | New ideas are shared | Eliminate unnecessary controls, invest instead in process quality and team autonomy and capability | |
| | Apply disciplined problem solving to complex systemic issues to identify strategic improvement themes and targets (strategy deployment), apply learning to update standard work | | | Best ideas win | | Budget for and allocate time for creativity | New ways of working are introduced | | |
| | Technical Leads are viewed as leaders and can work across the organization to move and idea to a shipped state | | | Developers continuously try to improve the approach to software development and deployment | | | Willingness to experiment with new ways of thinking and working | | |
| Phase IV | Design for constant evolution | | | | Deploys on demand | We have a process to get work done and everybody understands it | Bias towards the new | | |