RUP/XP 가이드라인: 페어 프로그래밍

Robert C. Martin
Object Mentor, Inc.

Rational Software 백서

TP 158, 3/01

Rational®
the software development company

목차

개요

페어링, 간략한 설명

페어 프로그래밍은 프로그래머가 페어를 이루어 프로젝트의 소프트웨어를 작성하는 설명입니다. 각 페어는 단일 워크스테이션에서 함께 작업합니다. 페어의 한 구성원은 워크스테이션을 드라이브하며 다른 구성원은 생성되고 있는 코드를 주의하여 관찰합니다. 드라이버는 현재 작성하고 있는 코드 행에 관심을 기울이며 전술적으로 생각합니다. 관찰자는 구문을 확인하고 전체 프로그램에 관해 전략적으로 생각합니다. 이들은 역할을 자주 서로 교환하며 한 사람이 수행하는 것보다 결과 코드가 빠르게 작성되고 결함이 덜 발견됩니다. 더욱이 코드는 최소한 두 명의 개발자에게 상세하게 알려집니다.

페어링 유스 케이스

일반적인 코드 검토 세션을 검토하십시오. 개발하는 데 한 사람 당 8 시간이 필요한 모듈을 여덟 사람이 1 시간에 검토합니다. 네트 결과는 시간당 16 명이 모듈에서 송신됩니다. 그러나 검토자는 코드에 익숙해지는 데 필요한 시간을 소요할 수 없으므로 검토는 상당히 피상적이 됩니다. 한 명의 개발자는 상세하게 친숙하지만 너무 친숙하여 결함의 벌크를 찾을 수 없습니다.

이것을 페어 프로그래밍 실행과 비교하여 보십시오. 페어로 모듈을 개발하는 데 8 시간이 필요한 경우 총 16 시간이 소요됩니다. 그러나 이 경우, *두 명의* 개발자는 코드에 대한 상세한 지식을 얻게 됩니다. 한 개발자에게 숨겨진 결함이 다른 개발자에게 보여질 수 있습니다.

페어 프로그래밍의 경우는 단순하지만 그 영향은 민감하며 광범위합니다. 페어 프로그래밍은 코드를 작성하고 검토하는 데 훨씬 더 효과적인 방법입니다. 모듈에 상세하게 친숙한 두 사람이 있으면 훨씬 적은 결함이 코드에 작성됩니다. 코드는 더 나은 구조를 갖게 되며 코드에 대한 상세한 지식은 많은 두뇌를 가진 것처럼 두 배가 됩니다. 이 점이 유일한 장점이더라도 충분하지만, 페어링 활동은 더 많은 장점을 제공합니다.

페어는 더욱 도전적입니다. 한 사람의 프로그래머는 시도를 꺼릴 수 있지만 페어로는 시도할 용기와 평가할 기술을 갖게 됩니다.

페어링은 팀워크를 조성합니다. 모듈은 한 사람이 작성한 것이 아니므로 코드는 특정 개발자가 아닌, 팀의 소유물이 됩니다.

페어링은 지식의 보급을 조성합니다. 더 많은 개발자가 서로 페어를 이룰수록 더 많은 시스템 지식이 전체 팀에 보급됩니다. 결과적으로 팀의 각 구성원은 그들의 특정 파트에 대한 지식뿐만 아니라 모든 시스템에 친숙하게 됩니다.

페어링은 생산성을 증진시킵니다. 혼자 프로그래밍하는 사람은 관련된 비활성 기간 다음의 에너지 버스트를 자세히 검토합니다. 페어는 서로 보조를 맞춥니다. 한 사람이 피곤해지면 역할을 바꿉니다. 일반적으로 한 사람이 허용할 수 있는 것보다 더 오래 효력를 유지하도록 관리합니다.

페어링은 즐겁습니다. 다른 개발자와 함께 일하면 교육적이고 활기가 있으며 즐겁습니다. 페어링은 작업 만족도 및 전체적인 의욕을 증가시킵니다.

실행

페어링

페어링(pairing)은 타스크를 담당하는 개발자가 누군가에게 도움을 요청할 때 시작됩니다. 규칙은 "요청했을 때 예라고 대답해야 합니다."입니다. 이것은 수행 중인 작업을 즉시 중단해야 함을 의미하지 않습니다. 오히려 도움을 제공할 수 있는 시간과 그 대신에 도움을 받을 수 있는 시간을 협의해야 함을 의미합니다.

페어를 이룬 동료는 타스크에 대한 책임을 맡지 않습니다. 그 책임은 여전히 타스크 소유자에게 있습니다. 또한 타스크가 완료될 때까지 페어를 이룬 동료는 소유자로 머무르도록 확정하지 않습니다. 페어를 이룬 동료는 도움 확인만 합니다.

페어의 한 구성원은 드라이버가 되고 다른 구성원은 관찰합니다. 드라이버는 코드를 입력하고 컴파일러를 실행하며 단위 테스트를 실행합니다. 관찰자는 각각의 키 누름, 명령 및 테스트 결과를 조사하고 도움 및 제안을 제공합니다. 두 사람은 항상 관련되어 있습니다.

때때로 드라이버가 수행할 작업에 대해 잘 알고 있고 관찰자는 단지 드라이버를 따르기만 합니다. 다른 경우에는 관찰자가 드라이버에게 수행할 작업에 대해 지시합니다. 때때로 드라이버가 부진해지면 관찰자에게 키보드를 건네주며 이로 인해 역할이 전환됩니다. 다른 경우, 관찰자는 키보드를 요청하고 역할을 전환합니다. 이는 페어링 세션에서 여러 번 발생하게 됩니다.

페어 변경

페어를 이룬 동료는 장기간 지속되지 않습니다. 일반적인 페어링 세션은 한나절 가량 지속됩니다. 또는 어떠한 이유로도 페어에서 나올 수 있습니다. 이런 경우에 타스크의 소유자는 다른 페어를 찾아야 합니다. 이것은 타스크 소유자가 지난 주에 함께 페어를 이루었던 누군가에게 호의를 갚아야 할 때임을 의미할 수 있습니다. 다른 한편으로는, 특히 까다로운 문제점을 도울 수 있도록 적절한 경험이 있는 사람을 요청해야 합니다.

이러한 페어 변경으로 인해 개발 팀 전체에 시스템 지식이 보급됩니다. 짧은 시간 내에 팀 구성원 모두는 거의 모든 시스템 파트에서 작업하는 데 시간을 소요하게 됩니다. 이는 프로젝트의 감도를 과감하게 회전율로 줄이며 모든 프로그래머가 더욱 자신있게 전체 시스템을 처리하도록 합니다.

협력 소유권

모든 직원이 모두 다른 시스템 모듈에서 작업하므로 아무도 특정 모듈을 소유하지 않습니다. 이는 시스템에 대한 책임이 모듈별 기반에 따라 나뉘어지지 않는 것을 의미합니다. 오히려 전체 팀은 공동으로 시스템 전체에 대한 책임이 있습니다. 팀 구성원은 어떠한 이유로 시스템에서 모듈을 점검하고 변경할 수 있습니다. 페어가 모듈 X 를 변경하고 이 변경사항이 모듈 Y 의 장치 테스트를 실패하게 하면 페어는 모듈 Y 를 수리합니다.

보조 맞춤 및 협력

페어 프로그래밍은 강력한 의사소통의 양식입니다. 언어로된 대화는 자주 충돌하며 외부 관찰자가 이해하는 데 문제가 있을 수 있습니다. 관찰자로서 페어는 다음과 같은 특이한 단어를 말하는 것을 들을 수 있습니다(예:“ 세미콜론” 또는 닫는 중괄호). 또는 프로그래머가 화면에 표시되는 컨텐츠에 동의 또는 반대하면서 불평하는 명료하지 않은 소리를 들을 수 있습니다. 이 둘은 많은 의사소통이 비언어적임을 나타내는 코드와 매우 직접적으로 연관되어 있습니다. 신체 언어는 중요한 부분을 담당합니다. 페어를 이룬 한 사람의 동료는 이러한 코드로 상대방이 불편한지를 말하지 않아도 알 수 있습니다. 얼굴을 찌푸림, 한숨, 불안하여 안절부절 못함 모두는 상호 작용을 하여 동료 간 의사소통의 대역폭을 늘립니다.

때때로 한 명의 동료는 마우스를 쥐고 다른 동료는 키보드를 조작합니다. 마우스를 잡은 사람은 작업이 발생하는 모듈 내에서 위치를 제어합니다. 키보드를 담당한 사람은 해당 위치에서 변경되거나 추가되는 컨텐츠를 제어합니다. 다른 경우에는 한 명의 동료는 입력을 하고 다른 동료는 발생할 함수 호출을 예견하고 코더에 스펙이 필요하면 API 문서를 올바른 페이지에서 엽니다.

한 명의 동료가 피곤해지면 다른 동료가 관찰자 역할을 수행하여 휴식을 취할 수 있도록 이끌어갈 수 있습니다. 다른 경우에는 두 동료 모두 높은 에너지가 있으며 키보드와 마우스를 자주 교환합니다.

요약하면, 몇몇 규칙 및 간단한 절차가 있습니다. 유일한 실제 제한조건은 두 사람 모두 관련되어 있어야 하며 이들 간의 의사소통은 강력해야 합니다. 한 동료는 입력 중이고 다른 사람은 창을 관찰 중인 페어는 정확하게 페어로 작업하는 것이 아닙니다.

개발자가 혼자 수행할 수 있는 것

항상 페어를 이룰 수는 없습니다. 일부 프로젝트(*eXtreme Programming*(XP)(참조서 [1] 참조) 프로세스를 채택하는 프로젝트)는 페어가 모든 제품 코드를 작성해야 한다는 규칙을 따릅니다. 이 경우, 페어를 이루지 않으면 전자 우편 확인, 새로운 설명 또는 API 로 읽기, 친숙하지 않은 코드를 통해 읽기 또는 현재 반복 또는 미래 계획에 대해 투자자와 상의할 수 있습니다. 실제로, 페어를 이룬 동료가 없는 경우 짧은 시간 내에 개발자가 찾아 수행할 수 있는 유익한 점이 항상 있습니다.

일부 프로젝트는 페어링에 덜 엄격합니다. 일부에서는 한 명의 개발자가 테스트를 작성할 수 있습니다. 기타 프로젝트에서는 한 명의 개발자가 추상 클래스 또는 인터페이스를 작성할 수 있습니다. 그러나 여전히 나머지 프로젝트에서 개발자는 페어링에 적합한 시기만을 결정할 수 있습니다. 그러나 한 가지 분명한 점은 —연구에서 페어링이 실행되면 결함율이 현저하게 떨어진다는 것을 보여 주었습니다.

페어링을 좋아하지 않는 사람

일부 직원은 페어링 개념을 불편하게 생각합니다. 경험상, 일주일 동안 실제로 페어링을 시도한 사람들은 불편함이 사라지고 이를 좋아하게 되며 그 유용함을 발견합니다. 소수만이 이 방법을 계속 좋아하지 않습니다. 따라서 대부분의 경우, 이는 단순히 시도 및 익숙해지는 것에 관한 문제입니다. 충실하게 시도해도 여전히 이 방법을 싫어하는 사람의 경우 팀은 이들이 수행할 수 있는 다른 적절한 방법을 찾아야 합니다.

설비, 기능 및 세부 계획

모니터 및 키보드 교체

설비 교체는 페어링을 완료하는 데 매우 중요합니다. 페어를 이룬 동료가 서로의 옆에 앉아 키보드를 재빨리 교환하지 못하면 페어를 이루어 작업을 제대로 할 수 없습니다. 규칙은 다음과 같습니다. 자리를 바꾸지 않고 키보드 및 마우스를 앞뒤로 건네줄 수 있어야 합니다.

일반적으로 가장 좋은 방법은 길고 평평한 테이블입니다. 모니터를 가운데에 놓고 두 개의 의자를 모니터를 보도록 둡니다. 모니터를 두 사람 사이에 두십시오. 두 사람 사이에서 키보드 및 마우스를 앞뒤로 이동하기 쉬운지 확인하십시오. 또한 키보드를 사용하는 동안 편안하며 똑바로 앉는지 확인하십시오. 모니터를 회전하지 않아도 두 동료 모두에게 잘 보이는지 확인하십시오.

Bullpen

페어를 이루는 동료 변경을 용이하게 하려면 종종 bullpen 조정으로 작업하는 것이 현명합니다. 몇몇 페어 부서를 하나의 방에 배치하십시오. 바퀴가 있는 의자 및 리놀륨 또는 타일 바닥을 사용하십시오. 페어가 서로를 마주보도록 워크스테이션을 배열하십시오. 이러한 목적은 의사소통을 증가시키는 것입니다. 때때로 가장 중요한 의사소통은 우연한 것입니다. 그러한 의사소통이 발생할 수 있는 기회를 증가시키고자 합니다.

내부 코너

요즘은 많은 칸막이에서 내부 코너에 워크스테이션을 배치해 두고 있습니다. 개발자는 모니터를 앞에 두고 칸막이의 코너를 마주보고 앉습니다. 이는 개별적인 작업에는 편리한 반면 이러한 환경에서 페어로 작업하는 것은 거의 불가능합니다. 내부 코너에 워크스테이션이 있는 칸막이인 경우에는 페어 부서를 회의실 같은 다른 장소에 배치하십시오. 랩탑을 사용하여 회의실 테이블에서 페어를 만들면 종종 상당히 효과적입니다.

문제점 및 고려사항

생산성을 반감하는 페어링

두 사람이 하나의 타스크에서 함께 일하면 한 사람이 같은 타스크에서 작업하는 것과 같은 인시의 두 배가 소비되는 것은 당연합니다. 이는 논리적이지만 같은 경우로 보이지 않습니다. 독립적인 연구(참조서 [2] 참조)에서는 페어로 작업하여 발생한 생산성 감소는 거의 없다고 보여주었습니다. 같은 연구에서 페어링은 대체로 결함 비율, *및 코드 크기*를 줄이는 반면, 작업의 즐거움은 증가시킨다고 보고했습니다.

페어를 이룬 동료 간 분쟁

타스크 소유자가 모든 설계 논쟁에 대해 마지막으로 언급하지만, 분쟁을 진정시키는 가장 좋은 방법은 두 아이디어를 모두 시도하여 가장 잘 작업하는 것을 선택하는 것입니다.

전문가

전통적인 지식은 특정 영역(예: 데이터베이스 또는 GUI)에 대해 전문화된 개발자는 그들의 노력을 단독으로 이러한 영역에 기울이도록 제안합니다. 그러나 페어 프로그래밍 환경에서 이런 전문가는 전문성을 공유하지 않는 다른 사람에게 조언자 역할을 하게 됩니다. 개발자는 전문 분야 이외의 타스크에 참가할 수 있으며 전문가로부터 도움을 구할 수 있습니다. 이러한 방법으로 전문 지식이 프로젝트 팀에 보급될 수 있으며 재편성에 대한 프로젝트의 민감성을 현저하게 줄여줍니다.

잡음

페어는 프로그램하면서 떠들 수 있습니다. 페어로 가득한 bullpen 에는 지속적이고 낮은 소음이 있게 됩니다. 이 소음을 귀찮고 산만하게 느끼는 사람도 있습니다. 이는 중요한 문제점은 아닙니다. 소음이 산만하게 한다고 생각하면 잠깐 회의실로 이동할 수 있습니다.

무법자

대부분의 팀에는 무법자 코더의 소유자임을 자랑스러워 하는 사람이 한두 명씩 있습니다. 이들은 다른 사람보다 작업을 빨리 완료하는 프로그래머로, 다른 사람과 작업할 수 없으며 아무도 그들의 코드를 읽지 못하거나 읽지 못하도록 합니다. 이러한 개발자에 대한 가장 좋은 방법은 이들을 프로젝트에서 제외하거나 제품 코드를 작성하지 않는 역할로 이동시키는 것입니다. 가능하다면 단기간 지속되는 도구를 작성하거나 전문가적인 힘든 테스트 작업을 수행할 수 있습니다.

실제 장애 및 장애 유형

일부 직원은 QWERTY 키보드를 사용합니다. 일부는 DVORAK 를 선호합니다. 일부는 특수 키보드, 마우스, 표시장치, 발 스위치 등이 필요합니다. 일부는 헤드폰과 큰 소리의 음악이 없으면 프로그램을 할 수 없습니다. 어떤 직원은 우스꽝스러운 빈 패키지로 주변을 둘러싸야 합니다. 일부 직원은 Emacs 를 좋아합니다. 다른 직원은 VI 를 좋아합니다. 게다가 다른 직원은 워드패드로 작업하고자 합니다.

실제로, 거론할 수 있는 장애는 셀 수 없이 많습니다. 그러나 이들 모두는 조금만 더 생각하고 타협할 능력이 있으면 해결될 수 있습니다. 이러한 문제점이 구성원을 방해하도록 하는 팀은 그들이 시도하는 어떤 노력에서도 성공하지 못하게 됩니다.

팀이 페어링을 계획하는 방법

페어로 타스크를 할당해야 한다고 생각하지 않습니다. 오히려 각각의 개발자가 타스크 세트를 담당해야 합니다. 비공식으로 페어를 구성하기를 원합니다. 각 개발자는 자신의 해당 업무를 수행하면서 다른 개발자에게 간단한 도움을 요청합니다. 타스크 소유자는 계속 소유권 및 책임을 유지합니다. 페어를 이룬 동료는 단지 협력자입니다.

각 개발자는 타스크에 대한 견적을 제공할 때 페어로 작업하게 될 총 시간을 고려해야 합니다.

결론

페어 프로그래밍은 코드 검토를 위해 충분히 테스트되고 적절하게 채택된 대안입니다. 또한 소프트웨어를 작성하는 방법과 근본적으로 다르지 않습니다. 그 장점은 생산성 및 품질 향상을 훨씬 뛰어넘는 것이며, 견고성 및 팀의 의욕과 같은 것에 영향을 줍니다.

참조서

[1] *eXtreme Programming eXplained*, Kent Beck, Addison Wesley, 2000.

[2] *Strengthening the Case for Pair Programming*, Laurie Williams, University of Utah, July/Aug 2000 IEEE Software.

Rational
the software development company

본사:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
전화번호: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
전화번호: (781) 676-2400

무료 전화번호: (800) 728-1212
전자 우편: info@rational.com
웹: www.rational.com
월드와이드: www.rational.com/worldwide