

集中型と分散型のバージョン管理システムは ソフトウェアの変更にどのような影響を与えるのか

How Do Centralized and Distributed Version Control Systems Impact Software Changes?

Caius Brindescu, Mihai Codoban, Sergii Shmarkatiuk, Danny Dig
School of EECS, Oregon State University
Corvallis, OR, USA
{brindesc,codobanm,shmarkas,digd}@eecs.oregonstate.edu

背景

- 従来の集中型バージョンシステム(CVCS)より分散型バージョン管理システム(DVCS)が主流になりつつある
 - 65%がDVCS、35%がCVCSを利用
 - 今回調査した820人の開発者中

DVCSの特徴

- ローカルにコピーできるので、サーバに繋がなくても編集できる
- 簡単にブランチやマージができる
- ファイル単位でコミットできる
 - CVCSは強制的に全ファイルがコミットされる

問題点

- **DVCSはCVCSより有益なのかあまりわかっていない**
 - **開発者&マネージャー**
 - 新しいツールに移行するのに労力と時間を費やしていいのか
 - **研究者**
 - DVCSの影響がわからないと研究で間違った結果を出す
 - **ツール開発者**
 - 開発者の行動がわからないと間違ったツールを作成する

目的

- DVCSがソフトウェアバージョン管理方法にどのように影響するか調査をする
 - DVCSは本当に有益なのか
 - 研究者、開発者、ツール開発者、マネージャーに見識を提供

RQ

Theme1

VCSの型と開発者の行動の関係

RQ1:コミットサイズ
RQ2:コミットを分割するか
RQ3:スカッシュの頻度と理由
RQ4:そのVCS型を好む理由
RQ5:コミット頻度

Theme2

開発チームの規模とVCS利用法の関係

RQ6:VCS型の選択
RQ7:ITSの利用
RQ8:コミットサイズ
RQ9:スカッシュ頻度

Theme3

VCSの型と開発プロセスの関係

RQ10:ITLがあるか
RQ11:ITLの数とコミットサイズの
関係
RQ12:コミットサイズ

アプローチ

アンケート調査

- 20問: 多肢選択型、自由選択型含む
- 820人: 発達したSNS(TwitterやGoogle+など)で募る
 - 商業的なソフトウェア開発者: 85%
 - 10年以上の開発経験がある: 56%
 - 6人以上の大規模なチームで開発: 51%

アンケート調査の
結果から選定

リポジトリマイニング

Table 2: Repository corpus.

Repo. Type	Repositories	Commits	Authors	Total LOC changed
SVN	52	95571	451	270M
Hybrid	29	151004	2249	89M
Git	51	111725	3190	50M
Total	132	358300	5890	409M

132リポジトリ

- SVN: SOURCE-FORGE
- Git: GITHUB

プロジェクトを3種類に分類
SVN: CVCS、Git: DVCS
Hybrid: **CVCS**→**DVCS**
(HybridSVNとHybridGitで区別)

結果

Theme1

VCSの型と開発者の行動の関係

RQ1: コミットサイズ

	Mean		Median		StdDev	
	LOC	files	LOC	files	LOC	files
Git	27.20	3.08	13.46	1.96	32.72	2.7
Svn	40.06	5.65	18.44	3.19	49.62	6.72
HybridGit	23.02	2.40	11.52	1.70	27.57	1.74
HybridSVN	25.72	2.82	12.61	1.96	31.24	2.15

- DVCSの方がCVCSよりコミットサイズが小さい
- Hybridリポジトリでは差がない

RQ2: コミットを分割するか

Practice	DVCS	CVCS	Overall
Split their changes	81.25	67.89	75.99
Group their changes	12.50	26.61	18.05
Other	6.25	5.50	5.96

- 76%がコミットを分割。DVCSの方がCVCSより割合が高い

Technique	DVCS	CVCS	Overall
Implementation details	37.01	21.85	32.03
Intent of change	45.13	62.251	50.76
Policy	6.17	5.30	5.88
Other	11.69	10.60	11.33

- 全体的に開発者は意図的にコミットを分割する
- DVCSの方が実装に基づいて頻繁にコミットを分割する

Theme1

VCSの型と開発者の行動の関係

RQ3: スカッシュの頻度と理由

Reason	DVCS	CVCS	Overall
Group similar changes	25.63	45.16	28.80
Intermediate steps are irrelevant	20	0	16.75
Remove mistakes	15	0	12.57
Keep history clean	26.88	6.45	23.56
Policy requirement	5.63	9.68	6.28
Other	6.88	38.71	12.04

Response	DVCS	CVCS	Overall
Yes	36.59	18.12	30.21
No	54.79	44.57	51.31
Not applicable	8.62	37.32	18.48

- スカッシュは頻繁にしない
- スカッシュはDVCS利用者に多い

RQ4: そのVCS型を好む理由

Reason	DVCS	CVCS	Overall
Killer feature	46.02	10.89	30.41
Old habit	22.88	41.58	30.41
Easy to use	19.79	41.58	27.14
Personal preference	2.06	0.99	2.04
Other	9.25	4.95	10

- DVCS: 機能が使いやすい
- CVCS: 慣れているから使いやすい

Theme1

VCSの型と開発者の行動の関係

RQ5:コミット頻度

	DVCS	CVCS	Overall
Once a minute	3.38	0.82	2.51
Once an hour	19.66	4.10	14.37
Several times a day	65.96	66.80	66.25
Once a day	7.19	14.75	9.76
Several times a week	1.90	9.43	4.46
Once a week	1.48	3.32	2.09
Once a month	0.42	0.82	0.56

- ほとんどの開発者はVCSの型によって似たような頻度でコミットする

Implications

● 開発者へ

- DVCSは**行単位**でのコミット分割が可能で変更がしやすい
 - 行単位:CVCSでは出来ない
- DVCSは1つの変更ごとにコミットできるので管理がしやすい
- **DVCSの学習コストは無駄じゃない**

● 研究者へ

- DVCSの方がコミットが小さいので分析しやすい
 - CVCSは1コミットに複数の変更内容が含まれている場合がある
- 分析する上での注意点
 - HybridリポジトリはCVCSのときの規約や習慣が残っている場合がある
 - **VCSの型によって開発者の振る舞いが違う**

Theme2

開発チームの規模とVCS利用法の関係

RQ6: VCS型の選択

VCS type	1	2-5	6 - 10	11 - 25	26 - 100	> 100
DVCS	82.22	72.07	62.30	65.22	60.00	70.97
CVCS	17.78	27.93	37.70	34.78	40.00	29.03

- どの開発チーム規模でもDVCSを選ぶ方が多い

RQ7: ITSの利用

Team size	Use an ITS	Don't use an ITS
1	63.04	36.96
2 - 5	97.88	2.12
6 - 10	92.23	7.77
10 - 25	95.24	4.76
26 - 100	97.01	2.99
Over 100	100.00	0.00
Overall	93.87	6.13

- ほとんどのプロジェクトがITSを使用している

Theme2

開発チームの規模とVCS利用法の関係

RQ8:コミットサイズ

Repository type	LOC	# of files
Git	-0.11	-0.09
SVN	-0.07	-0.16
Hybrid _{GitStage}	-0.01	0.16
Hybrid _{SVNStage}	-0.06	-0.35

- 相関があるとは言えない

RQ8:スカッシュ頻度

Team size	Squash	Don't squash	Not Applicable
1	17.39	56.52	26.09
2-5	27.27	54.55	18.18
6-10	42.25	29.50	28.06
11-25	30.16	50.79	19.05
26-100	40.00	46.15	13.85
Over 100	57.14	34.29	8.57

- 大きな規模の開発チームはスカッシュ頻度が多い

Implications

- 研究者へ
 - 大規模な開発チームはよりスカッシュをするので品質がいいが、スカッシュをするのでコミットが大きくなり複雑になりやすい
- マネージャーへ
 - たくさんの変更が必要な開発を行っている大規模なチームは、DVCSへの移行を考えるべき

Theme3

VCSの型と開発プロセスの関係

RQ10: ITLがあるか

- ITLがあるコミットは少数だが、DVCSの方が多い

RQ11: ITLの数とコミットサイズの関係

VCS type	number of issue references					corr.
	1	2	3	4	5	
SVN	33.04	35.69	54.56	31	80	0.68
Git	25.27	36.46	38.05	23	23	-0.38
Hybrid	27.67	31.66	37.74	83.2	62.14	0.81
All	28.59	34.72	39.91	57.78	60.08	0.97

- SVNとHybridでは正の相関
- Gitでは負の相関

RQ12: 期間ごとのコミットサイズ

- プロジェクトが長いほどコミットサイズは小さくなる

Implications

Theme3

VCSの型と開発プロセスの関係

- 研究者へ
 - プロジェクトが長いほどコミットサイズが小さくなるので、そのプロジェクトがどのステージにあるかわかる
 - DVCSはCVCSよりもITLがあるのでITSと対応させて研究しやすい
- マネージャーへ
 - 長いプロジェクトは量だけでなくソースコードの複雑度や重要性も見るようにする

まとめ

- この論文は初めてDVCSを大規模に調査したもの
- 全体的にDVCSは学習コストをかける価値がある
 - DVCSはコミットサイズが小さくなる
 - 特徴的機能より
 - どの規模の開発チームもDVCSを選択する方が多い
 - DVCSの方がITLがあることが多い
 - コミットが1つの変更に対応しているので、ITSとの対応付けがしやすい
- DVCSとCVCSの特徴を知っておく
 - 開発者、研究者、ツール開発者、マネージャー

所感

- RQが多くてうまくまとめられなかった
- 字が多い(工夫して読みやすくしてくれてはいるけど...)