Project: UnifiedDevOps OpenCoding ITE2 (master)

Report created by Jessica Diaz on 12/04/2022

Code Report – Grouped by: Code Groups

All (21) codes

AUTOMATION

2 Codes:

- **automated (continuous) everything**

  FROM build automation; testing automation; delivery automation; deployment automation; operational tasks automation (continuous feedback from operations to development) TO non-automation. Tools adoption or tool providing for deployment pipeline, monitoring, security, etc. Not to be confused with platform servicing; it only provides tools and pipelines for automated delivery/deployment in certain contexts. 07/03/2022 1:48:19, merged with automated (continuous) delivery 07/03/2022 1:48:19, merged with automated (continuous) deployment 07/03/2022 1:48:19, merged with automated (continuous) integration 07/03/2022 1:48:19, merged with automated (continuous) operation 07/03/2022 1:48:19, merged with automated (continuous) testing 12/03/2022 18:04:09, merged with tools adoption/providing

- **platform servicing**

  This code refers to the services offered or provided by an entity (e.g., a team, an external consultant, etc.) on DevOps platform, such as infrastructure automation (from low to high levels of automated infrastructure services aka. Infrastructure as Code, IaC), IT operation, CI/CD and release tools, etc.

CULTURE

4 Codes:

- **collaboration**

  From eventual collaboration, which may generate conflicts and disagreements on decisions, to daily collaboration (working together regularly on a daily basis).

- **communication**

  From poor/rare communication (and standardization) to frequent communication.

- **cultural silos/conflicts**

  From non-cultural barriers to existing cultural barriers. Not to be confused with organizational/silos conflicts. While the cultural silos/conflicts focus on practices and culture that leads/break to barriers, organization silos/conflicts focus on team structures.

- **values & best practices**

  This code refers to a generic concept to be used when a quotation explicitly refers to this generality such as DevOps best/good practices (continuous integration, continuous testing, continuous delivery and deployment, infrastructure as code, continuous monitoring, etc.), cultural values (collaboration, communication, transparency, etc.), principles (customer-centric action, create with the end in mind, end-to-end responsibility, cross-functional autonomous teams, continuous improvement, automate everything you can, among others). If a quotation refers to a specific practice, value, and principle, specific codes should be used.

MANAGEMENT

4 Codes:

- **leadership & management**

  From high to low levels of leadership, which may involve single to multiple managers 07/03/2022 2:02:26, merged with single-to-multiple management

- **rotary human resources**

  DevOps engineers are involved in product teams with exclusive dedication but limited in time, until product teams are capable of doing all their responsibilities

- **team self-organization & autonomy**

  From external and/or bureaucratic dependencies and approvals to high levels of team self-organization and autonomy (i.e., the team has freedom and autonomy to organize its tasks, budget, infrastructure, etc.). Alignment with the Amazon motto ‘‘You built it, you run it’’. 07/03/2022 0:24:35, merged with autonomy 07/03/2022 1:57:10, merged with external dependencies

- **transfer of work between teams**

  Transfer of work and responsibilities between teams (e.g., between development and infrastructure/operation teams). Hence, the definition of development "done" may go from committing a change to running the change in production-like environments. When there is transfer of work, depending on the high/poor trust/confidence on the other team, it may generate stress in the teams.

MONITORING

2 Codes:

- **delivery performance**

  From high (in short, high performers) to low delivery performance (deployment frequency, mean-time to recovery, lead time). Sometimes, deployment/delivery frequency is limited to external factors, such as periodic time slots (windows release).

- **end-to-end product vision**

  From waterfall and process-oriented models where each unit or individual works only for a particular role/function to overseeing the complete picture (focus on building working products so all employees need to share the engineering mindset that is required to envision and realize those products).

SHARING

3 Codes:

- **alignment of dev & ops goals**

  From misalignment among teams that have own interests and goals (local optimization) to alignment with business goals and global ones. A typical example is "developers want to deliver as much as possible, whereas operations target stability".

- **knowledge sharing**

  From high to low levels of knowledge sharing (e.g., developers may have knowledge about infrastructure/platform, or minimal or no awareness of what is happening on the other side of the wall, aka. wall of confusion).

- **responsibility/ownership sharing**

  From shared responsibility of the product (e.g., NFR shared responsibility, monitoring, and incident handling shared responsibility, etc.) and output artifacts (e.g., databases) to separated responsibilities and tasks (developers, infrastructure/staff and operators have different responsibilities and tasks). Thus, if there is no shared responsibility, there is necessarily a transfer of work development to production and operation (and vice versa). However, ownership sharing is related to a new definition of "done" (e.g., developers work doesn't finish with coding, but they support deployment in production).
  04/03/2022 0:08:18, merged with NFR shared responsibility

SKILLS & ROLES

3 Codes:

- **cross-functionality/skills**

  From multidisciplinary/poly-skilled teams (i.e., teams with all the necessary skills such as development, infrastructure, etc.) to teams with a lack of skills/knowledge/background. This can be addressed by product teams with their own infrastructure staff/engineers.
  13/03/2022 0:13:47, merged with need for dedicated infra engineers

- **evangelization and mentoring**

  DevOps evangelization and mentoring

- **role definition/attributions**

From "skills over roles" and T-shape engineers (aka. full stack engineers or devops engineers) to well-defined and differentiated roles (e.g., dev versus ops, so that they work together but in different tasks). Approaches with well-defined and differentiated roles may decrease collaboration and promote a transfer of responsibilities; or there can be collaboration and avoid conflicts over who is responsible for each task 02/03/2022 12:48:04, merged with well-defined and differentiated roles 07/03/2022 2:08:43, merged with T-shape engineers

---

STRUCTURE

3 Codes:

- **horizontal (platform) teams**

  This code refers to specific structures/teams that organizations create to satisfy some needs of product teams such as platform servicing, tools, consulting, evangelization, mentoring, human resources, etc. Thus, they behave as enabler teams by providing capabilities. These structures/teams are named in different ways, e.g., DevOps Centers of Excellence, chapters, guilds, platform teams, etc. This code should be used when a quotation explicitly mentions these new structures/teams or when a quotation describes the capabilities (at least three) of these enabler teams. 03/03/2022 23:59:06, merged with DevOps Center of Excellence (CoE) 03/03/2022 23:59:06, merged with DevOps Chapter 03/03/2022 23:59:06, merged with DevOps Platform Team

- **organizational silos/conflicts**

  From non-organizational silos/barriers to siloed departments and existing organizational barriers (segregated departments; frictions, conflicts, and disagreements among silos; silos that become bottlenecks; minimal or no awareness of what is happening on the other side of the wall).

- **small size teams (two pizza rule)**

  Agile team size (to promote communication over documentation).