

소프트웨어 프로젝트 관리

아주 기초 개념과 아주 기초 실무 이야기

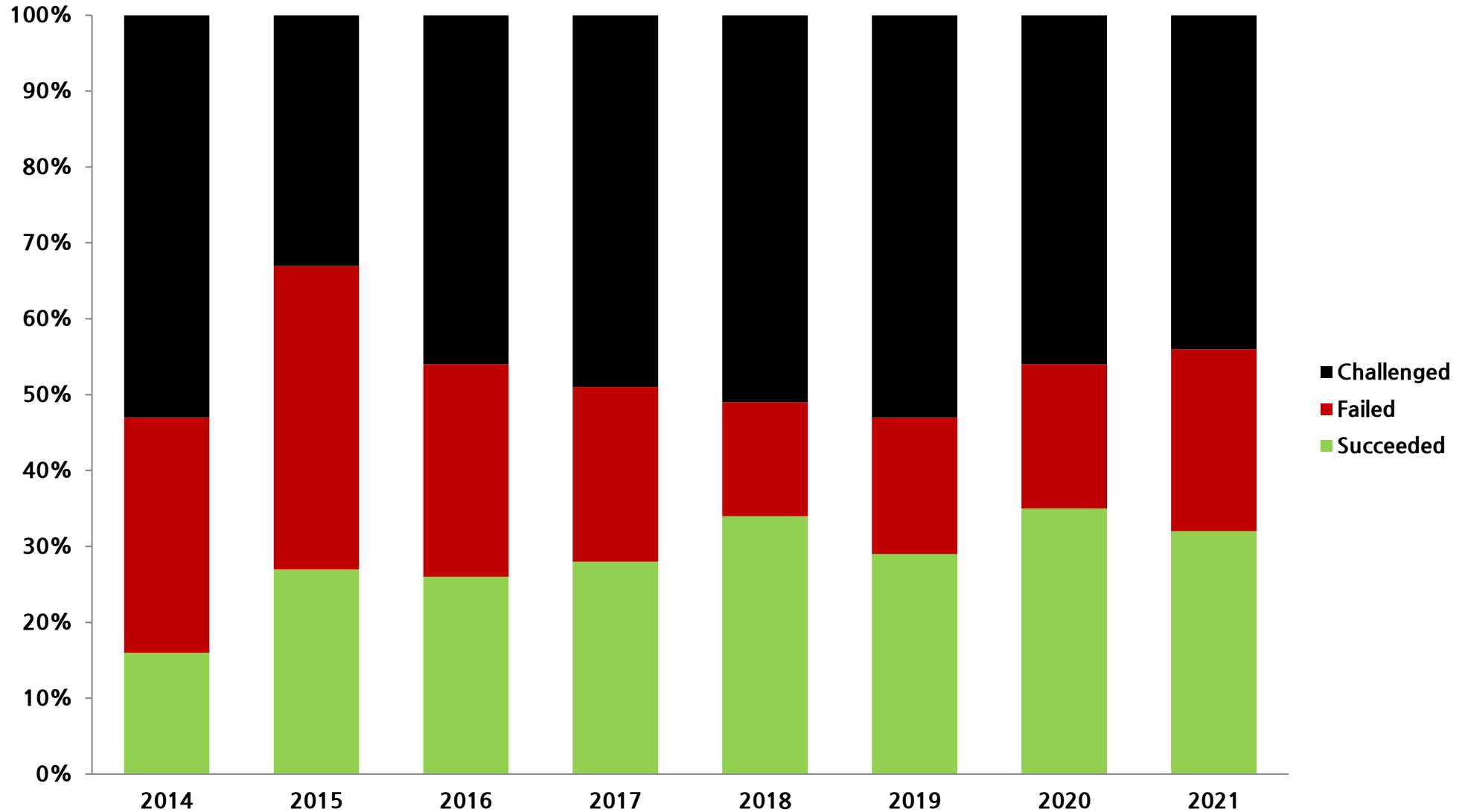
개념을 잘 잡읍시다

- SW
- SW Project
- SW Project 관리

프로젝트, 가만히 놔두면 亡한다

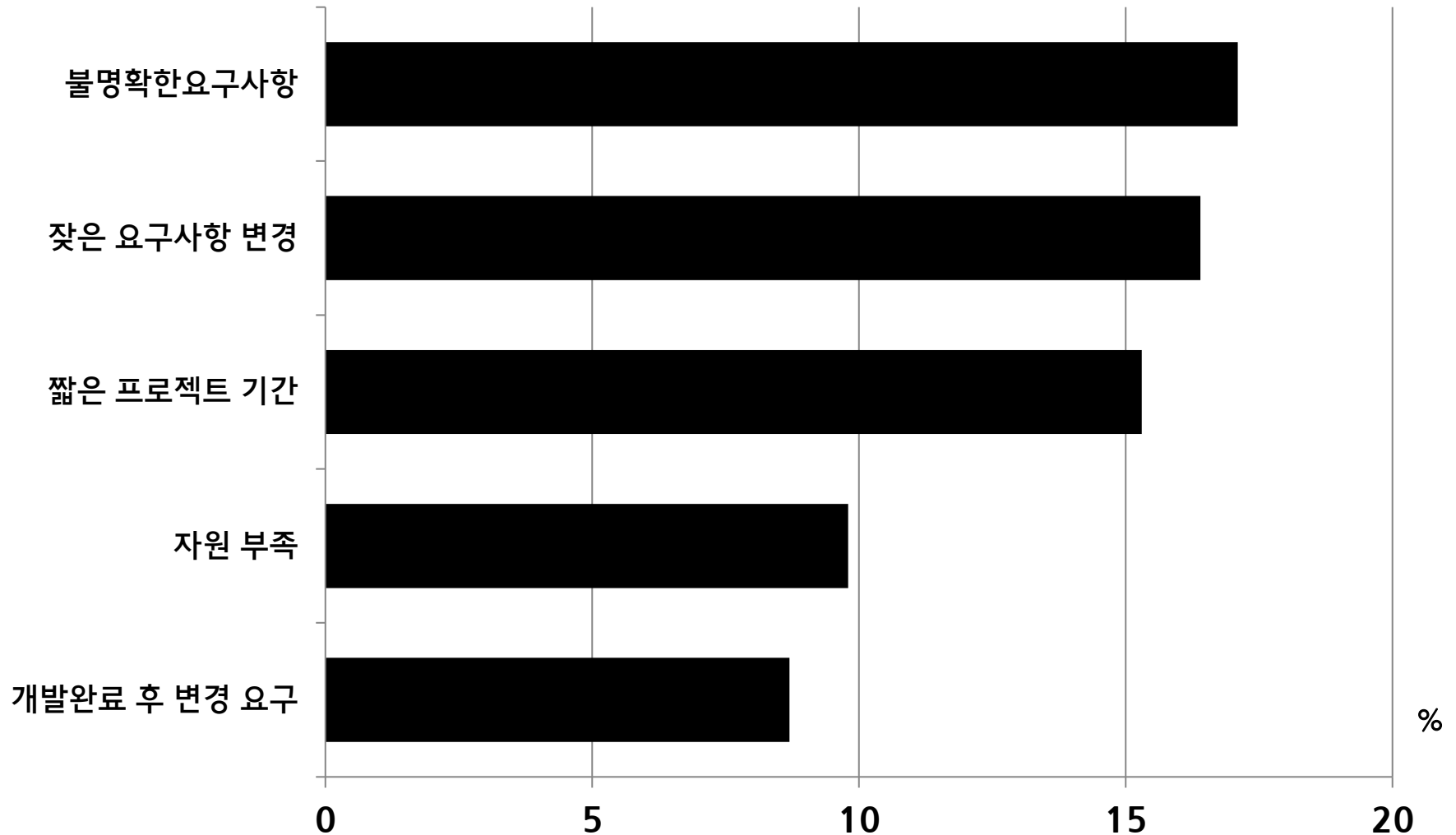
연도별 프로젝트 성공률 (Standish Group)

- Succeeded: 계획했던 예산, 일정, 요구사항을 준수
- Failed: 프로젝트가 완료되지 못하고 취소
- Challenged: 완료는 되었지만 예산이나 일정이 초과되거나 일부 요구사항을 준수하지 못함



프로젝트 실패의 원인

프로젝트 실패 원인 (NIPA, 2008)

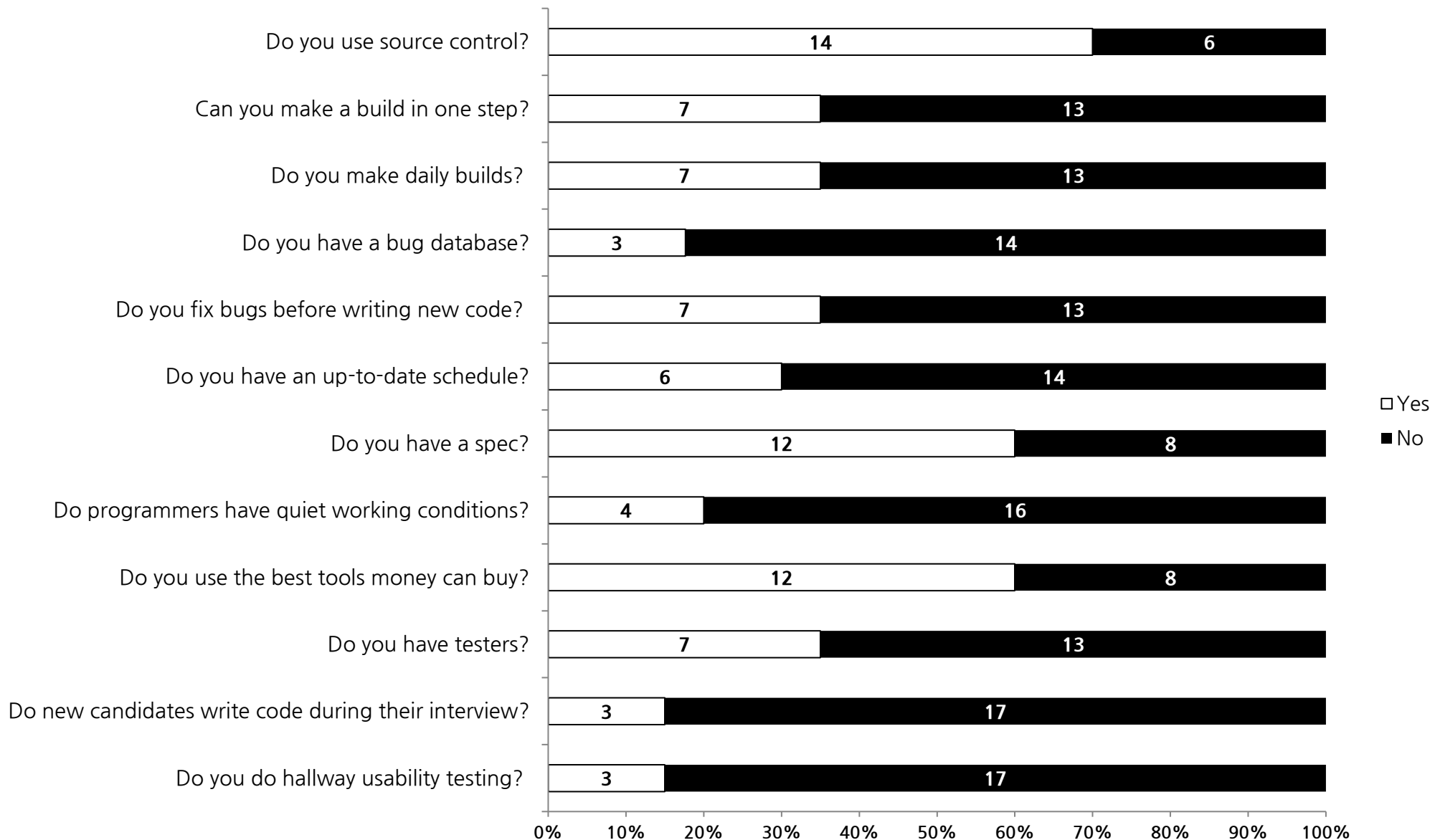


소프트웨어에 대한 이해로부터 관리가 시작됩니다

1. 소프트웨어 작업을 성공하려면 유능한 사람을 뽑으십시오.
2. 사람이 일을 잘 할 수 있는 환경(물리적, 문화적)을 만드십시오.
3. 의사소통을 어떻게 하면 잘 하게 할 것인가 고민하십시오.
4. 지적인 일에 집중하고 사무적인 일은 기계가 할 수 있도록 만드십시오.
5. 소프트웨어의 본질인 복잡성을 놓치지 마십시오.
 1. 신속한 f/b
 2. 실시간 알림
 3. 너무 멀리 가서 확인 하지 마십시오.

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

Joel Test



프레드의 하루

아침에 출근한 프레드는 월마가 보낸 이메일을 확인했습니다. 그녀가 지난 밤에 수정한 코드에 대한 글이었습니 다. 변경된 코드가 필요했기 때문에 프레드는 네트워크 드라이브에서 월마의 코드를 내려 받았습니다. 한 시간 정도 이리저리 만지작거리고 나서, 월마의 코드를 컴파일 되게끔 만들었습니다. 프레드는 몇 분 간 그녀가 수정한 코드를 두 번 확인했는데, 제대로 작동하는 듯했습니다. 그런 후에 프레드는 어제 써놓은 메모를 보고 기억을 되살렸습니다. 새 기능(Feature)을 거의 다 작성했습니다. 3일이나 작업한 끝에 마침내 오늘 점심 무렵에는 끝낼 수 있으리라 생각했습니다.

숫구치는 열정에, 프레드는 코드 에디터를 열었습니다. 그러나 소스 코드는 온데간데 없었습니다. 우울한 감정이 덮쳐 들었습니다. 프레드는 작업했던 결과를 월마의 코드로 덮어씌웠다는 걸 깨달았습니다. 3일 간의 작업이 30초 만에 날아갔고, 복구할 방법도 없었습니다. “젠장, 처음 있는 일도 아니지, 마지막일 리도 없고... 이 직업의 고난 중 하나일 뿐인데 뭘.”

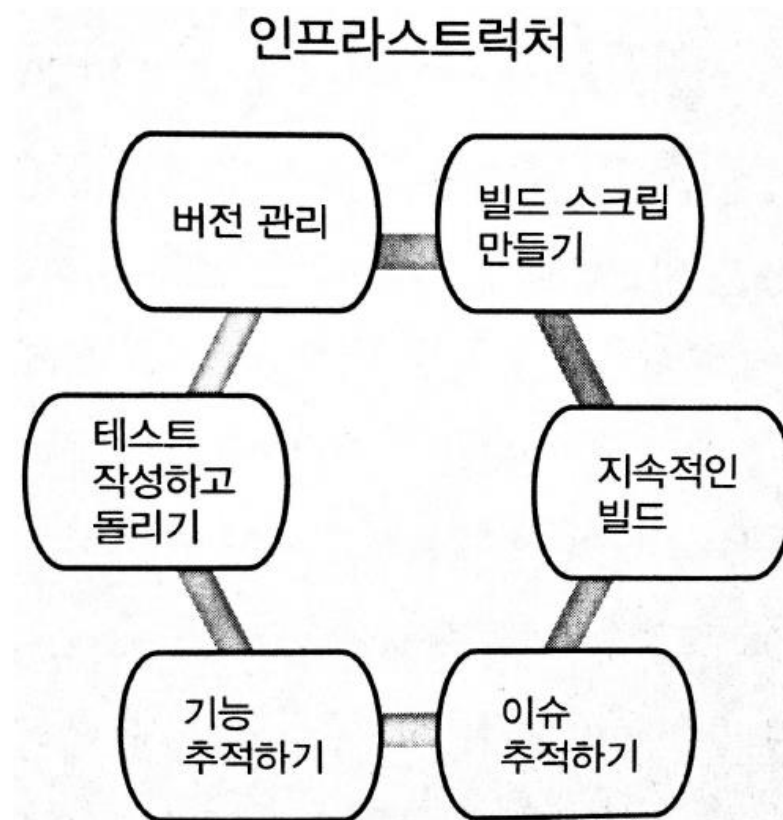
때마침 판매부서의 리처드가 지난주에 요청한 버그 수정 건이 얼마나 진전됐나 확인하려 들었습니다. 프레드는 움찔했습니다. 새 기능 추가 작업에 정신이 팔려 버그 건은 잊어버렸습니다. 리처드는 작업 지연을 탐탁지 않아 했고, 프레드는 오늘 내로 마무리하겠다고 약속했습니다.

프레드는 기한이 지난 버그를 오후 늦게서야 고칠 수 있었습니다. 고객에게 보낼 업데이트를 준비하다가, 고객은 자신이 수정한 버전이 아닌 이전 버전의 제품을 사용한다는 걸 깨달았습니다. 다른 직원들이 퇴근 준비를 시작할 무렵에 프레드는 아내에게 전화해서 저녁 식사 약속을 취소했습니다. 그런 후 수정된 코드를 적용하기 위해 네트워크 드라이브에서 이전 버전의 제품을 찾기 시작했습니다.

필요한 버전의 코드를 찾는 데 한 시간 정도가 걸렸고, 코드가 작동하게 하는 데 또한 시간이 걸렸습니다. 오래된 코드 베이스에 수정된 코드를 적용하는 데 한 시간을 더 쓰고 나서야 하루 일과가 끝났습니다.

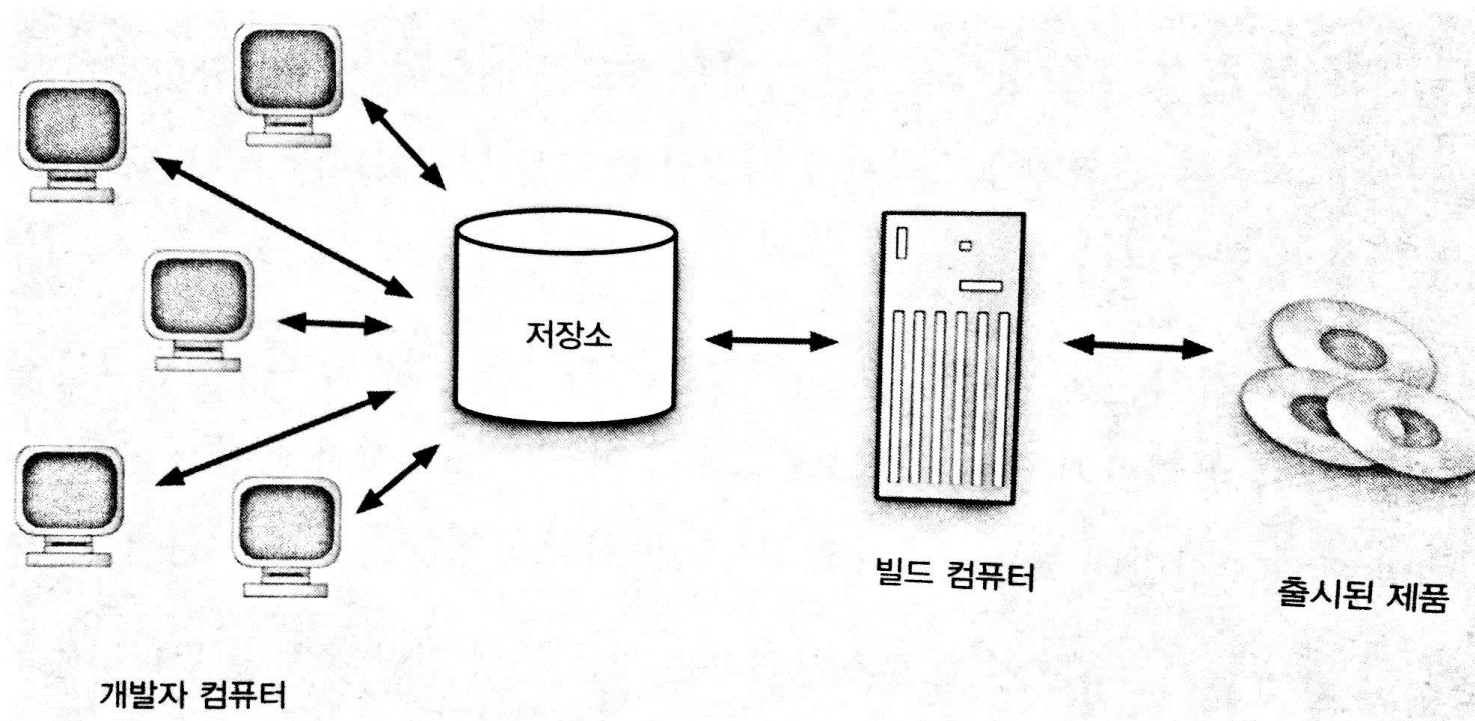
긴 하루였지만 프레드는 투자한 시간을 자랑스럽게 여겼습니다. 프레드는 회사의 성공을 위해 기꺼이 최선을 다해 일했습니다. 비록 그의 가족이 자신을 항상 이해해주는 것은 아니지만 말입니다.

다음날 고객은 프레드가 예전 버그를 다시 불러낸 데다가 새로운 버그를 두 개나 만들어냈다는 사실을 발견했습니다.



모래상자(Sandbox) 안에서 개발하기

사람이 서로에게 방해하지 않고 자신의 일에 집중할 수 있는 작업 환경을 만듭니다



1) 자산을 관리하세요

- SCM(Source Code Management) 시스템이 주는 유익

- 팀이 프로젝트 규모의 () 버튼을 갖게 됩니다. 어떤 것도 최종적이지 않고 실수를 쉽게 돌이킬 수 있습니다.
- 여러 명이 동시에 한 파일을 편집할 때 충돌을 조정할 수 있습니다.
- 여러 버전의 소프트웨어를 ()할 수 있습니다. 지난 출시 제품의 버그를 고치는 와중에도 다음 출시 제품에 기능을 추가할 수 있습니다.
- 어떤 파일이 변경됐는지 기록할 수 있습니다(언제 누가 고쳤는지도 기록할 수 있습니다).
- 특정 시점의 작업을 () 꺼내올 수 있습니다.

- 제품에 필요한 모든 것을 SCM에 보관하세요

자산 관리 시작하는 방법

1. 몇몇 SCM 시스템을 평가해보세요. Git, SVN 강추!!
2. SCM 사용법을 익히세요.
3. 흔히 쓰는 기능을 어떻게 사용하는지 알려줄 한 페이지짜리 quick-start guide를 만드세요
4. 팀원에게 시스템을 보여주세요.
5. 코드와 지원 파일을 임포트(import)하세요.
6. 모든 파일을 SCM으로 관리하세요.

내재화 점검 항목

- 코드 체크인, 체크아웃 간격이 하루 안에 이뤄집니까?
- 개인 PC가 망가져도 개발에 지장이 없습니까?
- 새 기계를 가져와 개발 준비를 하는데 시간이 별로 걸리지 않습니까?
- SCM은 최고의 서버에 설치되어 있습니까?
- SCM 저장소는 정기적으로 Backup을 받고 있습니까?

2) 빌드를 스크립트화 하세요

- () 에서 빌드하는 것을 의미하지 않습니다.
- () 은 OS patch 버전, IDE 링크, 라이브러리 버전 등에 따라 빌드되는 현상이 제각각 일 수 있습니다
- 수동 빌드는 사람이 실수할 수 있으며, 실수를 만회하기 위한 노력이 만만치 않게 들어갑니다
- 프로젝트 초반에 빌드를 () 해야 합니다
- 어떤 컴퓨터에서라도 빌드가 되어야 합니다

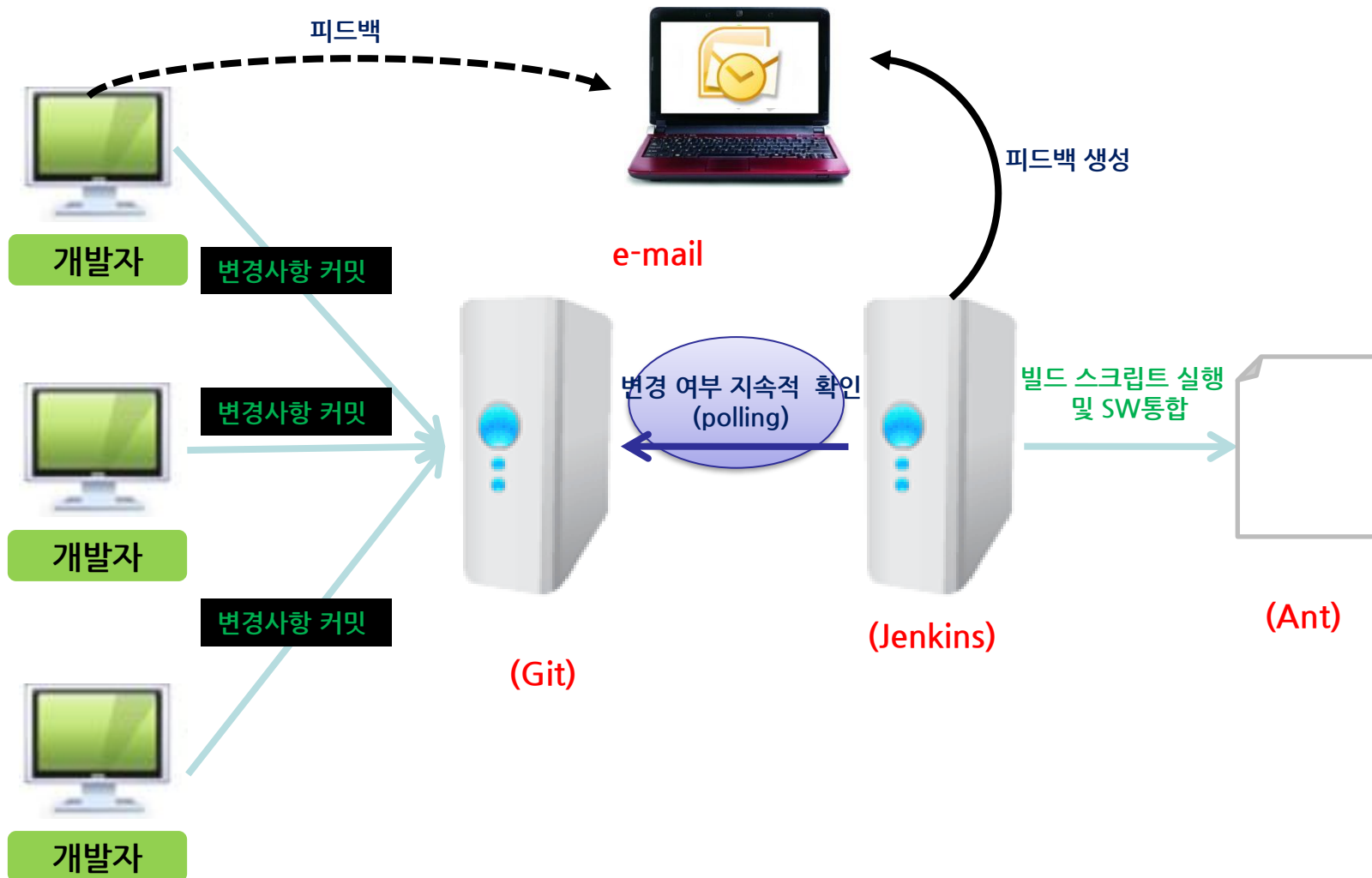
빌드 스크립트화를 시작하는 방법

1. 여러분이 스트립트를 작성하는 동안은 팀 구성원 한 명을 뽑아 수동으로 시스템을 빌드하게 하세요.
2. 각 빌드 단계를 정의하세요.
3. 빌드 도구를 선택하되 그 도구가 짐이 되면 다른 선택사항을 다시 검토할 준비를 하세요.
4. 각 단계를 점진적으로 스크립트화하세요. 수작업을 하나하나 제거해나가세요.
5. 다른 컴퓨터에서 스크립트를 돌려보세요. 특정 컴퓨터에서만 작동되는 코드를 찾아낼 수 있습니다.
6. 이제 여러분이 도와주지 않아도 다른 팀 구성원이 스크립트를 사용할 수 있는지 확인해보세요.

- 명령어 하나면 빌드가 됩니다.
- 소스 코드 관리 시스템(SCM)에서 코드를 내려 받아 빌드할 수 있습니다.
- 어느 컴퓨터에서라도 빌드가 됩니다.
- 외부 환경(특정 네트워크 드라이브 같은)에 의존하지 않고도 빌드가 됩니다.
- 빌드에 수작업 단계가 포함되면 안됩니다.
- 다른 컴퓨터에서 돌리려면 빌드 스크립트를 수정하면 안됩니다.
- 팀 구성원 중 몇 명만이 빌드 스크립트를 수정할 줄 알면 안됩니다.
- [빌드 스크립트 예제](#)

3) 자동으로 빌드하세요

- 빌드 스크립트화가 완료되어야 가능합니다.
- 자동 빌드는 기계가 빌드를 하므로 여러분은 ()에 집중하시면 됩니다.
- 자동으로 빌드할 수 있다면 일단 코드 변경 시 마다 다시 빌드해야 하며 이로 인해 어떤 변경 사항이 빌드를 깨뜨리는 지 즉시 알 수 있습니다.
- 너무 오래 지나 빌드가 깨진 걸 안다면 복잡성 때문에 원인을 찾고 수정하는데 많은 비용이 발생합니다.
- 이런 종류의 시스템을 지속적인 통합(Continuous Integration)이라 합니다.
- CI 도구는 누군가 코드를 커밋할 때마다 프로젝트를 다시 빌드 할 수 있습니다. 이로 인해 코드 베이스를 깨끗하게 유지합니다.
- 대표적인 CI 도구는 Jenkins 라는 오픈소스 도구 입니다.



CruiseControl - Builds - Windows Internet Explorer

http://163.239.99.229:8080/dashboard/tab/builds

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(D) 도움말(H)

즐거찾기 Google Gmail Calendar SEMOSS REVET Mantis 정보통신대학원 로컬라도서관 NFR workspace Daum 영머사전 beemp3.com osun.org

CruiseControl - Builds

Dashboard Server : Tensai-PC

Dashboard Builds Administration

Summary

0 project build(s) failed
 2 project build(s) succeeded

2 project build(s)
 0 project(s) building
 0 project(s) discontinued

0 project(s) inactive

100% of projects passing

Tools

RSS Feed
 for CCTray
 for CC-Config

connectfour passed about 9 hours ago

ReqTracer passed 14 minutes ago

완료 인터넷 100%

[2Team CruiseControl Build] ReqTracer build.5 Build Fixed

Inbox | X

☆ 2Team CruiseControl Machine to User, h.hyunjin, skib1, me, aha1970 [show details](#) Apr 11 [Reply](#)

BUILD COMPLETE - build.5

Date of build: 2010-04-10T15:30:27
 Time to build: 4 minutes 19 seconds
 Last changed: 2010-04-09T01:43:46
 Last log entry:

Errors/Warnings: (2)

Note: Some input files use or override a deprecated API.
 Note: Recompile with -Xlint:deprecation for details.

Unit Tests: (0)

No Tests Run

This project doesn't have any tests

Modifications since last successful build: (1)

change	User	force build/force build	2010-04-09T01:43:46
--------	------	-------------------------	---------------------

Deployments by this build: (1)

Building jar: c:\cruisecontrol\projects\ReqTracer\dist\archive\ReqTracer.jar

빌드 자동화를 시작하는 방법

1. 사용할 자동 빌드 시스템을 선택하세요.
2. 빌드 시스템을 돌릴 깨끗한 컴퓨터를 구하세요
3. 자동 빌드 시스템을 설치하고 여러분의 환경에 맞춰 설정하세요. 모든 설치 과정을 문서화해 놓으세요.

- 시스템에 테스트가 포함되어 있습니까?
- 시스템에 신경 쓰는 사람이 있습니까? 알림 시스템은 켜져 있습니까?
- 빌드가 깨지면 빨리 고쳐집니까? 아니면 며칠씩 그대로 방치됩니까?
- 합리적인 시간 내에 빌드가 끝납니까? 아니면 너무 오래 걸려서 끝날 기미가 안 보입니까?

4) 이슈를 추적하세요

- 누군가 버그나 다른 중요한 이슈를 보고하면 이를 잊지 않고 관리해야 합니다.
- 화이트보드나 포스트잇은 어느 순간 감당할 수 없게 됩니다. 소프트웨어가 점차 복잡해지기 때문이죠.
- 이슈가 남아내는 내용들
 - 어느 버전의 제품에 이슈가 있습니까?
 - 어느 고객이 그 이슈에 부딪혔습니까?
 - 얼마나 심각합니까?
 - 그 문제가 사내에서 재현되었습니까?
 - 고객의 환경은 무엇입니까?
 - 어느 버전의 제품에서 이슈가 처음 발생했습니까?
 - 어느 버전의 제품에서 이슈가 고쳐졌습니까?
 - 누가 이슈를 고쳤습니까?
 - 누가 수정 사항을 검증했습니까?
- 대표적인 이슈추적시스템은 Jira, Redmine 등이 있습니다

이슈 추적을 시작하는 방법

1. 이슈 추적 시스템을 고릅니다
2. 여러분이 사용할 이슈 추적 시스템을 구축하고 사용법을 익힙니다.
3. 내부 사용자를 위해 한 쪽짜리 quick-start guide를 만듭니다.
4. 이 시스템에 새 이슈를 보관하기 시작합니다.
5. 시간이 나면 이전 이슈를 새 시스템으로 옮깁니다.

- 아직 해결 못한 최우선사항의 목록을 생성해낼 수 있습니까? 두 번째 계층에서 발생한 문제는 어떻게 됐습니까?
- 지난 주에 고친 것들의 목록을 생성할 수 있습니까?
- 시스템이 해당 이슈를 수정한 코드를 참조할 수 있습니까?
- 기술 리더가 이 시스템을 사용해서 개발팀이 할 일을 목록으로 뽑아낼 수 있습니까?
- 시스템에서 정보를 가져오는 방법을 기술 지원 팀이 알고 있습니까?
- 어떤 이슈가 고쳐졌을 때 다른 부서가 알 수 있도록, 시스템이 ‘관심 있는 관계자’에게 통지해줄 수 있습니까?

5) 테스트 장비를 사용하세요

- 테스트 장비란 테스트를 만들고 실행시키는 데 사용할 수 있는 도구 또는 소프트웨어 툴킷을 말합니다.
- 테스트를 자동화하지 않았다면, 스크립트로 테스트를 돌릴 수가 없습니다. 테스트 수트를 돌릴 사람이 필요하고, 그러면 시간과 돈이 듭니다.
- CI 시스템 내부에서 돌아가는 좋은 자동화된 테스트 수트만큼 품질을 높일 수 있는 실천방법을 본 적이 없습니다.
- CI 시스템과 연계한 단위테스트프레임워크를 도입하세요. JUnit, NUnit, CppUnit 등이 대표적입니다.
- 시스템 테스트 또한 관련 test case를 CI와 연동하여 빌드할 때마다 테스트가 진행되도록 하세요.
- 오픈소스도구 중에서 코드 인스펙션도구, 정적분석도구, 코딩룰체크 도구등과도 연계하여 결과를 가공하여 정리하여 보세요.

테스트 장비 사용을 시작하는 방법

1. 테스트 도구와 툴킷을 선택하세요.
2. 문제 영역에 테스트를 추가하세요.
3. 테스트가 자동 빌드 시스템 내에서 돌아가도록 하세요.

- 테스트 수트가 효율적입니까? 테스트가 버그를 잡아내고 있나요?
- 코드 적용 범위의 숫자는 얼마인가요? 시간이 지남에 따라 숫자가 커지고 있나요?
- 여러분이 테스트하는 제품이 안정적입니까?
- 테스트가 자동으로 실행되고 있습니까?
- 테스트가 성공했는지 실패했는지 알려줍니까?
- 여러분 회사의 모든 사람이 테스트를 추가할 수 있습니까?

매일 인프라를 사용한 결과로 프로젝트 현황을 관리해야 합니다

The screenshot displays the SonarQube web interface in a browser window. The address bar shows the URL `220.85.176.7:9000/sonar/`. The interface includes a sidebar with navigation links: Home, Filters, Reviews, Dependencies, Motion chart, Radiator, and Views. The main content area is titled 'Projects' and shows a table of project quality metrics. The table columns are: Alert, Name, Lines of code, Classes, Methods, Line coverage, Branch coverage, Rules compliance, Duplicated lines (%), Complexity (class), and Complexity (method). The table lists six projects, with 'oasis_security_parent' and 'oasis_supplementary_parent' showing critical alerts (red exclamation marks) and low branch coverage (0.0%).

Alert	Name	Lines of code	Classes	Methods	Line coverage	Branch coverage	Rules compliance	Duplicated lines (%)	Complexity (class)	Complexity (method)
	QASIS_EPCNetwork - Parent	137	6	25	0.0%	0.0%	97.8%	0.0%	4.5	1.1
	oasis.edi.parent	3,372	127	273	69.6%	73.0%	96.7%	0.0%	3.9	1.8
	oasis.persistence.parent	1,095	28	31	56.3%		95.2%	0.0%	1.1	1.0
	oasis_security.parent	112	4	9	0.0%	0.0%	50.0%	0.0%	3.8	1.7
	oasis.servicencore.parent	4,250	108	320	34.1%	20.6%	98.4%	0.0%	6.2	2.1
	oasis.supplementary.parent	194	10	12	0.0%	0.0%	77.3%	0.0%	1.7	1.4

6 results [Alerts feed](#)

Powered by SonarSource - Open Source L&GPL - v.2.8 - Plugins - Documentation - Ask a question - Bug/feature request

소프트웨어 프로젝트 관리

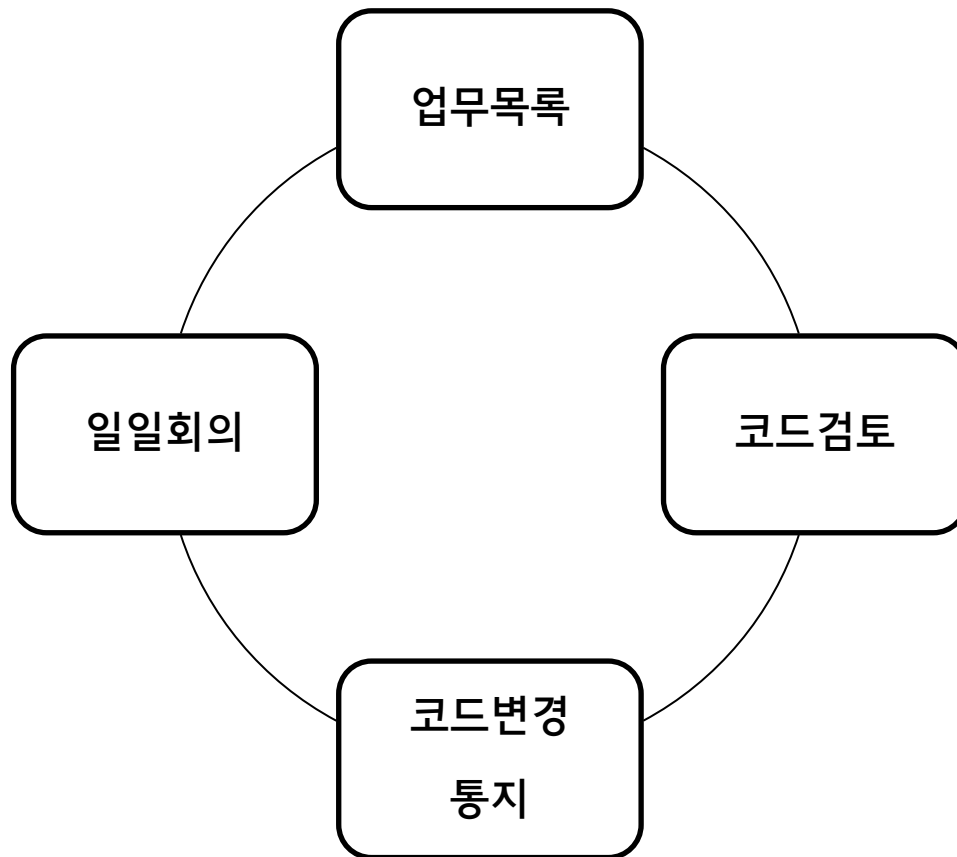
아주 기초적인 Practice(습관)

소프트웨어에 대한 이해로부터 관리가 시작됩니다

1. 소프트웨어 작업을 성공하려면 유능한 사람을 뽑으십시오.
2. 사람이 일을 잘 할 수 있는 환경(물리적, 문화적)을 만드십시오.
3. 의사소통을 어떻게 하면 잘 하게 할 것인가 고민하십시오.
4. 지적인 일에 집중하고 사무적인 일은 기계가 할 수 있도록 만드십시오.
5. 소프트웨어의 본질인 복잡성을 놓치지 마십시오.
 1. 신속한 f/b
 2. 실시간 알림
 3. 너무 멀리 가서 확인 하지 마십시오.

There is no silver bullet

There is only practice



목록에 따라 일하세요

- 개발할 시스템의 요구사항을 ()으로 만드세요
- 목록에 들어가야 할 요소
 - “누가, 언제까지, 무엇을, 얼마나 중요하게, 얼마 동안”
- 목록관리로 도구로 활용할 수 있는 것
 - Jira issue에서 subtask
 - Trello
- 기능마다 조리에 맞게 ()를 매긴 포괄적인 목록이 있으면, 여러분의 노력에 의존하는 팀과 관리자, 그리고 다른 팀도 자신감을 가질 수 있습니다.
- 목록을 사용하여 팀의 기민함이 엄청나게 증가합니다.
 - 제품을 기능 요구사항으로, 기능 요구사항을 목록의 항목으로 분해함으로써 기본적인 ()을 미리 끝내는 효과가 있습니다.

목록을 어떻게 사용해야 할까요?

1. 여러분이 개발하고 있는 모든 기능을 ()에 적어놓으세요. 이렇게 하려면 시간도 제법 걸립니다.
 2. 각 기능에 ()를 부여하세요. 이 과정에 적절한 **이해관계자**를 참여시키세요. 이 과정에 **팀 전체를** 참가시키는 게 이상적입니다.
 3. 모든 기능 요구사항을 우선순위에 따라 정렬해서 다시 쓰세요.
 4. 각 항목에 예측 시간을 다세요.
 - 1~2일 사이에 완료할 수 있는 수준으로 분할
- **현재의 상위 우선순위 항목들이 완료될 때까지 누구도 그보다 낮은 우선순위 항목을 처리해선 안됩니다.**

효과적으로 목록을 사용하는 규칙

- 누구나 사용할 수 있어야 합니다.
- 우선순위를 부여해야 합니다.
- 일정에 기반해야 합니다.
- 죽어 있는 문서가 아닌 살아있는 문서여야 합니다. (이메일 첨부로 보내는 습관을 버리세요. 링크를 걸어주세요.)
- 측정 가능해야 합니다.
- 목표가 있어야 합니다.

목록 예제

- 우선순위 별로 정렬된 개인용 목록 예제

1. 일별로 생산되는 위젯을 보여주는 새 보고서를 추가하기
2. 직원 별로 만들어내는 위젯을 보여주는 새 보고서를 추가하기
3. 버그 #12345 확인하기 (5달러짜리 보고서를 보여줄 때 월별 위젯이 0을 표시한다.)
4. 내 새 컴퓨터에 개발 도구를 설치하기
5. 멋진 새 그게뭐더라-보고서를 확인하기… 다음 버전의 보고서 시스템의 쓸만한 구성물이 될지도 모르겠다.

목록을 제대로 활용하지 못하는 증거

- “너무 바빠서” 목록에 작업을 넣지 못했습니다.
- 작업을 끝내는 일보다 ‘목록’을 갱신하는 데 더 많은 시간이 소모됩니다.
- 팀 구성원들이 개인용 목록에 있는 각 항목을 끝내는 데 몇 주 씩 걸립니다.
- ‘목록’이 일주일에 한 번도 갱신되지 않습니다.
- ‘목록’에 적힌 우선순위가 ‘진짜’ 우선순위와 일치하지 않습니다.
- ‘목록’이 엄중하게 비밀로 유지되어 팀 외부 사람은 보지 못합니다.
- 팀의 목록과는 다른 버전이 공개되어 있습니다.

소프트웨어에 대한 이해로부터 관리가 시작됩니다

1. 소프트웨어 작업을 성공하려면 유능한 사람을 뽑으십시오.
2. 사람이 일을 잘 할 수 있는 환경(물리적, 문화적)을 만드십시오.
3. 의사소통을 어떻게 하면 잘 하게 할 것인가 고민하십시오.
4. 지적인 일에 집중하고 사무적인 일은 기계가 할 수 있도록 만드십시오.
5. 소프트웨어의 본질인 복잡성을 놓치지 마십시오.
 1. 신속한 f/b
 2. 실시간 알림
 3. 너무 멀리 가서 확인 하지 마십시오.

일일 회의

- 주간회의: 공지사항, 지난 주의 업무, 다음 주 계획
- 일일 회의: 일정에 심각한 악영향을 주지 않고도 상호작용이나 의사소통을 장려하는 간단한 팀 회의
- 일일회의 Key point
 - 구성원이 무슨 일을 하고 있는지,
 - 어떤 문제에 부딪혔는지 간단하게(1인당 1~2분) 공유
 - 회의는 짧게 핵심만 다룹니다
- 일일회의가 필요한 이유
 - 의사소통의 실패를 해결하는 가장 쉬운 방법은 팀과 보다 자주 이야기를 나누는 것입니다.
 - 문제 발견 시간이 짧아지고, 고칠 기회가 더 많아집니다.

일일회의의 이득

- Is intended to fast update everyone on each team's progress
- For usually 15 to 20 minutes maximum
- Held at the same time and the same place by using the same method of communication with all participant's short speech
- During the meeting, each team member answers three questions:
 - What have you done since yesterday?
 - What are issues on what you did yesterday?
 - What are you planning to do by tomorrow?
- Be cautious!! Daily meeting is intended for synchronization, not(_____)!

이렇게 시작하세요

- 모든 사람이 을 알게 하세요.
- 모든 사람이 질문에 대답해야 합니다. 넘어갈 수도 없고 예외도 없습니다.
- 처음에 느긋하게 시간 제한을 두세요. 시작할 무렵엔 많은 정보가 오갈 테니, 자유롭게 의사 소통하게끔 내버려 두어야 합니다.
- 회의를 같은 (), 같은 () 에서 매일 여세요.
- 일일 회의를 일종의 ()처럼 만드세요. 어쩔 수 없이 계속해야 하는 허드렛일처럼 만들어선 안됩니다.
- 일일 회의 시간에 논의한 주제를 웹 페이지나 블로그에 올립니다.
- 한 사람을 골라 발표하게 하고, 시계 방향 또는 반 시계방향으로 돌아가며 발표하게 합니다. 무작위로 한 사람씩 발표하게 하면 사람들을 좀 더 쉽게 긴장시킬 수 있습니다.

이렇게 하고 있다면 제대로 하고 있는 겁니다

- 회의가 유용하다고 팀원들이 느낍니다.
- 매일 같은 장소, 같은 시간에서 열립니다.
- 회의가 열리지 않으면 사람들이 불평을 합니다. 귀한 정보를 공유할 수 있다는 경험을 갖게 되었기 때문입니다.

이렇다면 경고 신호입니다.

- 팀 구성원이 각자가 10분 이상을 씁니다.
- 단 한 명이 다른 모든 사람을 합친 것만큼의 시간을 씁니다.
- 서로를 비열하게 야유해댁니다.
- 허구한날 회의가 늦게 시작되거나 늦게 끝납니다.
- 개발자들이 “90% 끝냈습니다” 라던가 단순히 “인증 기능을 만들고 있습니다”라는 식의 주장만 하고, 알맹이는 없는 회의가 열립니다.
- 팀 구성원들이 자신이 한 일을 잊고 보고하지 않거나 두서없이 설명합니다.

소프트웨어에 대한 이해로부터 관리가 시작됩니다

1. 소프트웨어 작업을 성공하려면 유능한 사람을 뽑으십시오.
2. 사람이 일을 잘 할 수 있는 환경(물리적, 문화적)을 만드십시오.
3. 의사소통을 어떻게 하면 잘 하게 할 것인가 고민하십시오.
4. 지적인 일에 집중하고 사무적인 일은 기계가 할 수 있도록 만드십시오.
5. 소프트웨어의 본질인 복잡성을 놓치지 마십시오.
 1. 신속한 f/b
 2. 실시간 알림
 3. 너무 멀리 가서 확인 하지 마십시오.

코드 리뷰

- 코드리뷰는 소스 코드를 비판하고 검사하여 코드를 개선하는 것을 의미합니다.
- 우리(개발자)는 코드도 만들어 내고, ()도 만들어 냅니다.
- 만들어 낸 사람들이 코드를 검토하고(Code self inspection:CSI), 테스트를 하기 전에 관련자들이 모여서 검토하는 것은 결함을 찾는데 있어서 코드 검토가 매우 효율적입니다.
- 코드 검토는 버그를 찾는데 있어 으뜸가는 방법입니다. 이보다 나은 방법은 없습니다(“ship it” 中)
- 코드를 리뷰하지 않으면 소프트웨어의 결함 존재 확률이 급격히 증가한다.
- “프로그래머는 설계를 할 때 한 시간에 보통 1~3개의 결함을 만들어내고, 코드를 작성할 때는 5~8개의 결함을 만들어냅니다. 그리고 테스트를 할 때는 한 시간에 2~4개의 결함을 제거하지만, 코드 리뷰를 하는 동안에는 한 시간에 6~12개의 결함을 찾아낸다.”

코드 리뷰의 장점

- 소프트웨어의 품질을 높입니다
- 코더에게 스스로 만든 코드에 대한 책임감을 부여하며 () 문화 확립
- 전체 프로젝트의 코딩 스타일에 통일성을 부여합니다.
- 지식 전파 (코드의 핵심 부분의 내부 작동에 대한 지식)

이렇게 시작하세요

코드 검토는 강력한 도구입니다. 일단 습관만 들이며, 예전엔 어떻게 검토도 하지 않고 코드를 작성했었는지 의아해할 겁니다.

- 여러분이 계획하는 코드 검토가 어떤 건지 모든 사람이 이해해야 합니다. ()로 ()검토하세요. 몇 주씩 검토하지 않고, 수백 또는 수천 줄이나 되는 변경 내역을 쌓아놓지 마세요.
- 처음 몇 주나 몇 달간은 선임 개발자가 코드 검토에 매번 참석하게 하세요. 이렇게 하면 튼튼한 기반 위에서 지식을 공유하고 검토를 받기에 아주 좋습니다.
- 코드 검토는 가벼워야 합니다. 너무 많은 코드를 검토하기보단 차라리 ()을 검토하는 편이 낫습니다. 거창하게 한번 검토하는 것보단 겹치더라도 두 번 검토하는 편이 낫습니다.
- 이번 기회에 코드 변경 통지 시스템을 도입하세요.
- 팀 구성원 모두가 참여하기 전에 관리자 층의 지지를 이끌어내도록 하세요.

- 코드 검토를 거의 하지 않습니다.
- 코드 검토가 거의 매번 고통스럽습니다.
- 사람들이 코드 검토를 하기 싫어해서 코드를 체크인하길 기피합니다.
- 코드를 검토했던 팀 구성원들이 코드가 무슨 일을 하는지, 코드를 작성한 이유가 무엇인지 설명하지 못합니다.
- 신참 팀 구성원이 다른 신참의 코드만 검토합니다.
- 모든 사람이 한 사람에게만 검토를 맡깁니다.

소프트웨어에 대한 이해로부터 관리가 시작됩니다

1. 소프트웨어 작업을 성공하려면 유능한 사람을 뽑으십시오.
2. 사람이 일을 잘 할 수 있는 환경(물리적, 문화적)을 만드십시오.
3. 의사소통을 어떻게 하면 잘 하게 할 것인가 고민하십시오.
4. 지적인 일에 집중하고 사무적인 일은 기계가 할 수 있도록 만드십시오.
5. 소프트웨어의 본질인 복잡성을 놓치지 마십시오.
 1. 신속한 f/b
 2. 실시간 알림
 3. 너무 멀리 가서 확인 하지 마십시오.

코드 변경 통지 보내기

- 코드의 변경된 내역을 관련자들에게 보냅니다.
- 이런 지식의 공유는 정말 () 정도의 효과를 보입니다.
- 반드시 변경된 내역이 이메일로 () 전송되어야 합니다.
- 이는 소프트웨어 복잡성으로 인해 생기는 파급효과를 바로 차단하는 효과가 있습니다.
- 상당수의 사람들이 시도해보기도 전에 () 합니다.
- 한달 정도 지나면 가장 불만이 많은 사람들이 여러분에게 () 할 것입니다

전형적인 코드 변경 내역

DB 패키지 내의 TdDataFile에 createRecord()를 추가했다.

Index: com/tde/db/TdDataFile.java

```
=====
RCS file: c:\apps\cvs\cvsroot\WeissDB/com/tde/db/TdDataFile.java,v
retrieving revision 1.3
diff -r1.3 TdDataFile.java
381a382,401
```

```
> /**
>  * 이 테이블에 저장된 레코드 타입의 새 인스턴스를 반환한다.
>  * @return TdRecord - 레코드의 새 인스턴스
>  * @exception ClassNotFoundException - 저장된 클래스 이름을 찾지 못하면
>  * 이 예외를 던진다.
>  * @exception InstantiationException - 저장된 클래스 이름이 만들어지지 않으면
>  * 이 예외를 던진다.
>  * @exception IllegalAccessException - 저장된 클래스 이름을 대표하는 클래스가
>  * 잘못된 범위(scope)를 가지거나 매개변수를 안 갖는 생성자가 없다면
>  * 이 예외를 던진다.
>  */
> public TdRecord createRecord() throws ClassNotFoundException,
>     InstantiationException,
>     IllegalAccessException {
>     // 레코드의 새 인스턴스를 생성한다.
>     return (TdRecord) (Class.forName(getRecordName()).newInstance());
> }
```

- 변경 내역이 이메일로 자동으로 각 팀 구성원에게 전달
- 대부분의 자동화된 빌드 시스템은 변경 내역을 전송
- 소스코드 관리 시스템에 코드가 체크인될 때마다 변경 통지

코드 변경 통지 이메일 내역

- 검토자 이름
- 코드를 변경하거나 추가한 목적
- 새 코드와 예전 코드의 차이점 (유명한 소스 코드 관리 시스템이라면 차이점 보고서 생성 해줍니다)
- 너무 많은 양의 코드 블록을 완전히 다시 작성하는 바람에 변경 내역이 무의미해졌다면, 그냥 새 코드만 포함 시킵니다.

모든 사람에게 코드 통지를 보내세요

- 코드 통지는 팀 구성원 사이에 책임감을 조성하는 쉬운 방법 중 하나입니다.
- 코드 통지는 코드 검토를 받지 않거나, ‘목록’ 상의 버그나 기능도 아닌 코드를 추가하는 이단자 개발자를 잡아내는 데 도움이 됩니다.
- 누가 가장 활동적인 코드 작성자이고 검토자인지 아주 금방 눈치챌 수 있습니다.
- 누군가 일주일 동안 코드를 커밋하지 않으면, 기술 리더가 함께 상의해볼 수 있습니다.

APPENDIX

SCM(Source Configuration Management) 시스템

GitHub 기본 활용 방법 및 브랜치 전략



1. GitHub의 특징

1.1 GitHub의 개요

GitHub는 Git 서비스를 기반으로 한 SaaS 서비스로, 단순 소스 관리 뿐 아니라 프로젝트 관리에 필요한 이슈, 태그, 마일스톤, 변경관리, 프로젝트 관리, 릴리즈 관리를 포함하는 형상관리 포털 서비스.

- GitHub의 서비스 유형

- Public : 무료. GitHub에 등록된 소스를 모두 공개하는 조건
- Private : 유료. GitHub에 기업, 조직 별 저장소 운영을 비공개

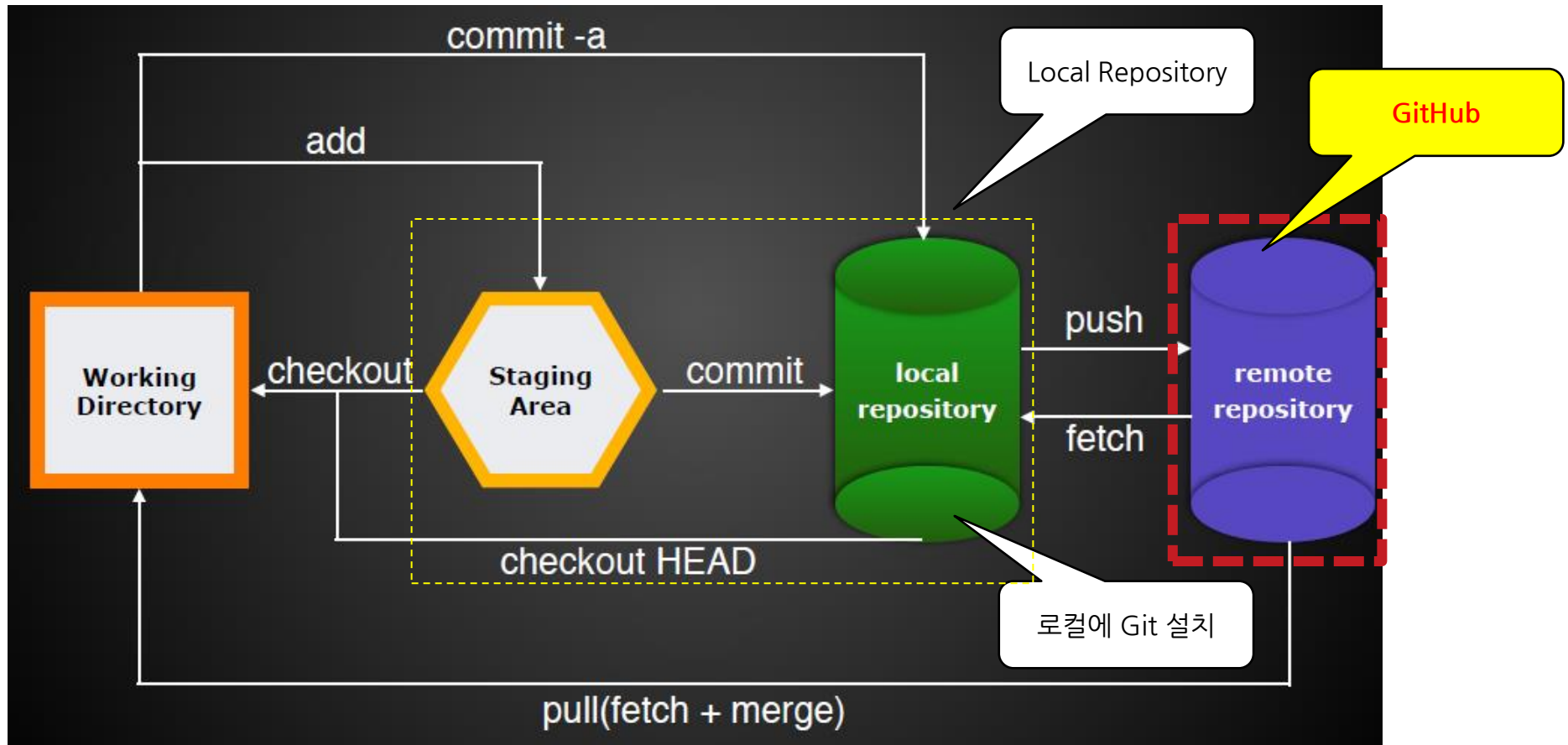
- GitHub의 서비스 플랜 (2019.1월 정책 변경 - private repository 무료화)

Free	Pro
\$0	\$7
Per month	Per month
The basics of GitHub for every developer	Pro tools for developers with advanced requirements
∞ Unlimited public repositories ∞ Unlimited private repositories ✓ 3 collaborators for private repositories ✓ Issues and bug tracking ✓ Project management	∞ Unlimited public repositories ∞ Unlimited private repositories ∞ Unlimited collaborators ✓ Issues and bug tracking ✓ Project management ✓ Advanced tools and insights
	Included free alongside other real-world development tools in the GitHub Student Developer Pack

Teams	
Team	Enterprise
\$9	Contact Sales for pricing
Per user / month	
Advanced collaboration and management tools for teams	Security, compliance, and deployment controls for organizations
∞ Unlimited public repositories ∞ Unlimited private repositories ✓ Team access controls ✓ User management and billing ✓ Issues and bug tracking ✓ Project management ✓ Advanced tools and insights	← Everything included in Team ✓ Self-hosted or cloud-hosted ✓ SAML single sign-on ✓ Access provisioning ✓ Simplified account administration ✓ Unified search and contributions ✓ Priority support ✓ 99.95% uptime SLA for Enterprise Cloud ✓ Invoice billing ✓ Advanced auditing
Starts at \$25 / month and includes your first 5 users	Questions? → Learn more about Enterprise
Free to academic faculty for teaching or non-profit research	Free for educational institutions participating in the GitHub Education program

1.2. GitHub 구성도

GitHub는 Git 서비스의 원격 저장소(Remote Repository)의 역할을 수행.



1.3. GitHub 지원 기능

GitHub는 Git 서비스를 기반으로 한 버전관리 및 프로젝트 관리 포털 SaaS 포털 서비스

The screenshot shows the GitHub interface for the repository 'seeg-1 / seeg1'. Several features are highlighted with callouts and red dashed boxes:

- 소스 버전관리** (Source Version Control): Points to the repository name.
- 코드 리뷰, 검토, 소스 통합 기능** (Code Review, Review, Source Integration Features): Points to the 'Pull requests' tab.
- 프로젝트관리 (단순 칸반, 버그처리 프로세스 등 활용)** (Project Management (Simple Kanban, Bug Handling Process, etc.)): Points to the 'Projects' tab.
- 멤버관리, 브랜지, 서비스 연계 설정 등** (Member Management, Branching, Service Integration Settings, etc.): Points to the 'Settings' tab.
- 이슈추적관리 (라벨, 마일스톤 포함)** (Issue Tracking Management (Labels, Milestones, etc.)): Points to the 'Issues' tab.
- 릴리즈 관리** (Release Management): Points to the '1 release' indicator.
- GitHub에서 소스 파일 관리** (Source File Management on GitHub): Points to the 'Create new file' and 'Upload files' buttons.

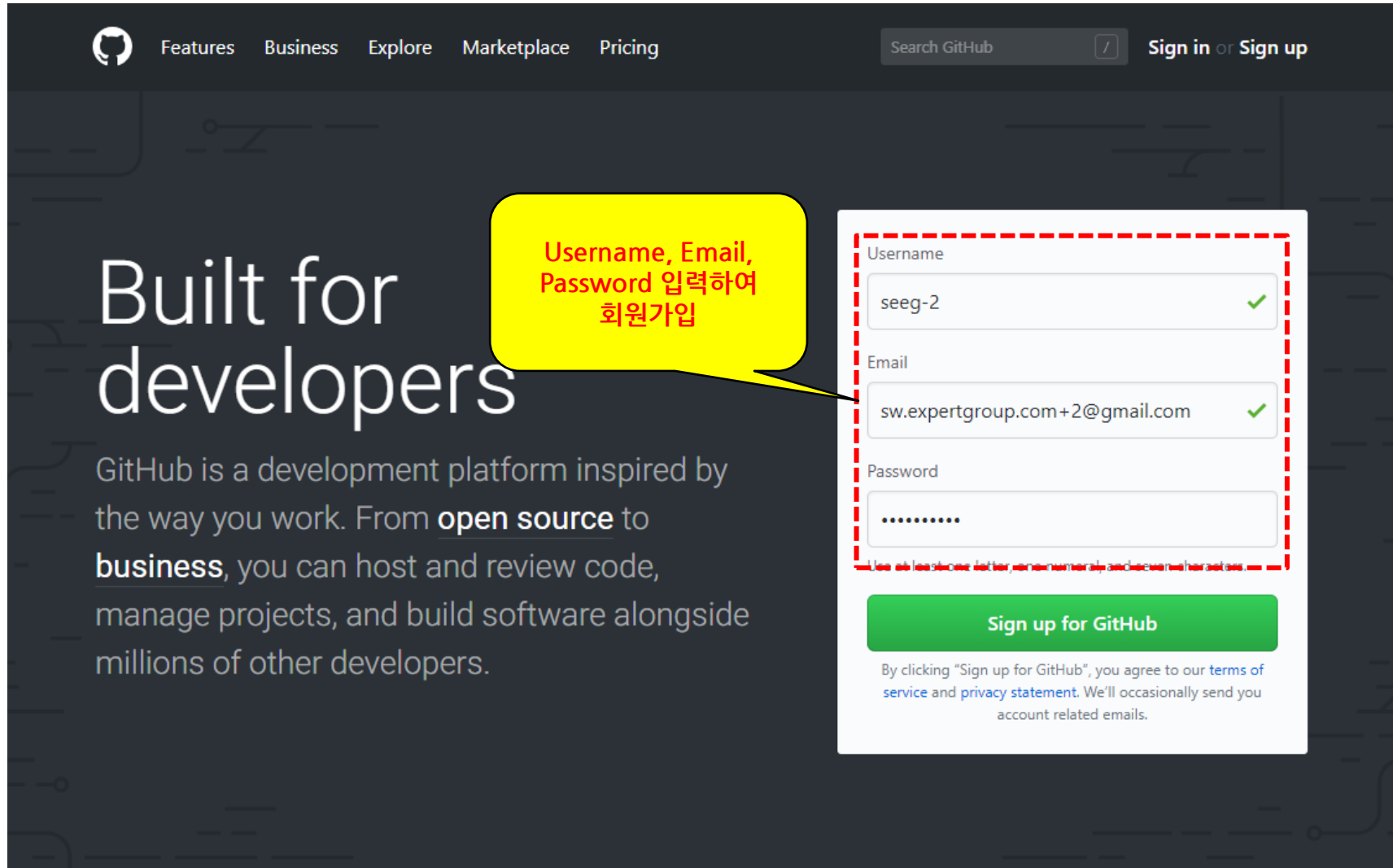
The repository statistics show 10 commits, 3 branches, 1 release, and 1 contributor. The file list includes:

File	Commit Message	Time
.gitignore	modified configuration (gitignore)	23 hours ago
README.md	modified 2 line	23 hours ago
main.c	develop 1st commit	23 hours ago
submod.c	iss1 #1 modified	8 hours ago
submodule.c	Create submodule.c	23 hours ago

2. GitHub의 기본 활용

2.1. GitHub 의 회원가입

GitHub의 On-Premise 버전인 Git Enterprise를 제외하고 SaaS 형태의 GitHub를 이용하기 위해서는 회원가입(Sign Up)이 필요. (<http://github.com/>)



The screenshot shows the GitHub homepage with a sign-up form on the right. A yellow callout box with a pointer indicates the fields for Username, Email, and Password. The form fields are as follows:

Field	Value	Status
Username	seeg-2	✓
Email	sw.expertgroup.com+2@gmail.com	✓
Password		

Below the form is a green button labeled "Sign up for GitHub". At the bottom, there is a disclaimer: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails."

2.2. GitHub 의 회원가입

무료로 무제한 저장소 사용 가능하나 무료의 경우에는 저장소가 모두에게 공개되므로, 일반적인 기업에서 저장소를 비공개로 사용하기 위해서는 유료 서비스 (매월 7\$/인 이상) 플랜 가입 필요

Welcome to GitHub

You've taken your first step into a larger world, @seeg-2.

Completed Set up a personal account

Step 2: Choose your plan

Choose your personal plan

☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month. (view in KRW)

Don't worry, you can cancel or upgrade at any time.

☐ Help me set up an organization next
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations](#)

☐ Send me updates on GitHub news, offers, and events
Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

Plans include:

- ✓ Collaborative code review

무료로 무제한 저장소 생성 가능. (한 저장소당 1GB의 용량 제한)
단, 저장소는 모두 공개되어 일반 기업에서 사용하기에는 부적절

기업에서 비공개 사용을 위해서는 유료 플랜 가입 필요. (매월 7\$/인 이상)

2.3. Git 클라이언트 SourceTree 설치

클라이언트에서 Git 을 사용하기 위해서는 CLI(Command Line Interface) 사용이 일반적이나, 비가시적인 단점으로 Git 클라이언트로 SourceTree 클라이언트 활용.

- SourceTree - <http://www.sourcetreeapp.com/>

The collage illustrates the SourceTree installation and setup process:

- SourceTree Website:** Shows the 'Download for Windows' button and the 'SourceTree' menu with options: 라이선스 계약 (License Agreement), 계정 등록 (Account Registration), and 도구 설치 (Install Tools).
- Account Registration:** Shows the '계정 등록' (Account Registration) screen. A red dashed box highlights the 'Atlassian 계정' (Atlassian account) option, with a yellow callout stating: **설치 시, Atlassian 계정 입력 필요** (Atlassian account input required during installation).
- Pick tools to download and install:** Shows the 'Pick tools to download and install' screen. A red dashed box highlights the 'Git' option (36.27 MB), with a yellow callout stating: **Git만 선택** (Select only Git).

2.4. 원격 저장소 (Remote Repository) 생성

“Start a project” 버튼 혹은 “New repository” 메뉴를 이용하여 원격 저장소를 생성한다.

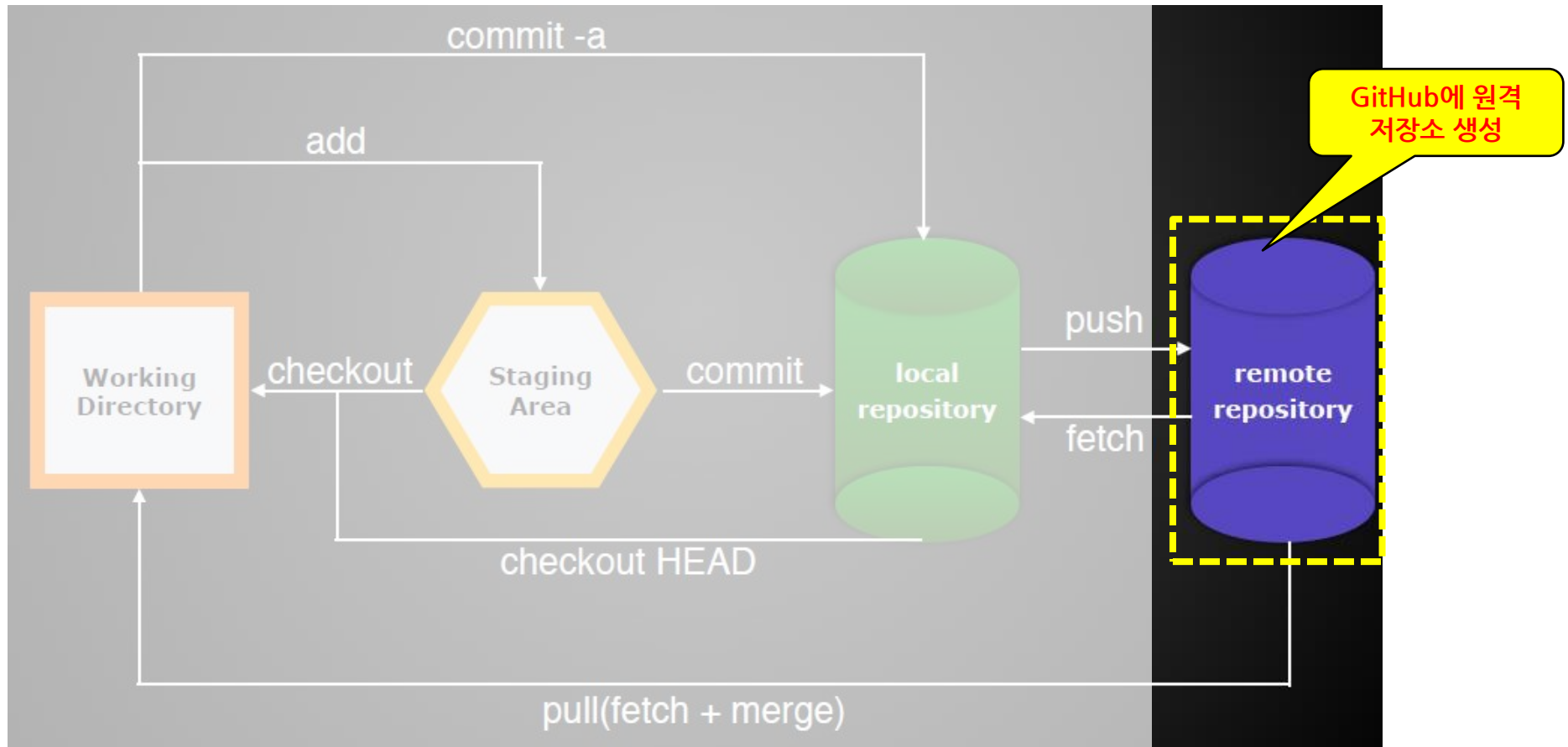
The screenshot illustrates the process of creating a new repository on GitHub. It features three overlapping window views:

- Top-left window:** Displays the GitHub landing page with the "Start a project" button highlighted by a red dashed box. A red arrow points from this button to the "New repository" menu in the bottom-left window.
- Bottom-left window:** Shows the "New repository" dropdown menu, with the "New repository" option highlighted by a red dashed box.
- Main window:** Shows the "Create a new repository" form. The "Repository name" field (containing "seeg1") and the "Public" visibility option (selected) are both highlighted with red dashed boxes. A red arrow points from the "New repository" menu to this form.

Additional details in the main window include:

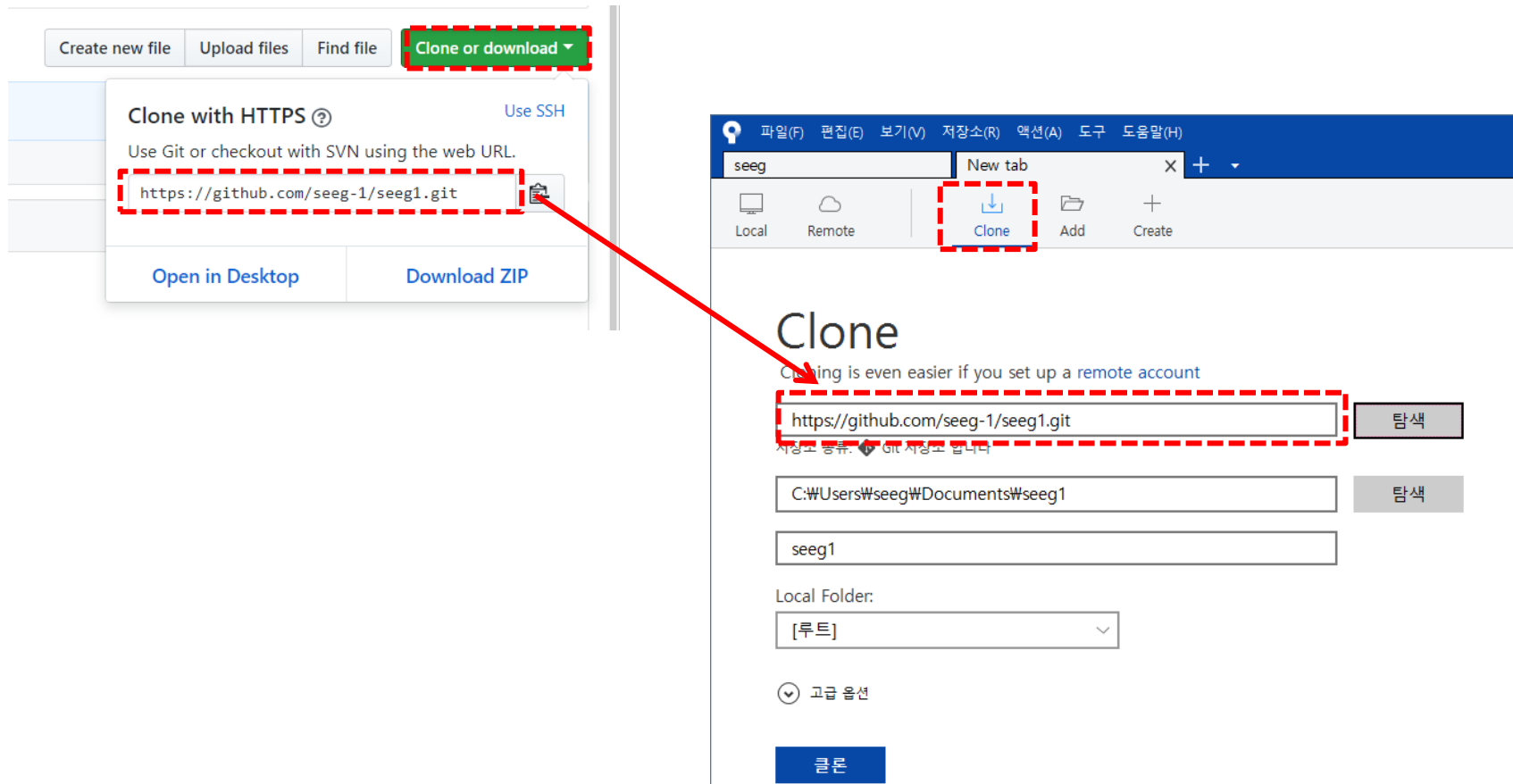
- Owner:** seeg-1
- Description (optional):** SEEG first Project
- Visibility:** Public (selected). A callout bubble next to it states: "Public 무료, Private 은 유료" (Public is free, Private is paid).
- Options:** "Initialize this repository with a README" is checked. There are dropdowns for ".gitignore" (set to None) and "Add a license" (set to None).
- Button:** A green "Create repository" button at the bottom.

2.5. 원격 저장소 (Remote Repository) 생성

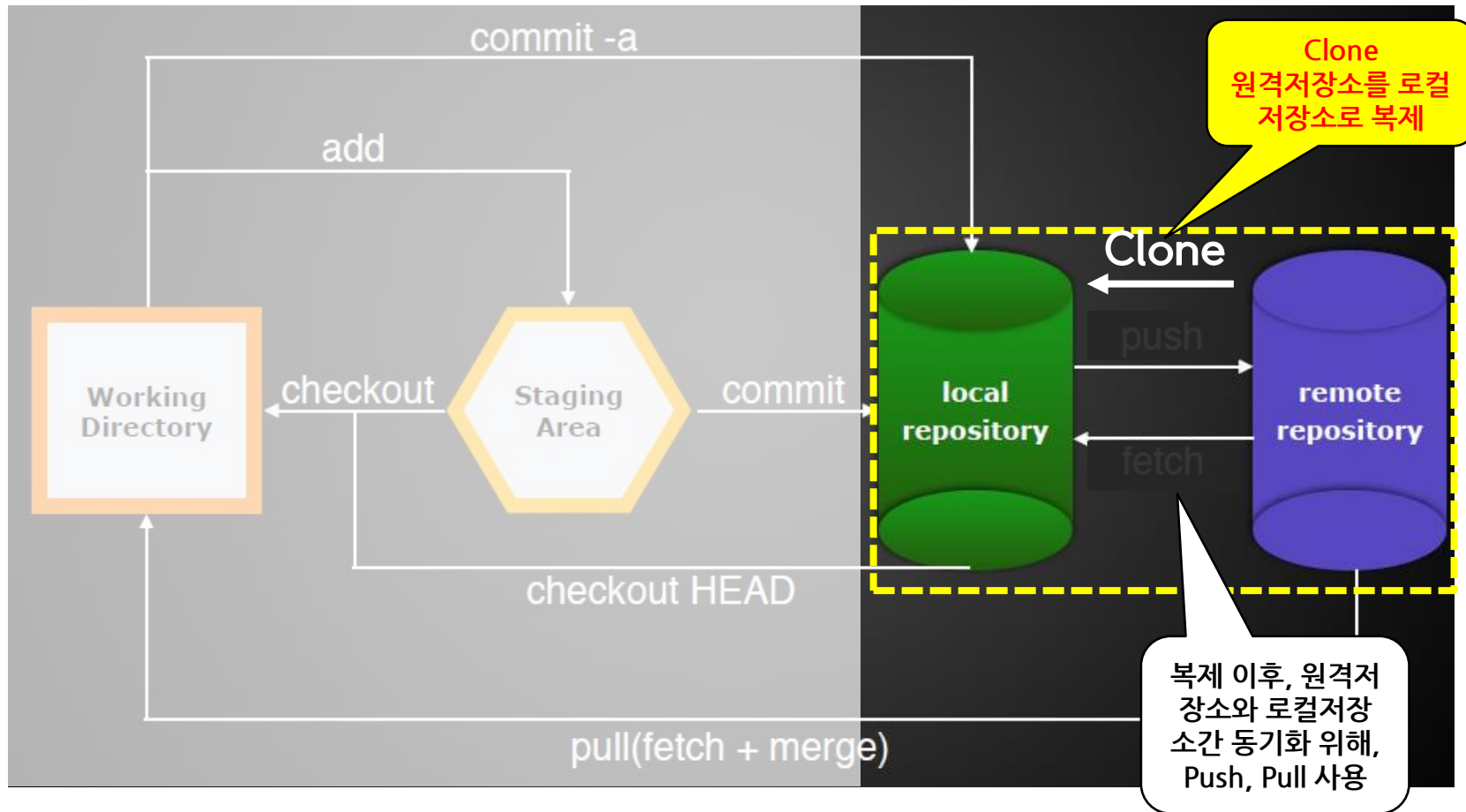


2.6. 저장소 복제 (Clone : Remote -> Local)

원격 저장소를 로컬 저장소로 복제. 이후 로컬에서의 작업은 로컬 저장소에만 작업하게 됨.

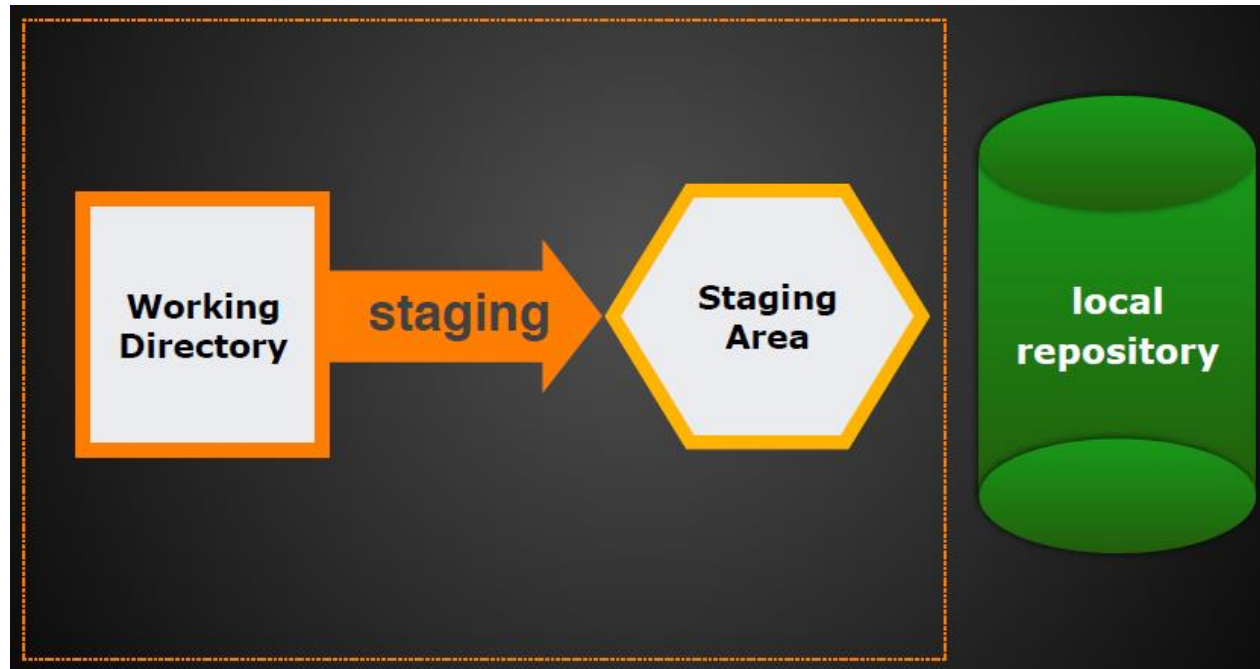


2.7. 저장소 복제 (Remote -> Local)



2.8. 파일 수정 후 Stage 올리기

작업 변경 파일을 커밋하기 전, 변경 파일 단위, 변경 부분 중 일부분만 골라서 커밋 대기 상태 가능



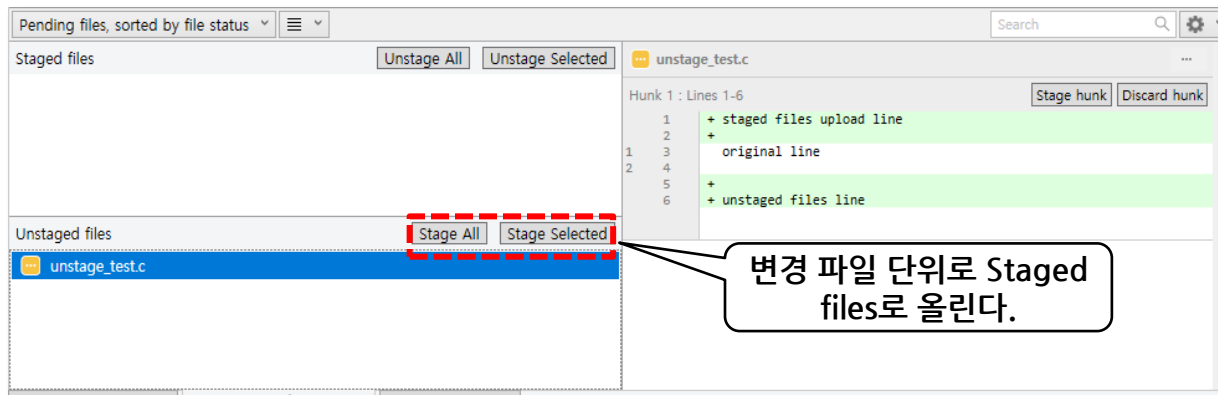
=



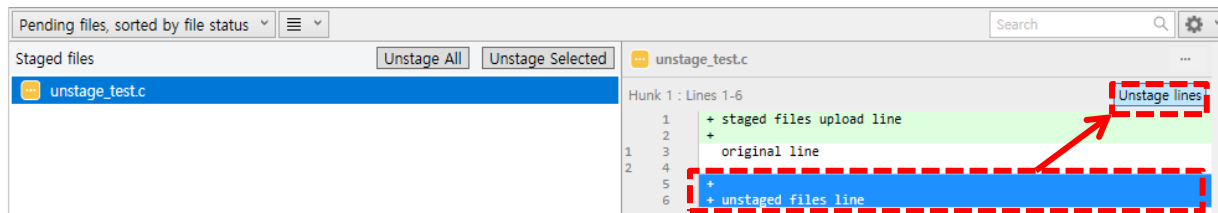
2.9. 파일 수정 후 Stage 올리기 (파일 일부분만 Staging Area에 반영)

파일 내 변경된 부분 중 일부분만 골라서 Staging Area에 추가하는 것도 쉽게 작업 가능

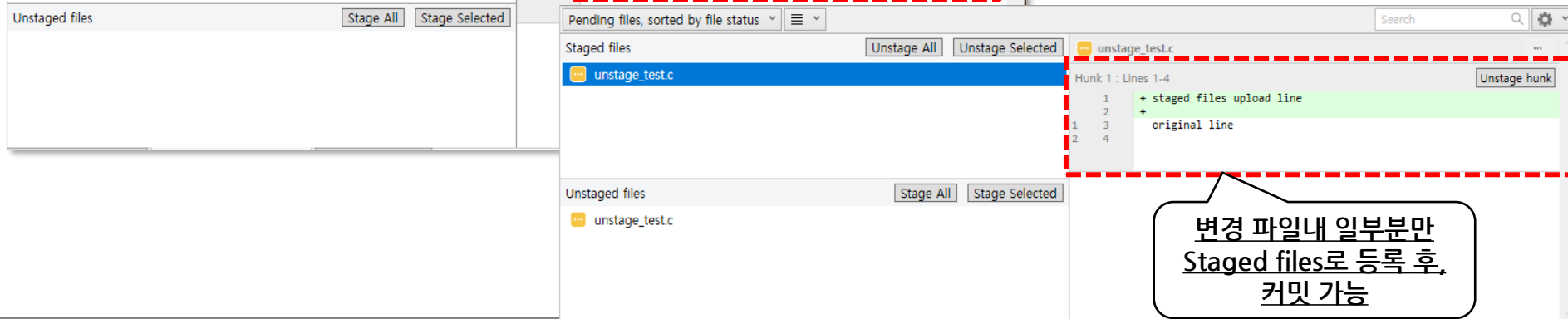
1) 먼저 변경 파일을 Staged files 로 올린다.



2) Staged files 의 우측 소스 화면에서, +, - 표시된 라인(변경된 라인)을 선택하고 'Unstage lines' 선택 -> Unstaged files 에 Staged files 에서 unstage lines 한 라인을 제외한 남은 파일이 표시된다.

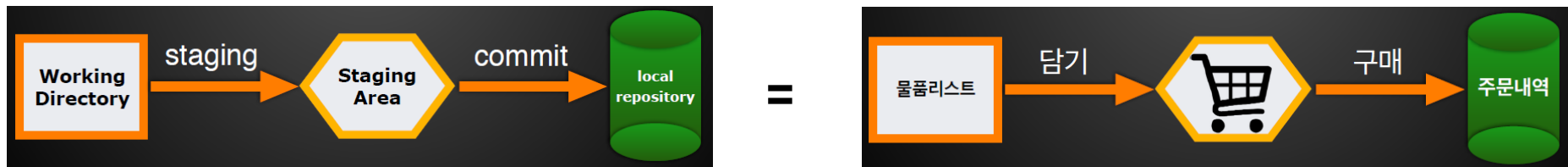
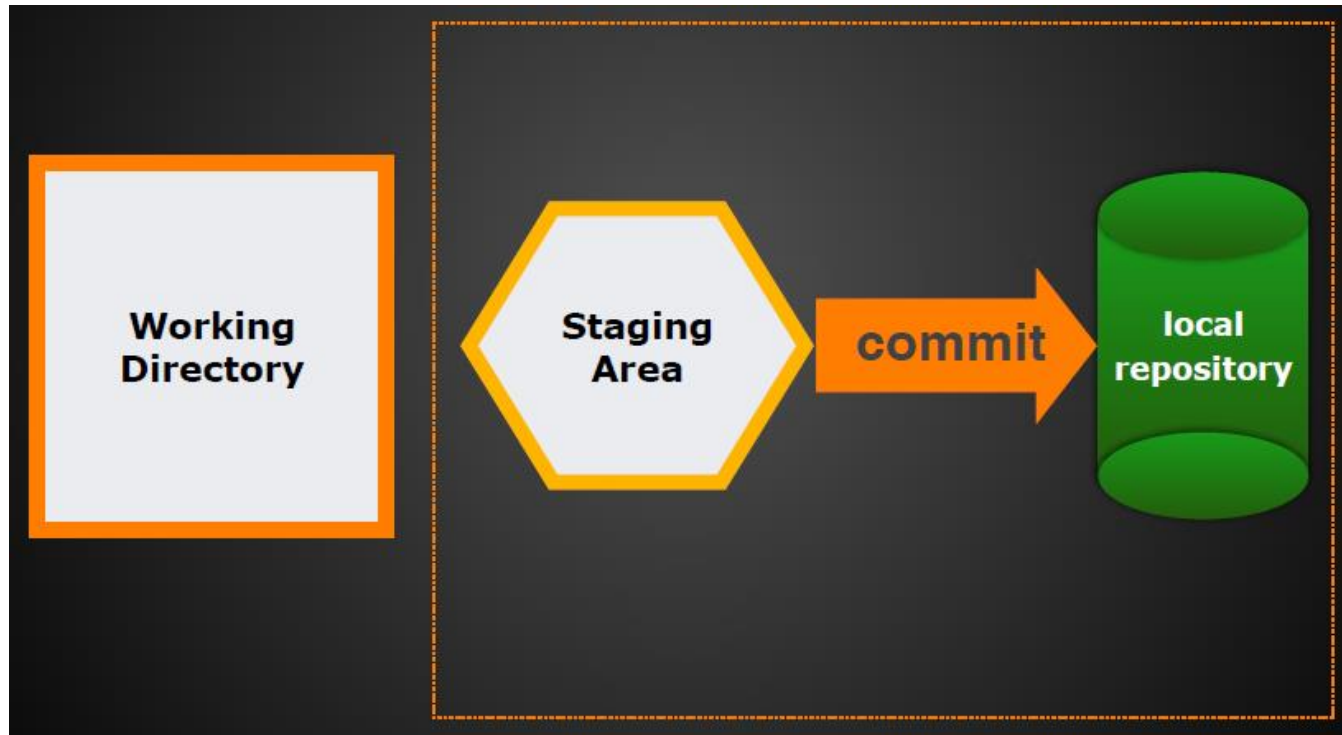


3) 파일 내 변경 부분 일부분만 Staged files 상태로 두고, commit 가능하다.



2.10. Commit

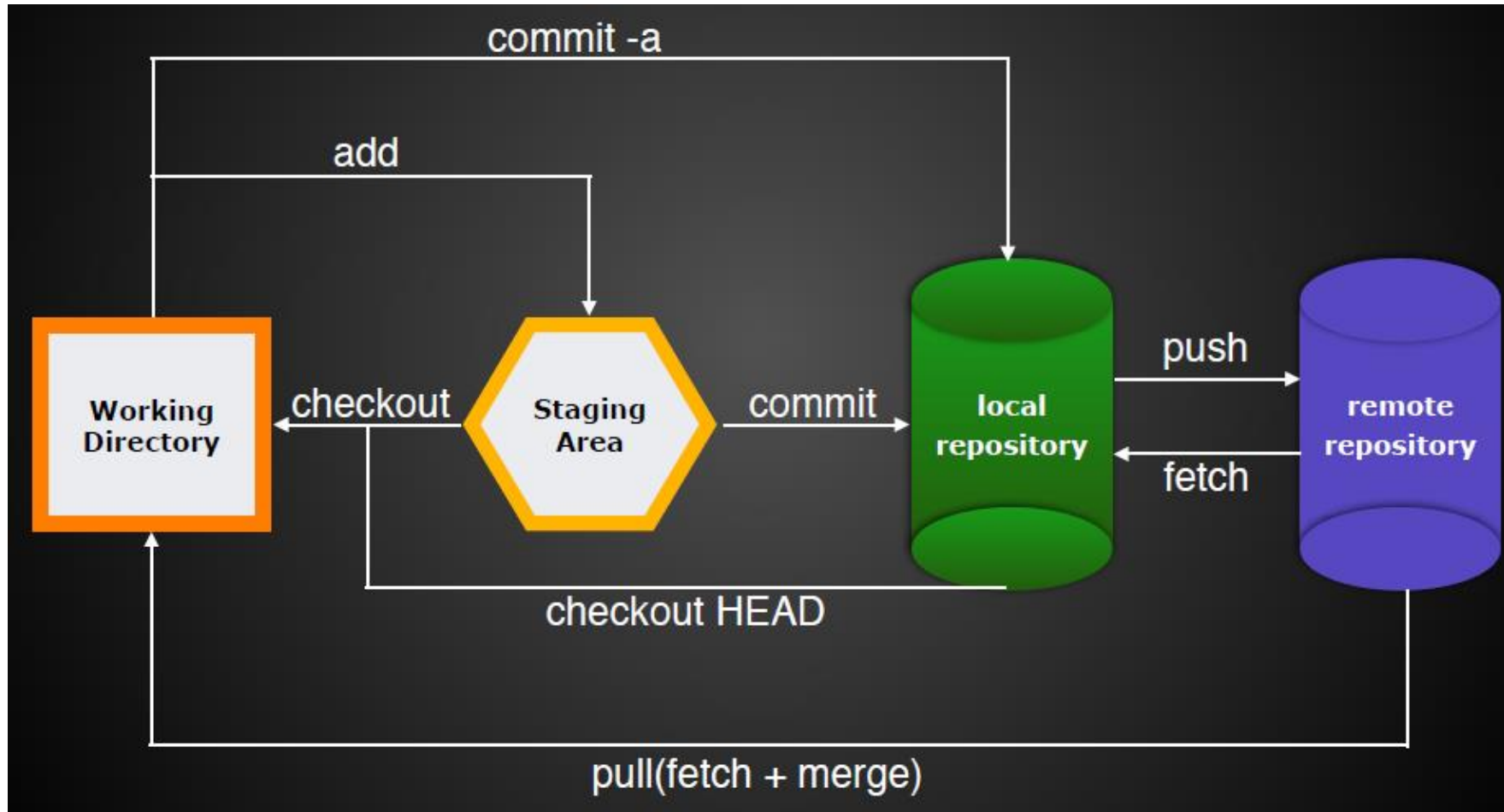
Stage 상태의 파일을 로컬 저장소에 커밋한다. 아직 원격 저장소에는 반영되지 않은 상태임.



2.11. Push, Pull

Push : 로컬 저장소의 내용을 원격 저장소에 반영하는 것

Pull : 원격 저장소의 변경 내용을 로컬 저장소에 동기화 (Fetch + Merge)



Task / Requirements Management 도구



1. Jira와 Confluence를 활용한 요구사항 관리

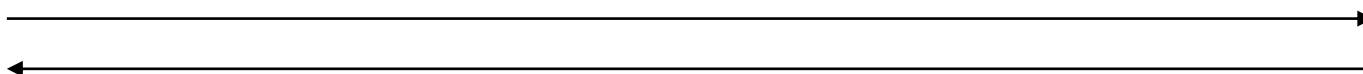
• Jira와 Confluence의 특징

	Jira	Confluence
주요 목적	이슈 추적, 계획, 릴리즈 관리	문서 생성, 논의, 토론
특징	• Ticket을 이용한 Task, Issue 관리 • Ticket에 하나의 Task, Issue등에 대한 상세 내용 작성 • 정의된 업무플로우를 활용하여 업무의 진행파악가능 • 큰틀(Epic)을 이용하여 이슈를 계층구조로 관리 가능	• Wiki 스타일의 중앙 저장소 • 피드백, 파일 버전관리, 변경 이력관리 제공 • Space 개념을 이용하여 팀/프로젝트 별 숙지해야할 사항 공유 • 필터를 이용하여 Jira 현황을 정렬하여 문서로 생성 가능(Jira Query Language) • Confluence의 내용을 Jira의 Ticket으로 즉시 생성 가능
한계	Ticket만 활용하여 거시적인 현황 파악은 어려움	지식생성(문서작성)에 특화 → 강제화 기능 없음
활용	유지보수, 변경요청 등 이벤트성 기능(요구사항) 변경	추상적인 요구사항으로부터 상세한 기능으로 구체화 활동이 필요한 경우

관리에 중점!



요구사항 조각(Ticket)을 기존 요구사항(Confluence page)에 Link 혹은 요구사항 상세 설명



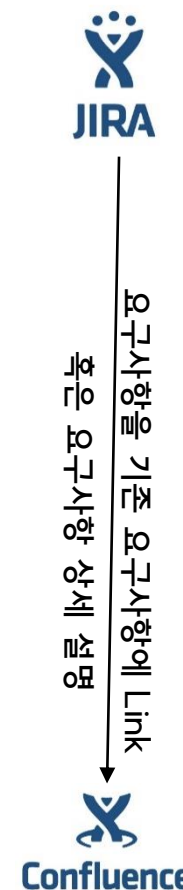
개념적인 요구사항을 개발가능한 수준으로 Breakdown

콘텐츠에 중점!



2. Jira 중심의 요구사항 관리하기

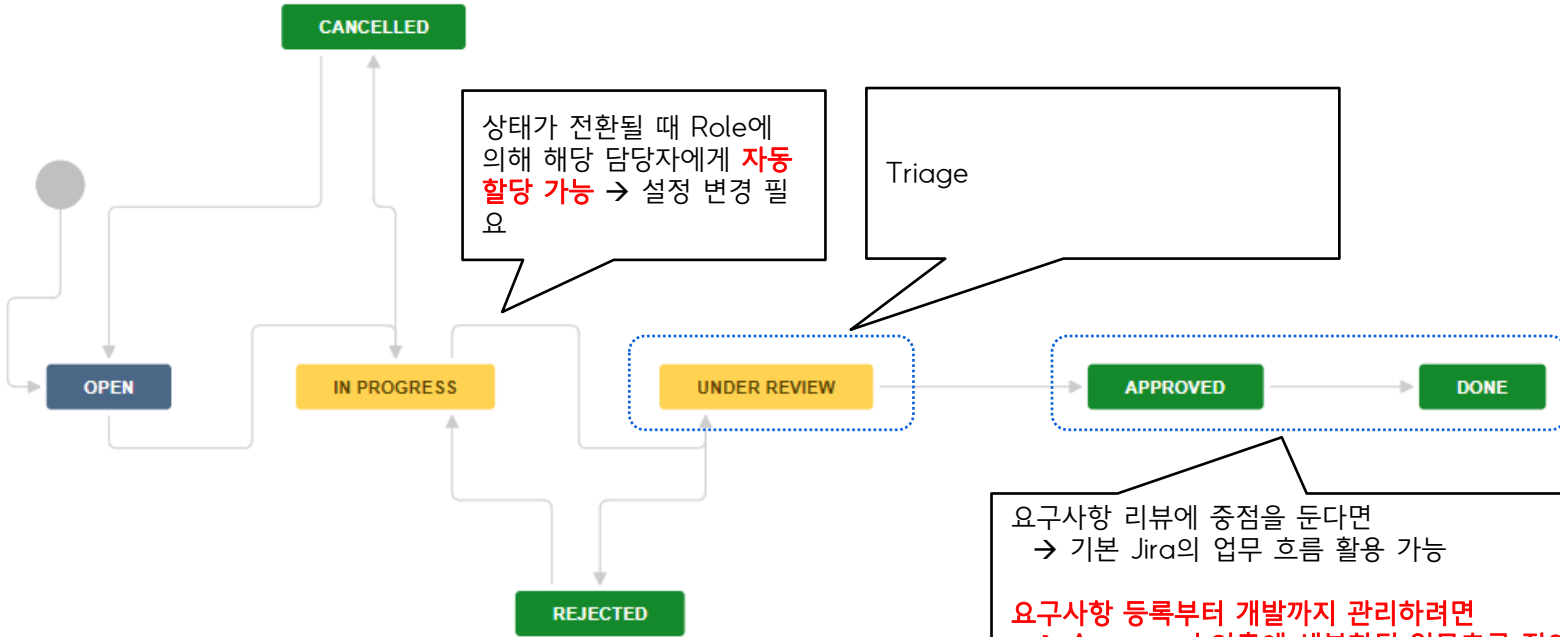
- Jira 중심의 요구사항관리에 적합한 사업 및 조직의 특성
 - SM 조직 (유지보수 중심의 프로젝트 진행)
 - Bug fix가 주 업무일때!
 - 새롭게 개발되는 기능이 거의 없는 프로젝트 (이미 솔루션을 가지고 있는 상태)
 - 비즈니스요구사항 혹은 사업목표가 문서화되어 있는 조직
- 필요한 요구사항 관리 활동
 - Ticket이 어디서 (사람? 회의록? 사이트? 기타 등등)왔는가?에 대한 관리
 - Review (리뷰의 포인트 : 개발할 수 있는가? Jira ticket의 완료를 확인할 수 있나?)
 - Inspection 까지는 아니지만 PM/PL급의 검토 필요 (영향도 평가)
 - 요구사항을 상세하게 작성해야 하는지 여부 검토
 - Ticket 템플릿
 - 기존 요구사항과의 연결성 확보 (Epic 혹은 비즈니스 요구사항 어느 것에 파생된 것인지 확인 필요)
 - 요구사항이 등록된 원인, 이유 등
 - 큰 단위의 요구사항과의 추적성
 - Jira Epic과 연결 혹은 Jira 요구사항에 대한 부연 설명(Confluence) 연결



2. Jira 중심의 요구사항 관리하기

- ‘Review’ 단계가 포함된 업무 흐름 활용 필요

[Jira에서 기본으로 제공하는 Review가 포함된 업무 흐름]



[고려해야할 사항]

1. Review가 포함된 업무 흐름 정의
2. 요구사항을 리뷰하기 위한 Role 생성
3. 리뷰의 주기 (등록될때마다? 일주일에 한 번?)
4. 개발할꺼야, 말꺼야? 중복이야 아니야? → Triage 할 시점

2. Jira 중심의 요구사항 관리하기

- Jira 요구사항 템플릿

- 요구사항을 서술하기 위한 템플릿 제시보다 요구사항 관리에 초점을 맞추어 템플릿 구성 필요
- 필수/선택 필드 선정, 사용자 정의 필드 등 정의 필요

요약 

보고자   sunshine0731@gmail.com

키보드 입력을 시작하면 일치하는 필터의 목록을 보여줍니다.

구성 요소 없음

설명 

스타일 ▾ B I U       + ▾ ≡

미리보기 텍스트 ↺ ↻

수정 버전 ▾

키보드 입력을 시작해서 사용 가능한 목록을 가져오거나 아래를 눌러 선택하세요.

우선순위 ↑ Medium ▾ ?

레이블 ▾

새 레이블을 만들거나 찾으려면 키보드 입력을 시작하세요. 추천 레이블을 보시려면 목록을 여세요.

연결된 이슈 blocks ▾

이슈 ▾ +

연결할 이슈를 찾기 위해 키보드 입력을 시작하십시오. 비워두면 연결을 만들지 않습니다.

담당자  자동 ▾

나에게 할당

큰틀 연결  ▾

+

Confluence 연결 필드

Review 요청 기한

2. Jira 중심의 요구사항 관리하기

- 요구사항 그룹핑

- 비즈니스요구사항 혹은 사업목표가 문서화되어 있지 않은 경우, Ticket으로 등록되는 요구사항을 그룹핑 하는 작업 필요
- Epic (큰 틀)을 만들거나, Jira Query Language를 활용하여 Confluence page를 생성 가능

- Jira Query Language

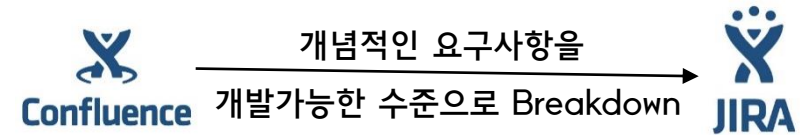
- Jira의 Database에 쿼리하여 원하는 정보 불러 올 수 있도록 지원하는 언어
- 필터를 통해 원하는 정보를 Table 형태로 디스플레이
- 예제) Scrum 프로젝트 중 제목에 “일괄등록된”이 포함된 이슈
 - Query : project =‘scrum’ and summary ~ “일괄등록된”

키	요약	유형	생성일	변경일	기한	담당자	Reporter	우선	상태	해결책
SCRUM-26	일괄등록된 Task-예자일 111	□	5월 02, 2018	5월 02, 2018		할당되지 않음	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-32	일괄등록된 Task-예자일 117	□	5월 02, 2018	5월 02, 2018		할당되지 않음	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-33	일괄등록된 Task-예자일 118	□	5월 02, 2018	5월 02, 2018		할당되지 않음	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-38	일괄등록된 Task-예자일 123	□	5월 02, 2018	5월 02, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-42	일괄등록된 Task-예자일 127	□	5월 02, 2018	5월 02, 2018		userpark	sunshine0731@gmail.com	↑	진행 중	미해결
SCRUM-34	일괄등록된 Task-예자일 119	□	5월 02, 2018	5월 02, 2018		sunshine0731@gmail.com	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-40	일괄등록된 Task-예자일 125	□	5월 02, 2018	5월 02, 2018		userpark	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-39	일괄등록된 Task-예자일 124	□	5월 02, 2018	5월 02, 2018		userpark	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-41	일괄등록된 Task-예자일 126	□	5월 02, 2018	5월 02, 2018		sunshine0731@gmail.com	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-29	일괄등록된 Task-예자일 114	□	5월 02, 2018	5월 02, 2018		userpark	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-35	일괄등록된 Task-예자일 120	□	5월 02, 2018	5월 02, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-37	일괄등록된 Task-예자일 122	□	5월 02, 2018	5월 02, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-27	일괄등록된 Task-예자일 112	□	5월 02, 2018	5월 02, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-28	일괄등록된 Task-예자일 113	□	5월 02, 2018	5월 02, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-31	일괄등록된 Task-예자일 116	□	5월 02, 2018	5월 02, 2018		sunshine0731@gmail.com	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-30	일괄등록된 Task-예자일 115	□	5월 02, 2018	5월 02, 2018		sunshine0731@gmail.com	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-36	일괄등록된 Task-예자일 121	□	5월 02, 2018	5월 02, 2018		userpark	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-43	일괄등록된 Task-예자일 300	□	5월 03, 2018	5월 03, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-45	일괄등록된 Task-예자일 303	□	5월 03, 2018	5월 03, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결
SCRUM-47	일괄등록된 Task-예자일 306	□	5월 03, 2018	5월 03, 2018		userkim	sunshine0731@gmail.com	↑	할 일	미해결

3. Confluence 중심의 요구사항 관리하기

- Confluence 중심의 요구사항관리에 적합한 사업 및 조직의 특성

- R&D 조직, 새롭게 개발되는 기능이 대다수인 프로젝트
- 정확한 사업범위와 목표가 있는 사업
- 구체적인 요구사항이 정의되지 않은 사업



- 필요한 요구사항 관리 활동

- 리뷰의 포인트

- JIRA의 하나의 Ticket으로 옮길 때, 단위 기간 동안에 개발 가능한 요구사항인가?
- 규모를 추정하는 근거(왜 스토리 포인트를 10으로 산정을 했는지?)에 대한 이유를 작성해둔다.
(어차피 처음에는 추정이 틀리니까 계속 조정해나가야지!) 이후 회고에서 스토리포인트의 추정과 실측값의 갭차
이 분석 및 추정 시 활용

