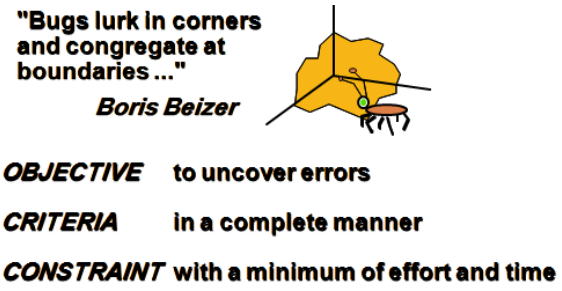
تقنيات فحص البرمجيات

Software Testing Technologies

تصميم حالات الاختبار Test Case Design (---- الموضوع الأهم في فحص البرمجيات ----)

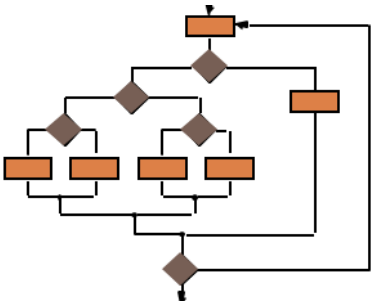
سنتعلم كيف نصمم حالات الاختبار



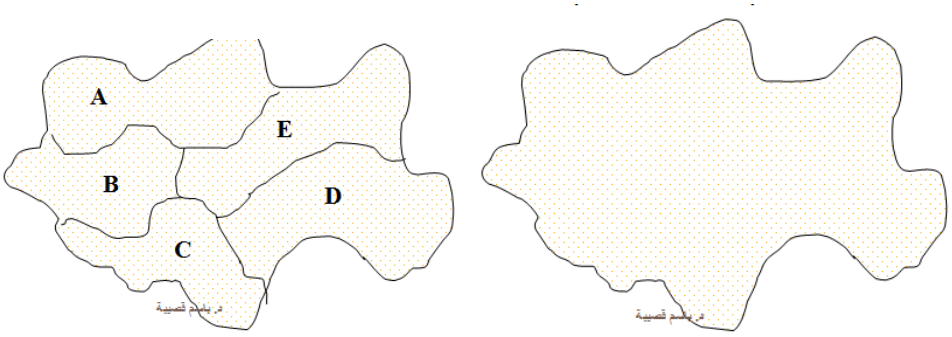
تاريخ: عندما تعطل أول حاسب (وكان عملاقا) جاء المهندسون لعند كبيرهم وسألهم ما الخطب فقالوا Bug أي أن حشرة كانت سبب التماس الذي عطل الحاسب وأصبحت مثلاً على سبيل الطرفة فعندما لا يعمل برنامج ما نقول Bug.

القدر: أن الحشرات تتجمع في زوايا المنزل وتمشي على الفاصل (الحدود) بين الأرض والجدار وكذلك أخطاء البرامج هي في المناطق الحدية (مثلاً .. عند نهاية مصفوفة : تعمل for ولكنها تتجاوز بعد المصفوفة بواحد) وهذا ما نسميه بالعيب Fault ولكن ليس من الضرورة أن يظهر فورا فإن كنت تعبر المصفوفة باحثاً عن عنصر موجود فيها فلن تصل إلى النهاية ولن ترى العيب .. وإن لم يكن العنصر غير موجود فستصل إلى ما بعد نهاية المصفوفة وينتج خطأ Error و لو أوقف هذا الخطأ البرنامج لأصبح إخفاق Failure

لذلك لاكتشاف الأخطاء نحتاج إلى تصميم حالات اختبار .. كل حالة اختبار هي مجموعة دخل للبرنامج (مثل في برنامج البحث عن عنصر من مصفوفة ... حالة الفحص = البحث عن عنصر غير موجود في المصفوفة .. للتأكد من أن حلقة عبور المصفوفة لا تتجاوز حدود المصفوفة)



إن أي برنامج يقبل عدد لا نهائي من تشكيلات الدخل (برنامج جمع عددين ..2+3 .. 8+5 ..22+44 .. إلخ) فمن المستحيل اختبار كل تشكيلة دخل ممكنة وبالتالي من المستحيل إختبار كل التشكيلات الممكنة.. بل أخذ تقسيم كل تشكيلات الدخل الممكنة إلى صفوف تكافؤ بحيث أي اختبار لنقطة (تشكيلة دخل) من هذا الصف التكافؤ كما لو أنك اختبرت كل نقاطه ..

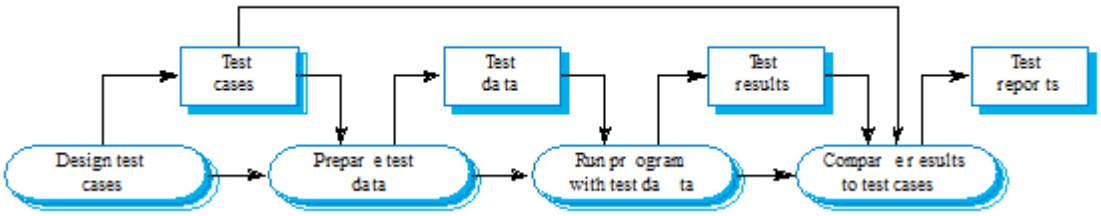


كيف نقسم فضاء تشكيلات الدخل (حالات الفحص) إلى صفوف تكافؤ من حالات الفحص (تشكيلات الدخل) بالحفاظ على معايير تصميم حالات الفحص:

- اكتشاف الأخطاء
- بشكل كامل (دون أن ننسى اكتشاف أحدها)
- بأقل جهد ووقت ممكن (بأقل عدد ممكن من حالات الفحص)

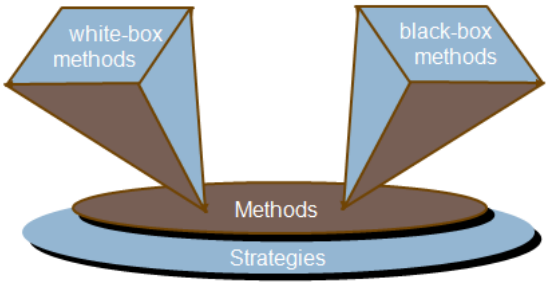
أي اختبار العدد الأصغري من حالات الفحص واكتشاف كل الأخطاء

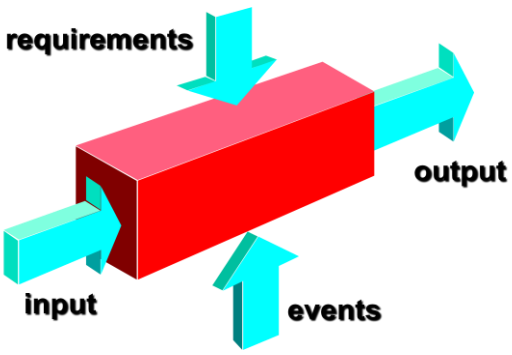
يبين الشكل التالي إجرائية تصميم حالات الاختبار واختبار النظام (واضحة)



لدينا منهجيتين لتصميم حالات الفحص:

- White Box: باختصار سنعتمد على الكود لاستنتاج حالات الفحص.
- Black Box: باختصار سنعتمد على توصيف وظائف النظام لاستنتاج حالات الفحص.

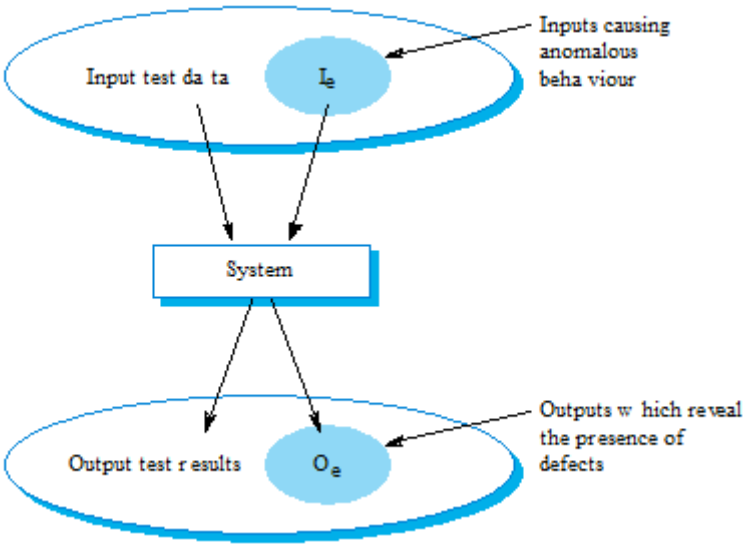




منهجية Black BOX

في هذه التقنية نرى البرنامج كصندوق أسود (له متطلبات تقول لنا من أجل دخل معين ما هو الخرج الذي يمكننا الحصول عليه) ولا تهمنا تفاصيل التنفيذ والتحقيق. ونركز فيه على وجهة نظر المستخدم.

عندما نهتم ببرمجيات الحاسوب، يشير اختبار الصندوق الأسود إلى الاختبارات التي تُجرى لواجهة البرمجيات. ومع أن اختبارات الصندوق الأسود مصممة لكشف الأخطاء، فإنها تستخدم لإثبات أن وظائف البرنامج تعمل، وأن المدخلات تُقبل قَبولاً صحيحاً وأن المخرجات تولّد توليداً صحيحاً، وأنّ سلامة المعلومات الخارجية (أي ملفات المعطيات) مصونة. يقوم اختبار الصندوق الأسود بتفحص بعض النواحي الأساسية للنظام، ولا يُلقى الضوء على البنية المنطقية الداخلية للبرمجيات إلا قليلاً.



لذلك نحاول في هذه الطريقة إيجاد الدخل الذي يسبب خرجاً شاذاً عن مواصفات عمل البرنامج (المتطلبات) كما هو مبين بالشكل السابق.

بالطبع، يجب أن تخطر ببالك كيف يمكن اختبار إقرار الصلاحية لبرنامج واختبار الأداء (كيف يمكن تقسيم الدخل إلى صفوف تكافؤ لإيجاد حالات اختبار جيدة؟):

- هل النظام حساس لبعض قيم الدخل (يجب إيجادها)؟
- كيف يمكن إيجاد الحدود بين صفوف مجموعات الدخل (تذكر الأخطاء تختبيء في الزوايا وتسير ما بين الحدود)؟
- ما هي تراكيب الدخل التي تؤثر على عمليات النظام؟
- ماهي حجوم ومعدلات المعطيات التي يمكن للنظام معالجتها ويبقى الأداء مقبولاً؟

لنأخذ مثالاً بسيطاً جداً **(تابع للبحث عن عنصر (عدد صحيح) ضمن مصفوفة أعداد صحيحة).**

ويعبر الشكل التالي عن توصيف لمتطلبات برنامج البحث عن عدد صحيح في مصفوفة للأعداد الصحيحة:

```
procedure Search (Key : INTEGER ; T: array 1..N of
INTEGER;  Found : BOOLEAN; L: 1..N) ;
Pre-condition
        -- the array has at least one element
        1 <= N
Post-condition
        -- the element is found and is referenced by L
        ( Found and T (L) = Key)
        or
        -- the element is not in the array
        ( not Found and
        not (exists i, 1 >= i >= N, T (i) = Key ))
```

لتقسيم الدخل إلى صفوف نستخدم المحاكمة التالية:

لدينا صنفين رئيسين للدخل:

- صف دخل لا يوافق للشروط المسبقة (في مسألتنا المصفوفة خالية)
- صف دخل موافق للشروط المسبقة (في مسألتنا المصفوفة غير خالية)
  - صف دخل يحقق نتيجة سلبية (في مسألتنا المصفوفة غير خالية ولا تحوي العنصر موضوع البحث)
    - عدد عناصر المصفوفة واحد فقط.
    - عدد عناصر المصفوفة أكبر من واحد.
  - صف دخل يحقق نتيجة إيجابية (في مسألتنا المصفوفة غير خالية وتحوي العنصر موضوع البحث)
    - العنصر موضوع البحث موجود في المصفوفة وعدد عناصر المصفوفة واحد فقط
    - العنصر موضوع البحث في أول المصفوفة وعدد عناصر المصفوفة أكبر من واحد
    - العنصر موضوع البحث في نهاية المصفوفة وعدد عناصر المصفوفة أكبر من واحد
    - العنصر موضوع البحث في منتصف المصفوفة وعدد عناصر المصفوفة أكبر من واحد

والشكل التالي يوضح ببساطة Black Box Testing لتابع البحث عن عدد صحيح في مصفوفة.

Array	Element
Single value	In array
Single value	Not in array
More than 1 value	First element in array
More than 1 value	Last element in array
More than 1 value	Middle element in array
More than 1 value	Not in array

Search Routine - Test Cases

Input array (T)	Key (Key)	Output (Found, L)
17	17	true, 1
17	0	false, ??
17, 29, 21, 23	17	true, 1
41, 18, 9, 31, 30, 16, 45	45	true, 6
17, 18, 21, 23, 29, 41, 38	23	true, 4
21, 23, 29, 33, 38	25	false, ??

لتحديد صفوف التكافؤ للدخل لدينا القواعد التالية:

- ☐ If an input condition specifies **a range**, one valid and two invalid equivalence classes are defined
  - **Input range: 1 – 10**
  - **Eq classes: {1..10},**
  - **{x < 1}, {x > 10}**
- ☐ If an input condition requires **a specific value**, one valid and two invalid equivalence classes are defined
  - **Input value: 250**
  - **Eq classes: {250},**
  - **{x < 250}, {x > 250}**

- If an input condition specifies **a member of a set,** one valid and one invalid equivalence class are defined
  - **Input set: {-2.5, 7.3, 8.4}**
  - **Eq classes: {-2.5, 7.3, 8.4},**
  - **{any other x}**
- If an input condition is **a Boolean value**, one valid and one invalid class are define
  - **Input: {true condition}**
  - **Eq classes: {true condition},**
  - **{false condition}**

على العكس من اختبار الصندوق الأبيض الذي ينفَّذ في مرحلة مبكرة من إجرائية الاختبار، ثمة نزعة إلى تطبيق اختبار الصندوق الأسود في مراحل متأخرة من الاختبار

يركّز اختبار الصندوق الأسود black-box testing الاهتمام على المتطلبات الوظيفية للبرمجيات، التي تمكّن مهندس البرمجيات من اشتقاق مجموعات من شروط الدخل التي تقوم بتجريب المتطلبات الوظيفية الكاملة للبرنامج. لا يعد اختبار الصندوق الأسود بديلاً من تقنيات الصندوق الأبيض، بل هو منهج متمم له، ويحتمل أن يعثر على صنف مختلف من الأخطاء التي لا تستطيع طرائق الصندوق الأبيض اكتشافها.

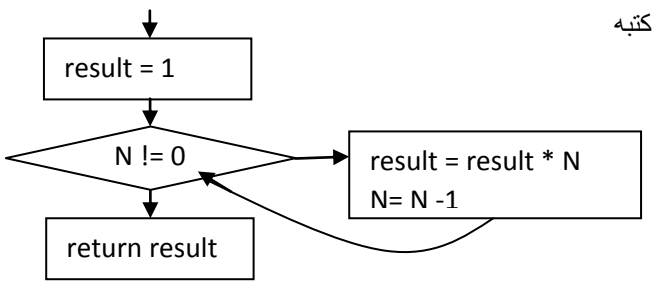
لشرح النقطة الأخيرة لنتذكر بأن اختبارات White Box Testing تعتمد على تفاصيل التنفيذ وبالتالي تعتمد على وجهة نظر المطور المبرمج من المسألة المراد حلها وقد لا تطابق هذه الوجهة تماماً مع المسألة (بل ضمن زاوية معينة) وبالتالي قد لا تساعدنا على اكتشاف كل الأخطاء الموجودة في البرنامج.

مثال: لنفرض أن مبرمج كتب برنامج لحساب العاملي كما يلي:

```
Factorial(int N){
        result = 1;
        while(N!=0){
                result = result * N
                N = N-1
        }
        return result
}
```

قام هذا المطور باختبار White Box للبرنامج الذي كتبه

الخطوة 1 : رسم Control Flow Graph



الخطوة 2 : تحديد المسارات وحالات الاختبار

لدينا مسارين: الأول دون الدخول بالحلقة (N=0 ➔ 1) والثاني بالدخول بالحلقة (N=3 ➔ 6 :Ex)

وبالتالي عند تجريب البرنامج مع الدخل N=0 و N=3 سيعطي البرنامج نتائج صحيحة تماما أي أن البرنامج اجتاز اختبارات الصندوق الأبيض وعليه أن يكون صحيح تماما ... ولكن هذا ليس صحيح تماما لأن المبرمج غاب عن نظره أن يختبر الدخل إن كان سالباً (ومن المعروف ألا عاملي للعدد السالب وهنا سيعطي الرنامج نتيجة خاطئة)

لماذا حدث هذا لأن اختبار الصندوق الأبيض يختبر زاوية المطور من المسألة وهنا كل مافعله المبرمج (ضمن هذه الزاوية) كان صحيحا واختبار الصندوق الأبيض أقر بذلك

<u>لنختبر البرنامج باستخدام الصندوق الأسود:</u>

إن الشروط الابتدائية لمسألة العاملي هو أن يكون الدخل عدداً صحيحاً موجباً

لدينا صنفين رئيسين للدخل:

- صف دخل لا يوافق للشروط المسبقة (في مسألتنا الدخل عدد سالب والنتيجة رسالة خطأ للمستخدم)
- صف دخل موافق للشروط المسبقة (في مسألتنا الدخل عدد صحيح موجب)
    - صف دخل يحقق نتيجة إيجابية)
        - العدد صفر (عند الحدود والنتيجة 1)
        - أي عدد موجب(وليكن 3 والنتيجة 6)

وهنا نرى بأن اختبار الصندوق الأسود سيكتشف خطأ عند إدخال عدد سالب (لن نحصل على رسالة خطأ وإنما سنحصل على رقم وهذا مخالف لمواصفات مسألة العاملي)