CLOCKWORK

Clockwork Glossary

Throughout the **Clockwork** site content, we use terms to identify specialist concepts and ideas which may not be familiar to all readers. This glossary is designed to help briefly explain the terminology in the context of the services provided by Clockwork members.

Architecture Decision Record

A document that captures the important architectural decisions made during the development of a software system. It provides context, rationale, and consequences of each decision, helping teams understand the system's design and evolution. ADRs serve as a valuable reference for future development, enabling teams to learn from past decisions and maintain consistency across projects.

Archipelago Architecture

A hybrid approach between monolithic and microservices architectures where each service manages a substantial domain but consists of multiple independent deployment units. These units can scale independently based on different metrics, while sharing a codebase and allowing direct data access without inter-service communication. Each service owns its data and is responsible for its infrastructure, combining scalability and flexibility with reduced complexity in communication.

Backoffice Gateway

A Backoffice Gateway is a centralised API proxy service which is deployed as a part of the support systems in an organisation. It provides a single interface through which engineering and support teams can interact with deployed services to access their live data, perform administrative tasks, and manage configurations. The Gateway is generally secured by a centralised authentication and authorisation system providing RBAC and access to the API is fully audited for traceability, compliance and security purposes.

Continuous Deployment

A software development practice where code changes are automatically built, tested, and deployed to production without manual intervention. After passing all automated tests, updates are immediately released to users, ensuring frequent and reliable software delivery. Continuous Deployment reduces the time between development and release, allowing teams to quickly respond to feedback and improve features. It relies on strong automation, monitoring, and testing to ensure code quality and system stability.

Contract Testing

A type of software testing that ensures the interaction between software interfaces (such as APIs or microservices) conforms to a defined contract. The contract specifies the expected input and output for a service, and contract tests validate that both the provider (service) and consumer (client) adhere to these expectations. This helps prevent integration issues by verifying that changes in one service do not break dependencies, improving reliability and communication between services in distributed systems.

Conway's Law

Conway's law describes the link between communication structure of organizations and the systems they design. It is named after the computer programmer Melvin Conway, who introduced the idea in 1967.

Event Storming

Event Storming is a collaborative workshop technique used to explore and model complex business processes or software systems. It focuses on identifying key domain events—things that happen in the system. Participants from various roles contribute their insights, mapping out events, actors, and workflows. This visual and interactive approach helps uncover knowledge gaps, align understanding, and improve communication among stakeholders, driving more effective domain-driven design.

Flow

Engineering Flow refers to the seamless, uninterrupted progress of software development, where engineers can work efficiently without delays, distractions, or bottlenecks. It encompasses optimized processes, tooling, and collaboration, allowing developers to focus on delivering value with minimal friction. Achieving flow involves automating repetitive tasks, ensuring clear communication, and reducing context switching, ultimately leading to faster, higher-quality output and greater team productivity.

Hexagonal Architecture

Also known as Ports and Adapters, Hexagonal a software design pattern that emphasizes separation of concerns by organizing code into loosely coupled components. It was introduced by Alistair Cockburn to make applications more maintainable, flexible, and testable.

Living Documentation

Dynamic and continuously updated form of documentation that reflects the current state of a software system, including its design, requirements, and implementation. Unlike traditional static documentation that can become outdated as the software evolves, living documentation is kept in sync with the system, typically through automated processes. It evolves alongside the codebase, ensuring it remains relevant, accurate, and useful to developers, stakeholders, and testers.

Mikado Method

The Mikado Method gets its name from the children's game of pick-up sticks. It provides a simple, iterative framework for breaking down complex changes to software code. At its core, you start by defining a goal, trying one change, and noting how that change causes other failures

Monorepo

A software development strategy where all code for multiple projects, services, or components is stored in a single shared repository. This approach simplifies code management, allows for shared dependencies, ensures consistency, and enables easier cross-team collaboration. In a monorepo, teams can work on different parts of the system while maintaining visibility and control over the entire codebase. However, it requires careful tooling and version control to manage scalability and complexity as the codebase grows.

Shift Left Engineering

A software development practice that emphasises addressing quality, security, and performance concerns early in the development process. By "shifting left," these tasks, typically handled later in the workflow, are integrated into the earlier stages, such as design, coding, and testing. This proactive approach aims to identify and resolve issues sooner, reducing defects, improving efficiency, and lowering costs associated with late-stage fixes. It promotes collaboration between development, QA, and security teams throughout the development lifecycle.

Strangler Pattern

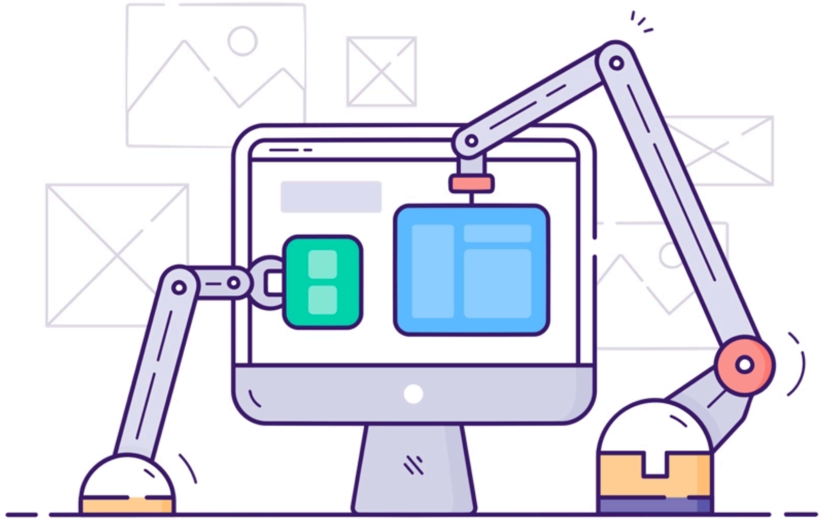
A software development and modernization strategy that involves incrementally replacing or refactoring parts of an existing legacy system by gradually building a new system around it. The term was inspired by the strangler fig plant, which grows around a tree, slowly replacing it over time..

Trunk Based Development

Trunk-Based Development is a software development practice where all developers integrate their work frequently into a shared main branch. This approach minimises the complexity of merging long-lived branches, encourages continuous integration, and enables rapid feedback, ensuring that the codebase remains stable and deployable at all times.

XP Development Practices

A set of software development practices aimed at improving software quality and responsiveness to customer requirements. Core practices include pair programming, test-driven development (TDD), continuous integration, small releases, and frequent customer feedback. XP encourages a high level of communication and collaboration between team members and customers, emphasising simplicity and adaptability. Its goal is to deliver high-quality software while responding to changing needs and fostering a sustainable pace of development.



Choose technology professionals not technology *pretenders*

Clockwork is an invite-only network of **world-class independent technology consultants** who help companies to deliver ***without the hefty markup*** of traditional consulting firms.

CLOCKWORK

https://clockwork.ing