Project: UnifiedDevOpsTeamStructures

Report created by Jessica Diaz on 07/03/2022

Code Report – Grouped by: Code Groups

All (23) codes

AUTOMATION

3 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. 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*

- **platform servicing**

  > *DevOps platform services, for example: infrastructure automation (from low to high levels of automated infrastructure services aka. Infrastructure as Code, IaC), IT operation, CI/CD and release tools and platforms, etc.*

- **tools adoption/providing**

  > *Not to be confused with platform servicing; it only provides tools and pipelines for automated delivery/deployment in certain contexts.*

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*

- **values & best practices**

  > *Best practices: continuous integration, continuous testing, continuous delivery and deployment, infrastructure as code, and continuous monitoring. Cultural values: collaboration, communication, transparency...*

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*

- **self-organization & autonomy**

  > *From external and/or bureaucratic dependencies and approvals to high levels of 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. 04/03/2022 0:08:18, merged with NFR shared responsibility*

MONITORING

2 Codes:

- **delivery performance**

  > *From high 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**

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 (developers, infrastructure/staff and operators have different responsibilities and tasks). Ownership sharing is also 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

4 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*

- **evangelization and mentoring**

  > *DevOps evangelization and mentoring*

- **need for dedicated infra engineers**

  > *Need or not need for product teams to have infrastructure engineers*

- **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 (cross) teams**

  > *DevOps Centers of Excellence, chapters, guilds, platform teams, etc. 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)**