테스트 주도 개발을 적용한 무기체계 소프트웨어 연구 개발 사례 연구

남성우*, 이계진*, 오선택*
*LIG 넥스원
sungwoo.nam@lignex1.com, kyejin.rhee@lignex1.com, suntaek.oh@lignex1.com

A Case Study on the Weapon System Software Development Appling Test-Driven Development

Sung-Woo Nam*, Kye-Jin Rhee*, Sun-Taek Oh*
*LIG Nex1

요 약

테스트 주도 개발(TDD: Test-Driven Development)의 가치와 원칙을 잘 이해하고 있고, TDD 를 적용한 개발 방식이 습관화되어 있으면 TDD 가 제공하는 다양한 장점들을 얻을 수 있다. 하지만 위와 같은 경우가 아니라면 TDD 를 적용하더라도 비용 대비 효과를 얻을 수 없다. 본 논문에서는 비용 대비 효과를 높일 수 있도록 TDD 를 상황에 맞게 테일러링하여 협업에 적용한 사례를 제시한다.

1. 서론

전통적인 개발방법론은 문제 해결 코드 작성에 초점을 맞추고, 테스트 코드는 문제 해결 코드 작성 후에 작성된다. 반면에 테스트 주도 개발(TDD: Test-Driven Development)에서는 요구사항 충족과 설계 개선에 초점을 맞춘다. 문제 해결 코드 작성 전에 테스트 코드를 먼저 작성함으로써 구현 전에 요구되는 기능에 대해 충분히 생각할 수 있도록 해준다. [1]

TDD 의 가치와 원칙을 잘 이해하고 있고, TDD 를 적용해서 개발하는 방식이 습관화되어 있으면 TDD 를 적용하는데 아무런 문제가 없다.

하지만 위와 같은 경우가 아니라면 TDD 를 적용한다고 하더라도 비용 대비 효과를 얻을 수 없다. 예를 들면 초급 개발자(TDD 입문자)에게 TDD 를 적용하도록 권장할 경우 코딩 전에 테스트 케이스를 먼저 작성하는 이유를 이해시키기가 쉽지 않고, 강제로 지시할 경우에는 올바른 테스트 케이스를 작성하는 방법부터 가르쳐야 하므로 생산성이 급격히 떨어지게 된다.

이와 같은 상황에서는 정의된 절차에 따라 TDD 를 적용하는 대신에 코딩 테스트와 유사한 방법을 적용하면 충분히 TDD 의 장점을 얻을 수 있다. 아래 본론에서는 위의 방법을 현업에 적용한 사례를 제시한다. 본문은 시간 순서대로 TDD 적용 전, TDD 적용 중, TDD 적용 후로 구성된다.

2. TDD 적용 전

고급 개발자(TDD 숙련자)는 구현이 필요한 코드의 인터페이스와 이에 대한 테스트 케이스를 작성하여 제공한 후, 초급 개발자(TDD 입문자)에게 정해진 인터페이스를 구현하고 테스트 케이스를 모두 패스하도록 지시했다.

<표 1> 고급 개발자(TDD 숙련자)가 제공한 구현이 필요한 코드의 인터페이스

```
#include "MessageIntegration.h"

MessageIntegration::MessageIntegration(MessageManagerInterface
&messageManager)
: messageManager(messageManager)
{
    endByte = '\n';
    PreviousUDPMessage = "";
}

MessageIntegration::~MessageIntegration()
{

}

void MessageIntegration::ReceiveUDPMessage(UDPMessage
udpMessage)
{
    //Todo
}

void MessageIntegration::Reset()
{
    //Todo
}
```

<표 2> 고급 개발자(TDD 숙련자)가 제공한 테스트 케이스

```
#include "MessageIntegrationInterface.h"
#include "MessageIntegration.h"
#include <iostream>
#include <string>

using namespace std;

#define TEST(x)\
if(x == true)\
{\
	cout << "Sucess : " << __func__ << "()" << endl;\
}\
else\
{\
	cout << "Fail	: " << __func__ << "()" << endl;\
}

//Valid Message Input : $PMHS,POS,-145.9*40\r\n

void TestBasicMessage()
{
	//Given
	MessageManagerTest messageManagerTest;
	MessageIntegration messageIntegration(messageManagerTest);
	MessageIntegrationInterface	&MessageIntegrationInterface	=
messageIntegration;

	//When
	string testString = "$PMHS,POS,-145.9*40\r\n";
	UDPMessage udpMessage;
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	//Then
	TEST(messageManagerTest.result.compare("$PMHS,POS,-
145.9*40\r\n") == 0);
}

void TestPatialMessage()
{
	//Given
	MessageManagerTest messageManagerTest;
	MessageIntegration messageIntegration(messageManagerTest);
	MessageIntegrationInterface	&MessageIntegrationInterface	=
messageIntegration;

	//When
	string testString = "$PMHS,POS,";
	UDPMessage udpMessage;
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	//Then
	TEST(messageManagerTest.result.compare("") == 0);
}

void TestConcatenatedMessage()
{
	//Given
	MessageManagerTest messageManagerTest;
	MessageIntegration messageIntegration(messageManagerTest);
	MessageIntegrationInterface	&MessageIntegrationInterface	=
messageIntegration;

	//When
	string testString = "$PMHS,POS,";
	UDPMessage udpMessage;
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	testString = "-145.9*40\r\n";
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	//Then
	TEST(messageManagerTest.result.compare("$PMHS,POS,-
145.9*40\r\n") == 0);
}

void TestReset()
{
	//Given
	MessageManagerTest messageManagerTest;
	MessageIntegration messageIntegration(messageManagerTest);
	MessageIntegrationInterface	&MessageIntegrationInterface	=
messageIntegration;

	//When
	string testString = "$PMHS,POS,";
	UDPMessage udpMessage;
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);
	MessageIntegrationInterface.Reset();

	testString = "-145.9*40\r\n";
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	//Then
	TEST(messageManagerTest.result.compare("") == 0);
}

void TestMessageSizeOverflow()
{
	//Given
	MessageManagerTest messageManagerTest;
	MessageIntegration messageIntegration(messageManagerTest);
	MessageIntegrationInterface	&MessageIntegrationInterface	=
messageIntegration;

	//When
	string testString = "$PMHS,POS,";
	UDPMessage udpMessage;
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	testString = "";
	for(int i = 0; i < 100; i++)
	{
		testString += "1";
	}
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);
```

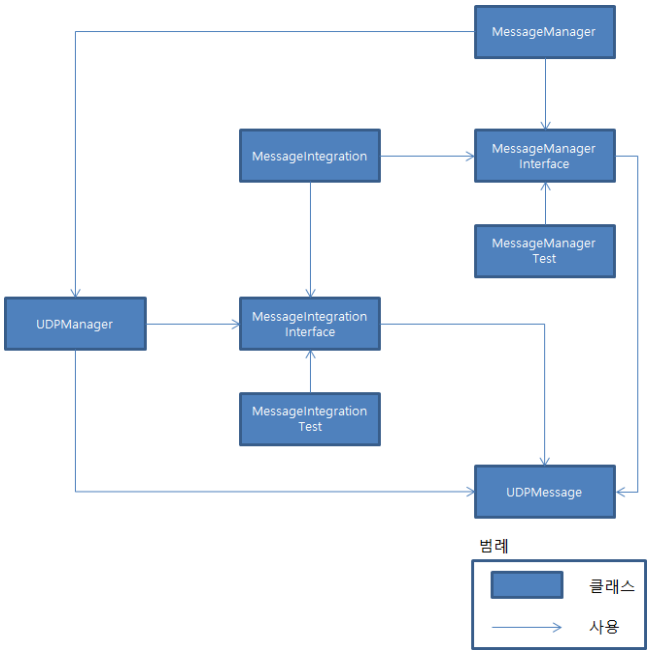
```
	//Then
	string result = "$PMHS,POS,";
	for(int i = 0; i < 90; i++)
	{
		result += "1";
	}
	TEST(messageManagerTest.result.compare(result) == 0);
}

void TestMessageSizeOverflowAndConcatenatedMessage()
{
	//Given
	MessageManagerTest messageManagerTest;
	MessageIntegration messageIntegration(messageManagerTest);
	MessageIntegrationInterface &MessageIntegrationInterface =
messageIntegration;

	//When
	string testString = "$PMHS,POS,";
	UDPMessage udpMessage;
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	testString = "";
	for(int i = 0; i < 100; i++)
	{
		testString += "1";
	}
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	testString = "-145.9*40\r\n";
	udpMessage.messageSize = testString.length();
	strcpy(udpMessage.messagebuffer, testString.c_str());
	MessageIntegrationInterface.ReceiveUDPMessage(udpMessage);

	//Then
	string result;
	for(int i = 0; i < 10; i++)
	{
		result += "1";
	}
	result += "-145.9*40\r\n";
	TEST(messageManagerTest.result.compare(result) == 0);
}

int main()
{
	TestBasicMessage();
	TestPatialMessage();
	TestConcatenatedMessage();
	TestReset();
	TestMessageSizeOverflow();
	TestMessageSizeOverflowAndConcatenatedMessage();

	return 0;
}
```

3. TDD 적용 중

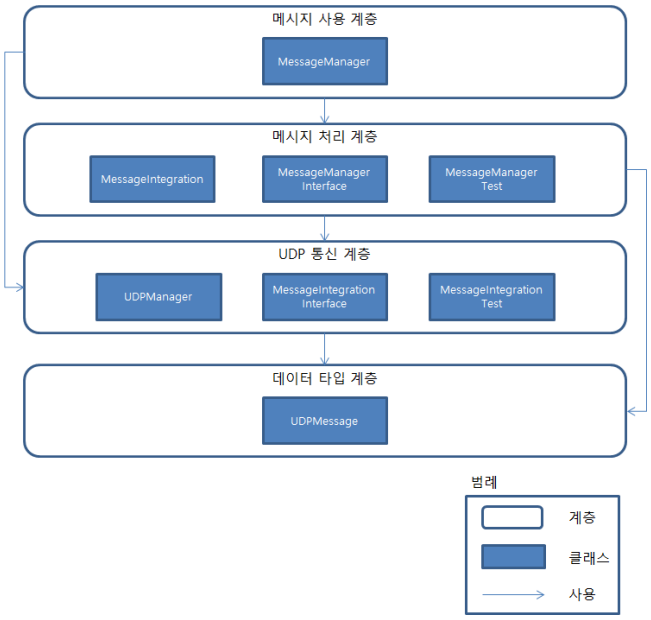
고급 개발자(TDD 숙련자)는 초급 개발자(TDD 입문자)가 설계를 제대로 이해하지 못하는 상태에서 구현 업무를 수행하고 있다고 생각해서 이해를 돕기 위해 아래와 같은 설계 자료를 작성한 후 초급 개발자(TDD 입문자)에게 제공했다. [2]

<표 3> 설계 개요

* 설계의 특징 1. 클래스 사이의 의존성을 제거하여 각 클래스는 단독으로 개발하고 테스트할 수 있음 2. 추후 클래스가 변경되더라도 인터페이스가 변경되지 않는다면 기존 클래스는 수정하거나 재컴파일할 필요가 없음 3. 테스트용 클래스를 별도로 제공하여 TDD 수행을 지원 * 적용된 설계 개념 1. 옵저버 패턴 2. 클래스의 생성자를 사용한 의존성 주입



(그림 1) uses style로 작성한 정적인 관점에서의 설계



(그림 2) layered style로 작성한 정적인 관점에서의 설계

4. TDD 적용 후

초급 개발자(TDD 입문자)는 코드를 구현 완료한 후 고급 개발자(TDD 숙련자)에게 전달했다. 고급 개발자(TDD 숙련자)는 초급 개발자(TDD 입문자)가 구현한 코드를 검토 후 아래와 같이 피드백을 주었다. [3]

<표 4> 피드백 결과

* 피드백 내용
  코드 검토 후 일부 로직에 오류가 발견되어 이를 검증하기 위한 테스트 케이스를 추가했고, 전체적으로 코드를 개선했습니다. 현재는 팀의 SW 개발 효율성보다는 개개인의 개발 역량 향상을 우선 순위로 두고 있습니다.
  아래의 변경 사항을 참고해서 앞으로 어떤 방향으로 개발하면 좋은지에 대해 생각해보시면 좋을 것 같습니다. 앞으로는 짧은 개발 일정으로 인해 지금처럼 개발하기는 어려울 것이라고 생각됩니다. 하지만 "알지만 어쩔 수 없이 못하는 것"과 "모르고 있어서 하지 않는 것"은 차이가 크다고 생각됩니다.

* 변경 사항
1. 불필요한 클래스 멤버 변수 삭제
2. 크기가 큰 함수는 기능의 역할에 따라 작은 함수로 분리
3. 함수 실행 전 사전 조건 검사 추가
4. 중첩된 조건문 최소화
5. 필요한 부분에 필요한 만큼의 주석 작성
6. 사용 의도를 명확히 드러내는 변수명 및 함수명 사용
7. 분할 가능한 로직은 공백으로 분리

* 변경 사항으로 인한 장점
1. 코드를 쉽게 이해할 수 있다.
2. 코드를 쉽게 디버깅할 수 있다.
3. 코드의 기능을 변경하기가 쉬워진다.
4. 코드에 새로운 기능을 추가하기가 쉬워진다.

<표 5> 피드백 결과를 반영하여 변경된 코드

```
#include "MessageIntegration.h"

MessageIntegration::MessageIntegration(MessageManagerInterface&
messageManager)
    : messageManager(messageManager)
{
    endByte = '\n';
    PreviousUDPMessage = "";
}

MessageIntegration::~MessageIntegration()
{

}

void MessageIntegration::ReceiveUDPMessage(UDPMessage
udpMessage)
{
    // 사전 조건 검사
    const int maxMessageSize = 100;
    if (udpMessage.messageSize > maxMessageSize)
        return;

    // 이전 메시지와 병합
    for (int i = 0; i < udpMessage.messageSize; i++)
    {
        PreviousUDPMessage += udpMessage.messagebuffer[i];
    }

    // 조건에 따라 메시지 분할 후 전송
    string udpMessageToSend;
    for (unsigned int i = 0; i < PreviousUDPMessage.length(); i++)
    {
        udpMessageToSend += PreviousUDPMessage[i];
        // 메시지 분할 조건 : 전송할 메시지의 크기가 최대
메시지 크기보다 크거나 종료 문자를 수신할 경우
        if (udpMessageToSend.length() >= maxMessageSize ||
PreviousUDPMessage[i] == endByte)
        {
            MakeSerialMessage(udpMessageToSend);
            udpMessageToSend.clear();
        }
    }

    // 잔여 메시지 저장
    PreviousUDPMessage = udpMessageToSend;
}

void MessageIntegration::MakeSerialMessage(string message)
{
    UDPMessage udpMessage;
    strncpy(udpMessage.messagebuffer, message.c_str(),
message.length());
    udpMessage.messageSize = message.length();
    messageManager.ReceiveSerialMessage(udpMessage);
}

void MessageIntegration::Reset()
{
    PreviousUDPMessage = "";
}
```

5. 결론

본 논문에서는 TDD 를 상황에 맞게 테일러링하여 적용한 사례를 제시했다. TDD 를 정의된 절차에 따라 적용했을 때보다는 가질 수 있는 장점들이 적었지만 비용 대비 효과는 얻을 수 있었다고 판단된다.

어떤 개발방법론을 얼마나 잘 적용하는지 보다는 적용하는 목적을 이해하고, 현재 환경에서는 어떻게 적용해야 비용 대비 효과를 높일 수 있는지에 대해 합리적으로 판단하는 과정이 중요하다고 생각한다.

참고문헌

[1] Jeff Langr, “Modern C++ Programming with Test-Driven Development”, Dallas, The Pragmatic Bookshelf, 2013
[2] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford, “Documenting Software Architectures Views and Beyond, Boston, Pearson Education, 2010
[3] Stephan Roth, Clean C++20, New York, Apress Media, 2021