LABORATORIUM 7
Walidacja i testy akceptacyjne systemu XR

testy jednostkowe • testy integracyjne • wydajność • użyteczność • smoke tests • przypadki brzegowe

Problem inżynierski

Czy system XR (wszystkie komponenty Lab 1-6 razem) działa poprawnie, szybko i zgodnie z oczekiwaniami użytkownika? Jak zmierzyć niezawodność i ustalić, że system jest gotowy do użytku w rzeczywistym środowisku?

Parametry laboratorium

Parametr	Wartość
Tryb pracy	zespoły 2-3 osób; test całego systemu pod nadzorem
Rezultat	raport walidacji, matryca testów, logika gotowości do produkcji
Oddanie	commit + evidence/lab07.md + raporty; FINALNE ODDANIE KURSU
Wersja	1.3 • 08 września 2026 r.

System oceniania

Ocena wynika z osiągnięcia kolejnych checkpointów.

Ocena	Główne wymagania
3.0	Testy dymne (smoke tests) - system startuje bez błędów
3.5	Testy funkcjonalne - wszystkie komponenty działają
4.0	Testy wydajności - FPS > 60, latency < 100ms, brak wycieków pamięci
4.5	Testy użyteczności - przypadki użytkownika, HCI, dostępność
5.0	Testy przypadków brzegowych, automatyzacja testów, raport produkcyjny

Walidacja: Numer indeksu i wariant diagnostyczny

Każdy student wpisuje numer indeksu. Wariant błędu wynika ze wzoru:

$$\text{wariantDiagnostyczny} = (\text{numer_indeksu_1} + \text{numer_indeksu_2}) \bmod 2$$

Wynik: wariant A lub B (każda para ma inny błąd systemowy do znalezienia)

Wymagania wstępne

Przed zajęciami

- Zaliczone Laboratorium 1-6 (lub funkcjonujące komponenty wszystkich lab)
- Ukończone wszystkie merge'ów gałęzi do main
- Unity z zainstalowanymi wszystkimi pakietami (XRI, AR Foundation, ROS 2, Analytics)
- Profiler Unity (Profiler, Memory, CPU Performance)
- Urządzenie testowe: Quest 3 LUB Android z ARCore LUB symulator
- Robot/symulator Gazebo dostępny w ROS 2 (jeśli Lab 6 używane)

Dostępne narzędzia testujące

Narzędzie	Zastosowanie	Wersja
Profiler Unity	CPU, GPU, Memory wydajności	Built-in
Memory Profiler	Analiza wycieków pamięci	1.1+
Frame Debugger	Wizualna analiza Draw Calls	Built-in
Console	Logowanie błędów i zdarzeń	Built-in
Unity Test Framework	Testy jednostkowe (opcjonalnie)	2.0+

Do Lab 7 będziesz konfigurować testy ręczne (głównie). Testy automatyczne są opcjonalnie dla Checkpoint 5.0.

Bezpieczeństwo i warunki testów

- Testy prowadź w kontrolowanym otoczeniu. Jeśli robot fizyczny - zawsze pod nadzorem.
- Nie testuj na danych produkcyjnych. Używaj symulatorów i testowych danych.
- Watchdog: jeśli system zawiedzie, powinien przejść do bezpiecznego stanu (stop ruchu, wyłączenie).

- Akumulacja błędów: jeśli jedno laboratorium nie działa, przejdź do następnego - nie blokuj walidacji całego systemu.
- Dokumentacja: każdy test udokumentuj - będzie potrzebna do raportu producenckiego.

Podstawy teoretyczne: Strategie testowania

4.1. Piramida testów

- Testy jednostkowe (Unit): pojedyncza klasa/metoda. Szybkie, izolowane, nie wymagają hardware'u.
- Testy integracyjne (Integration): wiele klas razem, interakcja z ROS 2/AR Framework.
- Testy E2E (End-to-End): cały system od A do Z. Wolne, wymagają hardware'u.

4.2. Testy dymne (smoke tests)

- Szybki test: czy aplikacja startuje?
- Czy główne funkcje są dostępne?
- Nie szukamy szczegółowych błędów, tylko czy system w ogóle działa.

4.3. Testy funkcjonalne

- Czy każda funkcja działa zgodnie ze specyfikacją?
- Dla Lab 2: czy chwyt działa?
- Dla Lab 3: czy rejestracja działa?
- Dla Lab 4: czy okluzja działa?

4.4. Testy wydajności

- FPS (frames per second) - czy jest >= 60 (VR) albo >= 30 (AR)?
- Latency - czy jest < 100ms? Memory - czy system wyciek pamięci?
- CPU - czy shader cost jest akceptowalny?

4.5. Testy użyteczności (UX)

- Czy użytkownik bez instrukcji potrafi użyć systemu?
- Czy jest dostępny (dla osób z niepełnosprawnościami)?
- Czy jest spójny (UI, terminologia)?

Plan pracy

Etap	Dowód ukończenia
Wejściówka i przygotowanie testów	Odpowiedzi + testy są przygotowane
Smoke tests - startowanie systemu	Brak błędów krytycznych, wszystkie komponenty inicjalizują
Testy funkcjonalne - Lab 1-6 oddzielnie	Każde laboratorium przechodzi testy podstawowe
Testy wydajności - Profiler	FPS, latency, Memory zmierzone
Testy użyteczności i UX	Scenariusze użytkownika przeprowadzone
Diagnoza błędu systemowego	Błąd znaleziony, przyczyna zidentyfikowana, wdrożona naprawa
Oddanie i raport finalny	Evidence/lab07.md + raporty

Checkpoint 3.0

Wymagania do zaliczenia

✓ Smoke tests: aplikacja startuje bez błędów krytycznych

✓ Konsola nie wyświetla RED errori (mogą być WARNINGi)

✓ Główne komponenty initializzują się: XR Origin, AR Manager, ROS 2 węzeł

✓ Testy funkcjonalne: co najmniej 70% Lab 1-6 przechodzi podstawowe testy

✓ Profiler: FPS zarejestrowany (może być < 60, ale musi być stabilny)

✓ Latency: zmierzony dla lab 3 albo lab 6 (jeśli dostępne)

✓ Memory: snapshot przed i po teście (brak drastycznych wycieków)

✓ Diagnoza: objaw, dwie hipotezy, test rozstrzygający

✓ Evidence/lab07.md zawiera: numery indeksów, wyniki testów, wariant diagnostyczny

✓ System jest stabilny i może zostać wdrożony (z zastrzeżeniami)

Część A. Przygotowanie testów

A.1. Wejściówka

1. Wyjaśnij różnicę między smoke tests i testami funkcjonalnymi.
2. Na jakim etapie projektu powinno się wykonywać testy wydajności?
3. Kiedy test zawodzi: czy powinno się bezpośrednio naprawiać kod czy zapisać bug?
4. Podaj trzy pomiary, które powinny być zawarte w raporcie walidacyjnym.
5. Co to jest "akceptowalny" FPS w kontekście VR vs AR?

A.2. Przygotowanie środowiska testów

1. git switch main i upewnij się, że wszystkie laboratoria są merged
2. Stwórz gałąź: git switch -c team-<nr>/lab07-<nazwiska>
3. Otwórz Profiler: Window → Analysis → Profiler
4. Przygotuj listę testów (patrz poniżej)
5. Ustaw urządzenie do testów: Quest 3 / Android / symulator Gazebo

A.3. Smoke Tests - startowanie systemu

```
// SMOKE TESTS - Checklist
[ ] Aplikacja startuje bez crash'u
[ ] Konsola nie wyświetla RED errors
[ ] XR Origin jest inicjalizowany (widoczny w Hierarchy)
[ ] Kamera główna odpowiada[ ] Jeśli Quest: kontrolery reagują na input
[ ] Jeśli AR: ARCore/ARKit inicjalizuje się
[ ] Jeśli ROS 2: węzeł Unity pojawia się w ros2 node list
[ ] Brak wycieków pamięci w pierwszych 30 sekundach
[ ] FPS nie spada poniżej 20 (nawet na słabszym hardware'u)
```

1. Uruchom Play Mode
2. Poczekaj 30 sekund bez interakcji
3. Zaznacz każdy punkt
4. Jeśli wszystko przechodzi: Smoke tests PASSED

PUNKT KONTROLNY A: Smoke tests przebiegły pomyślnie. System startuje bez krytycznych błędów.

Część B. Testy funkcjonalne poprzednich laboratorów

B.1. Matryca testów funkcjonalnych

```
// Matryca testów - Status Lab 1-6
Lab | Komponent | Test | Status | Uwagi
1   | XR Origin | Pozycja (1,1,1), skala (1,1,1) | PASS/FAIL | ___
1   | Geometria | Model widoczny, wymiary poprawne | PASS/FAIL | ___
1   | Audyt | FPS > 60 | PASS/FAIL | ___
2   | Chwyt | Direct Grab działa | PASS/FAIL | ___
2   | Panel | UI reaguje na ray | PASS/FAIL | ___
2   | Lokomocja | Teleportacja + snap turn | PASS/FAIL | ___
3   | Skanowanie | ARPlaneManager wykrywa | PASS/FAIL | ___
3   | Raycast | Umieszczanie modelu | PASS/FAIL | ___
3   | Rejestracja | Dwupunktowa wyrównana | PASS/FAIL | ___
4   | Depth API | Aktywna, zwraca dane | PASS/FAIL | ___
4   | Okluzja | Model okluzowany | PASS/FAIL | ___
4   | Light Estimation | Oświetlenie zmienia się | PASS/FAIL | ___
5   | Import CAD | Model załadowany | PASS/FAIL | ___
5   | LOD | Przejścia między LOD | PASS/FAIL | ___
5   | Benchmark | FPS zarejestrowany | PASS/FAIL | ___
6   | ROS 2 | Węzeł widoczny | PASS/FAIL | ___
6   | Topiki | Wszystkie dostępne | PASS/FAIL | ___
6   | Synchronizacja | Model śledzi robota | PASS/FAIL | ___
```

Wynik: __ / 18 testów przechodzą

1. Dla każdego laboratorium: przeprowadź podstawowy test
2. PASS: funkcja działa zgodnie ze specyfikacją
3. FAIL: funkcja nie działa albo zwraca błędy
4. Dla FAIL: zanotuj przyczynę (potrzebna jest naprawa?)
5. Oblicz procent przechodzących testów

PUNKT KONTROLNY B: Matryca testów funkcjonalnych wypełniona. >= 70% testów przechodzi.

Część C. Testy wydajności

C.1. Pomiar FPS i CPU/GPU

```
// Performance Measurement
Warunek | FPS | CPU [ms] | GPU [ms] | Memory [MB] | Status
Idle (nic się nie dzieje) | ___ | ___ | ___ | ___ | PASS/FAIL
Lab 1: audyt | ___ | ___ | ___ | ___ | PASS/FAIL
Lab 2: interakcja (chwyt) | ___ | ___ | ___ | ___ | PASS/FAIL
Lab 3: AR skanowanie | ___ | ___ | ___ | ___ | PASS/FAIL
Lab 4: okluzja + oświetlenie | ___ | ___ | ___ | ___ | PASS/FAIL
Lab 5: LOD aktywna | ___ | ___ | ___ | ___ | PASS/FAIL
Lab 6: ROS 2 synchronizacja | ___ | ___ | ___ | ___ | PASS/FAIL
Stres test (ciągły ruch 60s) | ___ | ___ | ___ | ___ | PASS/FAIL
Kryteria akceptacji:
- FPS >= 60 (VR) / >= 30 (AR)
- CPU < 15ms, GPU < 16.6ms (dla 60 FPS)
- Memory: bez drastycznych wzrostów (< 50MB wzrost w 2 minutach)
```

1. Otwórz Profiler (Window → Analysis → Profiler)
2. Dla każdego warunku: uruchom 30-60 sekund
3. Zanotuj medianę FPS, przeciętny CPU i GPU
4. Memory Profiler: snapshot na początku i na końcu testu
5. Oblicz przyrost pamięci

PUNKT KONTROLNY C: Wydajność zmierzona. FPS, CPU, GPU, Memory zanotowane dla każdego Lab.

Część D. Testy użyteczności i scenariusze użytkownika

D.1. Scenariusze użytkownika

Przeprowadź 3 scenariusze użytkownika bez instrukcji - obserwuj, czy użytkownik potrafi działać intuitywnie:

- Scenariusz 1 (montaż): Umieść obiekt wirtualny, połóż na powierzchni rzeczywistej, obróć go, zmień rozmiar
- Scenariusz 2 (diagnostyka): Zbadaj model robota, sprawdź jego stany, wyślij polecenie (jeśli Lab 6 dostępne)
- Scenariusz 3 (dokładność): Umieść model w rzeczywistej pozycji, zmierz błąd, sprawdź oświetlenie

1. Każdy scenariusz: czas na ukończenie, liczba błędów, liczba prób
2. Błąd = akcja nie wypadła tak, jak oczekiwał użytkownik
3. Wniosek: czy UI jest intuicyjny?

PUNKT KONTROLNY D: Testy użyteczności przeprowadzone. Scenariusze użytkownika zarejestrowane.

Część E. Diagnoza błędu systemowego

Wyznaczenie wariantu:

$$\text{wariant} = (\text{numer_indeksu_1} + \text{numer_indeksu_2}) \bmod 2$$

Wariant	Błąd systemowy	Hipotezy do rozróżnienia
A	Jedna z lab (losowo wybrana) nigdy nie startuje. Kto ma dostęp do tej gałęzi w Git?	Missing files / błędy merge'u / zła konfiguracja / brakujące pakiety
B	System startuje, ale wydajność drastycznie spada po 2 minutach. Memory leak albo zbieranie śmieci?	Brak Profiler danych / wyciek texture / akumulacja bufora / GC pause

1. Znajdź przypisany wariant
2. Odtwórz błąd
3. Opisz objaw dokładnie (np. "Lab 3 crashuje z NullReferenceException w RobotStateListener")
4. Sformułuj dwie hipotezy z różnymi metodami debugowania
5. Wykonaj test: zmień parametr, obserwuj wynik, log
6. Popraw błąd; powtórz test
7. git commit -m "fix(lab07): <przyczyna>"

PUNKT KONTROLNY E: Błąd systemowy znaleziony i naprawiony. Commit opisuje przyczynę.

Checkpoint 3.5

Wymagania dodatkowe (oprócz Checkpointu 3.0):

✓ Wszystkie funkcjonalne testy są PASS (100% Lab 1-6 działa)

✓ Wydajność: FPS >= 60 we wszystkich warunkach albo uzasadnienie, dlaczego nie

✓ Latency (Lab 3, 6): < 100ms albo zalecenie do optymalizacji

✓ Memory: brak drastycznych wycieków, stabilny wzrost

✓ Dostępność: system jest używalny dla użytkowników z różnymi umiejętnościami

Weryfikacja (prowadzący, max 90s):
✓ Checkpoint 3.0: wszystkie kryteria
✓ 100% testów funkcjonalnych PASS
✓ Wydajność akceptowalna
✓ Memory profile bez drastycznych wycieków
Zaliczenie: 2/4 dodatkowe kryteria (+ Checkpoint 3.0)

Checkpoint 4.0

Wymagania dodatkowe (oprócz checkpointów 3.0 + 3.5):

✓ Testy wydajności: pełna tabela CPU/GPU/Memory dla każdego Lab

✓ Analiza wydajności: bottleneck (CPU vs GPU), zalecenia optymalizacji

✓ Testy stresowe: system stabil przez 5 minut ciągłego ruchu

✓ Scenariusze użytkownika: 3 przeprowadzone, czasy zarejestrowane

✓ Raport: wnioski dotyczące gotowości do produkcji

Weryfikacja (prowadzący, max 90s):
✓ Checkpointy 3.0 + 3.5: wszystkie kryteria
✓ Tabela wydajności pełna
✓ Analiza bottleneck'u + zalecenia
✓ Stress test: stabilny
Zaliczenie: 2/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.0

Checkpoint 4.5

Wymagania dodatkowe (oprócz Checkpointów 3.0 + 4.0):

✓ Testy przypadków brzegowych: co się dzieje, gdy sieć pada? Sensor zawodzi? Input jest spamowany?

✓ Testy regresji: przechodzą testy ze starej wersji po zmianach?

✓ Przygotowana dokumentacja: design doc, deployment guide, troubleshooting

✓ Logika failover: jeśli Lab X zawiedzie, czy system dalej działa?

✓ Rekomendacje dla produkcji: hardware, sieć, setup, szkolenie użytkownika

Weryfikacja (prowadzący, max 90s):
✓ Checkpointy 3.0-4.0: wszystkie kryteria
✓ Przypadki brzegowe: testowane, rezultaty udokumentowane
✓ Regresja: powodzenie
✓ Dokumentacja + logika failover
Zaliczenie: 2/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.5

Checkpoint 5.0

Wymagania dodatkowe (oprócz Checkpointów 3.0 + 4.0 + 4.5):

✓ Testy automatyczne (Unity Test Framework): co najmniej 10 testów jednostkowych

✓ CI/CD pipeline: GitHub Actions albo Jenkins uruchamiający testy na każdy commit

✓ Raport produkcyjny: gotowy do oddania Stakeholderom

✓ Matryca oceny ryzyka: jakie są biggest risks w wdrożeniu?

✓ Harmonogram: kiedy system będzie gotowy do pełnego wdrożenia (benchmark: miesiące)

Weryfikacja (prowadzący, max 90s):
✓ Checkpointy 3.0-4.5: wszystkie kryteria
✓ Testy automatyczne: >= 10, > 80% coverage
✓ CI/CD: działający
✓ Raport produkcyjny: kompletny
Zaliczenie: 2/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 5.0

Oddanie

```
# Laboratorium 7
- RAPORT FINALNY <para / nazwiska>
## Identyfikacja pary
- Numer indeksu student 1: _____
- Numer indeksu student 2: _____
- Suma numerów: _____
- Przypisany wariant diagnostyczny: ___ (A/B)
- Data testu: _____
## Smoke Tests
- Aplikacja startuje: TAK / NIE
- RED errory w Konsoli: NIE (idealnie) / TAK (ile?)
- XR Origin inicjalizuje: TAK / NIE
- Główne komponenty działają: TAK / NIE
- **Wynik Smoke Tests: PASS / FAIL**
## Matryca testów funkcjonalnych (Lab 1-6)
| Lab | Test | Status | Uwagi |
|---|---|---|---|
| 1 | XR Origin + Geometria | PASS/FAIL | __ |
| 1 | Audyt FPS | PASS/FAIL | __ |
| 2 | Chwyt + Panel + Lokomocja | PASS/FAIL | __ |
| 3 | Skanowanie + Raycast + Rejestracja | PASS/FAIL | __ |
| 4 | Depth API + Okluzja + Oświetlenie | PASS/FAIL | __ |
| 5 | Import CAD + LOD + Benchmark | PASS/FAIL | __ |
| 6 | ROS 2 + Topiki + Synchronizacja | PASS/FAIL | __ |
- Procent przechodzących: __ %
- **Wynik funkcjonalny: PASS (>=70%) / FAIL (<70%)**
## Testy wydajności
| Warunek | FPS | CPU [ms] | GPU [ms] | Memory [MB] | Status |
|---|---|---|---|---|---|
| Idle | __ | __ | __ | __ | PASS/FAIL |
| Lab 1 | __ | __ | __ | __ | PASS/FAIL |
| Lab 2 | __ | __ | __ | __ | PASS/FAIL |
```

```
| Lab 3 | __ | __ | __ | __ | PASS/FAIL |
| Lab 4 | __ | __ | __ | __ | PASS/FAIL |
| Lab 5 | __ | __ | __ | __ | PASS/FAIL |
| Lab 6 | __ | __ | __ | __ | PASS/FAIL |
| Stress (60s) | __ | __ | __ | __ | PASS/FAIL |
- FPS kryteria: >= 60 (VR) / >= 30 (AR) → **PASS / FAIL**
- Memory: brak drastycznych wycieków → **PASS / FAIL**
- **Wynik wydajności: PASS / FAIL**
## Testy użyteczności
| Scenariusz | Czas [s] | Błędy | Próby | Status |
|---|---|---|---|---|
| Montaż | __ | __ | __ | PASS/FAIL |
| Diagnostyka | __ | __ | __ | PASS/FAIL |
| Dokładność | __ | __ | __ | PASS/FAIL |
- Średnia liczba błędów na scenariusz: __
- Rekomenduje system użytkownikowi: TAK / NIE
## Diagnoza błędu systemowego (wariant ___)
- Objaw: ___
- Hipoteza 1: ___
- Hipoteza 2: ___
- Test rozstrzygający: ___
- Przyczyna: ___
- Wynik przed / po: ___ / ___
## Raport gotowości do produkcji
- System jest stabilny: TAK / NIE
- Wymaga optymalizacji: TAK / NIE (wskaż obszary: ____)
- Dokumentacja: pełna / częściowa / brakująca
- Zagrożenia: główne ryzyko to ____
## Wnioski i rekomendacje
1. ___
2. ___
3. ___
## Zaliczenie
- Checkpoint 3.0 (PASS smoke + 70% funkcjonalnych): **TAK / NIE**
- Ocena finalna (jeśli 3.0): 3.0 / 3.5 / 4.0 / 4.5 / 5.0
**STUDENT 1 - Wkład w Lab 1-7: Głównie _____**
**STUDENT 2 - Wkład w Lab 1-7: Głównie _____**
![System gotowy do testów](img/lab07-final-system.png)
![Wyniki wydajności](img/lab07-performance.png)
```

Każda osoba w parze musi mieć co najmniej jeden commit dotyczący testów w Lab 7.

Lista końcowa studenta - przed oddaniem

☐ Smoke tests przeprowadzone: aplikacja startuje bez błędów krytycznych.

☐ Matryca testów funkcjonalnych: >= 70% Lab 1-6 przechodzi.

☐ Profiler: FPS, CPU, GPU, Memory zmierzone dla każdego Lab.

☐ Stress test: system stabilny przez 60 sekund.

☐ Testy użyteczności: 3 scenariusze użytkownika przeprowadzone.

☐ Błąd systemowy znaleziony i naprawiony.

☐ Evidence/lab07.md wypełniony z wszystkimi tabelami.

☐ Raporty (Profiler snapshots, logi) znajdują się w repozytorium.

☐ Każda osoba w parze wykonała co najmniej jeden commit do Lab 7.

☐ Wkład obu studentów jest porównywalny (nie ma 90/10).

☐ Repozytorium jest gotowe do finalnego merge'u na main.

Pytania kontrolne

1. Na czym polega różnica między smoke tests i testami funkcjonalnymi?
2. Kiedy test może przejść, a system wciąż być niestabilny?
3. Jaka jest dolna granica FPS dla VR bez sickness? (odpowiedź: 60 FPS)
4. Jak rozpoznać wyciek pamięci w Profilerze?
5. Które dane powinieneś dostarczyć Stakeholderom w raporcie produkcyjnym?
6. Jaki jest cel stress testu?
7. Czy może być PASS w Smoke Tests i FAIL w funkcjonalnych? Kiedy?

Exit ticket (odpowiedź indywidualnie) - FINALNE SŁOWO

Dwa zdania:

(1) Ile procent systemu, aby uważać go za gotowy do użytku?

(2) Jakimi trzema rzeczami byś się zajął najpierw, gdybyś miał 3 miesiące na optymalizację?

GRATULACJE Z UKOŃCZENIA PRZEDMIOTU

Wirtualna i Rozszerzona Rzeczywistość - Laboratorium 7 z 7