CSE 403
Software Engineering
Spring 2023

#16: Coverage-based Testing
## Logistics

### WEEK 6

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/01</td>
<td>L: Test Coverage</td>
<td></td>
</tr>
<tr>
<td>05/02</td>
<td>T:</td>
<td>DUE: TCC!!!</td>
</tr>
<tr>
<td>05/03</td>
<td>L: Mutation Testing</td>
<td>Alpha_Release (R1)</td>
</tr>
<tr>
<td>05/04</td>
<td>P:</td>
<td></td>
</tr>
<tr>
<td>05/05</td>
<td>LX: Code Defenders</td>
<td></td>
</tr>
</tbody>
</table>
This week: test efficacy and adequacy

- Coverage-based testing
- Mutation-based testing
- In-class exercise
Structural code coverage: motivating example

```shell
(LITW_API_BS) nigin@librarian-xps:~/WORKSPACE/LITW/litw-api $ PYTHONPATH=../src/ hatch run test:run
```

Test session starts

```
platform linux -- Python 3.9.12, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigin/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2, cov-4.0.0
collected 10 items

src/litw/api/tests/test_api.py ....
src/litw/api/tests/test_model.py ..
src/litw/api/tests/test_mongo.py ....
```

10 passed in 1.41s
test.py3.10

Test session starts

```
platform linux -- Python 3.10.7, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigin/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2, cov-4.0.0
collected 10 items

src/litw/api/tests/test_api.py ....
src/litw/api/tests/test_model.py ..
src/litw/api/tests/test_mongo.py ....
```

10 passed in 1.42s
Structural code coverage: motivating example

```
[tool.hatch.envs.test]
dependencies = [
    "coverage[toml]",
    "pytest"
]

[tool.coverage.run]
source = ["litw"]
omit = ["**/test**"]

[tool.hatch.envs.test.scripts]
run-coverage = "coverage run -m pytest; coverage report"
run = "pytest"
```
Structural code coverage: motivating example

```bash
(LITW_API_BS) nigini@librarian-xps:~/WORKSPACE/LITW/litw-api$ PYTHONPATH=. /src/hatch run test:run-coverage test.py3.9

--- test session starts ----------------------------------------
platform linux -- Python 3.9.12, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigini/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2, cov-4.0.0
collected 10 items

src/litw/api/tests/test_api.py ....
src/litw/api/tests/test_model.py ...
src/litw/api/tests/test_mongo.py ....

--- 10 passed in 1.60s -----------------------------------------

/home/nigini/.local/share/hatch/env/virtual/litw-api/tA-wS_ke/test.py-3.9/lib/python3.9/site-packages/coverage/file.py:32: UserWarning: module litw was previously imported, but not measured (module-not-measured)
    self.warn(msg, slug="module-not-measured")

<table>
<thead>
<tr>
<th>Name</th>
<th>Stmts</th>
<th>Miss</th>
<th>Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td>src/litw/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/<strong>about</strong>.py</td>
<td>1</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>src/litw/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api.py</td>
<td>45</td>
<td>4</td>
<td>91%</td>
</tr>
<tr>
<td>src/litw/api/data/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/data/model.py</td>
<td>28</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/data/mongo.py</td>
<td>90</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/security.py</td>
<td>92</td>
<td>15</td>
<td>84%</td>
</tr>
<tr>
<td>src/litw/api/tests/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/util.py</td>
<td>14</td>
<td>2</td>
<td>86%</td>
</tr>
<tr>
<td>src/litw/settings.py</td>
<td>11</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

---

TOTAL 281 22 92%
```
### Structural code coverage: motivating example

#### Coverage report: 92%

Coverage report created on 2023-04-30 14:16:07.000

<table>
<thead>
<tr>
<th>Module</th>
<th>statements</th>
<th>missing</th>
<th>excluded</th>
<th>coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>src/litw/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/<strong>about</strong>.py</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>src/litw/api/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/api.py</td>
<td>45</td>
<td>4</td>
<td>0</td>
<td>91%</td>
</tr>
<tr>
<td>src/litw/api/data/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/data/model.py</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/data/mongo.py</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/security.py</td>
<td>92</td>
<td>15</td>
<td>0</td>
<td>84%</td>
</tr>
<tr>
<td>src/litw/api/tests/<strong>init</strong>.py</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>src/litw/api/util.py</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>86%</td>
</tr>
<tr>
<td>src/litw/settings.py</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>281</strong></td>
<td><strong>22</strong></td>
<td><strong>0</strong></td>
<td><strong>92%</strong></td>
</tr>
</tbody>
</table>

Coverage report created on 2023-04-30 14:16:07.000
Structural code coverage: motivating example

```python
@app.post("/studies")
async def post_study(study_name: str, user: dict = Depends(user_authorization)):
    return {}  

@app.get("/studies/{study_id}")
async def get_studies(study_id: UUID, study: dict = Depends(study_authorization)):
    if study["id"] == str(study_id):
        return study
    else:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="You don't have access to the study: {}.".format(study_id)
        )

@app.post("/studies/{study_id}/data")
async def post_study_data(study_id: UUID, study_data: dict, study: dict = Depends(study_authorization)):
    if study["id"] == str(study_id):
        data_access = DataAccessFactory()
        study_data_access: StudyDataMongo = data_access.access_points[data_access.access_studies[study_id].data_access]
        result = study_data_access.add_data(str(study_id), study_data)
        return result
    else:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="You don't have access to the study: {}.".format(study_id)
        )
```

Code coverage metrics

- Statement coverage
- Branch coverage
  - Condition coverage
  - Decision coverage
  - Modified Condition/Decision coverage
public double avgAbs(double ... numbers) {

    // We expect the array to be non-null and non-empty
    if (numbers == null || numbers.length == 0) {
        throw new IllegalArgumentException("Array numbers must not be null or empty!");
    }

    double sum = 0;
    for (int i=0; i<numbers.length; ++i) {
        double d = numbers[i];
        if (d < 0) {
            sum -= d;
        } else {
            sum += d;
        }
    }

    return sum/numbers.length;
}
What’s the control flow graph (CFG) for this method?
Average of the absolute values of an array of doubles

```java
public double avgAbs(double ... numbers) {

    // We expect the array to be non-null and non-empty
    if (numbers == null || numbers.length == 0) {
        throw new IllegalArgumentException("Array numbers must not be null or empty!");
    }

    double sum = 0;
    for (int i=0; i<numbers.length; ++i) {
        double d = numbers[i];
        if (d < 0) {
            sum -= d;
        } else {
            sum += d;
        }
    }

    return sum/numbers.length;
}
```

What’s the control flow graph (CFG) for this method?
**Structural code coverage: the basics**

```
Entry point

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>a == null</td>
<td></td>
</tr>
<tr>
<td>sum = 0</td>
<td></td>
</tr>
<tr>
<td>i = 0</td>
<td></td>
</tr>
</tbody>
</table>

i < a.length

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>num = a[i]</td>
</tr>
<tr>
<td>false</td>
<td>return sum / a.length</td>
</tr>
</tbody>
</table>

num < 0

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>sum -= num</td>
</tr>
<tr>
<td>false</td>
<td>sum += num</td>
</tr>
</tbody>
</table>

++i

Normal exit

Exceptional exit
```
public double avgAbs(double ... numbers) {

    // We expect the array to be non-null and non-empty
    if (numbers == null || numbers.length == 0) {
        throw new IllegalArgumentException("Array numbers must not be null or empty!");
    }

    double sum = 0;
    for (int i=0; i<numbers.length; ++i) {
        double d = numbers[i];
        if (d < 0) {
            sum -= d;
        } else {
            sum += d;
        }
    }

    return sum/numbers.length;
}
Statement coverage

- Every statement in the program must be executed at least once.
```java
if (a == null || a.length == 0) {
    throw new IllegalArgumentException("Array a must not be null or empty!");
}
sum = 0
i = 0
while (i < a.length) {
    num = a[i]
    if (num < 0) {
        sum += num
    } else {
        sum -= num
    }
    ++i
}
return sum / a.length
```
Statement coverage

- **Every statement** in the program must be executed at least once.
- Given the control-flow graph (CFG), this is equivalent to node coverage.
Branch coverage: Condition vs. Decision
Branch coverage: Condition vs. Decision

Terminology

- **Condition**: a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).

- **Decision**: a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).

- **Example**: if \((a \mid b)\) { … }
  
  - *a and b are conditions.*
  
  - The boolean expression \(a \mid b\) is a decision.
Branch coverage: Condition vs. Decision

Terminology

- **Condition**: a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).

- **Decision**: a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).

- **Example**: if \((a \mid b)\) { … }
  
  - \(a\) and \(b\) are **conditions**.
  
  - The boolean expression \((a \mid b)\) is a **decision**.
Decision coverage

- **Every decision** in the program must take on all possible outcomes (true/false) at least once.
Decision coverage

Entry point

\[ a == \text{null} \quad \text{||} \quad a.\text{length} == 0 \]

sum = 0
i = 0

i < a.length

num = a[i]

num < 0

sum += num

++i

true

false

false

true

true

false

false

true

false

return sum/a.length

throw new IllegalArgumentException("Array a must not be null or empty!")

Exceptional exit

Normal exit
Branch coverage: **Condition** vs. **Decision**

**Terminology**

- **Condition**: a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).
- **Decision**: a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).
- **Example**: if \((a | b)\) { ... }
  - \(a\) and \(b\) are *conditions*.
  - The boolean expression \(a | b\) is a *decision*.
Condition coverage

- **Every condition** in the program must take on **all possible outcomes** (true/false) at least once.
Condition coverage

Entry point

\[ \text{a==null} \quad \text{||} \quad \text{a.length==0} \]

\[ \text{true} \quad \text{throw new IllegalArgumentException(} \quad \text{"Array a must not be null or empty!"}) \]

\[ \text{false} \quad \text{Exceptional exit} \]

\[ \text{sum = 0} \]
\[ \text{i = 0} \]

\[ \text{i< a.length} \]

\[ \text{false} \quad \text{return sum/a.length} \]

\[ \text{true} \quad \text{Normal exit} \]

\[ \text{num = a[i]} \]

\[ \text{num < 0} \]

\[ \text{false} \quad \text{sum += num} \]

\[ \text{true} \quad \text{sum -= num} \]

\[ \text{++i} \]
Code coverage metrics

- Statement coverage
- Branch coverage
  - Condition coverage
  - Decision coverage
  - Modified Condition/Decision coverage
Structural code coverage: subsumption

Given two coverage criteria A and B, A subsumes B iff satisfying A implies satisfying B

- Subsumption relationships (True or False):
  1. Does statement coverage subsume decision coverage?
  2. Does decision coverage subsume statement coverage?
  3. Does decision coverage subsume condition coverage?
  4. Does condition coverage subsume decision coverage?

https://tinyurl.com/cse403-cov
Structural code coverage: subsumption

Given two coverage criteria A and B, A subsumes B iff satisfying A implies satisfying B

Subsumption relationships (True or False):

1. Does **statement** coverage subsume **decision** coverage?
2. Does **decision** coverage subsume **statement** coverage?
3. Does **decision** coverage subsume **condition** coverage?
4. Does **condition** coverage subsume **decision** coverage?

The only correct statement in #2!!!
Decision coverage vs. condition coverage

4 possible tests for the decision $a \mid b$:

1. $a = 0, b = 0$
2. $a = 0, b = 1$
3. $a = 1, b = 0$
4. $a = 1, b = 1$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$a \mid b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Satisfies condition coverage but not decision coverage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$a \mid b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Does not satisfy condition coverage but decision coverage

Neither coverage criterion subsumes the other!
Modified Condition/Decision Coverage (MC/DC)
Modified Condition/Decision Coverage (MC/DC)

Do not confuse… 🤘
MCDC: Modified condition and decision coverage

- **Every decision** in the program must take on all possible outcomes (true/false) **at least once**
- **Every condition** in the program must take on all possible outcomes (true/false) **at least once**
- **Each condition** in a decision has been shown to **independently affect that decision’s outcome**. (A condition is shown to independently affect a decision’s outcome by: varying just that condition while holding fixed all other possible conditions.)

Required for safety critical systems (DO-178B/C)
MC/DC: an example

if (a | b)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

MCDC
- Decision coverage
- Condition coverage
- Each condition shown to independently affect outcome

Which tests (combinations of a and b) satisfy MCDC?
MC/DC: an example

if (a | b)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

MCDC
- Decision coverage
- Condition coverage
- Each condition shown to independently affect outcome

MCDC is still cheaper than testing all possible combinations.
MC/DC: another example

if (a || b)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**MCDC**
- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to independently affect outcome

Why is this example different?
MC/DC: another example

if (a || b)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

MCDC
- Decision coverage
- Condition coverage
- Each condition shown to independently affect outcome

Short-circuiting operators may not evaluate all conditions.
MC/DC: yet another example

if (!a) ... if (a || b)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

MCDC

- Decision coverage
- Condition coverage
- Each condition shown to independently affect outcome

What about this example?
MC/DC: another example

if (!a) ... if (a || b)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

MCDC
- Decision coverage
- Condition coverage
- Each condition shown to independently affect outcome

Not all combinations of conditions may be possible.
MCDC: complex expressions

Provide an MCDC-adequate test suite for:

1. $a \mid b \mid c$
2. $a \& b \& c$
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### a & b & c

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Structural code coverage: summary

- Code coverage is easy to compute.
- Code coverage has an intuitive interpretation.
- Code coverage in industry: Code coverage at Google
- Code coverage itself is not sufficient!