무인항공기체계 소프트웨어 결함 원인분석을 통한 소프트웨어 프로세스 개선 방안

(An Approach for Software Process Improvement based on Software Defect Analysis of UAS)

김영택 [a,b], 박지훈 [a], 전영미 [b], 백종문 [a], 배두환 [a]

[a] 한국과학기술원 전산학과, [b] 국방과학연구소
{ytkim, jhpark}@se.kaist.ac.kr, douner7198@add.re.kr,
jbaik@kaist.ac.kr, bae@se.kaist.ac.kr

요약: 무인항공기체계의 통합시험 시 발견되는 소프트웨어 결함을 줄이기 위해서는 사업계획 단계부터 신뢰성 계획을 수립하고 이를 관리하는 것이 중요하다. 본 논문에서는 무인항공기체계 개발 시 수행된 네 단계의 통합시험으로부터 수집된 결함 데이터의 중결함, 경결함 및 개선사항 수 및 이를 해결하는데 소요된 기간을 도출하고, 각 결함의 원인을 제공한 개발단계가 무엇인지 분석하였다. 이 결과, 조종사가 관여하는 UI 관련 설계 시 개발자간 협의를 충분히 할 경우 설계 결함을 줄일 수 있는 것으로 분석되었으며, 이러한 결과를 바탕으로 설계 단계에서 결함을 줄이기 위한 개발 프로세스 개선방안을 제안하였다.

핵심어: 통합시험, 결함 분석, 소프트웨어 프로세스

1. 서론

무인항공기체계(Unmanned Aircraft System, 이후 UAS 로 약칭)는 부여된 임무를 수행하는 무인항공기(Unmanned Aerial Vehicle, 이후 UAV 로 약칭)와 이 무인항공기의 상태를 모니터링하고 임무를 통제하는 지상제어시스템(Ground Control System, 이후 GCS 로 약칭) 및 기타 임무 관련 장비를 포함한 전체 체계를 말한다[1]. UAS 는 1991 년 걸프전 이후 정찰임무에 투입된 이래로 감시 및 전투 임무 등으로 확대되어 운용되고 있을 뿐만 아니라 민간분야에도 국경감시, 탐색 및 구조, 통신중계, 재난관리 및 과학연구 등 다양한 분야에 걸쳐 그 응용분야가 점점 늘어가고 있는 추세이다[2].

이와 같이 다양하게 활용되는 UAS 는 지상에서 운용되는 장비와 달리 임무 중 고장이 날 경우 UAV 추락으로 인한 인적, 물적 피해가 크다. 또한 유인항공기에 비하면 무인항공기의 신뢰성이 상당히 떨어지기 때문에[3], 개발 프로세스를 향상시켜 신뢰성에 영향을 미치는 결함을 제거해야 한다.

본 논문에서는 UAS 개발 시 네 단계의 통합시험을 통해 수집된 결함 데이터의 발생원인을 분석한 후, 통합시험 전에 결함을 줄이기 위한 개발 프로세스 개선방안을 제안하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 UAS 개발 시 적용된 네 단계의 통합시험 절차를 소개하고, 3 장에서는 통합시험을 통해 수집된 소프트웨어 결함 데이터를 분석한 후, 4 장에서는 3 장의 분석결과를 바탕으로 개발 프로세스 개선방안 제안, 5 장에서 관련 연구를 언급하고 6 장에서 결론을 맺는다.

2. 통합시험 절차

UAS 는 그림 1 과 같이 통상적인 V-모델 방법론을 적용하여 개발된다. UAS 를 구성하는 각 부체계들은 UAS 요구사항으로부터 설계 및 구현이 완료된 후 기본적인 기능시험을 수행한다. 이렇게 완성된 부체계들은 체계 SIL(System Integration Laboratory) 통합시험, 체계 HILS (Human In The Loop) 시험, 지상 통합시험 및 비행시험 등 네 단계의 UAS 통합시험을 통해 검증된다. 본 장에서는 각 통합시험 절차의 목적과 방법에 대해 설명한다.

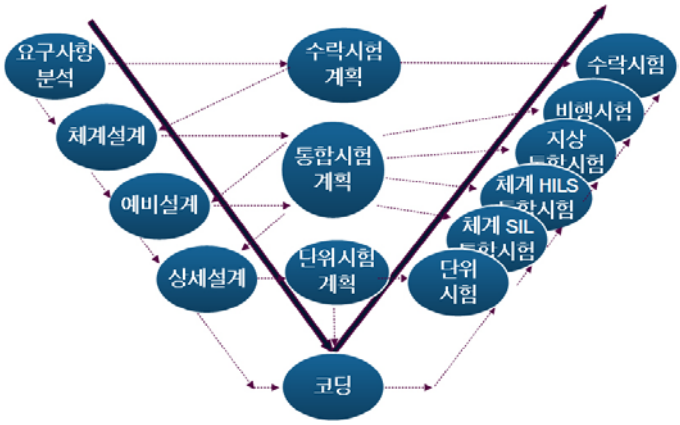

그림 1. UAS 개발 프로세스

2.1 체계 SIL 통합시험

체계 SIL 통합시험의 목적은 UAS 를 구성하는 각 부체계의 인터페이스와 기능을 확인하기 위한 시험으로, 자체 시험이 완료되지 않은 부체계는 모델이 사용된다. 부체계 개발에 직접 참여하지 않은 별도의 통합시험 팀이 시험을 주관하고, 부체계 개발자는 체계 SIL 통합시험 팀을 지원한다.

2.2 체계 HILS 시험

체계 HILS 시험은 비행모의 시험을 통하여 비행조종컴퓨터의 비행로직을 검증하고, 체계통합시험 시 오류의 원인을 분석하기 위하여 수행된다. 정확도가 높은 비행모의 시험을 위해서는 비행환경을 모사하는 기능을 제외하고 가능한 한 운용할 때와 동일한 실장비로 구성된다. 시험수행은 조종사가 포함된 별도의 시험 팀이 주관하고, 부체계 개발자는 결함발생 시 원인분석 및 시험을 지원하는 역할을 수행한다.

2.3 지상 통합시험

지상 통합시험은 UAS 가 운용하는 활주로 또는 주기장에서 UAS 의 기능을 확인하기 위해 수행하는 것으로 실형상의 UAV 와 GCS 를 대상으로 한다. 시험 수행은 별도의 시험 팀이 주관하고, 부체계 개발자는 시험 지원을 한다.

2.4 비행시험

비행시험은 UAV 가 비행 중 정상적인 임무를 수행할 수 있는지 확인하기 위한 시험으로, 조종사를 포함하는 별도의 비행시험 팀이 주관하고 부체계 개발자는 시험을 지원한다.

3. 소프트웨어 결함 데이터 분석

UAS 통합시험 각각은 표 1 에서 나타낸 기간만큼 수행되었으며, 각 시험 간 중복된 기간을 감안하면 총 XX 개월(국방과학연구소 보안성 검토에 의거 소요기간은 XX 개월로 표시)이 소요되었다.

표 1. 체계통합시험 수행기간

시험 구분	수행 기간
체계 SIL 통합시험	X 개월
체계 HILS 시험	X 개월
지상 통합시험	X 개월
비행시험	X 개월

시험기간 동안 발견된 소프트웨어 결함은 통합시험 단계별로 표 2 와 같이 수집되었다. 시험 수행 시 결함이 발견되면 결함분류는 다음과 같은 분류기준에 따라 개발자와 협의를 거쳐 결정되었다. 즉, 비행안전에 직접 영향이 있거나 소프트웨어 취약점에 의한 오류 발생시 체계에 치명적인 영향을 미치거나 오작동을 유발할 가능성이 있는 경우는 중결함(Major), 비행안전에 직접 영향은 없으나 소프트웨어의 가독성, 유지보수성 등 소프트웨어 품질 저하의 가능성이 있는 경우는 경결함 (Minor), 그리고 비행안전에 영향이 없으나, 조종사 및 운용요원에게 정보가 표시되는 위치 이동, 색깔 표시 수정 등과 같이 단순한 불편함을 주는 사항일 경우는 개선사항(Enhancement)으로 분류된다.

표 2. 시험단계별 결함 수

결함분류	체계 SIL 통합시험	체계 HILS 시험	지상 통합 시험	비행시험
중결함	42(11%)	25(66%)	2(3%)	9(7%)
경결함	250(63%)	13(34%)	59(74%)	104(86%)
개선사항	104(26%)	0(0%)	19(24%)	8(7%)

체계 SIL 통합시험 결과 모든 종류의 결함이 많이 발생하였는데 그 원인은 각 부체계를 인터페이스 해서 최초로 확인하는 시험이었기 때문인 것으로 분석된다.

한편, 체계 HILS 시험은 비행조종컴퓨터의 제어로직 검증에 주목적이 있기 때문에 다른 시험들보다 결함 수는 적었으나, 결함 발생 시 비행안전과 직결되는 결함이 대부분이었기 때문에 중결함 비율이 66%로 다른 시험들보다 상대적으로 높았다.

중결함 수는 전체적으로 시험이 진행됨에 따라 점점 감소하는 경향을 보였다. 그 이유는 중결함 관련 사항을 해결하기 위해 자원을 집중적으로 투입해서 충분한 시험을 수행하였기 때문에 이와 유사한 결함이 점점 줄어들기 때문인 것으로 분석된다. 이와는 대조적으로 경결함 수는 체계 SIL 통합시험에서 가장 많이 발생한 후 많이 줄어들었으나 체계 HILS 시험, 지상통합시험 및 비행시험으로 진행될수록 증가하는 경향을 보였다. 이러한 현상의 분석결과, 모의환경에서 최적의 조건에서 시험하는 체계 HILS 시험과 달리, 지상 통합시험 및 비행시험은 무선 데이터링크의 간헐적 끊김으로 인한 패킷 손실, 전송지연 등 실제 운용환경에서 UAS 가 겪게 되는 특성으로 인한 문제가 점점 많이 발생하는 것을 확인할 수 있었다.

결함을 해결하는데 소요되는 기간을 결함 유형별로 분석해 보면 표 3 과 같다.

체계 SIL 통합시험에서 발생된 중결함을 해결하는데 소요되는 기간은 다른 통합시험보다 훨씬 길었는데, 그 이유는 비행통제장비 및 데이터링크 이중화 로직 설계, 비행통제장비의 비정상적 종료 및 계통의

표 3. 결함유형별 평균 해결기간(미해결 항목 제외)

결함분류	체계 SIL 통합시험(일)	체계 HILS 시험(일)	지상 통합 시험(일)	비행시험 (일)
중결함	203	6	23	60
경결함	50	10	18	33
개선사항	75	N/A	39	13

화면 수정 사항 등 설계를 근본적으로 수정하거나, 부서간 협조가 많고 원인 분석이 어려운 항목이 많았기 때문으로 분석된다. 이와는 반대로 체계 HILS 시험 시 발생된 중결함의 해결기간은 경결함 해결기간보다 짧았는데, 주로 비행조종컴퓨터와 관련된 사항에 국한된 코딩 오류, 비행과 관련된 계기화면 개선 등 사안의 중대성에 비하여 단순한 내용이 많을 뿐만 아니라 해결해야 할 결함 수가 적은 것도 그 이유로 해석된다.

체계 SIL 통합시험 시 발생된 결함 해소기간이 다른 통합시험에서 발생된 결함 해소기간보다 대체적으로 길었는데, 이는 최초 인터페이스 시 발견된 결함 수가 절대적으로 많았기 때문에 한정된 자원으로, 여러 부서간 협조를 통해 단기간에 해결하기는 어려웠기 때문인 것으로 보인다.

각 결함에 대하여 그 결함의 원인이 어떤 개발단계로부터 유래되었는지를 분류하고, 결함의 원인이 되는 개발 단계에 따라 결함을 해결하는데 소요되는 기간을 분석해 보았다. 표 4 는 결함의 원인을 제공한 개발단계별 결함 수이고, 표 5 는 이를 해결하는데 걸리는 소요기간을 나타낸다.

표 4. 결함 원인제공 개발단계별 결함 수

결함원인 개발단계	체계 SIL 통합시험	체계 HILS 시험	지상 통합 시험	비행시험
요구사항	11(3%)	10(26%)	6(8%)	4(3%)
설계	279(70%)	17(45%)	52(65%)	63(52%)
구현	106(27%)	11(29%)	22(28%)	54(45%)

표 5. 결함 원인제공 개발단계별 결함 평균 해결기간

결함원인 개발단계	체계 SIL 통합시험(일)	체계 HILS 시험(일)	지상 통합 시험(일)	비행시험 (일)
요구사항	55	11	18	11
설계	64	7	18	22
구현	49	1	10	16

결함이 유래된 개발단계는 주로 설계단계였으며, 체계 SIL 통합시험 후 시험이 진행될수록 설계단계로부터 발생된 오류가 구현단계에서 발생된 결함 수보다 점점 더 많아지는 것을 확인할 수 있었다. 이러한 이유는 코딩오류는 시험이 진행됨에 따라 오류를 수정한 후 소프트웨어의 안정화가 높아지므로 결함 수가 점점 줄어드는 반면, 설계 오류는 비행통제장비의 이중화 오류, 고장판단 시간 초과, 데이터링크 안테나 탐색 오류 등과 같이 실제 환경에서 다양한 시험을 수행해야만 발견할 수 있는 오류가 점점 많이 발생하기 때문인 것으로 해석된다.

또한 결함이 유래된 개발단계별로 그 오류를 해결하는데 소요되는 기간을 분석한 결과, 체계 HILS 시험을 제외하고는 요구사항 단계의 오류로부터 발생된 결함을 해결하는데 걸리는 시간보다 설계 단계의 오류로부터 발생된 결함을 해결하는데 시간이 더 많이 소요되는 것을 알 수 있었다. 이러한 원인은 설계단계의 오류가 요구사항 및 구현 단계의 오류보다 더 많았기 때문이며, 이것을 다른 방향으로 보면 요구사항 및 구현단계에 투입되는 자원을 설계단계에 더 많이 투입함으로써 설계단계로부터 유래되는 결함을 더 줄일 수 있을 것으로도 해석할 수 있다.

조종사가 주로 보는 비행통제화면의 계기 및 비행상태정보와 관련된 사항은 UI(User Interface) 중에서도 가장 중요한 것으로, 조종사를 포함하여 연관 있는 개발자간 협의가 필수적이다. 그러나 한정된 자원과 일정으로 이러한 협의가 심도 있게 이뤄지지 않고 개발이 진행됨으로써 결함이 많이 발생한 것으로 분석되었으며, 분석 결과를 표 6 으로 정리하였다.

표 6. 사전협의로 결함 해결 가능한 결함 수

가능여부	체계 SIL 통합시험	체계 HILS 시험	지상 통합 시험	비행시험
가능	213(54%)	13(34%)	23(29%)	0(0%)
불가능	183(46%)	25(66%)	57(71%)	121(100%)

즉, 체계 SIL 통합시험, 체계 HILS 시험 및 지상 통합시험에서 각각 발견된 총 결함의 54%, 34%, 29% 는 개발자간 사전협의를 통했다면 통합시험 전에 해결할 수 있는 것으로 분석되며, 결과적으로 불필요한 업무에 자원의 낭비를 초래하였다고 볼 수도 있다.

4. 개발 프로세스 개선방안

결함 데이터를 분석한 결과, 전체적으로 설계의 문제로 만들어진 결함이 요구사항이나 구현 단계의 문제로 만들어진 결함보다 수가 많고, 결함을 고치는데 시간이 오래 걸리는 것을 확인할 수 있었다. 하지만 이러한 결함들의 상당부분이 사전 협의로 해결할 수 있다는 것을 알 수 있었다.

이는, 보통 무인항공체계의 UI 설계 요구사항은 한번에 설계하기에는 양이 많고, 또한 요구사항도 개념이 구체화되면서 설계 과정 중에 새로운 사항이 추가 또는 수정되는 경우가 많기 때문이다.

따라서 이러한 결함을 줄이기 위해, 본 논문에서는 폭포수 모델에 부분적으로 애자일 기법[5]을 도입한 새로운 프로세스를 그림 2 와 같이 제안한다.

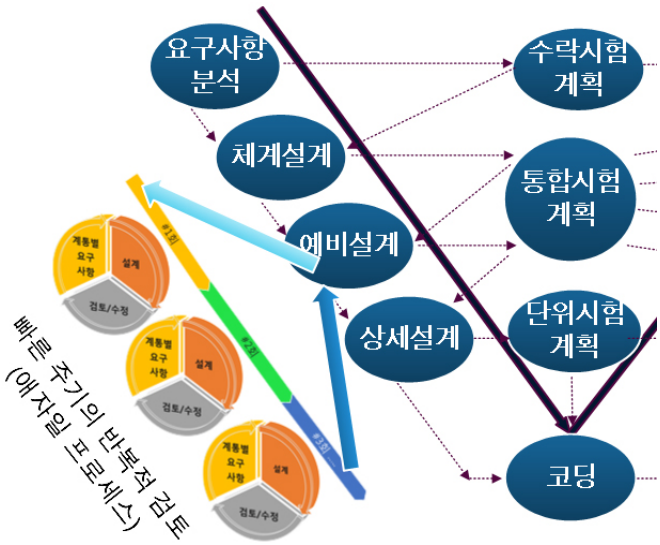

그림 2. 제안한 소프트웨어 프로세스

UAS 와 같이 UI 설계가 안전에 영향을 크게 미치는 사업은 이해당사자가 모두 포함된 협의체를 구성하고, 기본설계가 이루어지는 예비설계 단계에 UI 를 빠르게 구현할 수 있는 도구(예: VAPS[4])를 도입한 후, 애자일 프로세스[5]를 한시적으로 운영할 경우, UI 와 관련된 결함은 대부분 개발 초기에 해결할 수 있다. 즉, 예비설계 초기에 계통 별 설계 요구사항을 모두 수집한 후, UI 도구를 이용, 빠른 시간 안에 설계 및 구현하고, 구성원 모두 이에 대한 검토를 수행하면 이를 즉시 설계에 반영하는 과정을 반복하는 프로세스를 적용하면 기존의 폭포수 개발 프로세스를 보완할 수 있다. 본 프로세스를 이용할 경우, 표 6 의 결함들 중 체계 SIL 통합시험, 체계 HILS 시험, 지상 통합 시험에서 사전 협의를 통해 해결할 수 있었던 29% ~ 54%의 결함을 결함 삽입 전 미리 제거할 수 있을 것이다.

5. 관련 연구

다섯 개의 소프트웨어 프로젝트 데이터를 기반으로 한 Salo 의 연구[6]에서는, 애자일 프로세스가 실제로 소프트웨어 프로세스 개선에 도움이 될 수 있음을 보였다.

Tomohiro 의 연구[7]와 Karim 의 연구[8]에서는, 폭포수 모델에 애자일 기법을 부분적으로 도입하는 하이브리드 모델을 연구하였다. 본 연구에서는 예비 설계 단계에서 의사소통의 부족으로 인한 결함을 줄이기 위해 애자일 기법을 도입한 프로세스 모델을 제안하였으며, 버그 분석을 통해 실제 실행할 경우 효용성이 높을 수 있음을 보였다.

6. 결 론

본 연구에서는 UAS 개발 시 통합시험으로부터 수집된 소프트웨어 결함 데이터를 분석하고, 분석결과를 바탕으로 설계 결함을 줄이기 위한 소프트웨어 개발 프로세스를 제안했다. 실제 결함 데이터 분석을 통해 각 단계에서 발견되는 중결함/경결함/개선사항의 분포, 각 유형 별 해결 기간, 각 결함의 원인이 되는 단계, 각 원인제공 단계 별 평균 해결 기간, 사전 협의로 해결 가능한 결함 수를 밝혔다.

본 연구에서 제안하는 소프트웨어 프로세스는 UI 이해당사자로 구성된 협의체를 통해 애자일 프로세스를 한시적으로 운영하는 프로세스이다. 본 프로세스를 통해 사전 협의를 통해 줄일 수 있는 결함들을 최대한 줄일 수 있을 것이며, 이는 UAS 개발의 각 시험 단계에서 발견된 결함들 중 29%~54%에 해당하는 만큼, 결함을 사전에 제거하기 위해 큰 역할을 할 수 있을 것으로 보인다.

향후 연구로 더 많은 UAS 프로젝트 결함 데이터 분석을 통해 본 논문의 분석 결과가 다른 UAS 프로젝트에도 일반화 될 수 있는지 연구할 예정이다. 또한, 본 논문의 제안 프로세스를 실무에 적용해 도입 시 발생 가능한 문제점에 대해 연구해 볼 것이다.

참고문헌

[1] The Office of the Assistant Secretary of Defense, “DoD Dictionary of Military and Associated Terms,” Joint Publication 1-02, 2010

[2] http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[3] The Office of the Secretary of Defense, “Unmanned Aerial Vehicle Reliability Study,” Feb. 2003

[4] http://www.presagis.com/products_services/products/modeling-simulation/hmi/vaps_xt_simulation/

[5] http://en.wikipedia.org/wiki/Agile_software_development

[6] Salo, Outi, and Pekka Abrahamsson. "Integrating agile software development and software process improvement: a longitudinal case study." Empirical Software Engineering, 2005. International Symposium on. IEEE, 2005.

[7] T. Hayata, J. Han, “A Hybrid Model for IT Project with Scrum,” SOLI 2001, IEEE International Conference on

[8] K.M.Zaki, R.Moawad, “A Hybrid Disciplined Agile Software Process Model,” IEEE Informatics and Systems (INFOS), 2010 The 7th International Conference on