스쿼드(Squad) 조직 구조의 포괄적 분석: 정의, 운영 역학 및 실무적 함의에 관한 심층 연구 보고서

1. 서론: 애자일 스케일링과 조직 구조의 진화

현대 소프트웨어 엔지니어링과 프로덕트 개발 환경은 지난 10년간 급격한 패러다임의 전환을 겪어왔다. 전통적인 '명령과 통제(Command and Control)' 기반의 계층적 위계 조직과 폭포수(Waterfall) 모델은 시장의 불확실성과 기술의 복잡성을 감당하기에 한계를 드러냈다. 이에 대한 대안으로 등장한 것이 '스쿼드(Squad)' 모델이다. 2012년 헨릭 크니버그(Henrik Kniberg)와 앤더스 이바르손(Anders Ivarsson)이 Spotify의 엔지니어링 문화를 소개하며 널리 알려진 이 모델은, 단순한 팀 구성 방식을 넘어 조직의 의사결정 구조, 리더십의 정의, 그리고 성과 관리 체계 전반을 재정의하는 개념으로 자리 잡았다.[1]

본 보고서는 스쿼드 조직 구조에 대한 정의와 이론적 배경을 시작으로, 국내외 주요 테크 기업들의 도입 사례를 심층 분석한다. 특히 한국의 독특한 '서비스 기획자' 직군과 스쿼드 모델의 충돌, '사일로(Silo)', '셀(Cell)', '캠프(Camp)' 등 국내 기업들의 변용 사례를 상세히 다룬다. 나아가 인적 자원 관리(HR), 리소스 배분, 기술 부채 관리 등 실무 적용 시 발생하는 주요 '엣지 케이스(Edge Case)'를 규명하고, 거버넌스, 성과 측정, 조직 문화, 실패 유형 등 MECE(Mutually Exclusive Collectively Exhaustive) 관점에서 누락되기 쉬운 요소들을 포괄적으로 분석하여 조직 설계자와 리더십을 위한 실질적인 가이드를 제공하고자 한다.

2. 스쿼드 모델의 이론적 프레임워크와 구조적 해부

스쿼드 모델의 핵심 철학은 '정렬된 자율성(Aligned Autonomy)'이다. 이는 리더십이 '무엇(What)'과 '왜(Why)'에 대한 전략적 방향성을 제시하면, 실행 조직인 스쿼드가 '어떻게(How)'를 결정하는 구조를 의미한다.[1]

2.1 스쿼드(Squad): 가치 전달의 최소 단위

스쿼드는 비즈니스 가치를 창출하고 배포할 수 있는, 스타트업과 유사한 형태의 최소 조직 단위로 정의된다.

- 교차 기능성(Cross-functionality): 스쿼드는 기획, 디자인, 개발, 테스트, 배포 등 제품 개발의 전 주기를 소화할 수 있는 모든 역량을 내재화한다. 이는 기능 조직 간의 핸드오프(Hand-off)로 인한 병목 현상을 제거하고 리드 타임을 단축하는 데 핵심적인 역할을 한다.[1]
- 자율성과 자기 조직화: 스쿼드는 내부 프로세스(스크럼, 칸반 등), 도구 선택, 배포 주기에 대한 높은 결정권을 가진다. 공식적인 '스쿼드 리더'는 존재하지 않으며, 프로덕트 오너(PO)가 우선순위를 결정하고, 애자일 코치가 팀의 프로세스 개선을 지원하는 형태를 띤다.[1]

- 미션 중심: 각 스쿼드는 "검색 경험 최적화" 또는 "결제 전환율 증대"와 같은 장기적인 미션을 부여받으며, 이에 대한 끝단(End-to-End)의 책임을 진다.[6]

2.2 확장성을 위한 연결 구조: 트라이브, 챕터, 길드

수십, 수백 개의 자율적인 스쿼드가 조직 전체의 방향성과 일치하도록 돕기 위해, Spotify 모델은 매트릭스 형태의 연결 구조를 제안했다.

구조적 요소	정의 및 역할	규모 및 특징
트라이브 **(Tribe)**	관련된 비즈니스 영역(예: 뮤직 플레이어, 백엔드 인프라)에서 일하는 스쿼드들의 집합.	던바의 수(Dunbar's Number)에 근거하여 100명 미만으로 유지. 스쿼드 간의 의존성을 조율하고 물리적/가상적 협업 공간을 공유함.[3]
챕터 **(Chapter)**	동일한 트라이브 내에서 같은 직무(예: 백엔드 개발자, UX 디자이너)를 가진 구성원들의 모임.	기능적 전문성을 유지하고, 코드 품질 표준을 설정하며, 라인 매니지먼트(인사, 연봉, 코칭)를 담당함.[3]
길드 **(Guild)**	조직 전체를 가로지르는 관심사 기반의 커뮤니티(예: 자바 길드, 애자일 코치 길드).	지식 공유, 도구 표준화, 베스트 프랙티스 확산을 위한 자발적 모임. 조직의 경계를 넘어선 횡적 연결을 강화함.[2]
트리오/얼라이언스 **(Trio/Alliance)**	트라이브를 이끄는 프로덕트, 디자인, 기술 리더십 그룹 또는 트라이브 간의 연합체.	상위 수준의 전략적 정렬과 트라이브 간의 의존성 문제를 해결하는 거버넌스 기구.[9]

2.3 'Spotify 모델'의 허와 실: 카고 컬트(Cargo Cult)의 위험

연구 자료에 따르면, 업계에서 통용되는 'Spotify 모델'은 Spotify가 2012년 시점에 사용했던 방식의 스냅샷일 뿐, 현재의 Spotify조차 그대로 사용하지 않는다는 점을 명확히 해야 한다.[11] 많은 조직이 Spotify의 문화를 이해하지 못한 채 용어(스쿼드, 챕터 등)만 차용하여 기존의 위계 조직에 덧씌우는 '카고 컬트 애자일(Cargo Cult Agile)'의 오류를 범하고 있다. 이는 구조적 유연성을 확보하지 못한 채 혼란만 가중시키는 주요 원인이 된다.[11]

3. 글로벌 및 국내(한국) 도입 사례와 변용

스쿼드 모델은 각 기업의 비즈니스 환경과 문화적 특성에 따라 다양한 형태로 변용되어 적용되고 있다. 특히 한국의 테크 기업들은 고유의 '기획자' 문화와 빠른 실행력을 결합하여 독자적인 모델을 발전시켜 왔다.

3.1 해외 사례: ING 은행의 전사적 애자일 전환

금융권에서 스쿼드 모델을 가장 성공적으로 도입한 사례로 꼽히는 ING 은행은 IT와 비즈니스 조직의 장벽을 허물기 위해 이 모델을 채택했다.[14]

- 고객 여정(Customer Journey) 중심 재편: 기존의 기능 부서(여신, 수신 등)를 해체하고, '모기지 신청', '투자 관리' 등 고객의 여정에 맞춘 스쿼드를 구성했다.
- 오베야(Obeya) 룸 활용: 물리적인 시각화 공간을 통해 스쿼드 간의 의존성을 관리하고, 트라이브 리더와 스쿼드 구성원 간의 정기적인 소통을 구조화했다.
- 시사점: 규제가 강력한 금융 산업에서도 자율성을 강조한 조직 구조가 가능함을 증명했으나, 이를 위해 기존 중간 관리자 계층의 대대적인 역할 재정의가 필요했다.[14]

3.2 국내 사례: '네카라쿠배토'의 조직 구조 진화

한국의 주요 IT 기업들은 스쿼드 모델을 기반으로 하되, '속도'와 '책임'을 강화하는 방향으로 독자적인 진화를 거듭했다.

3.2.1 토스(Toss, 비바리퍼블리카): '사일로(Silo)'와 PO의 절대적 권한

토스는 스쿼드와 유사한 '사일로(Silo)'라는 명칭을 사용하며, 구성원 개개인의 높은 자율성과 책임을 극단적으로 강조한다.

- 미니 CEO로서의 PO: 토스의 PO는 단순한 우선순위 결정자를 넘어, 해당 사일로의 P&L(손익)까지 책임지는 '미니 CEO'의 역할을 수행한다. 이는 스쿼드 모델보다 훨씬 강력한 권한 위임을 의미한다.[15]
- 정보의 투명성과 승인 절차 제거: '권한 사일로 테스트(Authority Silos Test)'와 같이 의사결정 권한이 실무자에게 있음을 강조하며, 법무/보안 등 필수적인 절차를 제외한 모든 승인 과정을 최소화하여 속도를 극대화한다.[16]

3.2.2 우아한형제들(배달의민족): '캠프'와 '목적 조직'의 조화

우아한형제들은 기능 조직과 목적 조직(스쿼드)을 유연하게 배합하며 성장해왔다.

- 목적 조직으로서의 스쿼드: '배민1', 'B마트', '로봇 배달' 등 특정 비즈니스 목표를 가진 조직은 기획, 개발, 디자인이 통합된 스쿼드 형태(또는 더 큰 단위의 '캠프')로 운영된다. 예를 들어, 로봇 배달 서비스는 로봇 엔지니어와 서비스 기획자, 소프트웨어 개발자가 하나의 팀으로 움직이며 물리적 세계와 디지털 세계를 연결한다.[17]
- Tech HR의 존재: 스쿼드 구조에서 발생할 수 있는 개발자의 커리어 정체성 혼란을 막기 위해, 별도의 'Tech HR' 조직을 두어 개발 문화를 관리하고 성장을 지원한다. 이는 챕터의 역할을 별도 전문 조직으로 강화한 사례다.[19]

3.2.3 카카오(Kakao): '셀(Cell)'과 'TF'를 통한 기민성 확보

카카오는 정규 조직인 '셀'과 임시 조직인 'TF(Task Force)'를 혼합하여 운영한다.

- 유동적인 **TF** 운영**:** 카톡 개편이나 AI 신규 서비스 등 전략적 과제가 생기면, 기존 셀에서 인원을 차출하여 목적 중심의 TF를 신속하게 구성한다. 과제 종료 후에는 원래 조직으로 복귀하거나 새로운 셀로 분화한다.[20]
- **CIC(Company-In-Company)** 체제**:** 특정 셀이나 사업부의 규모가 커지면 CIC로 독립시켜 인사, 재무, 브랜딩의 독립권을 부여한다(예: 카카오게임즈, 카카오모빌리티 등). 이는 스쿼드의 자율성을 기업 단위로 확장한 개념이다.[22]

3.2.4 네이버(Naver): '팀'에서 '셀', 그리고 'CIC'로의 진화

네이버는 국내 기업 중 가장 먼저 직급 파괴와 셀 조직 도입을 시도한 기업 중 하나다.

- 책임 근무제와 셀 리더**:** 전통적인 '팀장'이 아닌 과업 중심의 '셀 리더' 체제를 도입하여, 직급이 아닌 역할 중심의 문화를 정착시켰다.[23]
- 내부 벤처 시스템**(CIC):** 네이버웹툰과 같이 셀 조직에서 시작해 CIC를 거쳐 별도 법인으로 분사하는 성공 방정식을 확립했다. 이는 스쿼드 모델이 단순한 개발 방법론이 아니라, 비즈니스 인큐베이팅 플랫폼으로 기능할 수 있음을 보여준다.[23]

4. 실무 적용 시 Edge Case 분석과 해결 방안

스쿼드 모델 도입 시 이론적 설명에는 나타나지 않는 다양한 실무적 충돌과 난관이 발생한다. 이를 HR, 리소스, 유지보수 측면에서 심층 분석한다.

4.1 HR 및 매트릭스 조직의 딜레마 (Two Bosses Problem)

스쿼드 모델의 가장 큰 구조적 긴장은 구성원이 '업무적 리더(PO)'와 '인사적 리더(챕터 리드)'라는 두 명의 상사를 모시게 되는 데서 발생한다.

- 평가와 보상의 불일치**:** PO는 기능 출시와 비즈니스 성과를 중시하는 반면, 챕터 리드는 코드 품질, 아키텍처 준수, 기술적 성장을 중시한다. 개발자가 기술 부채 해결을 위해 리팩토링에 집중할 경우, PO는 이를 '생산성 저하'로 평가할 수 있는 반면 챕터 리드는 '우수 성과'로 평가할 수 있다.[25]
- 해결 방안**:** 성공적인 조직은 챕터 리드가 평가 권한을 갖되, PO와 스쿼드 동료들로부터 피드백을 수집하는 '360도 다면 평가'를 제도화한다. 또한, 챕터 리드 자신이 스쿼드의 일원으로 실무에 참여함으로써('Playing Coach'), 현장 감각을 잃지 않도록 하는 것이 중요하다.[27]

4.2 리소스 배분의 비효율성과 '플로팅(Floating)' 인력

모든 스쿼드에 모든 직군을 배치하는 것은 리소스 효율성 측면에서 불가능하다.

- 희소 직군의 배치 문제**:** 보안 전문가, DBA, 모바일 아키텍트 등 희소한 인력은 모든 스쿼드에 상주할 수 없다. 이들을 억지로 스쿼드에 소속시키면 업무 부하가 불균형해지고,

반대로 중앙 집중화하면 스쿼드의 자율성이 저해된다.

- 해결 방안 - 인에이블링 팀**(Enabling Team):** '팀 토폴로지(Team Topologies)' 이론에서 제시하는 인에이블링 팀 개념을 도입하여, 특정 전문가들이 여러 스쿼드를 순회하며 기술을 전파하고 컨설팅하는 구조를 활용한다.[28]

4.3 기술 부채**(Technical Debt)**와 유지보수의 늪

"새로운 기능을 빨리 출시하라"는 스쿼드의 미션은 필연적으로 유지보수 소홀과 기술 부채 누적을 초래한다.

- 유지보수 비용의 기하급수적 증가**:** 연구에 따르면 프로덕트가 성숙해질수록 엔지니어링 리소스의 75%가 유지보수에 투입되어야 한다. 그러나 신규 기능 개발에만 KPI가 맞춰진 스쿼드는 이를 간과하기 쉽다.[30]
- 템플릿 표류**(Template Drift):** 각 스쿼드가 자율적으로 기술 스택을 선택할 경우, 조직 전체의 기술 파편화가 발생하여(예: React, Vue, Angular 혼용) 인력 이동과 유지보수가 불가능해지는 현상이 발생한다.[32]
- 레거시 모놀리스**(Legacy Monolith):** 스쿼드 모델은 마이크로서비스 아키텍처(MSA)를 전제로 한다. 그러나 거대한 레거시 시스템을 여러 스쿼드가 공유하는 환경에서는 배포 충돌과 통합 테스트 병목으로 인해 자율성이 훼손된다.[1]
- 해결 방안**:** 플랫폼 엔지니어링 팀을 통해 내부 개발자 플랫폼(Internal Developer Platform, IDP)을 구축하고, 표준화된 '골든 패스(Golden Path)'를 제공하여 스쿼드의 인지 부하를 줄여야 한다.[34]

5. MECE 관점의 심층 분석**:** 거버넌스**,** 측정**,** 문화**,** 실패 유형

기존 문헌에서 간과되기 쉬운 요소들을 MECE 관점에서 보완하여 분석한다.

5.1 거버넌스와 조율**: '**정렬된 자율성**'**의 구현

자율성은 무질서가 아니다. 스쿼드 간의 충돌을 방지하고 전사적 목표를 달성하기 위한 거버넌스 체계가 필수적이다.

- **OKRs (Objectives and Key Results):** 전사 목표가 트라이브와 스쿼드 단위로 캐스케이딩(Cascading)되어야 한다. 각 스쿼드의 미션이 상위 조직의 OKR과 어떻게 연결되는지 명확해야 '자율적 의사결정'이 전략적 방향성을 잃지 않는다.[35]
- 분기별 계획**(QBR/Big Room Planning):** 스쿼드 간의 의존성을 식별하기 위해 분기별로 모든 리더가 모여 로드맵을 공유하고 조율하는 의식이 필요하다. 이는 SAFe의 PI Planning과 유사하지만, 보다 유연하게 운영된다.[5]
- 내부 소싱**(InnerSource):** 타 스쿼드의 코드를 수정해야 할 때 요청하고 기다리는 대신, 직접 수정하고 풀 리퀘스트(PR)를 보내는 내부 오픈소스 문화를 정착시켜 의존성 병목을 해결한다.[1]

5.2 성과 측정 지표**:** 산출물**(Output)**에서 결과**(Outcome)**로

스쿼드의 성공은 '얼마나 많이 만들었는가'가 아니라 '어떤 가치를 창출했는가'로 측정되어야

한다.

지표 카테고리	세부 지표	목적 및 의미	출처
DORA 지표	배포 빈도, 변경 리드 타임, 서비스 복구 시간, 변경 실패율.	엔지니어링 조직의 속도와 안정성을 측정하는 표준 지표.	[36]
프로덕트 성과	활성 사용자 수(DAU/MAU), 전환율, NPS, 매출 기여도.	스쿼드가 비즈니스 목표를 달성했는지 확인하는 결과 중심 지표.	[1]
팀 건강도	스쿼드 건강 체크(신호등 시스템: 재미, 속도, 학습 용이성 등).	지속 가능한 속도로 일하고 있는지, 팀 내 문화적 문제는 없는지 정성적 측정.	[36]
기술적 건강도	기술 부채 비율, 코드 복잡도, 결함 밀도.	장기적인 유지보수성을 저해하는 요소 관리.	[37]

5.3 조직 문화와 변화 관리**:** 심리적 안전감

구조만 바꾼다고 조직이 변하지 않는다. 스쿼드 모델이 작동하기 위한 문화적 토양이 필요하다.

- 실패에 대한 안전지대**:** '빠른 실패(Fail Fast)'가 가능하려면, 실패했을 때 비난받지 않고 학습의 기회로 삼는 '심리적 안전감(Psychological Safety)'이 필수적이다. A/B 테스트 실패를 문책하면 스쿼드는 안전하고 지루한 기능만 개발하게 된다.[12]
- 투명성과 정보 공유**:** 모든 정보는 '기본적 공개(Default to Open)' 원칙을 따라야 한다. 스쿼드가 자율적으로 판단하려면 경영진과 동일한 수준의 정보에 접근할 수 있어야 한다.[38]

5.4 도입 실패 유형 분석

스쿼드 모델 도입이 실패로 돌아가는 주된 원인은 다음과 같다.

1. 한국형 '서비스 기획자'와 **PO**의 충돌**:** 한국 IT 특유의 '서비스 기획자'는 화면 설계(스토리보드)부터 로직까지 상세하게 정의하는 데 익숙하다. 이들이 스쿼드의 PO가 될 경우, 개발자에게 '구현'만을 지시하는 마이크로매니징이 발생하여 스쿼드의 자율성을 해친다. 반대로 서구형 PO를 도입하면 상세 기획서가 없어 개발자들이 혼란을 겪는 경우도 발생한다.[39]

2. 좀비 스쿼드**(Zombie Squad):** 제품 시장 적합성(PMF)을 찾지 못했음에도 불구하고, 팀 해체나 피봇팅에 대한 의사결정이 지연되어 리소스만 소모하는 현상. 한국의 경우 정부 과제나 대기업 내부 사정으로 인해 명목상 유지되는 '좀비 스타트업/팀' 사례가 빈번하다.[42]
3. 무늬만 애자일**(Fake Agile):** 경영진이 여전히 명령 하달 방식을 고수하면서 팀 이름만 스쿼드로 바꾼 경우. 이는 구성원의 냉소만 불러일으키며 기존 기능 조직보다 못한 효율을 낸다.[11]

6. 결론 및 제언**:** 스쿼드를 넘어 팀 토폴로지로

스쿼드 조직 구조는 기능 중심의 효율성 최적화에서 목적 중심의 속도 최적화로 나아가는 강력한 도구이다. 그러나 이는 만병통치약이 아니며, 'Spotify 모델'이라는 환상에서 벗어나 각 조직의 맥락에 맞게 재설계되어야 한다.

성공적인 도입을 위한 핵심 제언**:**

1. 맥락적 적용**:** Spotify를 맹목적으로 모방하지 말고, 조직의 규모, 기술적 성숙도, 규제 환경에 맞춰 자율성의 범위를 조정하라.
2. 플랫폼 엔지니어링 투자**:** 스쿼드가 비즈니스 로직에 집중할 수 있도록, 인프라와 배포 파이프라인을 서비스 형태로 제공하는 플랫폼 조직을 구축하라.
3. **PO**와 챕터 리드의 역할 명확화**:** 딜리버리(PO)와 역량 개발(챕터 리드) 사이의 긴장을 건전한 균형으로 승화시킬 수 있도록 역할 정의와 평가 보상 체계를 정교하게 설계하라.
4. 한국적 특수성 고려**:** 서비스 기획자의 역량을 PO의 전략적 역량으로 전환하거나, 디자이너/개발자가 기획의 일부를 흡수하도록 하여 '기획-개발' 간의 사일로를 타파하라.

결론적으로, 2025년 이후의 조직 구조 트렌드는 단순한 스쿼드 모델을 넘어, 스트림 정렬 팀(Stream-aligned Team), 플랫폼 팀(Platform Team), 인에이블링 팀(Enabling Team), 컴플리케이티드 서브시스템 팀(Complicated Subsystem Team)이 유기적으로 상호작용하는 '팀 토폴로지' 형태로 진화하고 있다.[29] 조직의 경쟁력은 구조 그 자체가 아니라, 그 구조 안에서 흐르는 신뢰와 협력의 문화에서 결정된다.

---

7. 부록**:** 상세 분석 및 데이터

7.1 스쿼드 내 역할 갈등 심층 분석**:** 한국의 **'**기획자**' vs** 글로벌 **'PO'**

한국의 IT 생태계에서 '기획자'는 독특한 위상을 가진다. 웹 에이전시 시절부터 발달한 이 직군은 디자이너와 개발자 사이에서 상세한 화면 설계서(SB)를 작성하며 프로젝트를 조율해왔다. 스쿼드 모델 도입 시 이 역할은 가장 큰 변화의 압력을 받는다.

구분	전통적 서비스 기획자 **(Planner)**	애자일 프로덕트 오너 **(PO)**	충돌 포인트 및 해결 방향

주요 산출물	상세 스토리보드 (PPT, Figma), 기능 명세서	백로그(Backlog), 유저 스토리, 우선순위 목록	충돌: 상세 명세가 없으면 개발이 안 되는 관성 vs 명세 작성하느라 시장 대응이 늦어지는 비효율. 해결: PO는 'Why/What'에 집중하고, 'How'에 해당하는 UI/UX 디테일은 프로덕트 디자이너와 개발자가 주도하도록 권한 위임.[39]
개발팀과의 관계	작업 지시 및 일정 관리 (WBS 기반)	비전 공유 및 협업, 스프린트 목표 설정	충돌: 상하 관계적 지시 vs 수평적 파트너십. 해결: PO를 '미니 CEO'로 정의하되, 기술적 구현 방식에는 개입하지 않는 원칙 수립.[15]
책임 범위	화면 설계 및 기능 구현 확인	프로덕트의 비즈니스 성과(KPI/OKR) 및 ROI	충돌: 기능 구현 자체를 성과로 인식 vs 비즈니스 임팩트를 성과로 인식. 해결: 성과 지표를 산출물(Output)에서 결과(Outcome)로 전환하여 평가.[44]

7.2 유지보수와 신규 개발의 균형: **75%**의 법칙 대응 전략

스쿼드 모델 운영 시 가장 간과되는 것이 '운영(Operation)' 업무다. 출시 초기에는 100% 신규 개발이지만, 서비스가 성장함에 따라 유지보수 비중이 늘어난다.

- 현상: 연구에 따르면 소프트웨어 생애주기 비용의 75%는 유지보수에서 발생한다. 스쿼드가 이를 감당하지 못하면 신규 기능 개발이 중단되고 팀의 사기가 저하된다.[30]
- 대응 전략:
  1. 혁신 스쿼드 vs 유지보수 스쿼드 분리? (비추천): 이는 '좋은 일'과 '나쁜 일'을 나누는 계급화를 초래한다.
  2. 로테이션 제도 (추천): 우아한형제들이나 카카오의 사례처럼, 일정 기간마다 구성원을 순환시켜 레거시 유지보수의 고통을 분담하고 다양한 도메인 지식을 습득하게 한다.[45]
  3. 데브옵스(DevOps) 내재화: 스쿼드 내에 운영 자동화 역량을 키워, 반복적인 유지보수 업무를 코드로 해결(Toil reduction)하도록 유도한다.[46]

7.3 스쿼드 모델의 진화: 팀 토폴로지(Team Topologies)로의 확장

단순히 스쿼드를 나누는 것을 넘어, 팀 간의 상호작용 방식(Interaction Mode)을 정의하는 것이 최신 트렌드이다.

- 스트림 정렬 팀 (Stream-aligned Team): 우리가 아는 일반적인 스쿼드. 비즈니스 흐름에 맞춰 가치를 전달한다.
- 플랫폼 팀 (Platform Team): 스트림 정렬 팀이 자율적으로 인프라를 사용하도록 내부 제품(플랫폼)을 제공한다. 이들은 스쿼드를 '내부 고객'으로 대우한다.[34]
- 인에이블링 팀 (Enabling Team): 기술 부채, 아키텍처, 테스트 자동화 등 특정 분야의 역량이 부족한 스쿼드에 한시적으로 합류하여 역량을 이식해주고 떠나는 특수 팀이다.[28]
- 복잡한 하위시스템 팀 (Complicated Subsystem Team): 영상 처리 코덱, AI 모델링 등 고도의 전문 지식이 필요하여 일반 스쿼드가 다루기 힘든 모듈을 전담한다. 이는 스쿼드의 인지 부하(Cognitive Load)를 줄여주는 역할을 한다.[43]

인용된 참고 문헌:
.1

참고 자료

1. Scaling Agile @ Spotify - Crisp's Blog, 1월 4, 2026에 액세스, https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf
2. What Is The Spotify Model? - Product School, 1월 4, 2026에 액세스, https://productschool.com/blog/product-fundamentals/spotify-model-scaling-agile
3. Introduction to spotify model | PDF - Slideshare, 1월 4, 2026에 액세스, https://www.slideshare.net/slideshow/introduction-to-spotify-model/251193113
4. Squad-Based Agile Development - Model Structure and Benefits - Resquad AI, 1월 4, 2026에 액세스, https://resquad.ai/squad-based-agile-development/
5. What is Spotify Model in Agile? - KnowledgeHut, 1월 4, 2026에 액세스, https://www.knowledgehut.com/blog/agile/spotify-model-agile
6. Agile Squad Model Explained - Daily.dev, 1월 4, 2026에 액세스, https://daily.dev/blog/agile-squad-model-explained
7. The Spotify Model: A Guide to Scaling Dev - Enyata, 1월 4, 2026에 액세스,

https://www.enyata.com/blog/the-spotify-model-a-guide-to-scaling-dev-teams
8. The Spotify Model: Magic Bullet or Overrated? - USU, 1월 4, 2026에 액세스, https://www.usu.com/en/blog/the-spotify-model-magic-bullet-or-overrated
9. What is the Agile Spotify Model? - Product HQ, 1월 4, 2026에 액세스, https://producthq.org/agile/agile-spotify-model/
10. 7 main elements of Spotify's Tribe Engineering Model in 2025 - Dworkz, 1월 4, 2026에 액세스, https://dworkz.com/article/7-main-elements-of-spotifys-tribe-engineering-model-in-2025/
11. The Spotify Model of Scaling - Spotify Doesn't Use It, Neither Should You - Agile Pain Relief, 1월 4, 2026에 액세스, https://agilepainrelief.com/blog/the-spotify-model-of-scaling-spotify-doesnt-use-it-neither-should-you/
12. Spotify Model - Agile Pain Relief, 1월 4, 2026에 액세스, https://agilepainrelief.com/glossary/spotify-model/
13. Why Copying Spotify's Squads and Tribes Model Probably Will Not Work for You - Medium, 1월 4, 2026에 액세스, https://medium.com/@a.thongkum/why-copying-spotifys-squads-and-tribes-model-probably-will-not-work-for-you-b0b0408ac883
14. The Power of Three: The Journey of an Agile Leadership Team, 1월 4, 2026에 액세스, https://agilealliance.org/resources/experience-reports/the-power-of-three-the-journey-of-an-agile-leadership-team/
15. 토스 채용, 1월 4, 2026에 액세스, https://toss.im/career/culture
16. Organizational Silos Are Bleeding Your Revenue: The 3-Question Reality Check, 1월 4, 2026에 액세스, https://www.agilemeridian.com/podcasts/meridian-point-2/episodes/2149040097
17. No dilly-dallying here: Baemin's delivery bot hits the streets of Gangnam, 1월 4, 2026에 액세스, https://koreajoongangdaily.joins.com/news/2025-02-25/business/industry/No-dillydallying-here-Baemins-delivery-bot-hits-the-streets-of-Gangnam/2249820
18. Woowa Brothers Establishes 'AI Way of Working'... One-Third of All Employees Use Agents, 1월 4, 2026에 액세스, https://www.newstheai.com/news/articleView.html?idxno=10197
19. Developers of Woowa, how do they grow? (a.k.a. system and culture of the development organization), 1월 4, 2026에 액세스, https://techblog.woowahan.com/10538/
20. The Stanford Question Answering Dataset, 1월 4, 2026에 액세스, https://rajpurkar.github.io/SQuAD-explorer/
21. 2022 Kakao's Commitment and Responsibility - 카카오, 1월 4, 2026에 액세스, https://www.kakaocorp.com/media/esg-resource/pdf/ESGReport2022_EN.pdf
22. Kakao ESG Report, 1월 4, 2026에 액세스, https://www.kakaocorp.com/media/esg-resource/pdf/Kakao_ESGReport2024_EN.pdf
23. (PDF) Towards Entrepreneurial Organization: From the case of Organizational

Process Innovation in Naver - ResearchGate, 1월 4, 2026에 액세스, https://www.researchgate.net/publication/321750468_Towards_Entrepreneurial_Organization_From_the_case_of_Organizational_Process_Innovation_in_Naver
24. 1 1447818 1275170068 | PDF - Scribd, 1월 4, 2026에 액세스, https://www.scribd.com/document/919006498/1188811-2169015-1-1447818-1275170068
25. Spotify Agile Model: What it is and How it Works | TCGen - Product Development Experts, 1월 4, 2026에 액세스, https://www.tcgen.com/agile/spotify-agile-model/
26. Overcoming the Pitfalls of the Spotify Model | by Shubham Sharma | Medium, 1월 4, 2026에 액세스, https://medium.com/@ss-tech/overcoming-the-pitfalls-of-the-spotify-model-8e09edc9583b
27. Experimenting With Spotify Inspired Chapters - Next Chapter Consulting, 1월 4, 2026에 액세스, https://nextchapterconsulting.ca/2024/03/31/experimenting-with-spotify-inspired-chapters/
28. Agile Teams: From Squads to Scaled Frameworks – Software Engineering, 1월 4, 2026에 액세스, https://softengbook.org/articles/agile-teams
29. Platform Engineering vs DevOps: who to hire and why? | DistantJob, 1월 4, 2026에 액세스, https://distantjob.com/blog/platform-engineering-vs-devops/
30. The ugly truth about why your product development team is so slow, 1월 4, 2026에 액세스, https://productdirection.co/the-ugly-truth-about-why-your-product-development-team-is-so-slow/
31. How much time do you spend building new stuff vs maintaining existing stuff? - Reddit, 1월 4, 2026에 액세스, https://www.reddit.com/r/ProductManagement/comments/164d9v1/how_much_time_do_you_spend_building_new_stuff_vs/
32. Eliminating Template Drift and Reducing Technical Debt in Backend Development, 1월 4, 2026에 액세스, https://amplication.com/blog/eliminating-template-drift-and-reducing-technical-debt-in-backend-development
33. Modernizing legacy systems: 3 strategies, building business-case & embedding security, 1월 4, 2026에 액세스, https://inwedo.com/blog/modernizing-legacy-systems/
34. Platform engineering vs. DevOps - Red Hat, 1월 4, 2026에 액세스, https://www.redhat.com/en/topics/platform-engineering/platform-engineering-vs-devops
35. Spotify Squad Framework - CREATEQ, 1월 4, 2026에 액세스, https://www.createq.com/en/software-engineering-hub/spotify-squad-framework
36. Agile Spotify Model: Squads, Tribes, Chapters & Guilds Explained - Echometer, 1월 4, 2026에 액세스, https://echometerapp.com/en/agile-spotify-model-squads-tribes-chapters-and-guilds-explained/

37. What is Technical Debt? Learn How to Manage and Reduce Your Tech Debt | O8, 1월 4, 2026에 액세스, https://www.o8.agency/blog/what-technical-debt-learn-how-manage-and-reduce
38. The Spotify Playbook | The Agile Warrior - WordPress.com, 1월 4, 2026에 액세스, https://agilewarrior.wordpress.com/2017/11/21/the-spotify-playbook/
39. Product Manager vs Product Owner. “What is the difference between a... | by Melissa Perri | Medium, 1월 4, 2026에 액세스, https://medium.com/@melissaperri/product-manager-vs-product-owner-57ff829aa74d
40. Who really owns the squad? (Spoiler...it's not the product manager), 1월 4, 2026에 액세스, https://www.mindtheproduct.com/who-really-owns-the-squad-spoiler-its-not-the-pm/
41. “Service Planner” — The Korean version of a Product Manager/UX Designer, 1월 4, 2026에 액세스, https://namyoonkim.medium.com/service-planner-the-korean-version-of-a-product-manager-ux-designer-d2f42b90d543
42. Zombie Startups Phenomenon: Nearly Half of Korea's Sanctioned R&D Firms Collapsed, 1월 4, 2026에 액세스, https://koreatechdesk.com/zombie-startups-korea-nearly-half-sanctioned-firms-collapsed
43. How to Choose the Right Engineering Team Structure for Your Organization - Jellyfish.co, 1월 4, 2026에 액세스, https://jellyfish.co/blog/engineering-organization-structure/
44. Product Owner vs. Product Manager: Which Role Drives Your Product Forward? - BairesDev, 1월 4, 2026에 액세스, https://www.bairesdev.com/blog/product-owner-vs-product-manager/
45. Scaling Beyond Bezos' "2-Pizza Team" Rule | Startup Grind, 1월 4, 2026에 액세스, https://www.startupgrind.com/blog/to-scale-past-bezoss-2-pizza-team-rule-try-switching-teams-every-few-months/
46. Mapping DevOps learnings to management - Spotify Engineering, 1월 4, 2026에 액세스, https://engineering.atspotify.com/devops-management
47. Platform Engineering vs. DevOps: Choosing the Right Approach for Scaling Modern Development - Cloud2, 1월 4, 2026에 액세스, https://cloud2.net/blogi/platform-engineering-vs-devops
48. Spotify model at tech11: An agile working method, 1월 4, 2026에 액세스, https://tech11.com/en/blog/spotifymodel
49. Don't Follow Spotify's Agile Model - Joey Guerra, 1월 4, 2026에 액세스, https://joeyguerra.com/blog/2018/spotify-agile-engineering-culture.html
50. Baemin Story 2022, 1월 4, 2026에 액세스, https://woowahan-cdn.woowahan.com/file/story2022/Baemin_Story_2022.pdf
51. How to Build an Engineering Culture | by Tom Drapeau | Codifying - Medium, 1월 4, 2026에 액세스, https://medium.com/codifying/how-to-build-an-engineering-culture-6c6b04a671

7
52. Agile Squads - CGI.com, 1월 4, 2026에 액세스, https://www.cgi.com/sites/default/files/2023-03/brochure_cgi_agile_squads2023_en.pdf
53. What are Squads in Agile: Master the Framework That Drives Innovation - Dart AI, 1월 4, 2026에 액세스, https://www.dartai.com/blog/what-are-squads-in-agile
54. SRE vs DevOps: Key Differences for Improved Collaboration | Atlassian, 1월 4, 2026에 액세스, https://www.atlassian.com/devops/frameworks/sre-vs-devops
55. DevOps vs. SRE: Bridging the Gap, Not Building Walls (Part 1) - Redgate Software, 1월 4, 2026에 액세스, https://www.red-gate.com/simple-talk/devops/devops-vs-sre-bridging-the-gap-not-building-walls-part-1/
56. Technology that opens a new world - Naver Corp, 1월 4, 2026에 액세스, https://www.navercorp.com/en/storyDetail?seq=32477
57. The Product Owner role should be scrapped. : r/agile - Reddit, 1월 4, 2026에 액세스, https://www.reddit.com/r/agile/comments/1l2aja9/the_product_owner_role_should_be_scrapped/
58. SQUAD vs Traditional Model : How Agile Teams Drive Fast ROI - Nimblechapps, 1월 4, 2026에 액세스, https://www.nimblechapps.com/blog/benefits-of-squad-model-over-traditional-hire-models
59. Crafting the Core: Structuring Your Engineering Team for Scalable Success - CodePath, 1월 4, 2026에 액세스, https://www.codepath.org/news/eng-team-structure
60. An Engineering Manager's Guide to Team Structure and Organization - Revelo, 1월 4, 2026에 액세스, https://www.revelo.com/blog/team-structure
61. Why companies fail in international markets | by Yong Su Kim - Medium, 1월 4, 2026에 액세스, https://medium.com/@yongsu/why-companies-fail-in-international-markets-4b177891d4ce
62. Spotify's Failed #SquadGoals - Jeremiah Lee, 1월 4, 2026에 액세스, https://www.jeremiahlee.com/posts/failed-squad-goals/