5.4. Quản lý cấu hình (Configuration Management) ⚙️

Khái niệm cốt lõi

Quản lý cấu hình (Configuration Management - CM) là **kỷ luật nhận diện, kiểm soát và theo dõi** các sản phẩm công việc (work products) trong kiểm thử.

Hãy tưởng tượng bạn đang viết luận văn tốt nghiệp cùng nhóm 5 người. Mỗi người sửa một phần, gửi qua gửi lại qua email. Đến ngày nộp, bạn có:

- `LuanVan_final.docx`
- `LuanVan_final_v2.docx`
- `LuanVan_final_v2_SỬA_LẠI.docx`
- `LuanVan_THẬT_SỰ_CUỐI_CÙNG.docx`

→ **Không ai biết bản nào là bản đúng.** Đây chính là hậu quả khi **không có quản lý cấu hình**.

Quản lý cấu hình (CM) giống như việc cả nhóm thống nhất dùng **Google Docs** hoặc **Git** — mọi thay đổi đều được:

- **Đánh số phiên bản** rõ ràng (v1.0, v1.1, v2.0...)
- **Ghi lại ai sửa gì, lúc nào**
- **Muốn sửa phải xin phép** (không phải ai cũng tự ý sửa được)
- **Quay lại bản cũ** bất cứ lúc nào

Các sản phẩm công việc được quản lý bao gồm:

Trong kiểm thử, có rất nhiều "tài liệu" cần quản lý — gọi chung là **mục cấu hình (configuration item)**:

Bạn có thể hình dung như...	Trong kiểm thử gọi là	Ví dụ
📋 Kế hoạch làm việc của nhóm	Kế hoạch kiểm thử (Test Plan)	"Sprint 5: test chức năng thanh toán, 3 ngày, 2 tester"
✅ Danh sách bài tập cần làm	Kịch bản kiểm thử (Test Case)	"Nhập sai mật khẩu 3 lần → hệ thống khóa tài khoản"
🤖 Code chạy tự động	Kịch bản tự động (Test Script)	File `login_test.py` chạy trên Jenkins
📊 Bảng điểm sau khi chấm	Kết quả kiểm thử (Test Results)	"50 test case: 47 pass, 3 fail"
🖥️ Phòng máy thi	Môi trường kiểm thử (Test Environment)	Windows 11 + Chrome 120 + Database v3.2

CM đảm bảo **tất cả những thứ trên** đều được quản lý phiên bản, không bị lẫn lộn.

Đường cơ sở (Baseline) và kiểm soát thay đổi

Khi mọi thứ đã được kiểm tra, phê duyệt xong → đóng dấu "OK" → gọi là **baseline**.

> 📌 *Giống như bản hợp đồng đã ký: muốn sửa phải làm **phụ lục**, không được tự tay tẩy xóa.*

Ví dụ thực tế:

Môi trường kiểm thử v2.1 gồm:

- Windows Server 2022
- SQL Server 2019
- API Gateway v3.0

Đây là **baseline**. Bây giờ tester muốn nâng SQL Server lên 2022 → **không được tự ý sửa**. Phải:

1. Gửi yêu cầu thay đổi
2. Người có thẩm quyền đánh giá tác động
3. Phê duyệt
4. Cập nhật → tạo baseline mới (v2.2)

Baseline cũ (v2.1) vẫn được **lưu lại** → sau này nếu cần chạy lại test cũ, có thể khôi phục đúng môi trường đó.

Truy xuất nguồn gốc (Traceability) — "Kéo sợi dây là ra hết"

CM yêu cầu mọi thứ phải **liên kết với nhau**:

```
Yêu cầu REQ-018: "Khóa tài khoản sau 3 lần đăng nhập sai"
       ↕
Test Case TC-042 v1.2: "Nhập sai mật khẩu 3 lần, kiểm tra tài khoản bị khóa"
       ↕
Bug BUG-203: "Tài khoản không bị khóa sau 3 lần sai"
```

→ Khi yêu cầu REQ-018 thay đổi, bạn biết ngay TC-042 cần cập nhật. Khi BUG-203 được sửa, bạn biết cần chạy lại TC-042.

CM đảm bảo gì cho kiểm thử?

1. Định danh duy nhất + kiểm soát phiên bản + truy xuất nguồn gốc (traceability)

Mọi mục cấu hình — kể cả các **mục kiểm thử (test items)**, tức các phần riêng lẻ của đối tượng kiểm thử — đều phải được:

- **Định danh duy nhất** (unique ID)
- **Kiểm soát phiên bản** (version control)
- **Theo dõi thay đổi** (change tracking)
- **Liên kết** với các mục cấu hình khác

→ Duy trì **tính truy xuất nguồn gốc (traceability)** xuyên suốt quá trình kiểm thử.

> 📌 ***Ví dụ:*** *Test case TC-042 v1.2 liên kết với yêu cầu REQ-018 và lỗi BUG-203. Khi REQ-018 thay đổi → biết ngay TC-042 cần cập nhật.*

2. Tham chiếu không mơ hồ (unambiguous reference)

Tất cả tài liệu và thành phần phần mềm đã được nhận diện phải được tham chiếu **rõ ràng, chính xác** trong các sản phẩm kiểm thử (testware).

> 📌 ***Ví dụ:*** *Thay vì ghi "dùng bản mới nhất của module thanh toán", phải ghi rõ "Payment Module v2.3.1, build #1847".*

CM trong DevOps 🔄

Tích hợp liên tục (CI), chuyển giao liên tục (CD), triển khai liên tục (Continuous Deployment) và các hoạt động kiểm thử liên quan thường được triển khai trong **luồng công việc DevOps tự động** — trong đó **CM tự động** thường được tích hợp sẵn.

> 📌 ***Ví dụ:*** *Pipeline CI/CD trên Jenkins/GitLab tự động gắn tag version cho mỗi build, lưu trữ artifact, ghi log môi trường test → đây chính là CM tự động.*

Tóm tắt nhanh

Khái niệm	Ý nghĩa	Ví dụ
Mục cấu hình (Configuration Item)	Bất kỳ sản phẩm nào cần được quản lý phiên bản	Test plan, test case, test script, môi trường test
Đường cơ sở (Baseline)	Phiên bản được phê duyệt, "đóng băng"	Môi trường test v2.1 đã approved
Kiểm soát thay đổi (Change Control)	Quy trình chính quy để sửa baseline	Yêu cầu thay đổi → đánh giá → phê duyệt → cập nhật
Truy xuất nguồn gốc (Traceability)	Liên kết giữa các mục cấu hình	TC-042 ↔ REQ-018 ↔ BUG-203
CM tự động (Automated CM)	CM tích hợp trong pipeline DevOps	Git tags, CI/CD artifacts, version pinning

---

Một số ghi chú

- Tài liệu dịch từ sách gốc ISTQB Foudation v4.0.1 sang tiếng Việt, nhằm mục đích giảm rào cản tiếp cận các kiến thức về testing tới cộng đồng tester Việt Nam nói chung và các anh chị em muốn tìm hiểu về testing nói riêng
- Tài liệu cố gắng dịch nhiều nhất các từ tiếng Việt để bạn đọc không phải tra từ điển trong quá trình đọc (VD: defects dịch là khuyết tật phần mềm).
- Dự án phi lợi nhuận, bạn có thể thoải mái sử dụng bản dịch, chia sẻ và sửa đổi nếu cần thiết.
- Có góp ý cho dự án, bạn có thể submit góp ý qua link này nha: https://go.betterbytesvn.com/sharing-documentation-feedback.
- Bản dịch được thực hiện bởi tác giả Đỗ Minh Phong. Bạn có thể gửi lời cảm ơn/feedback/ donate tới tác giả thông qua các hình thức:
  - Đăng ký kênh Youtube: https://www.youtube.com/@hoctest_com
  - Follow Fanpage: https://www.facebook.com/hoctest/
  - Tham gia group Playwright Việt Nam trên Facebook: https://www.facebook.com/groups/playwright.automation.test
  - Donate cho tác giả: https://academy.betterbytesvn.com/donate-cho-chung-minh/
  - Facebook cá nhân tác giả: https://www.facebook.com/dominhphong.18/

Xin chân thành cảm ơn bạn, vì đã quan tâm tới tài liệu ^^.