학습자료

임베디드 SW 엔지니어링

■ 학습모듈

임베디드 시스템 테스팅
(분류번호 2001020309)

18. 임베디드 시스템 변경 관리

한국산업인력공단
HUMAN RESOURCES DEVELOPMENT SERVICE OF KOREA

18. 임베디드 시스템 변경 관리

NCS 기본 정보

- 세분류 : 시스템 엔지니어링
- 능력단위 : 임베디드 시스템 테스팅
- 학습모듈 : 임베디드 시스템 테스팅
- 분류번호 : 2001020309
- 능력단위요소 : 임베디드 시스템 버그 수정하기
  임베디드 시스템 변경 관리하기

학습목표

1. 테스트 설계를 통한 테스트 케이스의 실행과 실행 결과를 분석하여 버그를 발견할 수 있다.
2. 테스트 실행결과에 따라 시스템이 정상동작 될 수 있도록 버그의 원인을 분석하여 원인이 식별된 버그는 수정하고 회귀테스트를 수행할 수 있다.
3. 형상관리 지침에 따라 소프트웨어 변경에 대한 사용자의 요구사항을 확인하고 변경 작업을 수행할 수 있다.
4. 형상관리 지침에 따라 수정된 소프트웨어의 검토된 변경 내용을 다음 배포판에 반영될 수 있도록 조치할 수 있다.

학습내용

- 임베디드 시스템 버그 수정하기
- 임베디드 시스템 변경 관리하기

1. 임베디드 시스템 버그 수정하기

1 임베디드 시스템 버그 수정

1. 테스트 관리 및 결함(defect) 관리의 필요성 이해

(1) 하나의 생명 주기와 프로세스를 갖는 테스트는 임베디드 시스템 개발을 수행하면서 결함(defect)를 발견하고 이를 수정함에 있다. 이해를 돕기 위해 몇 가지 결함(defect)과 관련된 용어를 정의함

- 버그(bug) : 소프트웨어 프로그램을 의도하지 않거나 예측하지 않은 방향으로 수행되도록 하는 결함(defect)
- 결함(defect) : 필요한 기능 수행을 방해하는 소프트웨어 또는 시스템 상의 결점으로 개발 산출물이나 소프트웨어 및 시스템에서 발생하며 Bug, Fault라고도 함
- 오류(error) : 부정확한 결과를 초래하는 인간의 활동, 실수를 말함
- 장애(failure) : 소프트웨어 컴포넌트나 시스템이 예상 기대 결과 또는 서비스와 실제적으로 편차를 보여 오작동을 발생하는 것을 말함

(2) 결함 유형 분류

조직의 품질 정책과 테스트 대상에 따라 달리 사용될 수 있으며 일반적으로 다음과 같이 정의되고 사용함

- 기능 결함 : 소프트웨어 프로그램 또는 시스템의 동작 결함
- 성능 결함 : 하드웨어 자원 또는 반응 시간과 관련된 결함
- 데이터 결함 : 연산 처리 결과의 출력 값에 대한 결함
- 요구 사항 결함 : 불명확한 요구 사항 또는 모순으로 발생한 결함
- 인터페이스 결함 : 기능 간 또는 시스템 간 연동과 관련된 결함
- UI 결함 : User Interface 검증 처리와 관련된 결함

(3) 결함 심각도 분류

일반적으로 조직의 품질(테스트) 정책에 따라 정의하고 사용됨

- 치명적(critical) 결함 : 시스템(소프트웨어/하드웨어) 운영이 중단된 결함. 데이터가 훼손 및 손실된 결함
- 주요(major) 결함 : 기능 중 핵심 기능이 작동하지 않거나 정확히 동작하지 않아 더 이상 작업 수행이 불가능한 결함. 요구 사항 성능에 부합되지 않은 결함. 필수적인 예외 상황 처리가 누락된 결함
- 사소한(minor) 결함 : 다른 대안이 있는 미미한 수준의 결함. 기능 수행은 대부분 가능하나 사용자의 불편함을 유발하거나 거슬리는 결함

(4) 우선순위에 따른 결함 분류

발견된 결함을 수정 시 조직에 따라 아래와 같이 우선순위를 정하여 긴급한 결함부터 수정 및 해결하도록 함

- 긴급(resolve immediately) : 즉시 수정해야 하며 필요 시 모든 개발 과정의 프로세스를 멈추고 시급히 수정할 결함
- 높은(give high attention) : 대체로 우선순위가 높은 결함
- 보통(normal queue) : 수정에 다소 시간적 여유가 있는 결함
- 낮음(low priority) : 수정이 꼭 필요하지 않은 결함

2. 결함(defect) 관리 프로세스 이해

조직(역할)	프로세서	설명
테스터	결함 등록	발견된 결함 정보 등록(open)
개발PM / 팀장	결함 할당	결함으로 등록된 결함을 수정할 개발자에게할당(assign)
개발 담당자	결함 수정	할당 받은 결함 수정(resolve) 테스터
테스터	결함 종료	수정된 결함 확인 및 종료 처리(closed)

[표] 역할별 기본적인 결함 관리 프로세스

(1) 결함 관리 필요성과 프로세스의 이해

개발 전 과정에서 테스트 활동을 통해 결함이 발견되어 등록되고 종료되기까지 결함을 추적하고 이해 관계자와 공유하여 정해진 시간과 비용을 들여 프로젝트를 효율적으로 성공시키기 위함

- 결함 관리에 필요한 요소

  (가) 결함 생명 주기(defect life cycle)

  - 결함 등록에서 결함 종료까지 전 과정의 절차와 역할이 정의되어야 함

  (나) 결함 관리 항목

  - 결함 관리용 정보로서 결함 지표 및 결함 목적, 활동에 따라 결함 관리 항목을 선정할 수 있음

  (다) 결함 관리 지표(metrics)

  - 결함 관리 중 획득한 데이터를 분석하고 가공하여 목적에 맞게 활동

(2) 결함 관리 도구와 필요성 이해

일반적으로 결함 관리는 아래와 같이 목적과 환경에 따라 적절히 활용할 수 있음

항목		Bugzilla	Trac	Mantis
주요 기능		이슈 관리	이슈 관리	이슈 관리
비용		무료	무료	무료
사용 환경	운영체제	Unix, Linux	Windows	Unix, Linux
	DB	My SQL Oracle	My SQL	My SQL MS SQL
	UI	Web	Web	Web
개발언어		Perl	Python	PHP
버전 관리 지원		지원	지원	지원

[표] 역할별 기본적인 결함 관리 프로세스(출처: wikipedia.org. 편집)

(3) 결함 제거를 위한 회귀적 테스팅(regression test)접근 방법

- 일반적으로 테스트 작업은 소프트웨어 개발 프로세스에서 한 번 수행되고 마는 작업이 아님
- 테스트 수행 중 또는 유지 보수 과정에서도 소프트웨어에 문제가 발생할 수 있는데, 개발 라이프 사이클 전체에서 시스템과 소프트웨어가 정상적으로 변경되었는지 확인, 변경 과정에서 의도치 않게 유입되는 이러한 결함 또는 오류를 발견하고 제거, 개선하기 위해 코드의 일부를 수정하는 작업이 진행됨. 이 과정을 디버깅 작업이라 함
- 디버깅 작업으로 인해 유입 가능한 새로운 오류를 검출하기 위해 회귀 테스트라는 작업으로 반복적인 시험을 진행하게 되는데 회귀 테스트의 경우는 시간을 절약하는 것이 필수
- 회귀시험과 같이 반복적인 작업은 반드시 자동화가 되어 바로 리포트를 확인할 수 있도록 되어야 함

(4) 디버깅(debugging) 기법의 이해

- 행위(behaviour) 기반의 디버깅 방법

  (가) 메모리덤프 디버깅

  - 문제의 원인을 메모리 정보에서 일일이 분석하여 찾는 방법

  (나) 출력물 삽입에 의한 디버깅

  - 프로그램의 이곳 저곳에 임의 출력 코드를 삽입하여 수행시켜 문제의 원인을 찾는 방법

(다) 자동화 도구를 이용한 디버깅

- 수행 과정을 추적해 놓고 특정 명령문이 수행될 때 수행을 중단시켜 문제의 원인을 찾는 방법

• 관점(view) 기반의 디버깅

(가) 귀납적 관점에 따른 디버깅

- 주변의 관련된 정보를 수집하여 문제의 원인들을 하나씩 따지면서 결함을 해결해 나가는 방법

(나) 연역접 관점에 따른 디버깅

- 어떤 가정하에 문제를 발생시켰을 법한 모든 원인들을 나열한 후 하나씩 그 가능성을 따져가며 불가능한 원인들을 제거시키고 나머지 원인들에 대해서 가정을 수집한 다음 결함을 해결해 나가는 방법

(다) 역행 조사에 따른 디버깅

- 결함 발견 지점으로부터 발생시킨 지점을 찾기 위해 원시 소스코드를 거슬러 올라가면서 결함을 찾아 가는 방법

2 임베디드 시스템 버그(결함)분석하기

1. 재료·자료

(1) 시스템 테스트의 대상 유닛

(2) 개발 관련된 문서(요구 사항 명세서, 상위/상세 설계서, 테스트 계획서/절차서)

(3) 테스트 자동화 도구(JTAC, 에뮬레이터 및 상용 도구)

2. 수행 순서

(1) 아래와 같이 소스 내에서 결함(defect) 및 버그(bug)를 찾아내고 원인을 분석

• 버퍼 오버플로우(buffer overflow)

- 이미 할당된 버퍼 메모리를 초과해서 데이터를 읽거나 쓸 때 발생하는 결함 또는 버그

```
int i;
int one_array[10]; ----------> 배열 선언
for(i=0;i<10;1++)
   one_array[i]=0;
 one_array[i]=200;   --------> 버퍼 오버프로우 오류 발생(삭제 코드)
```

1. 임베디드 시스템 버그 수정하기

- 메모리 누수(memory leak)
  - 메모리할당 후 반환(return)하지 않거나 메모리 크기할당 변경에 실패했을 때 발생하는 결함 또는 버그

```
char*  getHostName()
{
 char*  buff  =  (char*)malloc(HostName_SIZE);
 if(!buff)
   {
    return  NULL;
   }
    if(write_HostName(buff,  HostName_SiZE)  !=  HostName_SiZE)
   {
  free(buff);   <------------ 아래 경우로 삽입된 코드
     return NULL; ------->할당 후 해제하지 않아 메모리 누수 발생 가능
   }
 return  buff;
}
```

- 널 포인터 역참조(null pointer dereference)
  - 포인터 값이 널(null)로 정의한 후 널(null) 포인터 값에 접근할 때 발생하는 결함 또는 버그

```
struct  s  *ptr;
*ptr=Null;
if(ptr!=NULL)  ----------------------> NULL 조건 확인 로직 추가
{
ptr->field=val;     ------------------> 널(Null)포인터 역참조 발생
}
```

1. 임베디드 시스템 버그 수정하기

- 중복 해제(double free)
  - 이미 해제된 메모리 혹은 파일을 다시 해제할 때 발생하는 결함 또는 버그

```
int stud(void *pt)
{
 if(ting_error())
   {
     free(pt);    <---------- 이미 메모리 해제
          return 1; <---------- 1을 반환
   }
 return 0;
}
void main()
{
 void*p=malloc(60);
  if(stud(p)<0)
  {
   free(p);    ----------> 위에서 중복 해제 발생으로 free 함수 삭제
  }
  s_use(p);
}
```

(2) 결함 보고서 작성

- 먼저 조직 내에서 결함으로 관리할 항목을 정리
- 상황에 맞는 개발 및 테스팅 중 발견된 결함을 등록(open)
- 등록된 결함을 할당(assign)
- 할당된 결함을 수정(resolved)
- 수정된 결함 내용을 확인하고 종료 처리(closed)

1. 임베디드 시스템 버그 수정하기

	결함ID	DT_002	프로젝트명	HSET_F
기본 항목 정보	결함 제목	터치 패드 작업하던 도중 통화가 왔을 때 터치 패드 Lock-up 발생		
	결함 등록자	이발견 주임	등록일자	2015.3.4.
	결함 유형	기능 결함	테스트 케이스 ID	TC_002
	결함 심각도	Critical	우선순위	긴급
	결함 내용	터치 패드 작업하던 도중 통화가 왔을 때 터치 반응이 없는 결함		
	테스트 조건	1. 테스트 에뮬레이터에 스크립트 명령어로 터치 작업 수행 중, 동시에 여러 이벤트 처리 상태로 통화 환경 명령 조건 수행		

[표] 결함 기본 항목 정보(예시)

상태 항목 정보	결함 진행 상태	등록	할당	수정	종료
	결함 할당자	박완벽(PM)		결함 할당일	2015.3.5.

[표] 결함 상태 항목 정보(예시)

처리 항목 정보	결함 수정자	최수정(개발자)	결함 수정일	2015.3.7.
	결함 처리 내용	원인: 이벤트 핸들러 함수에 터치 패드에서 통화, 통화에서 터치 패드로 전환하는 함수의 변수 선언 오류 발생 수정 내용: 데이터 타입 변수 변경(int형 -> double 형)		
	결함 내용 확인자	이발견 주임	확인 결과: 재현 오류 발생 없음 결함 수정 확인 완료	
	결함 처리	결함 종료(closed)		

[표] 결함 처리 항목 할당·종료 정보(예시)

2. 임베디드 시스템 변경 관리하기

1 임베디드 시스템 변경 관리 수행

1. 형상 관리(configuration management) 이해

(1) 개념

- 개발 프로세스상의 시스템 및 소프트웨어 라이프 사이클 기간 동안 개발되는 모든 제품의 무결성을 유지하는 것으로 그 대상은 소스코드, 문서, 인터페이스 등 각종 결과물에 대해 형상 항목을 만들고 이들 형상에 대한 변경을 체계적으로 관리, 제어하기 위한 활동

(2) 의의

- 프로젝트 팀에서 만들어지는 시스템 제품의 오류 또는 실수를 최소화함으로써 생산성을 높이는 것

(3) 범위

- 형상 관리는 일반적으로 버전 관리, 변경 관리, 소스 관리, 빌드 관리, 이슈 관리 등을 모두 포함

2. 형상 관리(configuration management) 프로세스 이해

(1) 형상 식별(configuration identification)

형상 관리의 대상을 식별하고 베이스라인의 기준을 정하는 활동

- 세부 활동

  (가) 형상 항목 선정

  - 개발 프로세스에서 생산되거나 사용되는 작업 산출물 또는 작업 산출물을 구분하고 이중에 변경에 대한 통제가 필요한 산출물을 선정하는 활동

  (나) 베이스라인 기준 선정

  - 시스템 소프트웨어 개발의 특정 시점에서 형상 항목이 하나의 완전한 산출물로 공식적으로 검토, 승인된 후에 개발의 기준으로 활용되고 공식적인 변경 관리 절차를 통해서만 변경 관리될 수 있는 명세서나 제품

(2) 형상 통제(configuration control)

시스템 및 소프트웨어의 형상 변경 제안을 검토하고 승인하여 기준선(baseline)에 반영될 수 있도록 통제하는 활동세부 활동

(3) 형상 감사(configuration audit)

형상 항목의 변경이 제대로 이루어졌는지 검토, 승인하고 개발자와 유지 보수자들만의 검열이 아닌 객관적인 검증, 확인 과정을 거침으로써 새로운 형상의 무결성을 확보하는 활등

2. 임베디드 시스템 변경 관리하기

3. 시스템, 소프트웨어 형상 관리(configuration management)의 필요성

(1) 이전 리비전이나 버전에 대한 정보에 언제든지 접근할 수 있어야 함

(2) 동일한 프로젝트에 대해서 여러 개발자가 동시에 개발할 수 있어야 함

(3) 개발 소스의 접근을 제한하여 관리할 수 있어야 함

(4) 에러가 발생했을 때 빠른 시간 내에 수정할 수 있어야 함

(5) 사용자의 요구에 따라 적시에 최상의 시스템을 공급할 수 있어야 함

4. 형상 관리(configuration management) 자동화 도구의 필요성

시스템 및 소프트웨어의 형상을 수작업으로 관리하려면 많은 시간과 노력이 필요함

데이터의 일관성을 유지하고 프로젝트의 원활한 업무 수행과 생산성을 높이기 위해서는 자동화된 도구를 사용하는 것이 효율적임

분류		주요기능	제품
기본 항목 정보	Lower-End	기본적인 버전 관리 병렬 개발 지원	Visual Source Safe
	Higher-End	빌드 자동화 지원 통합 환경 문제/변경 추적 이기종 환경 지원 향상된 분기/병합 기능 지원 프로모션 모델 지원	PVCS StarTeam ClearCase Sourceintegrity Visual Enabler
프로세스 중심	생산성 중심의 기능 형상 관리 기능, 프로세스 관리 기능 추가 사용자의 역할과 책임 명시 승인 절차 추가 변경 관리(긴급 오류 수정, 변경) 플로우 추가		Harvest(CCC) PCMS PCMS(PVCS Dimensions) CM(Continuous)

[표] 형상 관리 자동화 도구의 주요 기능 및 제품(예시)

정리하기

1. 임베디드 시스템 버그 수정하기

결함과 관련된 용어

- 버그(bug): 소프트웨어 프로그램을 의도, 예측하지 않은 방향으로 수행되도록 하는 것
- 결함(defect): 필요한 기능 수행을 방해하는 소프트웨어/시스템 상 결점으로 개발 산출물이나 소프트웨어 및 시스템에서 발생
- 오류(error): 부정확한 결과를 초래하는 사람의 어떤 활동이나 실수
- 장애(failure): 소프트웨어 컴포넌트나 시스템이 예상 기대 결과 또는 서비스와 실제적으로 편차를 보여서 오작동을 하는 것

2. 임베디드 시스템 변경 관리하기

형상 관리 프로세스

- 형상 식별(configuration identification): 형상 관리의 대상을 식별하고 베이스라인의 기준을 정하는 활동
- 형상 통제(configuration control): 시스템 및 소프트웨어의 형상 변경 제안을 검토하고 승인하여 기준선(baseline)에 반영될 수 있도록 통제하는 활동
- 형상 상태(기록) 보고(configuration status accounting): 시스템 및 소프트웨어 개발 요소 및 변경 항목들이 제대로 반영되었는지 모든 변경 처리 과정에서의 변경 상태를 기록하고 보고하는 절차
- 형상 감사(Configuration Audit): 형상 항목의 변경이 제대로 이루어졌는지 검토, 승인하고 개발자와 유지 보수자들만의 검열이 아닌 객관적인 검증, 확인 과정을 거침으로써 새로운 형상의 무결성을 확보하는 활동