University of Jordan

King Abdullah II School of Information Technology

Business Information Technology Department

IT Project Management | 1904472

Change & Configuration Management

Student Name	**Student ID**
Shaima Aqeel	0228333
Tala Taha	2221281
Ksandra Haddad	0227097
Shahed Al Eiman	2221209
Fedaa Abd Elfattah	2210211

Supervised by: Prof Ahmad H. Kheraisat

1. Introduction

Change Management focuses on handling proposed changes to the system in a structured manner. It ensures that each change is necessary and feasible before implementation [9]. Typically, a change request is submitted, its impact on cost, schedule, and risk is analyzed, and a decision is made by the Change Control Board (CCB). Approved changes are then implemented, verified, and properly documented [9], [10].

Configuration Management (CM) has grown into an essential field in IT and software development, ensuring the integrity, reliability, and efficiency of complex systems. Fundamentally, CM involves systematically identifying, tracking, and controlling configuration items, which include hardware, software, documentation, and their interdependencies. Originally aimed at maintaining consistency and reducing risks related to system changes, CM has evolved to keep pace with rapid technological advances. For example, in large-scale cloud infrastructure projects, CM tools like Ansible or Puppet automate the tracking and deployment of thousands of servers and network devices, ensuring all components remain in sync [1].

Software Configuration Management (SCM), a key area within CM, manages the evolution of software systems by providing mechanisms for version control, building management, and maintaining consistent configurations. SCM supports multiple development workflows—such as workspace control and long transactions—to facilitate parallel development and collaboration across multiple teams. A common example is the use of Git in software development, where branching and merging strategies allow different teams to develop features concurrently and integrate them systematically. While many versioning models exist, the integration of robust process support in SCM is increasingly emphasized to prevent integration conflicts and deployment errors [2].

1.1 Importance in IT Projects

The importance of CM and Change Management is especially clear in IT projects where software systems continuously evolve. These practices help maintain consistency, traceability, and control of system artifacts throughout the lifecycle. SCM tools and processes are critical to managing risks, supporting reliable versioning, and ensuring smooth coordination in complex projects. For example, NASA's software development for space missions relies heavily on strict CM processes to track every software and hardware change, minimizing errors that could risk mission success [2].

1.2 Problem Statement

Despite its importance, CM is often overlooked or poorly implemented in many IS/IT projects. A lack of systematic and discipline change handling frequently leads to missed requirements and unintended impacts. For example, a financial services firm might release a software update without proper CM and testing, leading to system downtime and compliance violations. Effective implementation of CM processes is therefore essential to deliver successful, high-quality IT projects.

2. Change Management

Change Management is a structured process used to plan, evaluate, implement, and monitor changes that affect a project's scope, schedule, cost, or core components, to ensure that the project remains aligned with its objectives. It consists of two critical aspects:

Procedural Aspect: This focuses on managing changes to project plans, including scope, schedule, and budget, in a controlled way so that changes do not compromise the project's goals [11].

Human Aspect: This focuses on preparing and supporting individuals and the organization as they transition from the current state to the future state defined by the project outcomes. It ensures that stakeholders understand, accept, and adopt the change, which increases its overall effectiveness [12].

2.1 Change Management Process in Project Management

To manage changes effectively, organizations typically follow a structured change management process, which includes the following steps:

1. **Request to Change:** A formal change request is logged, including who is requesting the change, why it is needed, and how urgent it is. The change is categorized (for example, standard or emergency) and prioritized based on its potential impact and urgency. An example is when a client requests a new feature late in the project; a formal change request is submitted, and its impact on the schedule and budget is analyzed before any decision is made.
2. **Impact Analysis:** The proposed change is evaluated in terms of its impact on cost, time, quality, risk, and scope, to determine whether it is feasible and acceptable for the project.
3. **Approved or Denied:** Based on the impact analysis, the change request is either approved or denied. If it is approved, it is handed over to the appropriate team for implementation; if it is denied, the reasons are documented.

4. **Implement Change:** The approved change is implemented according to plan. If new information appears during implementation, the original Request for Change (RFC) may be adjusted or, in some cases, resubmitted.
5. **Review / Reporting:** After implementation, the change is reviewed to assess whether it was successful and whether it met its intended objectives. A retrospective or lessons-learned session may be conducted to capture insights that can improve future change management efforts.

3. Configuration Management

Configuration Management is concerned with organizing and controlling all software-related items such as source code, documents, datasets, and libraries. Its main goal is to avoid version conflicts, maintain consistency, and ensure that every change can be traced [7], [8].

3.1 Configuration Management Process in Project Management

To manage configurations effectively, organizations typically follow a structured configuration management process, which includes the following steps:

1. **Planning:** Planning in configuration management defines processes, roles, and responsibilities, ensuring alignment with organizational goals and compliance standards. It helps identify risks early, streamline execution, and minimizes disruptions. Proper planning also selects tools and resources, promoting consistency and supporting system stability while reducing errors.
2. **Configuration Identification:** Configuration identification involves defining and labeling system components, including hardware, software, documentation, and data. It ensures that all components are tracked, and no unauthorized changes occur. This step is crucial for successful Configuration Management, as it allows teams to track, reference, and manage changes to system components over time [4].
3. **Configuration Control:** Configuration control manages changes to configuration items (CIs) in a structured manner, ensuring changes are requested, evaluated, and approved before implementation. It minimizes risks of unauthorized modifications and helps prevent configuration drift, maintaining system stability. Configuration control is closely tied to changing management, ensuring changes are evaluated, approved, and implemented in a controlled way
4. **Configuration Status Accounting:** Configuration status accounting tracks the status and history of configuration items. It records all changes and versions of configuration items, providing visibility into the evolution of the system. This practice ensures that all stakeholders

can easily access the most current and accurate information about system configurations, facilitating audits and ensuring traceability [6].

5. **Configuration Verification and Audit:** Verification and auditing processes are used to ensure that the configuration items are implemented correctly and meet their requirements. This involves regular reviews and inspections to verify that the system's components conform to the defined configurations. Audits also ensure compliance with organizational policies and industry standards, further enhancing the stability and security of the system [5].

4.Relationship Between Change Management and Configuration Management

Change Management and Configuration Management are complementary and operationally interdependent disciplines that control how systems evolve over time. Change Management provides the governance layer that evaluates proposed changes, analyzes impact and risk, and authorizes what should be changed, while Configuration Management provides the control layer that ensures approved changes are implemented on the correct configuration items and documented accurately [9]**.** When both are aligned, organizations can introduce new features and fixes without losing consistency, traceability, or accountability across project deliverables [10].

In practice, the relationship becomes strongest when configuration data actively supports change decisions. Accurate configuration identification (CIs), relationships, and baselines enable better impact analysis because teams can see what components will be affected and what dependencies might break. After approval, configuration control and status accounting ensure that the change is applied to the right versions and environments and that records (code, documents, and configuration files) reflect the new approved state, which prevents “configuration drift” and reduces rework and integration conflicts. This is why many sources emphasize integrating process-oriented change workflows with configuration practices rather than treating them as separate activities.

Without this integration, projects commonly face risks such as implementing the right change on the wrong baseline, deploying inconsistent configurations across development, testing, and production, or losing evidence needed for audits and compliance. A practical example is approving a new feature through Change Management, then using Configuration Management to ensure the correct versions of source code, documentation, and configuration files are updated and baselined so the change remains traceable and reversible if issues occur. Overall, Change Management controls authorization and prioritization, while Configuration Management ensures correct implementation and reliable records—together they reduce project risk and keep system evolution controlled.

5. Tools, Challenges and Best practice

5.1 Tools

In modern IT environments, Change Management (CM) and Configuration Management (ConfigM) rely on integrated tools that support automation, traceability, and version control throughout system change cycles [1]. Platforms such as Jira, ServiceNow, and BMC Remedy help organizations submit change requests, manage approvals, assess risks, and coordinate implementation efforts [1]. These tools also provide audit trails essential for compliance and accountability [5].

Configuration Management depends on tools that manage software versions and automate infrastructure tasks [3]. Version control systems like Git support parallel development and maintain complete histories of code evolution [3]. Automation tools such as Ansible, Puppet, and Chef enable consistent, code-defined configurations across multiple servers, reducing manual errors, and improving reproducibility [4].

5.2 Common Challenges

Despite these capabilities, organizations face recurring challenges when applying CM and ConfigM. Resistance to new processes is common, especially when teams rely on informal workflows or lack proper training. Weak documentation leads to poor traceability and unclear system status. Version conflicts arise when parallel work is not coordinated, and unauthorized changes occur when approval processes are not enforced. Additionally, coordination gaps between development, operations, and security teams hinder consistent CM and ConfigM practices [4].

5.3 Solutions and Best Practices

To address these issues, the literature identifies several effective practices. Automation reduces human error, accelerates deployments, and enforces consistent procedures [4]. CI/CD pipelines ensure continuous testing, validation, and monitoring of changes before reaching production [1]. Clear versioning rules, such as semantic versioning, enhance release communication, while structured communication channels keep stakeholders informed [5]. Well-defined rollback plans allow organizations to restore stable configurations quickly when changes fail, reducing downtime and operational risk [1].

References

[1] Farayola, O. A., Hassan, A. O., Adaramodu, O. R., Fakeyede, O. G., & Oladeinde, M. (2023). *Configuration management in the modern era: Best practices, innovations, and challenges*. Financial Technology and Analytics Department, Naveen Jindal School of Management, Dallas, Texas, USA.

[2] Joeris, G. *Change management needs integrated process and configuration management*. Intelligent Systems Department, Center for Computing Technology, University of Bremen, Bremen, Germany.

[3] Tichy, W. F. (1995). *RCS: A system for version control.* Software: Practice and Experience.

[4] Estublier, J. (2000). *Software configuration management: A roadmap.* International Conference on Software Engineering.

[5] Hass, A. M. J. (2003). *Configuration Management Principles and Practice.* Addison-Wesley.

[6] Cooper, R., & Westfechtel, B. (1997). *A framework for software configuration management.* ACM Computing Surveys.

[7] M. Haug and E. W. Olsen, *Software Configuration and Change Management*. Springer, 2001.

[8] P. Álvarez, et al., "Configuration Management as a Foundation of Engineering Processes," CMS Open Access, 2020.

[9] integration of Change and Configuration Management," ResearchGate

[10] Best Practices in Configuration and Change Management," IEEE Xplore.

[11] APMG International. (n.d.). *What is change management?* APMG International.

[12] ISO. (n.d.). *Information security – IT change management*. ISO.