**7. Popclient가 Fetchmail이 되다.**

처음의 Popclient는 너무 많은 기능을 가지려고 했다. MTA(Mail Transport Agent)와 MDA (Mail Delivery Agent)의 기능을 모두 가지도록 설계되었다. 하지만 Harry Hochheiser가 보내준 SMTP 코드를 봤을 때 생각을 달리 해야 했다. 이 기능을 구현할 수 있다면 MDA기능이 필요 없는 순수한 MTA가 될 수 있었다. **좋은 아이디어를 생각하는 것 다음으로 중요한 것은 사용자들의 아이디어를 깨닫는 것이다. 가끔은 더 나을 수도 있다.**

Fetchmail 에서 MDA기능을 제거하자 많은 이점들이 나타났다. 드라이버 코드 중 가장 힘든 부분이 사라졌으며 OS가 파일잠금을 지원하는지 걱정할 필요도 없어졌다. 낡아서 뒤떨어진 기능이라면 효율적으로 제거할 수 있을 때 제거해버려야 한다. **설계에 있어서 완벽함이란 더 이상 추가할 것이 없을 때가 아니라 더 이상 버릴 것이 없을 때다.**

Fetchmail은 Popclient와 다르게 순수한 MTA가 되었다. Fetchmail은 자신만의 정체성을 얻었다.

**8. Fetchmail의 성장**

Fetchmail이 성공함에 따라서 Fetchmail을 좀 더 완벽하게 만들어야 했다. 나 자신의 필요성만이 아니라 나와 상관없는 다른 사람들에게 필수적인 기능을 지원하고 추가해야 했다. 이것을 깨닫고 추가한 첫번째 기능은 멀티드롭이였다. 멀티드롭 기능을 추가하기로 한 데에는 몇몇 사용자들이 이 기능을 원하는 것도 있었지만 가장 큰 이유는 멀티드롭 어드레싱을 구현함으로써 싱글드롭의 버그를 잡아낼 수 있을 거라고 생각했기 때문이다. 그리고 그렇게 되었다. **어떤 도구여도 기대하는 방법으로 쓸모가 있어야 하지만 정말 위대한 도구는 사용자가 전혀 기대하지 않았던 용도에 알맞게 된다.**

**9. Fetchmail에서 배울점**

Fetchmail의 rc파일의 선언들이 명령형 소언어를 얼마나 많이 닮아가고 있었다. 명령형 소언어를 더 영어처럼 만들면 사용하기 쉬울 것이라고 생각할 수 있다. 하지만 프로그래머들은 정확하고 짧으며 중복을 허용하지 않는 제어구문을 선호하는 경향이 있다. 하지만 영어는 50%정도의 중복을 허용하므로 대단히 부적절한 모델로 보인다. Fetchmail 제어구문은 이런 문제를 피하려고 했다. 언어의 영역이 매우 제한되어 있기 때문이다. **언어가 튜링완전하지 않다면 구문상의 유연함이 필요하다**.

또 하나의 교훈은 불투명함에 의한 보안에 대해서이다. Fetchmail 사용자들 중에는 rc파일에 있는 패스워드를 암호화하자고 이야기하는 사람들이 있었다. 하지만 그렇게 한다고 해서 보안이 강화되지 않기 때문에 그 이야기는 받아들여지지 않았다. Rc파일의 읽기 퍼미션을 얻은 사람이라면 사용자와 마찬가지로 Fetchmail을 실행시킬 수도 있고 디버깅을 통해서 코드를 뽑아낼 수도 있기 때문이다. **보안시스템은 그것이 보호하려고 하는 비밀만큼만 안전하다. 가짜비밀에 주의할 것**