# 5. Managing the Test Activities

## 5.1. Test Planning

### 5.1.1. Purpose and Content of a Test Plan

**# Purpose and Role of a Test Plan**

A test plan describes the test objectives, resources and processes for a test project. A test plan:

• Documents the means and schedule for achieving test objectives

• Helps to ensure that the performed test activities will meet the established criteria

• Serves as a means of communication with team members and other stakeholders

• Demonstrates that testing will adhere to the existing test policy and test strategy (or explains why the testing will deviate from them)

**# Test Planning and Its Components**

Test planning guides the testers’ thinking and forces the testers to confront the future challenges related to risks, schedules, people, tools, costs, effort, etc. The process of preparing a test plan is a useful way to think through the efforts needed to achieve the test objectives.

The typical content of a test plan includes:

• Context of testing (e.g., test scope, test objectives, test basis)

• Assumptions and constraints of the test project

• Stakeholders (e.g., roles, responsibilities, relevance to testing, hiring and training needs)

• Communication (e.g., forms and frequency of communication, documentation templates)

• Risk register (e.g., product risks, project risks)

• Test approach (e.g., test levels, test types, test techniques, test deliverables, entry criteria and

exit criteria, independence of testing, metrics to be collected, test data requirements, test

environment requirements, deviations from the test policy and test strategy)

• Budget and schedule

More details about the test plan and its content can be found in the ISO/IEC/IEEE 29119-3 standard.

### 5.1.2. Tester's Contribution to Iteration and Release Planning

**# iterative SDLCs**

In iterative SDLCs, typically two kinds of planning occur: release planning and iteration planning.

**# release planning**

**Release planning** looks ahead to the release of a product, defines and re-defines the product backlog, and may involve refining larger user stories into a set of smaller user stories. It also serves as the basis for the test approach and test plan across all iterations. Testers involved in release planning participate in writing testable user stories and acceptance criteria (see section 4.5), participate in project and quality risk analyses (see section 5.2), estimate test effort associated with user stories (see section 5.1.4), determine the test approach, and plan the testing for the release.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

تمام ✅ خليني أشرح لك النص ده بشكل مبسط وواضح:

**📌 معنى Release Planning (تخطيط الإصدار):**

* هو عملية **النظر للمستقبل نحو إصدار المنتج** (النسخة أو الإصدار القادم).
* خلال التخطيط يتم **تحديد أو إعادة تحديد الـ Product Backlog** (قائمة المتطلبات أو قصص المستخدم المراد تنفيذها).
* أحيانًا يتم **تفكيك القصص الكبيرة (Epics)** إلى **قصص أصغر (User Stories)** عشان يسهل تنفيذها واختبارها.

**📌 علاقته بالاختبار:**

* تخطيط الإصدار يعتبر **الأساس** اللي بيتبني عليه **خطة الاختبار (Test Plan)** وطريقة الاختبار (Test Approach) في كل الـ Iterations.
* يعني من خلاله بيتحدد: *إيه اللي هيتعمل – إزاي – إمتى – وبأنهي أولوية*.

**📌 دور المختبرين (Testers) في Release Planning:**

1. **كتابة قصص المستخدم بشكل قابل للاختبار** (Testable User Stories) مع وضع معايير قبول واضحة (Acceptance Criteria).
2. **المشاركة في تحليل المخاطر** الخاصة بالمشروع وجودة المنتج.
3. **تقدير جهد الاختبار** المطلوب لكل User Story (قد إيه وقت/موارد محتاجين).
4. **تحديد طريقة الاختبار** (Levels, Types, Techniques).
5. **التخطيط للاختبارات الخاصة بالإصدار** بالكامل (مش بس لكل Sprint).

🔑 **الخلاصة:**  
تخطيط الإصدار (Release Planning) مش بس بيخص المطورين أو مديري المشاريع، لكن كمان بيساعد المختبرين إنهم يجهزوا نفسهم بدري للاختبارات، ويضمن إن كل حاجة (قصص – معايير قبول – مخاطر – جهد – خطة اختبار) واضحة ومترابطة قبل ما يبدأ التنفيذ الفعلي.

**# Iteration planning**

**Iteration planning** looks ahead to the end of a single iteration and is concerned with the iteration backlog. Testers involved in iteration planning participate in the detailed risk analysis of user stories, determine the testability of user stories, break down user stories into tasks (particularly testing tasks), estimate test effort for all testing tasks, and identify and refine functional and non-functional aspects of the test object.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

تمام ✅ خليني أبسطلك فكرة **Iteration Planning (تخطيط التكرار)**:

**📌 ما هو Iteration Planning؟**

* هو التخطيط الخاص **بمدة تكرار واحدة (Iteration/Sprint)**.
* يركز على **Iteration Backlog** (المهام/قصص المستخدم اللي هتتنفذ جوه التكرار ده).
* الهدف: نحدد *بالضبط* إيه اللي هيتعمل في التكرار وكيفية تنفيذه.

**📌 دور المختبرين (Testers) في Iteration Planning:**

1. **المشاركة في تحليل المخاطر** الخاصة بقصص المستخدم (User Stories) بشكل مفصل.  
   → يساعد على تحديد الأجزاء الأكثر خطورة أو أهمية اللي لازم تتغطى بالاختبارات.
2. **تحديد قابلية اختبار قصص المستخدم** (Testability).  
   → هل القصة مكتوبة بشكل يسمح نعمل لها اختبارات واضحة؟ هل لها Acceptance Criteria واضحة؟
3. **تفكيك قصص المستخدم إلى مهام أصغر** (خصوصًا مهام الاختبار).  
   → مثلاً: إعداد بيانات اختبار، كتابة حالات اختبار، تنفيذ اختبارات وظيفية أو غير وظيفية.
4. **تقدير جهد الاختبار** لكل مهمة.  
   → قد إيه وقت أو موارد محتاجين لتنفيذ مهام الاختبار.
5. **تحديد وصقل الجوانب الوظيفية وغير الوظيفية** للاختبار.  
   → اختبارات الأداء، الأمان، سهولة الاستخدام… إلخ بجانب الوظائف العادية.

**🔑 الخلاصة:**

* **Release Planning** = نظرة واسعة على الإصدار ككل.
* **Iteration Planning** = نظرة مركزة على فترة قصيرة (Sprint/Iteration).
* المختبرين في Iteration Planning بيهتموا بالتفاصيل اليومية: *تحليل المخاطر – اختبارية القصص – تقسيم المهام – تقدير الجهد – تحديد الاختبارات الوظيفية وغير الوظيفية*.

### 5.1.3. Entry Criteria and Exit Criteria

**# Entry and Exit Criteria in Testing**

**Entry criteria** define the preconditions for undertaking a given activity. If entry criteria are not met, it is likely that the activity will prove to be more difficult, time-consuming, costly, and riskier. **Exit criteria** define what must be achieved to declare an activity completed. Entry criteria and exit criteria should be defined for each test level, and will differ based on the test objectives.

**# Examples of Entry Criteria**

Typical entry criteria include:

* + availability of resources (e.g., people, tools, environments, test data, budget, time),
  + availability of testware (e.g., test basis, testable requirements, user stories, test cases), and
  + initial quality level of a test object (e.g., all smoke tests have passed).

**# Examples of Exit Criteria**

Typical exit criteria include:

* + measures of thoroughness (e.g., achieved level of coverage, number of unresolved defects, defect density, number of failed test cases), and
  + binary “yes/no” criteria (e.g., planned tests have been executed, static testing has been performed, all defects found are reported, all regression tests are automated).

Running out of time or budget can also be viewed as valid exit criteria. Even without other exit criteria being satisfied, it can be acceptable to end testing under such circumstances, if the stakeholders have reviewed and accepted the risk to go live without further testing.

**# Entry and Exit Criteria in Agile**

In Agile software development, exit criteria are often called **Definition of Done**, defining the team’s objective metrics for a releasable item. Entry criteria that a user story must fulfill to start the development and/or testing activities are called **Definition of Ready**.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

تمام ✅ خليني أشرحلك النقطة دي بشكل مبسط وواضح:

**📌 في منهجية Agile:**

* **Exit Criteria** 👉 بتتسمى **Definition of Done (DoD)**.  
  يعني: إيه الشروط أو المعايير اللي لازم تتحقق عشان نقول إن القصة (User Story) أو المهمة خلصت وجاهزة فعلاً للإصدار.  
  🔹 أمثلة:
  + كل الـ Acceptance Criteria اتحققت.
  + الكود اتراجع (Code Review).
  + كل الاختبارات (Functional + Regression) نجحت.
  + التوثيق الأساسي اتعمل.
* **Entry Criteria** 👉 بتتسمى **Definition of Ready (DoR)**.  
  يعني: إيه الشروط اللي لازم القصة (User Story) تفي بيها عشان تبدأ فيها عملية التطوير أو الاختبار.  
  🔹 أمثلة:
  + القصة مكتوبة بشكل واضح ومفهومة.
  + في معايير قبول (Acceptance Criteria) محددة.
  + البيانات أو البيئة المطلوبة للاختبار متاحة.
  + القصة متقدّرة (Estimated).

**🔑 الخلاصة:**

* **DoR = جاهزية للبدء** → متى تكون القصة مستعدة ليدخلها الفريق في Sprint.
* **DoD = جاهزية للانتهاء** → متى نقدر نقول إن القصة خلصت وجاهزة للتسليم/الإصدار.

### 5.1.4. Estimation Techniques

**# Test Effort Estimation**

Test effort estimation involves predicting the amount of test-related work needed to meet the test objectives of a test project. It is important to make it clear to the stakeholders that the estimate is based on a number of assumptions and is always subject to estimation error. Estimation for small tasks is usually more accurate than for the large ones. Therefore, when estimating a large task, it can be decomposed into a set of smaller tasks which then in turn can be estimated.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

تمام ✅ خليني أشرحلك مفهوم تقدير جهد الاختبار (Test Effort Estimation) بشكل مبسط:

📌 ما هو تقدير جهد الاختبار؟

هو عملية توقّع حجم العمل الخاص بالاختبارات المطلوب عشان نحقق أهداف الاختبار في مشروع معين.  
يعني: نحاول نجاوب على سؤال “قد إيه وقت وجهد وموارد هنحتاج للاختبار؟”

📌 نقاط أساسية:

التقدير قائم على افتراضات

يعني النتيجة مش 100% دقيقة، لأن دايمًا في عوامل غير متوقعة.

لذلك لازم نشرح الافتراضات دي لأصحاب المصلحة (Stakeholders).

التقدير مش خالي من الخطأ

أي تقدير بيبقى فيه نسبة خطأ (Estimation Error).

عشان كده لازم نكون واقعيين ونوضح إن الرقم تقريبي مش نهائي.

المهام الصغيرة أسهل في التقدير

لما تبقى المهمة صغيرة، بنقدر نحدد وقتها ومجهودها بشكل أدق.

أما المهام الكبيرة فالتقدير بيكون أصعب.

الحل مع المهام الكبيرة → تقسيمها

لو عندنا مهمة كبيرة (مثلاً: "اختبار النظام بالكامل")، الأفضل نقسمها لمهام أصغر (زي: "اختبار تسجيل الدخول"، "اختبار الدفع"، "اختبار الأداء").

بعد التقسيم، بنقدّر كل مهمة صغيرة، وبعدين نجمعهم عشان نوصل لتقدير أدق للمهمة الكبيرة.

🔑 الخلاصة:

تقدير جهد الاختبار = توقّع المجهود المطلوب، مع إدراك إنه تقريبي ومبني على افتراضات، والدقة بتزيد لما نقسم المهام الكبيرة لمهام أصغر.

**# four estimation techniques**

In this syllabus, the following four estimation techniques are described.

**# Estimation based on ratios**

**Estimation based on ratios**. In this metrics-based technique, figures are collected from previous projects within the organization, which makes it possible to derive “standard” ratios for similar projects. The ratios of an organization’s own projects (e.g., taken from historical data) are generally the best source to use in the estimation process. These standard ratios can then be used to estimate the test effort for the new project. For example, if in the previous project the development-to-test effort ratio was 3:2, and in the current project the development effort is expected to be 600 person-days, the test effort can be estimated to be 400 person-days.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

تمام ✅ خليني أشرحلك مفهوم **التقدير باستخدام النِّسَب (Estimation based on Ratios)** بشكل مبسط وواضح:

**📌 الفكرة الأساسية:**

* هو **أسلوب تقدير مبني على البيانات السابقة** (metrics-based).
* بنجمع أرقام وإحصائيات من مشاريع قديمة داخل نفس المنظمة.
* بعد كده بنستخرج منها **نِسَب قياسية** نقدر نطبّقها على المشاريع الجديدة.

**📌 كيف يتم؟**

1. **نجمع بيانات تاريخية**
   * من مشاريع سابقة، زي: عدد أيام التطوير مقابل عدد أيام الاختبار.
2. **نحسب النسبة**
   * مثلاً: في مشروع قديم كان عندنا جهد التطوير إلى جهد الاختبار = **3:2**.
3. **نطبّق النسبة على المشروع الجديد**



* + لو متوقع إن جهد التطوير = **600 يوم عمل**.
  + يبقى جهد الاختبار = 23×600=400\frac{2}{3} \times 600 = 400 يوم عمل.

**📌 المميزات:**

* بيعتمد على **بيانات حقيقية** من نفس المنظمة، وبالتالي بيكون أكثر واقعية.
* مفيد جدًا في المشاريع المتشابهة (نفس نطاق العمل أو نفس نوع النظام).

**📌 التحديات:**

* لازم تكون عندنا **بيانات تاريخية دقيقة** ومخزنة كويس.
* لو المشروع الجديد مختلف تمامًا عن المشاريع السابقة، النسبة ممكن تكون مضللة.

**🔑 الخلاصة:**

**التقدير باستخدام النِّسَب** = الاعتماد على خبرات سابقة وأرقام تاريخية، وتحويلها لنسبة قياسية (Ratio) نستخدمها لتقدير جهد الاختبار في المشاريع الجديدة.

تحب أرتبلك الموضوع في شكل **خطوات عملية + مثال بالأرقام** بحيث تستخدمه كمذكرة جاهزة؟

**# Extrapolation**

**Extrapolation**. In this metrics-based technique, measurements are made as early as possible in the current project to gather the data. Having enough observations, the effort required for the remaining work can be approximated by extrapolating this data (usually by applying a mathematical model). This method is very suitable in iterative SDLCs. For example, the team may extrapolate the test effort in the forthcoming iteration as the averaged effort from the last three iterations.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

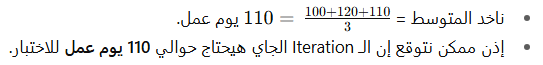
تمام ✅ خليني أشرحلك ببساطة مفهوم **التقدير بالاستقراء (Extrapolation)**:

**📌 الفكرة الأساسية:**

* هو أسلوب تقدير **مبني على القياسات داخل المشروع الحالي** (مش بس على بيانات تاريخية).
* يعني: بنبدأ المشروع، نجمع بيانات فعلية عن الجهد المبذول، وبعدين نستخدمها للتنبؤ بالجهد المتبقي.
* غالبًا بيتطبق باستخدام **نماذج رياضية أو إحصائية بسيطة** (زي المتوسط الحسابي أو المنحنيات).

**📌 كيف يتم؟**

1. **نجمع بيانات مبكرًا**:
   * أول ما يبدأ المشروع، الفريق يسجل الوقت/الجهد اللي بيصرفه على مهام الاختبار.
2. **نحدد نمط الأداء**:
   * مثلًا: في أول 3 Iterations، الجهد المبذول للاختبار كان (100، 120، 110) يوم عمل.
3. **نستنتج (Extrapolate) للمستقبل**:



* + ناخد المتوسط = 100+120+1103=110\frac{100 + 120 + 110}{3} = 110 يوم عمل.
  + إذن ممكن نتوقع إن الـ Iteration الجاي هيحتاج حوالي **110 يوم عمل** للاختبار.

**📌 المميزات:**

* **عملي جدًا** في البيئات التكرارية (Agile / Iterative SDLCs).
* بيعتمد على **بيانات حقيقية من نفس المشروع**، فبيكون أدق من الاعتماد على مشاريع سابقة.

**📌 التحديات:**

* محتاج **بيانات كافية** قبل ما يبدأ يبقى مفيد (لازم يكون فيه Iterations سابقة).
* الدقة ممكن تقل لو طبيعة المهام بتتغير بشكل كبير بين Iteration والتانية.

**🔑 الخلاصة:**

**Extrapolation** = التقدير عن طريق **جمع بيانات فعلية مبكرًا** من المشروع الحالي، وبعدين استخدامها للتنبؤ بالجهد المستقبلي.  
مثال: متوسط جهد الاختبار في آخر 3 Iterations = **110 يوم عمل** → نتوقع إن الـ Iteration القادم يحتاج **110 يوم عمل**.

**# Wideband Delphi**

**Wideband Delphi**. In this iterative, expert-based technique, experts make experience-based estimations. Each expert, in isolation, estimates the effort. The results are collected and if there are deviations of an expert’s estimate that are out of range of the agreed upon boundaries, the experts discuss their current estimates. Each expert is then asked to make a new estimation based on that feedback, again in isolation. This process is repeated until a consensus is reached. **Planning Poker** is a variant of Wideband Delphi, commonly used in Agile software development. In Planning Poker, estimates are usually made using cards with numbers that represent the effort size.

تمام ✅ خليني أوضحلك ببساطة مفهوم **Wideband Delphi** ونسخته في Agile (Planning Poker):

**📌 ما هو Wideband Delphi؟**

* هو **أسلوب تقدير تكراري Iterative** يعتمد على **خبرة الخبراء (Expert-based)**.
* الهدف منه: الوصول إلى **تقدير أكثر دقة** من خلال **مشاركة عدة خبراء** بشكل منظم.

**📌 خطواته:**

1. **كل خبير يقدّر بشكل منفصل** (بدون ما يتأثر برأي الآخرين).
2. **تُجمع التقديرات** ويتم تحديد هل في تقديرات خارج النطاق المتفق عليه.
3. **مناقشة جماعية** بين الخبراء لمراجعة الأسباب وراء التقديرات المختلفة.
4. **إعادة التقدير**: كل خبير يعدل تقديره (مرة تانية في عزلة).
5. **تتكرر العملية** لحد ما الفريق يوصل إلى **إجماع (Consensus)**.

**📌 ليه مفيد؟**

* يقلل من **التحيز الفردي** أو ضغط المجموعة.
* بيجمع **رؤى متعددة**، فبيكون التقدير أدق.
* مرن ويتكرر لحد ما الفريق يكون متفق على نتيجة واقعية.

**📌 Planning Poker (نسخة Agile من Wideband Delphi):**

* بيستخدم في Agile كتقنية عملية للتقدير.
* الفريق بيستعمل **كروت مرقمة** (مثلاً: 1, 2, 3, 5, 8, 13 … غالبًا متسلسلة فيبوناتشي).
* كل عضو يختار كارت سرًا يمثل حجم الجهد المطلوب للقصة (User Story).
* بعد كده الكل يكشف الكروت في نفس الوقت → لو في فروق كبيرة، الفريق يناقش الأسباب → ثم يُعاد التقدير.
* الهدف: الوصول لتقدير جماعي متفق عليه.

**🔑 الخلاصة:**

* **Wideband Delphi** = خبراء يقدّروا بشكل مستقل → مناقشة → إعادة تقدير → تكرار لحد الاتفاق.
* **Planning Poker** = تطبيق عملي داخل Agile بنفس الفكرة، لكن بشكل أبسط وأسرع باستخدام كروت للأرقام.

**# Three-point estimation**

**Three-point estimation**. In this expert-based technique, three estimations are made by the experts: the most optimistic estimation (a), the most likely estimation (m) and the most pessimistic estimation (b). The final estimate (E) is their weighted arithmetic mean. In the most popular version of this technique, the estimate is calculated as E = (a + 4\*m + b) / 6. The advantage of this technique is that it allows the experts to calculate the measurement error: SD = (b – a) / 6. For example, if the estimates (in personhours) are: a=6, m=9 and b=18, then the final estimation is 10±2 person-hours (i.e., between 8 and 12 person-hours), because E = (6 + 4\*9 + 18) / 6 = 10 and SD = (18 – 6) / 6 = 2.

See (Kan 2003, Koomen 2006, Westfall 2009) for these and many other test estimation techniques.

تمام ✅ خليني أشرحلك ببساطة مفهوم **Three-Point Estimation**:

**📌 الفكرة الأساسية:**

* هو أسلوب **خبيرّي Expert-based** لتقدير الجهد/الوقت.
* بدل ما ندي رقم واحد كـ “تقدير”، بنفكر في **3 احتمالات**:
  1. **a = Optimistic (أفضل حالة)**: أقل وقت ممكن.
  2. **m = Most Likely (السيناريو الأكثر احتمالاً)**: الوقت المتوقع غالبًا.
  3. **b = Pessimistic (أسوأ حالة)**: أطول وقت ممكن.

**📌 طريقة الحساب:**

* التقدير النهائي (E) يُحسب بالمتوسط المرجّح:

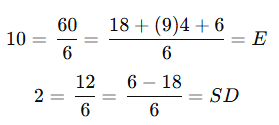
E=a+4m+b6E = \frac{a + 4m + b}{6}

* الانحراف المعياري (SD) يقيس **الخطأ/عدم اليقين في التقدير**:

SD=b−a6SD = \frac{b - a}{6}

**📌 مثال عملي:**

* لو:
  + a = 6 ساعات (متفائل)
  + m = 9 ساعات (الأكثر احتمالًا)
  + b = 18 ساعة (متشائم)
* نحسب:



E=6+4(9)+186=606=10E = \frac{6 + 4(9) + 18}{6} = \frac{60}{6} = 10 SD=18−66=126=2SD = \frac{18 - 6}{6} = \frac{12}{6} = 2

* إذن: **التقدير = 10 ± 2 ساعات**  
  → يعني المجهود المتوقع يتراوح بين **8 و 12 ساعة**.

**📌 المميزات:**

* يوازن بين التفاؤل والتشاؤم والواقع.
* يوضح **نطاق زمني متوقع** بدل رقم واحد فقط.
* يساعد في حساب **درجة عدم اليقين** (Estimation Error).

**🔑 الخلاصة:**

**Three-Point Estimation** = طريقة تقدير مبنية على 3 سيناريوهات (متفائل – محتمل – متشائم) → نحسب متوسط مرجّح + انحراف معياري → يطلع تقدير أدق مع نطاق ثقة.

### 5.1.5. Test Case Prioritization

**# Test Case Prioritization**

Once the test cases and test procedures are specified and assembled into test suites, these test suites can be arranged in a test execution schedule that defines the order in which they are to be run. When prioritizing test cases, different factors can be taken into account. The most commonly used test case prioritization strategies are as follows:

• **Risk-based prioritization**, where the order of test execution is based on the results of risk analysis (see section 5.2.3). Test cases covering the most important risks are executed first.

• **Coverage-based prioritization**, where the order of test execution is based on coverage (e.g.,

statement coverage). Test cases achieving the highest coverage are executed first. In another variant, called **additional coverage prioritization**, the test case achieving the highest coverage is

executed first; each subsequent test case is the one that achieves the highest additional coverage.

• **Requirements-based prioritization**, where the order of test execution is based on the priorities of the requirements traced back to the corresponding test cases. Requirement priorities are defined by stakeholders. Test cases related to the most important requirements are executed first.

**# Test Case Prioritization with Dependencies**

Ideally, test cases would be ordered to run based on their priority levels, using, for example, one of the above-mentioned prioritization strategies. However, this practice may not work if the test cases or the features being tested have **dependencies**. If a test case with a higher priority is dependent on a test case with a lower priority, the lower priority test case must be executed first.

**# Resource-Aware Test Execution Prioritization**

The order of test execution must also take into account the availability of resources. For example, the required test tools, test environments or people that may only be available for a specific time window.

### 5.1.6. Test Pyramid

**# Test Pyramid**

The **test pyramid** is a model showing that different tests may have different granularity. The test pyramid model supports the team in test automation and in test effort allocation by showing that different test objectives are supported by different levels of test automation. The pyramid layers represent groups of tests. The higher the layer, the lower the test granularity, the lower the test isolation (i.e., the degree of dependency on other elements of the system) and the higher the test execution time. Tests in the bottom layer are small, isolated, fast, and check a small piece of functionality, so usually a lot of them are needed to achieve a reasonable coverage. The top layer represents complex, high-level, end-to-end tests. These high-level tests are generally slower than the tests from the lower layers, and they typically check a large piece of functionality, so usually just a few of them are needed to achieve a reasonable level of coverage. The number and naming of the layers may differ. For example, the original test pyramid model (Cohn 2009) defines three layers: “unit tests”, “service tests” and “UI tests”. Another popular model defines unit (component) tests, integration (component integration) tests, and end-to-end tests. Other test levels (see section 2.2.1) can also be used.

### 5.1.7. Testing Quadrants

**# Testing Quadrants**

The testing quadrants, defined by Brian Marick (Marick 2003, Crispin 2008), group the test levels with the appropriate test types, activities, test techniques and work products in the Agile software development. The model supports test management in visualizing these to ensure that all appropriate test types and test levels are included in the SDLC and in understanding that some test types are more relevant to certain test levels than others. This model also provides a way to differentiate and describe the test types to all stakeholders, including developers, testers, and business representatives.

In this model, tests can be business facing or technology facing. Tests can also support the team (i.e., guide the development) or critique the product (i.e., measure its behavior against the expectations). The combination of these two viewpoints determines the four quadrants:

• Quadrant Q1 (technology facing, support the team). This quadrant contains component tests and component integration tests. These tests should be automated and included in the CI process.

• Quadrant Q2 (business facing, support the team). This quadrant contains functional tests,

examples, user story tests, user experience prototypes, API testing, and simulations. These tests

check the acceptance criteria and can be manual or automated.

• Quadrant Q3 (business facing, critique the product). This quadrant contains exploratory testing, usability testing, user acceptance testing. These tests are user-oriented and often manual.

• Quadrant Q4 (technology facing, critique the product). This quadrant contains smoke tests and non-functional tests (except usability tests). These tests are often automated.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/14**/2025]**

تمام 👌 خليني أبسطلك موضوع **اختبار الـ Quadrants (مربعات الاختبار في الأجايل)** خطوة بخطوة:

**🔹 الفكرة الأساسية:**

الموديل بتاع **Testing Quadrants** (اللي وضعه Brian Marick وطوّرته Lisa Crispin) هو طريقة **لتقسيم الاختبارات** في بيئة الـ **Agile** على أساس:

1. هل الاختبار **موجه للأعمال (Business-facing)** ولا **موجه للتكنولوجيا (Technology-facing)**.
2. هل الهدف منه **دعم الفريق (Support the team)** أثناء التطوير ولا **انتقاد المنتج (Critique the product)** بعد بنائه.

بمزج الاتجاهين دول، بنحصل على **أربع مربعات (Q1, Q2, Q3, Q4)**.

**🔸 المربعات الأربعة:**

**✅ Q1: (Technology Facing – Support the Team)**

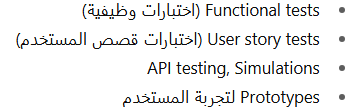
* يركز على **الاختبارات التقنية اللي تساعد الفريق يطوّر بشكل أسرع**.
* أمثلة:



* + Component tests (اختبارات الوحدات/الكومبوننت)
  + Component integration tests (تكامل المكونات)
* غالبًا **مؤتمتة** وبتدخل في الـ **CI/CD**.
* الهدف → يضمن إن الكود الجديد شغال ومتكامل مع القديم من بدري.

**✅ Q2: (Business Facing – Support the Team)**

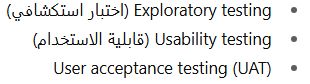
* يركز على **التحقق من متطلبات العمل (User Stories, Acceptance Criteria)**.
* أمثلة:



* + Functional tests (اختبارات وظيفية)
  + User story tests (اختبارات قصص المستخدم)
  + API testing, Simulations
  + Prototypes لتجربة المستخدم
* ممكن تكون **manual أو automated**.
* الهدف → يتأكد إن الفريق بيبني اللي العميل عايزه بالضبط.

**✅ Q3: (Business Facing – Critique the Product)**

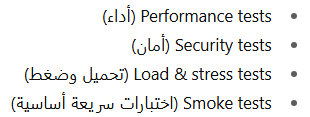
* يركز على **اختبارات من وجهة نظر المستخدم النهائي** لتقييم المنتج.
* أمثلة:



* + Exploratory testing (اختبار استكشافي)
  + Usability testing (قابلية الاستخدام)
  + User acceptance testing (UAT)
* غالبًا **يدوي**.
* الهدف → نتاكد المنتج مناسب وقابل للاستخدام من منظور العميل/المستخدم.

**✅ Q4: (Technology Facing – Critique the Product)**

* يركز على **اختبارات غير وظيفية (Non-functional tests)**.
* أمثلة:



* + Performance tests (أداء)
  + Security tests (أمان)
  + Load & stress tests (تحميل وضغط)
  + Smoke tests (اختبارات سريعة أساسية)
* غالبًا **مؤتمتة**.
* الهدف → يختبر المنتج تحت ظروف مختلفة ويكشف عيوب الجودة غير الوظيفية.

**🔹 الخلاصة:**

* **Q1 + Q2 = دعم الفريق (Guiding Development)**
* **Q3 + Q4 = نقد المنتج (Critiquing the Product)**
* **Q1 + Q4 = اختبارات تقنية**
* **Q2 + Q3 = اختبارات موجهة للأعمال**

## 5.2. Risk Management

**# Risk Management and Risk-Based Testing**

Organizations face many internal and external factors that make it uncertain whether and when they will achieve their objectives (ISO 31000). **Risk management** allows the organizations to increase the likelihood of achieving objectives, improve the quality of their products and increase the stakeholders’confidence and trust.

The main risk management activities are:

• **Risk analysis** (consisting of risk identification and risk assessment; see section 5.2.3)

• **Risk control** (consisting of risk mitigation and risk monitoring; see section 5.2.4)

The test approach, in which test activities are selected, prioritized, and managed based on risk analysis and risk control, is called **risk-based testing**.

### 5.2.1. Risk Definition and Risk Attributes

**# Risk Definition and Risk Attributes**

Risk is a potential event, hazard, threat, or situation whose occurrence causes an adverse effect. A risk can be characterized by two factors:

• **Risk likelihood** – the probability of the risk occurrence (greater than zero and less than one)

• **Risk impact (harm)** – the consequences of this occurrence

These two factors express the risk level, which is a measure for the risk. The higher the risk level, the more important is its treatment.

### 5.2.2. Project Risks and Product Risks

**# Project Risks and Product Risks**

In software testing one is generally concerned with two types of risks: project risks and product

risks. **Project risks** are related to the management and control of the project. Project risks include:

• Organizational issues (e.g., delays in work products deliveries, inaccurate estimates, cost cutting)

• People issues (e.g., insufficient skills, conflicts, communication problems, shortage of staff)

• Technical issues (e.g., scope creep, poor tool support)

• Supplier issues (e.g., third-party delivery failure, bankruptcy of the supporting company)

**Project risks**, when they occur, may have an impact on the project schedule, budget or scope, which affects the project's ability to achieve its objectives.

**Product risks** are related to the product quality characteristics (e.g., described in the ISO 25010 quality model). Examples of product risks include:

* + missing or wrong functionality,
  + incorrect calculations,
  + runtime errors,
  + poor architecture,
  + inefficient algorithms,
  + inadequate response time,
  + poor user experience,
  + security vulnerabilities.

**Product risks**, when they occur, may result in various negative consequences, including:

• User dissatisfaction

• Loss of revenue, trust, reputation

• Damage to third parties

• High maintenance costs, overload of the help desk

• Criminal penalties

• In extreme cases, physical damage, injuries or even death

### 5.2.3. Product Risk Analysis

**# Product Risk Analysis**

From a testing perspective, the goal of **product risk analysis** is to provide an awareness of product risk to focus the test effort in a way that minimizes the residual level of product risk. Ideally, product risk analysis begins early in the SDLC.

**# risk identification and risk assessment**

Product risk analysis consists of **risk identification** and **risk assessment**. **Risk identification** is about generating a comprehensive list of risks. Stakeholders can identify risks by using various techniques and tools, e.g., brainstorming, workshops, interviews, or cause-effect diagrams. Risk assessment involves:

categorization of identified risks, determining their risk likelihood, risk impact and risk level, prioritizing, and proposing ways to handle them. Categorization helps in assigning mitigation actions, because usually risks falling into the same category can be mitigated using a similar approach.

**Risk assessment** can use a quantitative or qualitative approach, or a mix of them. In the quantitative approach the risk level is calculated as the multiplication of risk likelihood and risk impact. In the qualitative approach the risk level can be determined using a risk matrix.

**# Impact of Product Risk Analysis on Testing**

Product risk analysis may influence the thoroughness and test scope. Its results are used to:

• Determine the test scope to be carried out

• Determine the particular test levels and propose test types to be performed

• Determine the test techniques to be employed and the coverage to be achieved

• Estimate the test effort required for each task

• Prioritize testing in an attempt to find the critical defects as early as possible

• Determine whether any activities in addition to testing could be employed to reduce risk

### 5.2.4. Product Risk Control

**# Product risk control**

**Product risk control** comprises all measures that are taken in response to identified and assessed product risks. Product risk control consists of **risk mitigation** and **risk monitoring**. Risk mitigation involves implementing the actions proposed in risk assessment to reduce the risk level. The aim of risk monitoring is to ensure that the mitigation actions are effective, to obtain further information to improve risk assessment, and to identify emerging risks.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/15**/2025]**

تمام ✅ خليني أشرح لك الكلام خطوة بخطوة بالعربي وبأسلوب مبسّط:

**التحكم في مخاطر المنتج (Product Risk Control)**

هو عبارة عن **جميع الإجراءات** اللي بتتعمل عشان نتعامل مع المخاطر اللي تم اكتشافها وتقييمها في المنتج.

يعني بعد ما نحدد المخاطر (Risk Identification) ونقيّمها (Risk Assessment)، نبدأ نعمل **تحكم** في هذه المخاطر.

**التحكم في المخاطر بيتكون من جزئين رئيسيين:**

1. **تخفيف المخاطر (Risk Mitigation)** 🔧
   * معناه: تنفيذ الإجراءات اللي اتخططت في مرحلة التقييم عشان نقلل من مستوى الخطر.
   * الهدف: إما نقلل احتمال حدوث الخطر أو نقلل تأثيره لو حصل.
   * مثال: لو في خطر إن السيرفر يقع وقت الضغط → نعمل نسخ احتياطية + موازنة أحمال.
2. **مراقبة المخاطر (Risk Monitoring)** 👀
   * معناه: متابعة وتقييم هل إجراءات التخفيف اللي عملناها شغالة فعلاً ولا لأ.
   * الهدف:
     + نتأكد إن الخطر فعلاً اتقلص.
     + نجيب معلومات جديدة تساعدنا نحسن عملية التقييم (Risk Assessment).
     + نكتشف مخاطر جديدة ممكن تظهر أثناء المشروع.
   * مثال: بعد إضافة النسخ الاحتياطية، نتابع الأداء ونشوف إذا في مشاكل جديدة ظهرت.

**📌 الخلاصة**

* **Risk Mitigation** = نعمل خطوات عملية لتقليل الخطر.
* **Risk Monitoring** = نتابع ونراجع باستمرار عشان نتأكد إن الخطوات دي فعالة، ونكتشف أي مخاطر جديدة.

**# Risk Mitigation Strategies in Testing**

With respect to product risk control, once a risk has been analyzed, several response options to risk are possible, e.g., **risk mitigation** by testing, risk acceptance, risk transfer, or a contingency plan (Veenendaal 2012). Actions that can be taken to mitigate the product risks by testing are as follows:

• Select the testers with the right level of experience and skills, suitable for a given risk type

• Apply an appropriate level of independence of testing

• Perform reviews and static analysis

• Apply the appropriate test techniques and coverage levels

• Apply the appropriate test types addressing the affected quality characteristics

• Perform dynamic testing, including regression testing

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/15**/2025]**

تمام 👌 خليني أوضحلك موضوع **خيارات الاستجابة للمخاطر (Risk Responses) مع التركيز على التخفيف بالممارسات الاختبارية** خطوة بخطوة:

**🔹 بعد تحليل المخاطر (Risk Analysis)**

بيكون قدام الفريق عدة طرق للتعامل مع المخاطر (Product Risks):

1. **Risk Mitigation (التخفيف)** → تقليل الاحتمال أو الأثر.
2. **Risk Acceptance (القبول)** → تقبل الخطر كما هو بدون إجراء.
3. **Risk Transfer (التحويل)** → تحويل المسؤولية لطرف آخر (مثلاً التأمين أو طرف ثالث).
4. **Contingency Plan (خطة الطوارئ)** → تجهيز خطة بديلة لو حصل الخطر.

**🔸 تخفيف مخاطر المنتج بالاختبار (Risk Mitigation by Testing)**

الاختبار هنا بيكون أداة قوية لتقليل مستوى المخاطر عن طريق:

1. **اختيار المختبرين المناسبين**
   * الاستعانة بأشخاص خبراء في المجال أو النوعية المعينة من المخاطر.
   * مثال: لو المخاطر أمنية → نجيب Tester عنده خبرة في Security Testing.
2. **تطبيق مستوى مناسب من استقلالية الاختبار**
   * أحيانًا نحتاج اختبار مستقل (مثلاً فريق خارجي) عشان نضمن حيادية وموضوعية النتائج.
3. **إجراء المراجعات والتحليل الثابت (Static Analysis & Reviews)**
   * مراجعة المتطلبات، التصميم، والكود مبكرًا → يساعد نكتشف عيوب قبل ما تدخل مرحلة التنفيذ.
4. **اختيار تقنيات وأساليب اختبار مناسبة**
   * حسب طبيعة المخاطر: ممكن نستخدم Boundary Testing، Decision Table، أو Exploratory Testing.
5. **تطبيق أنواع الاختبارات المناسبة**
   * مثلاً:
     + لو الخطر متعلق بالأداء → Performance Testing.
     + لو الخطر متعلق بالأمان → Security Testing.
     + لو الخطر متعلق بالواجهة → Usability Testing.
6. **إجراء الاختبارات الديناميكية (Dynamic Testing)**
   * تشغيل النظام فعليًا واكتشاف العيوب أثناء التشغيل.
   * **Regression Testing** مهم جدًا → يتأكد إن التغييرات ما سببتش مشاكل جديدة.

**🔑 الخلاصة:**

* المنتج فيه مخاطر دايمًا (وظيفية، تقنية، أمنية، أداء...).
* بعد تحليلها، نقرر **إيه نوع الاستجابة** (تخفيف، قبول، تحويل، خطة بديلة).
* الاختبار (بأنواعه وأساليبه) يعتبر **أهم وسيلة لتخفيف مخاطر المنتج** والتقليل من احتمالية أو أثر حدوث العيوب.

## 5.3. Test Monitoring, Test Control and Test Completion

**# Test monitoring**

**Test monitoring** is concerned with gathering information about testing. This information is used to assess test progress and to measure whether the exit criteria or the test tasks associated with the exit criteria are satisfied, such as meeting the targets for coverage of product risks, requirements, or acceptance criteria.

**# Test control**

**Test control** uses the information from test monitoring to provide, in a form of the control directives, guidance and the necessary corrective actions to achieve the most effective and efficient testing. Examples of control directives include:

• Reprioritizing tests when an identified risk becomes an issue

• Re-evaluating whether a test item meets entry criteria or exit criteria due to rework

• Adjusting the test schedule to address a delay in the delivery of the test environment

• Adding new resources when and where needed

**# Test completion**

**Test completion** collects data from completed test activities to consolidate experience, testware, and any other relevant information. Test completion activities occur at project milestones such as when a test level is completed, an agile iteration is finished, a test project is completed (or cancelled), a software system is released, or a maintenance release is completed.

### 5.3.1. Metrics used in Testing

**# Test metrics**

**Test metrics** are gathered to show progress against the planned test schedule and budget, the current quality of the test object, and the effectiveness of the test activities with respect to the test objectives or an iteration goal. Test monitoring gathers a variety of metrics to support the test control and test completion.

Common test metrics include:

• **Project progress metrics** (e.g., task completion, resource usage, test effort)

• **Test progress metrics** (e.g., test case implementation progress, test environment preparation

progress, number of test cases run/not run, passed/failed, test execution time)

• **Product quality metrics** (e.g., availability, response time, mean time to failure)

• **Defect metrics** (e.g., number and priorities of defects found/fixed, defect density, defect detection

percentage)

• **Risk metrics** (e.g., residual risk level)

• **Coverage metrics** (e.g., requirements coverage, code coverage)

• **Cost metrics** (e.g., cost of testing, organizational cost of quality)

### 5.3.2. Purpose, Content and Audience for Test Reports

**# Test Reporting**

**Test reporting** summarizes and communicates test information during and after testing. Test progress reports support the ongoing test control and must provide enough information to make modifications to the test schedule, resources, or test plan, when such changes are needed due to deviation from the plan or changed circumstances. Test completion reports summarize a specific test activity (e.g., test level, test cycle, iteration) and can give information for subsequent testing. During test monitoring and test control, the test team generates test progress reports for stakeholders to keep them informed. Test progress reports are usually generated on a regular basis (e.g., daily, weekly, etc.) and include:

• Testing period

• Test progress (e.g., ahead or behind schedule), including any notable deviations

• Impediments for testing, and their workarounds

• Test metrics (see section 5.3.1 for examples)

• New and changed risks within testing period

• Testing planned for the next period

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/15**/2025]**

تمام ✅ خليني أشرح لك الكلام بالعربي بشكل مبسط:

**تقرير الاختبار (Test Reporting)**

المقصود هنا إننا نعمل **ملخص ونوصل المعلومات الخاصة بالاختبار** (Testing) أثناء الاختبار وبعده. الهدف إن كل الأطراف المعنية (Stakeholders) يكون عندهم صورة واضحة عن:

* مدى التقدم في الاختبار.
* المشاكل أو العوائق اللي ظهرت.
* المخاطر الجديدة.
* الخطة القادمة للاختبار.

**أنواع تقارير الاختبار**

1. **تقارير تقدم الاختبار (Test Progress Reports)**
   * يتم إعدادها أثناء تنفيذ الاختبار.
   * تساعد في **التحكم المستمر بالاختبارات (Test Control)**.
   * لازم تحتوي معلومات كافية عشان نقدر نعدّل في **الجدول الزمني، الموارد، أو خطة الاختبار** إذا حصلت انحرافات أو تغييرات.
   * تُرسل بانتظام (يوميًا، أسبوعيًا، …).

**تشمل عادةً:**

* + فترة الاختبار (Testing period).
  + التقدم في الاختبار (هل نحن أسرع أو أبطأ من الخطة).
  + العوائق (Impediments) والحلول البديلة (Workarounds).
  + مقاييس الاختبار (Test Metrics) مثل عدد الحالات المنفذة/المرفوضة.
  + المخاطر الجديدة أو المتغيرة.
  + الأنشطة المخطط لها في الفترة القادمة.

1. **تقارير إكمال الاختبار (Test Completion Reports)**
   * يتم إعدادها عند انتهاء نشاط اختبار محدد (مستوى اختبار، دورة اختبار، Iteration).
   * هدفها تلخيص ما تم إنجازه وتعطي معلومات تساعد في **الاختبارات المستقبلية**.

**الخلاصة**

* **Test Reporting** = وسيلة لتجميع وتوصيل معلومات عن التقدم والجودة والمشاكل في الاختبار.
* **Test Progress Reports** = متابعة مستمرة أثناء التنفيذ.
* **Test Completion Reports** = ملخص عند انتهاء النشاط.

**# Test Completion Reporting**

A test completion report is prepared during test completion, when a project, test level, or test type is complete and when, ideally, its exit criteria have been met. This report uses test progress reports and other data. Typical test completion reports include:

• Test summary

• Testing and product quality evaluation based on the original test plan (i.e., test objectives and exit criteria)

• Deviations from the test plan (e.g., differences from the planned test schedule, duration, and

effort).

• Testing impediments and workarounds

• Test metrics based on test progress reports

• Unmitigated risks, defects not fixed

• Lessons learned that are relevant to the testing

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/15**/2025]**

تمام ✅ خليني أشرح لك هذا الكلام بالعربي وبشكل مبسّط خطوة خطوة:

**تقرير إكمال الاختبار (Test Completion Report)**

**متى يُعدّ؟**

* يتم إعداده **عند انتهاء نشاط اختبار محدد**، سواء كان:
  + مشروع كامل
  + مستوى اختبار معين (Test Level)
  + نوع اختبار محدد (Test Type)
* يفضل أن يكون التقرير بعد أن **يتم تحقيق المعايير النهائية (Exit Criteria)** الخاصة بالاختبار، أي بعد أن نكون متأكدين أن كل الأهداف المطلوبة للاختبار تحققت أو تمت مراجعتها.

**ما الذي يحتويه التقرير؟**

1. **ملخص الاختبار (Test Summary):**
   * لمحة عامة عن ما تم تنفيذه في الاختبار.
2. **تقييم الجودة للمنتج والاختبار (Testing and Product Quality Evaluation):**
   * تقييم إذا كانت جودة المنتج متوافقة مع خطة الاختبار الأصلية (Objectives & Exit Criteria).
3. **الانحرافات عن خطة الاختبار (Deviations from Test Plan):**
   * أي اختلافات عن ما كان مخططًا له، مثل:
     + الجدول الزمني
     + مدة الاختبار
     + الجهد المبذول
4. **العوائق والحلول البديلة (Testing Impediments and Workarounds):**
   * المشاكل التي واجهت الفريق أثناء الاختبار وكيف تم التعامل معها.
5. **مقاييس الاختبار (Test Metrics):**
   * أرقام وبيانات كمية تم جمعها من تقارير تقدم الاختبار (Progress Reports)، مثل عدد الحالات المنفذة أو الفاشلة.
6. **المخاطر غير المعالجة والعيوب غير المصححة (Unmitigated Risks, Defects Not Fixed):**
   * أي مشاكل أو مخاطر لم يتم حلها أثناء النشاط.
7. **الدروس المستفادة (Lessons Learned):**
   * الخبرات والملاحظات التي يمكن أن تساعد في تحسين الاختبارات القادمة أو تطوير العمليات المستقبلية.

**الخلاصة:**

* تقرير إكمال الاختبار هو **ملخص نهائي للنشاط الاختباري**.
* يعتمد على **تقارير تقدم الاختبار والبيانات الأخرى**.
* يساعد الفرق على **تقييم جودة المنتج، التعرف على الانحرافات، توثيق المخاطر، وتطبيق الدروس المستفادة** في المستقبل.

**# Audience-Specific Test Reporting**

Different audiences require different information in the reports and influence the degree of formality and the frequency of test reporting. Test progress reporting to others in the same team is often frequent and informal, while test completion reporting follows a set template and occurs only once.

The ISO/IEC/IEEE 29119-3 standard includes templates and examples for test progress reports (called test status reports) and test completion reports.

### 5.3.3. Communicating the Status of Testing

**# Test Status Communication Methods**

The best means of communicating test status varies, depending on test management concerns,

organizational test strategies, regulatory standards, or, in the case of self-organizing teams (see section 1.5.2), on the team itself. The options include:

• Verbal communication with team members and other stakeholders

• Dashboards (e.g., CI/CD dashboards, task boards, and burn-down charts)

• Electronic communication channels (e.g., email, chat)

• Online documentation

• Formal test reports (see section 5.3.2)

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/15**/2025]**

تمام ✅ خليني أشرح لك هذا الكلام بالعربي وبشكل مبسط:

**طرق التواصل حول حالة الاختبار (Test Status Communication Methods)**

هذا الجزء يتكلم عن **أفضل الطرق لنقل معلومات حول تقدم الاختبارات وحالتها**، ويشير إلى أن الطريقة تختلف حسب:

* **متطلبات إدارة الاختبار (Test Management Concerns)**: أي ما إذا كانت هناك حاجة لتقارير رسمية أو متابعة يومية.
* **استراتيجيات الاختبار في المؤسسة (Organizational Test Strategies)**: بعض المؤسسات تعتمد على أدوات معينة، وبعضها على الاجتماعات اليومية.
* **المعايير التنظيمية أو القانونية (Regulatory Standards)**: بعض الصناعات تتطلب توثيق رسمي معين.
* **نوع الفريق (Team Type)**: مثل الفرق ذاتية التنظيم (Self-Organizing Teams) قد تفضل أساليب أكثر مرونة وغير رسمية.

**خيارات التواصل الشائعة**

1. **التواصل الشفهي (Verbal Communication)**
   * مثل الاجتماعات اليومية، أو المناقشات مع أعضاء الفريق وأصحاب المصلحة.
   * **ميزة:** سريع ومرن، يسمح بالتفاعل الفوري.
2. **لوحات المعلومات (Dashboards)**
   * مثل:
     + **CI/CD Dashboards** لمتابعة حالة الاختبارات التلقائية.
     + **Task Boards** أو لوحات المهام.
     + **Burn-down Charts** لتتبع تقدم العمل.
   * **ميزة:** تعرض تقدم الاختبارات بشكل بصري وواضح.
3. **قنوات الاتصال الإلكترونية (Electronic Communication Channels)**
   * مثل البريد الإلكتروني أو برامج المحادثة (Chat).
   * **ميزة:** يمكن الوصول إليها بسهولة وتوثيق النقاشات.
4. **التوثيق عبر الإنترنت (Online Documentation)**
   * تخزين المستندات والتقارير على منصات مشتركة مثل Confluence أو SharePoint.
   * **ميزة:** توفر مصدر دائم للوصول للمعلومات.
5. **التقارير الرسمية للاختبار (Formal Test Reports)**
   * تقارير منظمة ومهيكلة حسب معايير المؤسسة أو المعايير الدولية مثل ISO/IEC/IEEE 29119-3.
   * **ميزة:** تستخدم للأغراض الرسمية، مثل التدقيق أو تقديم معلومات إلى إدارة عليا أو جهات تنظيمية.

**الخلاصة**

* **لا توجد طريقة واحدة مناسبة لكل الحالات**؛ اختيار طريقة التواصل يعتمد على **الفريق، المؤسسة، متطلبات الاختبار والمعايير التنظيمية**.
* غالبًا يتم **مزج عدة طرق** للحصول على أفضل تغطية ووضوح للمعلومات.

**# Tailoring Test Communication to Stakeholders**

One or more of these options can be used. More formal communication may be more appropriate for distributed teams where direct face-to-face communication is not always possible due to geographical distance or time differences. Typically, different stakeholders are interested in different types of information, so communication should be tailored accordingly.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/15**/2025]**

تمام ✅ خليني أشرح لك هذا الكلام بالعربي وبشكل مبسّط:

**تكييف طرق التواصل حول الاختبار حسب الفريق وأصحاب المصلحة**

1. **استخدام أكثر من طريقة واحدة:**
   * لا يقتصر الأمر على وسيلة واحدة للتواصل، بل يمكن استخدام **عدة طرق معًا** مثل الاجتماعات، البريد الإلكتروني، لوحات المعلومات، والتقارير الرسمية.
   * الهدف هو التأكد من أن جميع المعلومات المهمة تصل لأصحاب المصلحة المناسبين بطريقة فعّالة.
2. **التواصل الرسمي للفِرق الموزعة (Distributed Teams):**
   * إذا كان الفريق **موزع جغرافيًا** أو يعمل عبر مناطق زمنية مختلفة، غالبًا **التواصل الرسمي يكون أفضل**.
   * السبب: لا يمكن دائمًا التواصل وجهًا لوجه، لذا تحتاج الرسائل الرسمية أو التوثيق لضمان نقل المعلومات بشكل واضح ودقيق.
3. **تكييف المعلومات حسب أصحاب المصلحة:**
   * كل **أصحاب المصلحة** يهتمون بمعلومات مختلفة.
     + مثال: مدير المشروع يريد نظرة عامة على التقدم، بينما فريق الاختبار يحتاج تفاصيل دقيقة عن الأخطاء والاختبارات المنفذة.
   * لذلك يجب **تخصيص محتوى وطريقة التواصل حسب من سيستلم المعلومات**.

**الخلاصة:**

* استخدم **مزيج من وسائل التواصل** حسب الحاجة.
* للفرق الموزعة، من الأفضل استخدام **التواصل الرسمي والموثق**.
* **تكييف المعلومات حسب المستفيدين** يجعل التواصل أكثر فاعلية ويضمن فهم كل طرف لما يحتاجه.

## 5.4. Configuration Management

**# Configuration Management**

In testing, **configuration management (CM)** provides a discipline for identifying, controlling, and tracking work products such as test plans, test strategies, test conditions, test cases, test scripts, test results, test logs, and test reports as configuration items.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/16**/2025]**

تمام ✅ خليني أبسطها:

**إدارة التهيئة (Configuration Management - CM) في الاختبار** معناها إنك تتعامل مع كل مستند أو ملف أو مخرَج من أنشطة الاختبار كأنه **عنصر مُدار ومراقَب**.

🔹 يعني إيه؟

* كل حاجة بتتعمل في الاختبار (زي **خطط الاختبار، استراتيجيات الاختبار، شروط الاختبار، حالات الاختبار، سكربتات الاختبار، النتائج، السجلات، والتقارير**) بتتسجل وتتاخد لها نسخة مُعرَّفة.
* بيبقى في **هوية مميزة (ID)** لكل عنصر علشان مايحصلش لَخبطة.
* كل تغيير يحصل لأي عنصر بيكون **مراقَب ومسجَّل** (مين غيّر؟ إمتى؟ وليه؟).
* ممكن ترجع لأي إصدار قديم علشان تعيد إنتاج نفس نتائج الاختبار لو احتجت.
* بيبقى في **علاقات واضحة** بين العناصر (مثلاً: خطة الاختبار مرتبطة بحالات الاختبار، اللي مرتبطة بسكربتات الاختبار).

🔧 الهدف:

* ضمان **الاتساق (consistency)**.
* الحفاظ على **التتبع (traceability)**.
* منع الأخطاء اللي بتحصل من استخدام نسخ مختلفة أو غير معتمدة.

📌 مثال بسيط:  
تخيل إنك شغال في مشروع وعندك **خطة اختبار** و**100 حالة اختبار**. لو فريقك عدّل على بعض الحالات من غير ما يسجل، ممكن تعمل اختبارات على نسخة قديمة وتجيب نتائج مختلفة. CM يمنع ده، لأنه بيسجل كل تعديل ويحدد النسخة المعتمدة (baseline).

الخلاصة ✅

**إدارة التهيئة (CM) في الاختبار** = نظام بيراقب ويتحكم في كل مستند أو مخرَج من أنشطة الاختبار.

🔑 أهم النقاط:

* كل عنصر له **هوية مميزة** ونسخة معرّفة.
* أي تعديل بيتسجل (مين/إمتى/ليه).
* ممكن ترجع لنسخة قديمة لإعادة النتائج.
* في **ترابط واضح** بين العناصر.

🎯 الهدف:

* ضمان **الاتساق**.
* الحفاظ على **التتبع**.
* تجنّب أخطاء استخدام نسخ غير معتمدة.

**# Configuration Baseline**

For a complex configuration item (e.g., a test environment), CM records the items it consists of, their relationships, and versions. If the configuration item is approved for testing, it becomes a baseline and can only be changed through a formal change control process.

Configuration management keeps a record of changed configuration items when a new baseline is created. It is possible to revert to a previous baseline to reproduce previous test results.

To properly support testing, CM ensures the following:

• All configuration items, including test items (individual parts of the test object), are uniquely

identified, version controlled, tracked for changes, and related to other configuration items so that traceability can be maintained throughout the test process

• All identified documentation and software items are referenced unambiguously in testware

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/16**/2025]**

تمام ✅ خليني أشرح الجزء ده خطوة بخطوة وبطريقة مبسطة:

**1. العنصر المعقّد (Complex Configuration Item)**

مثال: **بيئة الاختبار (Test Environment)**.  
البيئة دي مش حاجة واحدة، دي عبارة عن:

* سيرفرات
* قواعد بيانات
* نسخ من التطبيق
* أدوات اختبار
* إعدادات معينة

إدارة التهيئة (CM) بتسجّل كل ده: **مكوّنات البيئة، إصداراتها، والعلاقات اللي بينهم**.

**2. خط الأساس (Baseline)**

* لما البيئة (أو أي عنصر معقّد) يتم اعتماده للاختبار، بيتحوّل إلى **Baseline**.
* الـ Baseline معناها نسخة ثابتة ومعتمدة، مش مسموح تتغيّر إلا من خلال **عملية رسمية لتغيير التهيئة (Change Control Process)**.

**3. تسجيل التغييرات**

* كل مرة يتعمل تغيير (مثلاً تحديث قاعدة بيانات أو نسخة برنامج)، CM بيسجّل التغيير ويعمل **Baseline جديد**.
* لو حصلت مشكلة أو عايزين نعيد اختبار زي ما كان قبل التغيير، نقدر نرجع للـ **Baseline القديم** ونجيب نفس النتائج بالظبط.

**4. اللي CM بيضمنه لدعم الاختبار**

🔹 **تحديد فريد (Unique Identification):**  
كل عنصر (خطة اختبار، حالة اختبار، كود، مستند) بيبقى له معرف (ID) واضح ومميز.

🔹 **التحكم في النسخ (Version Control):**  
أي تعديل بيتسجل كنسخة جديدة، مع إمكانية الرجوع للقديمة.

🔹 **تتبع التغييرات (Change Tracking):**  
نعرف مين غيّر إيه، ومتى، وليه.

🔹 **العلاقات (Traceability):**  
العناصر مرتبطة ببعض (مثلاً: حالة اختبار مرتبطة بشرط اختبار، ودي مرتبطة بمتطلب). ده بيضمن إنك تعرف أثر أي تغيير.

🔹 **مرجع واضح (Unambiguous Reference):**  
كل مستند أو برنامج مستخدم في الاختبار بيتحدد بشكل واضح في الـ Testware، علشان مفيش التباس بين نسخ مختلفة.

**5. مثال عملي**

افترض عندك **بيئة اختبار** فيها:

* قاعدة بيانات بإصدار 12
* تطبيق بإصدار 3.5
* 50 حالة اختبار مرتبطة بالمتطلبات

📌 CM يسجل الكلام ده كـ Baseline رقم 1.  
لو حد عدّل التطبيق إلى الإصدار 3.6، CM يعمل Baseline رقم 2.  
لو لقيت Bug ورجعت تجرب على الإصدار القديم 3.5، ترجع بسهولة للـ Baseline رقم 1 علشان تعيد النتائج.

🔑 **الخلاصة:**  
إدارة التهيئة (CM) في الاختبار بتديك **تحكم كامل، إمكانية تتبع، والقدرة على إعادة الاختبارات بدقة**.

**# Automated Configuration Management in CI/CD**

Continuous integration, continuous delivery, continuous deployment and the associated testing are typically implemented as part of an automated DevOps pipeline (see section 2.1.4), in which automated CM is normally included.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/16**/2025]**

تمام ✅ خليني أوضحها بشكل مبسط:

**1. الممارسات المستمرة في DevOps**

* **الدمج المستمر (Continuous Integration - CI):**  
  المطورين كل شوية يدمجوا (يضمّوا) التعديلات الجديدة في الكود الرئيسي، والدمج ده بيكون آلي مع بناء الكود وتشغيل اختبارات سريعة.
* **التسليم المستمر (Continuous Delivery - CD):**  
  بعد الدمج والبناء والاختبار، المنتج بيبقى جاهز يتسلّم في أي وقت (لكن النشر الفعلي لبيئة الإنتاج لسه بقرار بشري).
* **النشر المستمر (Continuous Deployment):**  
  الخطوة الأوتوماتيكية اللي بعد التسليم المستمر → أي كود ناجح في الاختبارات بيتنشر مباشرة في بيئة الإنتاج **من غير تدخل بشري**.

**2. الاختبارات المرافقة**

كل خطوة في CI/CD بيكون فيها **اختبارات آلية** (مثل اختبارات الوحدة، التكامل، الأداء) علشان نضمن إن الكود صالح قبل ما يتحرك للخطوة اللي بعدها.

**3. إدارة التهيئة الآلية (Automated CM)**

* في الـ DevOps pipeline، إدارة التهيئة (CM) بتكون أوتوماتيكية.
* يعني أي تغييرات في الكود، الإعدادات، أو البُنى التحتية بتتسجل وتتتبع آليًا.
* ده بيضمن إنك تعرف إيه النسخة اللي اتبنت، اتجرّبت، أو اتنشرت، وبالتالي تحافظ على **التتبع (traceability)** و**التحكم (control)**.

📌 **الخلاصة:**  
الـ CI/CD/CD + الاختبارات = عملية أوتوماتيكية بالكامل في DevOps pipeline، والـ CM بيدخل أوتوماتيكي فيها علشان يراقب ويتحكم في الإصدارات والتغييرات بشكل سلس.

تحب أعملك **رسم مبسط (Pipeline Flow)** يوضح إزاي CI → CD → CD + CM متوصلين مع بعض؟

## 5.5. Defect Management

**# defect management**

Since one of the major test objectives is to find defects, an established **defect management** process is essential. Although we refer to "defects" here, the reported anomalies may turn out to be real defects or something else (e.g., false–positive result, change request) - this is resolved during the process of dealing with the defect reports. Anomalies may be reported during any phase of the SDLC and the form depends on the SDLC. At a minimum, the defect management process includes a workflow for handling individual defects or anomalies from their discovery to their closure and rules for their classification. The workflow typically comprises activities to log the reported anomalies, analyze and classify them, decide on a suitable response such as to fix or keep it as it is and finally to close the defect report. The process must be followed by all involved stakeholders. It is advisable to handle defects from static testing (especially static analysis) in a similar way.

Typical defect reports have the following objectives:

• Provide those responsible for handling and resolving reported defects with sufficient information to resolve the issue

• Provide a means of tracking the quality of the work product

• Provide ideas for improvement of the development and test process

A defect report logged during dynamic testing typically includes:

• Unique identifier

• Title with a short summary of the anomaly being reported

• Date when the anomaly was observed, issuing organization, and author, including their role

• Identification of the test object and test environment

• Context of the defect (e.g., test case being run, test activity being performed, SDLC phase, and

other relevant information such as the test technique, checklist or test data being used)

• Description of the failure to enable reproduction and resolution including the test steps that

detected the anomaly, and any relevant test logs, database dumps, screenshots, or recordings

• Expected results and actual results

• Severity of the defect (degree of impact) on the interests of stakeholders or requirements

• Priority to fix

• Status of the defect (e.g., open, deferred, duplicate, waiting to be fixed, awaiting confirmation

testing, re-opened, closed, rejected)

• References (e.g., to the test case)

Some of this data may be automatically included when using defect management tools (e.g., identifier, date, author and initial status). Document templates for a defect report and example defect reports can be found in the ISO/IEC/IEEE 29119-3 standard, which refers to defect reports as incident reports.

**# Source**: **Chatgpt (GPT-5-Standard) at [**9/16**/2025]**

تمام ✅ خليني أبسطلك الكلام ده خطوة بخطوة:

**1. الهدف من إدارة العيوب (Defect Management Process)**

واحد من أهم أهداف الاختبار هو **اكتشاف العيوب (Defects)**. علشان كده لازم يكون فيه **عملية مُعتمدة لإدارة العيوب**.

لكن:

* مش كل حاجة يتم التبليغ عنها لازم تكون **عيب فعلي**.
* ممكن يكون:
  + **عيب حقيقي** في النظام.
  + **إنذار كاذب (False Positive)**.
  + **طلب تغيير (Change Request)**.  
    👉 العملية نفسها هي اللي بتحدد وتفرز وتوضح ده.

**2. أين ومتى يتم التبليغ عن العيوب؟**

* العيوب أو الشذوذات (Anomalies) ممكن يتم اكتشافها في أي مرحلة من **SDLC** (دورة حياة تطوير البرمجيات).
* شكل تقرير العيب بيتغيّر حسب المرحلة.

**3. ما تحتويه عملية إدارة العيوب**

العملية تشمل على الأقل:

1. **تسجيل (Log)** العيوب اللي تم الإبلاغ عنها.
2. **تحليل وتصنيف (Analyze & Classify)** العيوب (مثلاً حسب النوع أو الخطورة).
3. **تحديد الاستجابة المناسبة (Response Decision):**
   * يتصلح.
   * يفضل زي ما هو (مش عيب).
4. **إغلاق التقرير (Close Report)** بعد التعامل معاه.

🔑 القاعدة: العملية لازم يلتزم بيها **كل الأطراف المعنية** (مطورين، مختبرين، مدراء مشاريع...).

**4. أهداف تقرير العيب (Defect Report Objectives)**

* تدي معلومات كافية للناس المسؤولة علشان يقدروا **يحلّوا العيب**.
* تدي وسيلة لمتابعة **جودة المنتج**.
* تساعد بأفكار لتحسين **عملية التطوير والاختبار**.

**5. محتويات تقرير العيب (Typical Defect Report)**

📋 التقرير الجيد لازم يحتوي على:

* **معرّف فريد (ID)** للعيب.
* **عنوان مختصر** يوضح العيب.
* **تاريخ** وبيانات الجهة أو الشخص اللي اكتشفه.
* **تحديد عنصر الاختبار (Test Object)** وبيئة الاختبار.
* **سياق العيب** (في أي اختبار/نشاط/مرحلة تم اكتشافه).
* **وصف للفشل** مع خطوات إعادة الإنتاج (Reproduction Steps).
* **النتائج المتوقعة** والنتائج الفعلية.
* **درجة الخطورة (Severity)** = قد إيه العيب مؤثر.
* **الأولوية (Priority)** = مدى السرعة المطلوبة لإصلاحه.
* **الحالة (Status):** مفتوح، مؤجل، مكرر، تحت الإصلاح، مغلق...
* **مراجع (References):** زي رقم حالة الاختبار.

**6. أتمتة وإدارة الأدوات**

بعض الأدوات بتعمل **ملء تلقائي** لبعض البيانات (زي: ID، التاريخ، الشخص المُبلّغ، الحالة الأولية).

**7. المعايير القياسية**

المعايير الدولية (زي **ISO/IEC/IEEE 29119-3**) عندها قوالب جاهزة ونماذج لتقارير العيوب.

* في المعايير دي، تقرير العيب بيتسمى أحيانًا: **Incident Report**.

📌 **الخلاصة:**

* إدارة العيوب = عملية منظمة للتعامل مع أي شذوذ/مشكلة من لحظة اكتشافها لحد ما تتقفل.
* تقرير العيب لازم يكون شامل علشان يساعد في **الإصلاح، المتابعة، والتحسين**.
* الأدوات والمعايير بتساعد في عمل العملية بشكل أوتوماتيكي ومنظم.