

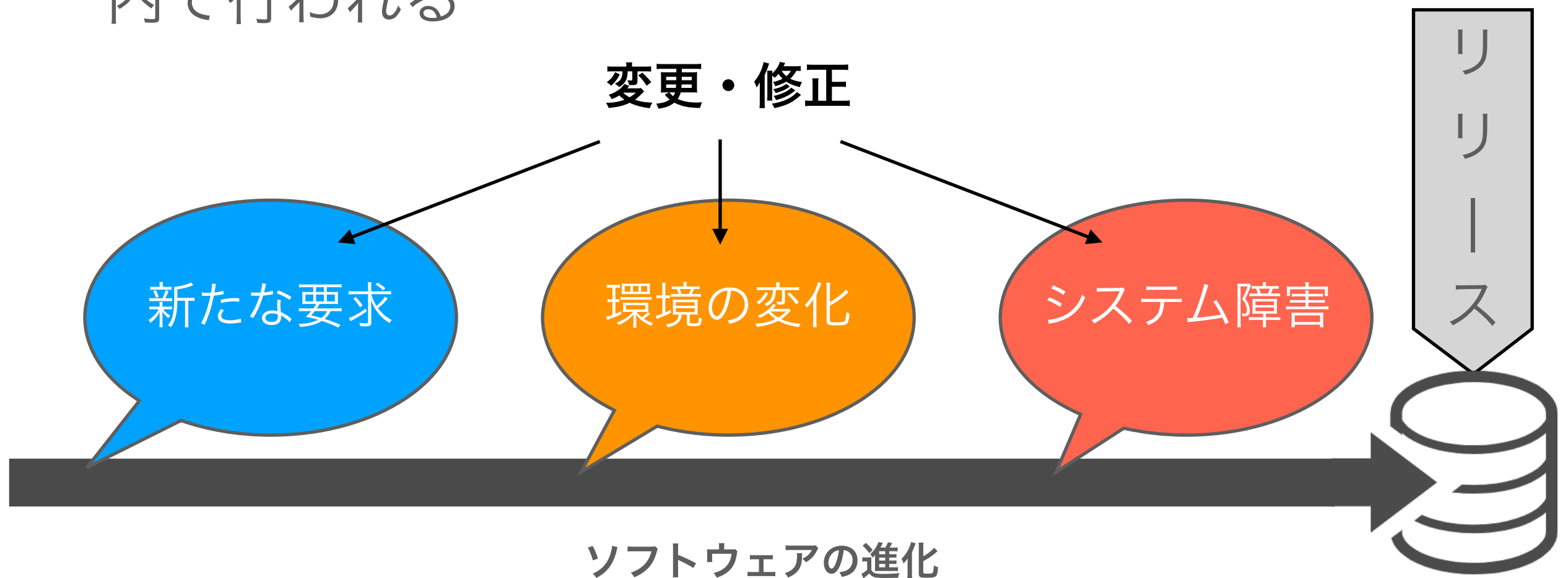
Developer-Driven Code Smell Prioritization

Fabiano Pecorelli,¹ Fabio Palomba,¹ Foutse Khomh,² Andrea De Lucia¹

MSR'20 (Mining Software Repositories Conference)

背景

- ソフトウェアの進化に合わせて継続的に変更を加える必要がある
- しかし、これらの活動はリリース締切や制約の範囲内で行われる



Technical Debt

- TD (Technical Debt)
 - 短期的には有用だが、ソフトウェア品質の低下を招く
最適でない実装
- 多くのソフトウェアプロジェクトにはリリース時期が定められている
- そのため、開発者は実装を優先し TD を混入させてしまうことがある

Code Smell

- 複雑なクラス, 冗長なクラス, 結合度が高い など
粗末な設計・実装のこと
- ソースコードの可読性^[1], 保守性^[38,59], 試験容易性^[28]
の低減を招き, 保守コストを増加させる^[80]
- Code Smell の自動検出が盛んに研究されている

目的

- 既存の Code Smell 自動検出ツールの問題点
 1. かなりの量の Code Smell が検出されるが、最適なリファクタリング方法がわからない
 2. 開発者が重要視していない Code Smell が大部分を占める

**開発者が認識する重要度で
Code Smell インスタンスの優先順位付を行う**

対象プロジェクト

- Apache, Eclipse の 9 プロジェクト
 - Java, クラス数が 500 以上, 変更履歴が 5 年以上, コミット数が 1,000 以上, 貢献者数が 20 以上

Table 1: Software Projects in Our Dataset.

Project	#Commits	#Devs	#Classes	KLOCs
Apache Mahout	3,054	55	813	204
Apache Cassandra	2,026	128	586	111
Apache Lucene	3,784	62	5,506	142
Apache Cayenne	3,472	21	2,854	542
Apache Pig	2,432	24	826	372
Apache Jackrabbit	2,924	22	872	527
Apache Jena	1,489	38	663	231
Eclipse CDT	5,961	31	1,415	249
Eclipse CFX	2,276	21	655	106
Overall	32,889	436	5,506	542

Code Smell Type とツール

- Decore
 - Blob, Complex Class, Spaghetti Code 検出可
- Hist
 - Shotgun Surgery 検出可

Code Smell	説明
Blob (God Class)	いろんな機能が詰め込まれたクラス
Complex Class	Cyclomatic複雑度が高いクラス
Spaghetti Code	オブジェクト指向を無視したクラス
Shotgun Surgery	結合度が高いクラス

Code Smell の重要度 (1)

- 選定したプロジェクトを 6ヶ月間モニタリング
- Code Smell に関連する変更が加えられるたびに開発者に以下のメールを送信
 1. Code Smell の存在を認識していたかどうか
 2. もし認識していたなら, どのような Code Smell のタイプと認識していたか
 3. もし認識していたなら, その重要度はどの程度か
 - ✓ 1 (low) ~ 5 (high) : Likert スケールで分類

Code Smell の重要度 (2)

1. Code Smell の存在を認識していたかどうか

➡ Yes : 1,332, No : 91, No-reply : 310

- 最終的に得られた実際の Code Smell

Code Smell	件数
Blob (God Class)	341
Complex Class	349
Spaghetti Code	313
Shotgun Surgery	329

Research Question

- RQ1
 - 開発者が認識する重要度を予測できるか
- RQ2 :
 - どの特徴が提案手法の予測性能を引き上げているのか
- RQ3 :
 - 他の手法に比べて提案手法はどの程度か

提案手法で用いられるメトリクス

Table 2: Software metrics used as independent variables split by categories - The motivations for their use are also reported.

Metric	Acronym	Description
Product metrics. Cohesion, coupling, and complexity may lower code quality and affect the perceived criticality of code smells [61, 84, 85].		
Lines of Code	LOC	Amount of lines of code of a class excluding white spaces and comments.
Lack of Cohesion of Methods [15]	LCOM5	Number of method pairs in a class having no common attribute references.
Conceptual Cohesion of Classes [48]	C3	Average cosine similarity [5] computed among all method pairs of a class.
Coupling between Object Classes [15]	CBO	Number of classes in the system that call methods or access attributes of a class.
Message Passing Coupling [15]	MPC	Number of method calls made by a class to external classes of the system.
Response for a Class [15]	RFC	Sum of the methods of a class, <i>i.e.</i> , number of methods that can potentially be executed in response to a message received by an object of a class.
Weighed Methods per Class [15]	WMC	Sum of the McCabe cyclomatic complexity [50] computed on all methods of a class.
Readability [11]	Read.	Measure of source code readability based on 25 features, <i>e.g.</i> , number of parentheses per lines of code.
Process metrics. The amount of activities made on smelly classes may affect the developer's perceived criticality [42, 45, 74].		
Average Commit Size	AVG_CS	Average number of classes that co-changed in commits involving a class.
Number of Changes	NC	Number of commits in the change history of the system involving a class.
Number of Bug Fixes	NF	Number of bug fixing activities performed on a class in the change history of the system.
Number of Committers	NCOM	Number of distinct developers who performed commits on a class in the change history of the system.
Developer-related metrics. Experience and workload of developers may affect the perceived criticality of code smells [8, 12, 14, 91].		
Developer's Experience [8]	EXP	Average number of commits of the committers of a class.
Developer's Scattering Changes [19]	DSC	Average number of distinct subsystems in which the committers of a class made changes.
Development Change Entropy [33]	CE	Shannon's entropy [77] computed on the number of changes of a class in the change history of the system.
Code Ownership [8]	OWN	Number of commits of the major contributor of a class over the total number of commits for that class.
Code smell-related metrics. Persistence of code smells and availability of refactoring opportunities may affect the developer's perception [58, 68].		
Persistence	Pers.	Number of subsequent major/minor releases in which a certain smell affects a class.
Intensity [49]	Sev.	Average distance between the actual metric values used for the detection of code smells and the corresponding thresholds considered by the detectors to distinguish smelly and non-smelly elements.
Refactorable	Ref.	Existence of refactoring opportunities for a class, as detected by automated tools.
Number of Refactoring Actions	NR	Number of previous refactoring operations made by developers on a class.

RQ1 · RQ3

開発者が認識する重要度を予測できるか・他の手法に比べて提案手法はどの程度か

- Our approach
 - **Random Forest**, Logistic Regression, Naive Bayes etc.
 - F値 : 72 ~ 85%
- Baseline ^[25]
 - F値 : 35 ~ 84%

Table 4: RQ₁ - RQ₃. Weighted Average of the performance achieved by the experimented models against our dataset.

Code smell	Model	Prec.	Rec.	F-Meas.	MCC	AUC-ROC
Blob	Our approach	86%	85%	85%	75%	89%
	Baseline	66%	69%	66%	44%	78%
Complex Class	Our approach	79%	81%	80%	71%	89%
	Baseline	62%	63%	63%	33%	76%
Spaghetti Code	Our approach	90%	88%	89%	83%	92%
	Baseline	83%	85%	84%	77%	89%
Shotgun Surgery	Our approach	74%	71%	72%	61%	78%
	Baseline	33%	40%	35%	32%	61%

RQ1

開発者が認識する重要度を予測できるか

- インスタンスごとにデータを分けたのちに、インスタンスの重要度を分類
- 誤分類率が 10% ~ 27%
- インスタンスの分類精度が向上すれば、より良いモデルになる可能性がある

Table 3: RQ₁ - RQ₃. Confusion matrices obtained when running the proposed model against our dataset.

Model	Class/Smell	Blob			Complex Class			Spaghetti Code			Shotgun Surgery		
		Non-severe	Medium	Severe	Non-severe	Medium	Severe	Non-severe	Medium	Severe	Non-severe	Medium	Severe
Our approach	Non-severe	54	10	6	46	17	27	57	11	2	46	37	13
	Medium	6	171	18	14	176	0	5	151	6	8	134	10
	Severe	1	10	65	6	1	62	0	7	74	5	16	60

誤分類 15% (51) 19% (65) 10% (31) 27% (89)

RQ2

どの特徴が提案手法の予測性能を引き上げているのか

Table 5: RQ₂. Information Gain of the independent variables of our approach. For space limits, only metrics providing significant contributions are reported.

Code smell	Metric	Mean	SK-ESD
Blob	RFC	0.65	68
	LCOM5	0.57	66
	NF	0.56	66
	DSC	0.55	64
	CBO	0.45	64
	WMC	0.42	64
	C3	0.35	45
	LOC	0.34	41
Complex Class	CBO	0.59	71
	WMC	0.54	69
	LCOM5	0.54	69
	Read.	0.54	69
	NC	0.50	54
	DSC.	0.49	51
	EXP	0.27	33
	RFC.	0.25	31
Spaghetti Code	Read.	0.65	53
	NF	0.57	46
	C3	0.38	41
Shotgun Surgery	NC	0.32	44
	LCOM5	0.31	39
	AVG_CS	0.24	33
	Pers.	0.17	21

メトリクス



Scott-Knott Effect Size Difference test
でメトリクスのランキング付け

RQ2

どの特徴が提案手法の予測性能を引き上げているのか

Table 5: RQ₂. Information Gain of the independent variables of our approach. For space limits, only metrics providing significant contributions are reported.

Code smell	Metric	Mean	SK-ESD
Blob	RFC	0.65	68
	LCOM5	0.57	66
	NF	0.56	66
	DSC	0.55	64
	CBO	0.45	64
	WMC	0.42	64
	C3	0.35	45
	LOC	0.34	41
Complex Class	CBO	0.59	71
	WMC	0.54	69
	LCOM5	0.54	69
	Read.	0.54	69
	NC	0.50	54
	DSC.	0.49	51
	EXP	0.27	33
	RFC.	0.25	31
Spaghetti Code	Read.	0.65	53
	NF	0.57	46
	C3	0.38	41
Shotgun Surgery	NC	0.32	44
	LCOM5	0.31	39
	AVG_CS	0.24	33
	Pers.	0.17	21

ほとんどがソースコードに関連するメトリクス

RQ2

どの特徴が提案手法の予測性能を引き上げているのか

Table 5: RQ₂. Information Gain of the independent variables of our approach. For space limits, only metrics providing significant contributions are reported.

Code smell	Metric	Mean	SK-ESD
Blob	RFC	0.65	68
	LCOM5	0.57	66
	NF	0.56	66
	DSC	0.55	64
	CBO	0.45	64
	WMC	0.42	64
	C3	0.35	45
	LOC	0.34	41
Complex Class	CBO	0.59	71
	WMC	0.54	69
	LCOM5	0.54	69
	Read.	0.54	69
	NC	0.50	54
	DSC.	0.49	51
	EXP	0.27	33
	RFC.	0.25	31
Spaghetti Code	Read.	0.65	53
	NF	0.57	46
	C3	0.38	41
Shotgun Surgery	NC	0.32	44
	LCOM5	0.31	39
	AVG_CS	0.24	33
	Pers.	0.17	21

プロダクト構造に関連する
メトリクスに加えて、開発者の
経験などが有効

RQ2

どの特徴が提案手法の予測性能を引き上げているのか

Table 5: RQ₂. Information Gain of the independent variables of our approach. For space limits, only metrics providing significant contributions are reported.

Code smell	Metric	Mean	SK-ESD
Blob	RFC	0.65	68
	LCOM5	0.57	66
	NF	0.56	66
	DSC	0.55	64
	CBO	0.45	64
	WMC	0.42	64
	C3	0.35	45
	LOC	0.34	41
Complex Class	CBO	0.59	71
	WMC	0.54	69
	LCOM5	0.54	69
	Read.	0.54	69
	NC	0.50	54
	DSC.	0.49	51
	EXP	0.27	33
	RFC.	0.25	31
Spaghetti Code	Read.	0.65	53
	NF	0.57	46
	C3	0.38	41
Shotgun Surgery	NC	0.32	44
	LCOM5	0.31	39
	AVG_CS	0.24	33
	Pers.	0.17	21

可読性や欠陥の数

RQ2

どの特徴が提案手法の予測性能を引き上げているのか

Table 5: RQ₂. Information Gain of the independent variables of our approach. For space limits, only metrics providing significant contributions are reported.

Code smell	Metric	Mean	SK-ESD
Blob	RFC	0.65	68
	LCOM5	0.57	66
	NF	0.56	66
	DSC	0.55	64
	CBO	0.45	64
	WMC	0.42	64
	C3	0.35	45
	LOC	0.34	41
Complex Class	CBO	0.59	71
	WMC	0.54	69
	LCOM5	0.54	69
	Read.	0.54	69
	NC	0.50	54
	DSC.	0.49	51
	EXP	0.27	33
	RFC.	0.25	31
Spaghetti Code	Read.	0.65	53
	NF	0.57	46
	C3	0.38	41
Shotgun Surgery	NC	0.32	44
	LCOM5	0.31	39
	AVG_CS	0.24	33
	Pers.	0.17	21

コミット時のファイル変更数
変更量など

まとめ・所感

- 4つの Code Smell インスタンスを用いて，開発者駆動の自動優先順位付の手法を提案
 - Code Smell インスタンスの分類ではそこそこ高い分類精度が得られた
 - Code Smell インスタンスごとに重要度分類した場合，高い分類精度が得られる
- 開発者がつけた重要度の理由があれば，より開発者の視点に沿った優先順位付の設計ができるのでは？