A description of team topologies in the development process about their goals, tasks, interaction modes, and dynamics

Dipl.- Ing. Stefan Peil

Zur Eiche 14

Weimar (Lahn) 35096 Germany)

stefan.peil@stpse.de

Summary

In the modern software development process, the organization of teams is a key success factor. The concept of team topologies provides a structured model for organizing teams to work together efficiently and respond to business requirements. This approach defines specific team types, modes of interaction, and team APIs to clarify roles, tasks, and responsibilities. The key concepts of team topologies are described below: Team Types, Interaction Modes, and Team APIs, as well as their goals and tasks with real-world examples.

1. Team types

The team topologies classify teams into four main types, each of which pursues specific tasks and goals:

1. Stream-aligned Team (SA)

- **Goal:** End-to-end functionality along a specific value stream (e.g., a product or service).
- **Tasks:**
  - Development and operation of functions and features.
  - Direct interaction with stakeholders to understand requirements.
  - Ensuring a high-quality and fast release cycle.

Example: A stream-aligned team would be responsible for the functional development of a mobile app.

2. Enabling team (ET)

- **Goal: To** support other teams through advice, training and the contribution of specific expertise.
- **Tasks:**
  - Communication of cross-industry best practices.
  - Removal of technical obstacles.
  - Building skills in other teams.

Example: A team specializing in Continuous Integration and Continuous Delivery (CI/CD) and supports other teams with implementation.

3. Complicated Subsystem Team (CT)

- **Goal:** Solving problems or developing highly complex solutions that require in-depth specialist knowledge.
- **Tasks:**
    - Focus on technical challenges like algorithms, data processing, and security mechanisms.
    - Provision of specific subsystems for other teams.

Example: A team responsible for developing a scalable machine learning model.

4. Platform Team (PT)

- **Goal: To** provide digitalization strategies, platforms, and tools that make work easier for other teams and increase productivity.
- **Tasks:**
    - Development and maintenance of an internal platform (e.g., cloud environment).
    - Digitization and optimization of processes for stream-aligned teams.
    - Ensuring the reusability of tools and services.

Example: A team that develops a development platform on which stream-aligned teams run their applications.

2. Interaction modes

Interaction between teams takes place in three main modes to optimize collaboration and knowledge transfer:

1. Collaboration:

- Working together on a task for a limited period of time.
- Example: An enabling team collaborates with a stream-aligned team to implement a new monitoring solution.

2. Facilitation (support)

- Provision of guidelines or expertise without direct responsibility for implementation.

Example: A platform team provides documentation and APIs so stream-aligned teams can efficiently deploy their applications.

3. Service (service):

- A team provides a reusable service that other teams can use autonomously.

Example: A platform team offers a CI/CD pipeline as a service.

3. Team APIs

Teams work together in three different ways to promote seamless interaction and knowledge sharing:

- Collaboration: Teams work intensively on a common goal for a limited time.

  Example: An enabling team supports a stream-aligned team in integrating new KPI monitoring solutions.

- Support: Teams offer advice and expertise without taking responsibility for implementation.

  Example: A platform team documents APIs to optimize deployment processes.

- Service: Teams provide reusable services that are used autonomously by others.

  Example: A platform team offers CI/CD pipelines as a service.

4. Digitalization through platform teams

A key objective of Platform Teams is the digitalization and automation of processes to increase the productivity of stream-aligned teams. This can be implemented as follows:

Example 1: A platform team develops a self-service portal where stream-aligned teams can independently provide test environments.

Example 2: By providing an automated CI/CD pipeline, stream-aligned teams can deploy quickly and securely without solving infrastructure problems.

5. Example I: Dynamic team customization

Compliance with the new Cyber Resilience Act (CRA) requirements requires specific development activities. The following table shows extended activities, the team types involved, their tasks and team API, as additional add-ons in the process landscape:

Activity	Team type	Tasks	Team APIs
Requirements analysis	Stream-aligned team	Identification and documentation of additional security requirements	Documentation, stakeholder communication
Advice on safety standards	Enabling Team	Training on CRA-relevant standards and best practices, e.g., on risk management (HAZOP)	Workshops, training documents
Development of secure subsystems	Complicated subsystem	Design and implementation of cryptographic algorithms	Interface documentation
Building a development platform	Platform Team	Provision of secure development and test environments	CI/CD pipeline APIs, self-service portal
Carrying out security tests	Stream-aligned team	Implementation and execution of penetration tests	Test tools and frameworks provided by the Platform Team

Monitoring and maintenance	Platform Team	Ensuring system stability through continuous monitoring	Monitoring dashboards, APIs for incident reporting
Optimization of security processes	Enabling Team	Advice on optimization strategies and adaptation of existing processes	Process guidelines, KPI tracking

Table: Example of a dynamic team adjustment due to new requirements according to CRA

4

6. Example II: The Software Development Cycle according to ISO 15288 and Team Topologies

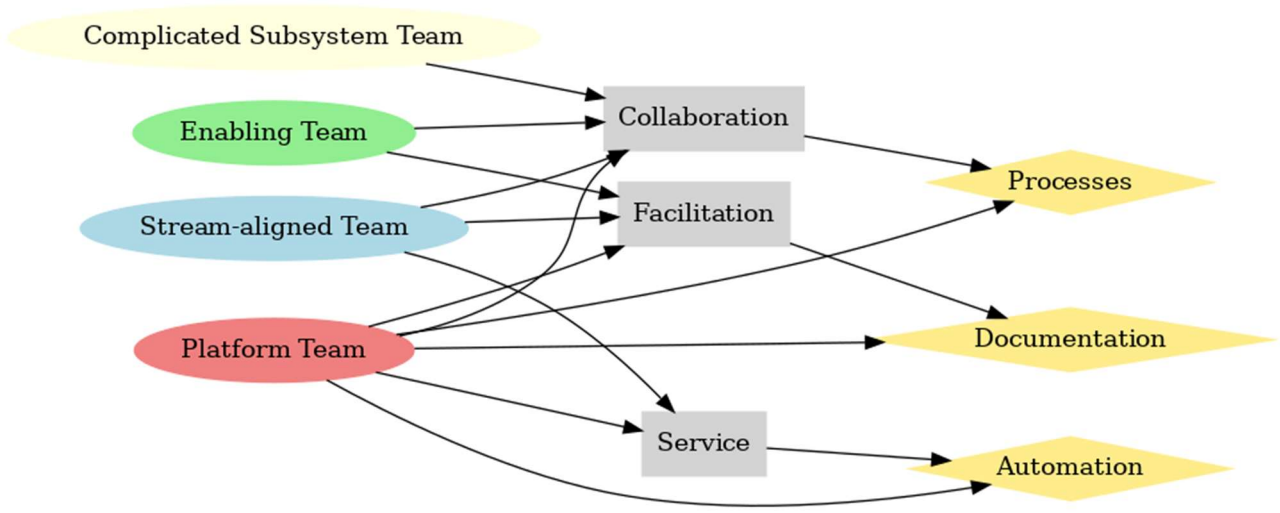
Software development, according to ISO 15288, requires specific development activities. The following table shows these activities, their inputs, outputs, team types involved, tasks, and APIs:

Phase	Activity	Input	Tasks	Output	Team type	Team APIs
Concept phase	Needs analysis	Stakeholder requirements	Analysis of requirements and creation of a concept	Requirement specification	Stream-aligned team	Documentation, stakeholder communication
Development phase	Architectural design	Requirement specification	Creation of a scalable system architecture	Architecture diagrams	Complicated subsystem	Interface documentation
	Implementation	Architecture diagrams, platform tools	Development or outsourcing of software components	Functioning software modules	Stream-aligned team	Test tools and frameworks provided by the Platform Team
Verification phase	Integration tests	Software modules, test plans	Implementation of automated integration tests	Test results	Platform Team	CI/CD pipeline APIs, self-service portal
	Acceptance tests	Test results, user requirements	Validation against stakeholder requirements	Acceptance report	Enabling Team	Process guidelines, KPI tracking
Operating phase	Deployment	Software packages, platform APIs	Provision on production platforms	Live systems	Platform Team	Monitoring dashboards, APIs for incident reporting
	Operation and monitoring	Live systems	Continuous monitoring and maintenance	Monitoring reports	Platform Team	Monitoring dashboards, APIs for incident reporting
Security optimization	Optimization of security processes	Monitoring reports	Advice on optimization strategies and adaptation of existing processes	Improved safety standards	Enabling Team	Process guidelines, KPI tracking

Table: Team topologies in a software life cycle according to ISO 15288

Dipl.- Ing. Stefan Peil | Zur Eiche 14 | 35096 Weimar (Lahn) | Germany

7. Metamodel of the Team Topologies



Picture: Team Topolgy Metamodel (Example)

Explanation of the metamodel graphic

The example metamodel graphically depicts the main components of the **Team Topologies methodology** and their relationships. Here is a detailed explanation of the components and their links:

1. Team types

The oval elements represent the four main types of teams defined in the Team Topologies methodology:

1. **Stream-aligned Team (SA)**:
    - Focus: Development and operation of functions along a specific value stream (e.g. a product).
    - Connections: This team interacts directly via the Collaboration, Facilitation and Service modes.
2. **Enabling Team (ET)**:
    - Focus: Supporting other teams with specialist knowledge and best practices.
    - Connections: Often works in Collaboration and Facilitation mode.
3. **Complicated Subsystem Team (CT)**:
    - Focus: Processing of highly complex technical tasks or subsystems.
    - Connections: Works mainly in Collaboration.
4. **Platform Team (PT)**:
    - Focus: Provision of tools and platforms that support other teams.
    - Connections: Acts in all modes (collaboration, facilitation, and service).

2. Interaction modes

The rectangular boxes represent the **interaction modes** that control the collaboration between the teams:

1. Collaboration (joint goals)

- Teams work together on a task for a limited period of time.
- Example: An enabling team helps a stream-aligned team set up a new pipeline.

2. Facilitation (support)

- A team provides guidelines or expertise but does not assume direct responsibility.
- Example: A platform team provides APIs to simplify deployment.

3.. Service (service)

- A team offers a reusable service that is used autonomously by other teams.
- Example: A platform team offers CI/CD tools as a service.

3. Team APIs

The rhombus-shaped elements represent **team APIs** that define interfaces between teams:

1. Processes:

- Standards and processes that control collaboration.
- Example: Weekly meetings for coordination between teams.

2. Documentation:

- Clear instructions and expectations for other teams.
- Example: Documentation that explains the use of a development environment.

3. Automation:

- Tools and systems that make teams independent and efficient.
- Example: Self-service portals for test environments.

4. Relationships between components

- **Team types↔ Interaction modes**:
  - The lines show how teams interact in different modes. For example, a stream-aligned team works with an enabling team in Collaboration mode.
- **Interaction modes↔ APIs**:
  - The connections show which APIs are used in which mode. For example, providing automation (e.g. CI/CD) is crucial in "Service" mode.
- **Platform Team↔ APIs**:
  - The Platform Team is central to processes, documentation, and automation, providing tools and standards used by other teams.

6. Conclusion

Team topologies provide a clear framework for organizing teams in the development process. By defining specific team types, interaction modes, and APIs, teams can collaborate more efficiently and achieve business goals faster. In particular, the digitization of processes through Platform Teams enables stream-aligned teams to focus on delivering end-user value. Targeted implementation of team topologies leads to clear responsibilities, better collaboration, and greater agility.

Dipl.- Ing. Stefan Peil | Zur Eiche 14 | 35096 Weimar (Lahn) | Germany