

최규식

건양대학교

che@konyang.ac.kr

A Study on the Effect of Imperfect Debugging of Developing Software

Che Gyu Shik

Konyang University

요 약

본 논문에서는 결함제거효율을 소프트웨어 신뢰도평가에 적용하여 연구하기로 한다. 결함제거효율은 소프트웨어 개발의 모든 과정에서 매우 유용한 척도이며, 개발자가 디버깅 효율을 평가하는데 크게 도움이 되며 이에 추가로 소요되는 작업량을 민감시켜 준다. 본 논문에서는 불완전 디버깅이라는 것이 디버깅 기간 동안 새로운 결함이 소프트웨어에 도입되며 따라서 검출된 결함이 완벽하게 제거되지 않을 수도 있다는 의미로 고려한다. 결함제거 효율, 고장율, 소프트웨어에로의 새로운 결함 도입율을 집합시키는 하나의 모델을 제안한다. 기존의 전통적인 신뢰도 척도에 추가하여 제안된 본 모델은 개발팀이 좀더 나은 판단을 할 수 있도록 유용한 척도를 개발자에게 제공한다. 실제 현장 적용에서 해석적인 변과 예측적인 변 모두에 대해서 제안된 모델을 예시하기 위해 수집된 소프트웨어 테스트데이터를 사용한다. 잔여결함의 기대치 및 소프트웨어 고장율도 나타낸다.

1. 서론

비록 약간의 소프트웨어 신뢰도 연구에서 불완전 디버깅 현상에 대해서 언급을 했지만 그들 대부분은 기존의 결함을 제거하는 중에 새로운 결함이 도입된 가능성에 대해서만 고려를 하였다. 그러나, 불완전 디버깅은 검출된 결함이 100%의 효율이 아닌 불완전 제거가 된 것을 의미한다. 조운스는 결함 제거 효율이 소프트웨어 품질과 공정관리에서 중요한 인자임을 지적하였다. 이는 소프트웨어 개발자에게 테스트 효율성을 산출하고 기타 추가 테스트노력의 필요하다는 논거를 제공해준다. 더욱 더 한 것은 결함 제거 효율이 보통은 100% 미만이라는 사실이다. (25%-55%이다). 실제로 소프트웨어 결함 디버깅은 매우 복잡한 공정이다. 보통은 테스트자가 소프트웨어 요건과 벗어난 것을 발견했을 때 그들은 수정요구를 한다. 그러면 검토 분과위원들은 이 요구를 특수 개발자에게 할당한다. 개발자가 소프트웨어 결함을 연구한 후에 그것을 교정하기 위한 코드변경을 제출한다. 변경된 코드는 보고된 문제를 해결하기 위해 다시 여러 가지 테스트를 통하여 과정을 거쳐야 한다. (유니트 테스트, 집적 테스트, 시스템 테스트) 이렇게 고치는 것은 이러한 테스트를 거치지 않을 수도 있고 때로는 이러한 테스트를 거쳤다 하더라도 그테스트 환경이 고객의 환경과 동일하지 않아서 그 사실 때문에 결함이 완벽하게 제거되지 않을

수도 있다. 소프트웨어 개발팀이 문제의 결함이 최종적으로 제거되기 전에 여러 번 보고된 소프트웨어 결함이라는 것을 발견하는 것이 별로 특별한 일이 아니다. 어떤 결함들은 고객이 현장에서 사용할 때만 나타나는 것도 있다. 그러므로, 결함제거효율은 소프트웨어 신뢰도 계산에서 중요한 인자이고 소프트웨어 사업관리에도 중요하다.

2. 소프트웨어 신뢰도 모델링

본 항에서는 결함 제거 효율을 가진 NHPP를 제시한다. 아래에서 얻기한 것은 이러한 모델에 대한 가정사항들이다.

- 1) 소프트웨어 고장의 발생은 NHPP를 따른다.
- 2) 임의의 시간에 있어서 소프트웨어 고장율은 결함 검출비이고 그 시간에 존재하는 잔여결함의 수이다.
- 3) 소프트웨어 고장이 발생되면 확률 p 로 디버깅 노력이 즉시 시작된다. 이 디버깅은 소프트웨어 고장의 각 위치에서 s-독립이다.
- 4) 각 디버깅 노력에 대하여 그 결함이 성공적으로 제거되건 아니건 새로운 결함들이 확률 $\beta(t)$ 로 소프트웨어 시스템에 도입된다. $\beta(t) \ll p$

A. 결함 제거 효율을 가진 일반적인 NHPP 소프트웨어 신뢰도 모델

본 항에서는 결함 제거 효율과 결함 도입비를