CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

The Evolution of Role-Stereotypes and Related Design (Anti)Patterns

Bachelor of Science Thesis in Software Engineering and Management

DUY NGUYEN NGOC
FABIAN FRÖDING

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

The Evolution of Role-Stereotypes and Related Design (Anti)Patterns

Supervisor: MICHEL R. V. CHAUDRON
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

The Evolution of Role-Stereotypes and Related Design (Anti)Patterns

Fabian Fröding
University of Gothenburg
Gothenburg, Sweden
gusfrodfa@student.gu.se

Duy Nguyen Ngoc
University of Gothenburg
Gothenburg, Sweden
gusngudu@student.gu.se

***Abstract*—This paper presents a study on how classes based on the role-stereotypes defined by Wirfs-Brock, change over time in software systems, and how the occurrence of anti-patterns change over time in relation to these roles. The aim of the study is to gain an understanding on how role-stereotypes change as software-systems evolve, and if these changes have possible correlations to certain anti-patterns.**

With an exploratory approach, we performed studies on the evolution of role-stereotypes and anti-patterns in three open source projects: Bitcoin Wallet, K9 Mail and Sweet Home 3D. By using descriptive graphs and through observation, we demonstrate how the distribution of role-stereotypes and the distribution of anti-patterns evolve over a selected number of versions of the three projects. Furthermore, we also analyzed the changes in role-stereotypes in relation to the the occurrence of anti-patterns in these roles. Additionally, we analyzed if there are certain roles that are more prone to switch to other roles.

We found that some changes in the occurrence of anti-patterns seems to be reflective to the changes in the distribution of role-stereotypes, and that the occurrence of anti-patterns in specific role-stereotypes seems to have more in common with the occurrence of anti-patterns in different roles in the same project, rather than with the occurrence of anti-patterns in the same roles in different projects. We also found that certain role-stereotypes are more prone to change role to other certain roles.

Therefore this study brings new insight to software developers and designers on the behaviour and nature of role-stereotypes and anti-patterns, when using classes designed based on role-stereotypes.

***Keywords*-Role-stereotypes, anti-patterns, software design, software evolution.**

I. Introduction

Responsibility-driven design is a technique introduced by Wirfs-Brock *et al.* [1] used to enact certain responsibilities and procedures to OO-classes (object-oriented) in software systems to improve aspects such as reusability, maintenance and expansion.

Role-stereotypes are roles that are generalized into predefined responsibilities in software design. Wirfs-Brock *et al.* [2] defined six roles for OO-classes (see Section II) as an approach for responsibility-driven design. This approach aids in the understanding of what work objects do [3], thus enhance coherence in the architectural facet of a software project. Role-stereotypes also play a significant part in the understanding of UML-diagrams (Unified modeling language), as discovered by Kuzniarz *et al.* [4].

Software *design patterns* are descriptions of objects and classes that attempt to solve common design problems in specific contexts, and aids in aspects such as code reusability, documentation and software maintenance [5]. On the contrary, *anti-patterns* are problematic structures in code design and impede software systems from being understandable, maintained and evolved [6], [7]. Preventing anti-patterns is important when developing and making changes to high-quality software systems [5]. Thus, neglecting the presence of anti-patterns can make software systems incomprehensible and is not a good long-term approach.

A practical approach that prevents anti-patterns from emerging in a system is to investigate the correlation between design (anti)-patterns and role-stereotypes. The study conducted by Chaudron *et al.* [8] was able to confirm such correlation, thus further benefit designers in paying extra attention to classes assigned with certain responsibilities. However, as a software system evolves, it is difficult for designers to predict on which degree the numbers of certain roles, along with patterns, increase when adding more classes or making changes to classes. Furthermore, some classes might change responsibilities after several development updates. For example, with the roles defined by Wirfs-Brock *et al.* [2], a class might switch from the role *Structurer*, *Interfacer*, *Coordinator*, or *Controller* to either *Information Holder* or *Service Provider*. As proved by Chaudron *et al.* [8], the formerly mentioned roles are less prone to contain anti-patterns than the latter ones. Such changes in roles are not able to explain whether correlating patterns will appear in the adjusted classes.

The purpose of our study is to provide insight to software designers and developers of how role-stereotypes and related anti-patterns evolve over time. Moreover, our study in combination with the results presented by Chaudron *et al.* [8], can present new results that will allow software designers and developers to become aware of the potential emergence of anti-patterns in related role-stereotypes when developing and making changes to software systems. Moreover, a survey-based study conducted by Yamashita *et al.* [9] found that the majority of software developers (68%), are aware and cautious of the consequences of anti-patterns. Therefore our study is primarily aimed to help the mentioned designers and developers.

The remainder of this paper is structured as follows. Section

II describes the cases on which the study is conducted. Section III defines the research methodology of the study, including the introduction of the research questions, the data collection methods and how the collected data is analyzed. Section IV illustrates the results and accordingly Section V discusses the implications and the impacts of the results with relation to the research questions. Section VI describes threats to the validity of our study. Section VII presents related work. Lastly, Section VIII states the conclusion to the study and proposes prospective work.

II. Case Description

The study was conducted on three open-source software projects, and analyzed the classes of those projects. Since the study is conducted on three different project, it is a multi-case study, and since it evaluates multiple versions of each projects over the course of several years, it is also longitudinal.

The specific projects that the study will be conducted on are (1) Bitcoin Wallet[1], an Andriod-application for bitcoin paymenys, (2) K9 Mail[2], an open-source client for emails on Android-systems, and (3) Sweet Home 3D[3], an application for interior-design that allows preview of the final design in 3D. These projects are the same systems that were analyzed by Chaudron *et al.* [8]. We chose to analyze the same systems because they have a validated ground truth established for the role-stereotypes.

All of the projects are written in Java, but the results are applicable to other programming languages as well (more on this in section IV and VI).

The six most common role-stereotypes are Information Holder, Structurer, Service Provider, Controller, Coordinator, and Interfacer [2], and are the ones that will are used in our study. The responsibilities of the role-stereotypes are as follows; Information Holder stores and provides information, Structurer manages relationships between objects and information related to these relationships, Service Provider computes different task and performs work, Coordinator delegates work to other artefacts by reacting to events, Controller is responsible for making decisions and directs the actions of other artefacts, Interfacer handles and transforms requests and information between different parts of a system.

III. Methodology

The study uses an exploratory and descriptive approach to observe and describe the gathered data. The aim of the study is to observe how role-stereotypes and anti-patterns evolve over time. To observe this phenomenon we formulated the research questions as follows:

- **RQ1a**: *How does the distribution of role-stereotypes change over time?*
- **RQ1b**: *How does the distribution of anti-patterns change over time?*

[1]https://github.com/bitcoin-wallet/bitcoin-wallet
[2]https://github.com/k9mail/k-9
[3]https://sourceforge.net/projects/sweethome3d/

TABLE I
Confusion matrix of classifier predictions

Actual stereotype \ Predicted stereotype	IH	ST	SP	CT	CO	IT
IH	**463**	13	47	2	2	14
ST	21	**24**	41	1	0	13
SP	35	10	**448**	2	6	38
CT	5	1	19	**29**	1	8
CO	17	2	38	4	**27**	14
IT	10	3	42	1	2	**144**

- **RQ2**: *How does the occurrence of anti-patterns in specific role-stereotypes change over time?*
- **RQ3**: *Which role-stereotypes are more prone to change roles over time, and to which roles do they change?*

The following steps describes the process of answering the stated research questions.

A. Selecting a relevant number of versions for each of the three projects

Firstly, data was gathered from multiple versions of each of the projects. For each project, we used the version that had the ground truth established by Chaudron *et al.* [8] as a pivot and selected all versions after that and several versions before, each with an average interval of 3 months in between. The interval might slightly vary depending on if commits were available in the version-control system of the projects on the exact date based on the interval. We ended up with 36 versions for BitcoinWallet, since it had not existed long enough to provide additional versions based on the interval. To keep things comparable, we selected 37 versions for K9Mail and SweetHome3D.

Based on this selection strategy, the time period for the selected versions spans from the beginning of 2011 to the beginning of 2020. Specifically, it includes versions of Bitcoin Wallet starting from March 2011 to January 2020, K9 Mail starting from February 2011 to February 2020, and Sweet Home 3D starting from January 2011 to January 2020.

B. Answering RQ1a: How does the distribution of role-stereotypes change over time?

To gain data on the distribution of role-stereotypes in the different projects we used CRI (Class-role identifier), a tool that uses machine learning to classify role-stereotypes [10], [11], [12]. This tool has undergone several improvements and offers 74%-98% classification accuracy [10] (see Table I for stereotype-prediction results). Since these mentioned studies are highly relevant to the work in our study, and has gone through several iterations of improvements, we find it suited for use for our case. The tool was executed on all selected versions of the projects, resulting in 110 individual executions. CRI has a number of different classifier-models, and we used the ”rf-smote-three-cases-model-0202.sav” model to classify the data, which was trained on all three of the selected projects, and contains the largest training set of all the models.

TABLE II
ANTI-PATTERNS DETECTED BY PTIDEJ

Anti-pattern
AntiSingleton
BaseClassKnowsDerivedClass
BaseClassShouldBeAbstract
Blob
ClassDataShouldBePrivate
ComplexClass
FunctionalDecomposition
LargeClass
LazyClass
LongMethod
LongParameterList
ManyFieldAttributesButNotComplex
MessageChains
RefusedParentBequest
SpaghettiCode
SpeculativeGenerality
SwissArmyKnife
TraditionBreaker

The raw data produced by the CRI-tool was parsed using APCRM[4] (anti-pattern class-role mapper), which was developed to automate various tasks specifically for the purposes in our study. Based on the parsed data, we produced graphs that illustrates the frequency of the different roles over the time period of the selected versions of each project (see Figures 1, 2 and 3).

C. Answering RQ1b: How does the distribution of anti-patterns change over time?

To gain data of the anti-patterns in the systems, we used Ptidej [13]. Ptidej is a software tool that performs detection of design patterns and anti-patterns by using detection algorithms and source-code metrics. The tool offers up to 80% accuracy on average and a recall rate of 100% [14], [15]. Ptidej is executed from the Eclipse IDE, and requires that the target project is built as an Eclipse project with generated Java class files. Each selected version of the three projects underwent this procedure. Ptidej detects 18 different anti-patterns, as shown by Table II.

Similarly to RQ1a, the raw data produced by Ptidej was parsed using APCRM, and allowed us to produce graphs to illustrate the frequency of the anti-patterns over the time period of the selected versions of each project (see Figures 7, 8 and 9).

D. Answering RQ2: How does the occurrence of anti-patterns in specific role-stereotypes change over time?

To answer RQ2, we used the APCRM tool to map which roles are involved in which anti-patterns. We developed an algorithm in APCRM that iterates through each class in a project, assigns a role-stereotype to that class based on the same data used in RQ1a, then assigns the related anti-patterns to that class based on the data from RQ1b. Finally, it loops through the classes and counts the anti-patterns assigned to the role of that class. 18 graphs were produced as a result, each illustrating the frequency of anti-patterns for a specific role-stereotype in each project (see Figures 10 - 27). These graphs can be studied to observe how anti-patterns occur in specific role-stereotypes throughout the evolution of the three projects.

[4]https://github.com/fabianfroding/apcrm

E. Answering RQ3: Which role-stereotypes are more prone to change roles over time, and to which roles do they change?

Since the CRI tool can assign a role to specific classes in a Java project, we were able to compare two versions of a project to check for changes in roles in each Java class. We developed an algorithm in the APCRM tool to iterate through all selected versions and automate the process of counting number of changes in roles in each class. Furthermore, APCRM also specifies which role a class changed from and to. As a result, the data produced by APCRM allowed us to create bubble charts (see Figures 28, 29 and 30) to highlight which role-stereotypes are more prone to change and be changed to over time, hence answer RQ3. The data, which consists of mappings between role-stereotypes and classes, generated by the CRI tool also allows us to produce timeline graphs of how each individual class changes roles throughout the whole time period (see Appendix B). These graphs may help us traces whether certain roles are falsely assigned to classes due to misclassification of the role-stereotypes detection tool. For example, some classes might transition between a pair of roles multiple times throughout the time period, thus causes suspicion toward the detection tool. The timeline graphs also provides information on how many roles a class has changed to, and how many role changes a class has gone through, as illustrated in Tables III, IV and V.

F. Cleaning the data

Roles and anti-patterns that were non-existent throughout the entirety of the selected periods were removed from the figures. As such, figures belonging to the same research question may have different roles and anti-patterns present.

Test classes such as JUnit-classes were removed. We choose to remove test classes since they could create a false representation of the projects' total roles and anti-patterns and their relation to other roles and anti-patterns. We manually identified and removed those classes from each individual version of the three projects. This removal was done on the CSV-files produced by the classification of the CRI-tool, by using the "Find & Select"-functionality in Microsoft Excel. We manually confirmed that each row containing the "test"-keyword actually belonged in a test-directory.

One of the selected versions of Bitcoin Wallet, specifically the version from 2011-10-03, was removed. This version included some sub-directories in the source-code in "src\com\google\bitcoin" called "bouncycastle", "core", "discovery", and "store". These directories contained a large number of added Java-classes, and were not included in any of the other selected versions of Bitcoin Wallet. This caused a massive spike at that time-point in the graphs related to

Bitcoin Wallet and increased the scale of the graphs to the degree that the rest of the data became insignificant. We choose to remove the data from this version for the sake of readability. The removal does not affect the overall picture of the history of the Bitcoin Wallet project. This concerns Figures 1, 7, 10, 13, 16, 19, 22 and 25 in section IV, as can be seen by the missing dots in the third "column" in beginning of those figures. The high frequency of Information Holders and Service Providers in the beginning (2012-01-01 to 2012-10-02) of Figure 1, and the high frequency of various anti-patterns in the beginning (2012-01-01 to 2012-10-02) of Figures 7, 10, 16, are likely to be remnants of the removed version, and should be interpreted as such. For the original graphs including the data in the version from 2011-10-03, see Figures 31, 33, 34, 35, 36, 37, and 38 in the Appendix.

SweetHome3D had a separate sub-project included in the project directory. This separate project is called FurnitureLibraryEditor and was removed from our data collection, since we choose to focus on the main application. In addition, another separate sub-project called TextureLibraryEditor was introduced in the later versions of the selected versions, and was also removed for the same reason as the FurnitureLibraryEditor, but also because it would not have made any sense to suddenly include a separate sub-project halfway trough the selection of the versions.

All the raw data extracted from the tools can be found in the Github repository[5] of the APCRM tool.

IV. Results

RQ1a: *How does the distribution of role-stereotypes change over time?*

1) Bitcoin Wallet: In Bitcoin Wallet, the most common role is Service Provider (see Figure 1). Information Holder and Interfacer are also very common, both growing steadily throughout the time-period. The number of Information Holders increase drastically at the beginning of 2018 (2018-04-01), and causes the distribution of roles to make a noticeable change, making Information Holder the most common role instead of Service Provider.

2) K9 Mail: In K9 Mail, the most common role is Service Provider (see Figure 2). Information Holder is also very common. All roles except for Controller make a noticeable increase in the beginning of 2013 (2013-02-02) and then decreases again halfway through 2014 (2014-05-02).

3) Sweet Home 3D: In Sweet Home 3D, Information Holder is the most common role (see Figure 3). Service Provider is also very common. The distribution of the roles in Sweet Home 3D remain mostly unchanged throughout the time-period.

The three projects share some characteristics but also have some significant differences in terms of the distribution of roles over time. Bitcoin Wallet and K9Mail both have Service Provider as the most common role (Figures 1, 2), while Sweet Home 3D has Information Holder as the most common role (Figure 3). In K9 Mail and Sweet Home 3D the distribution of roles is mostly unchanging throughout the entire time period, while the roles in Bitcoin Wallet seems to fluctuate a bit. Bitcoin Wallet has an unusual high amount of Interfacers in relation to the other roles compared to K9 Mail and Sweet Home 3D.

All three projects have a somewhat low amount of Structurers, Controllers and Coordinators in relation to the other roles. In Sweet Home 3D, the Coordinator is the least common role, which is not the case in Bitcoin Wallet and K9 Mail that instead has the Controller as the least common role.

RQ1b: *How does the distribution of anti-patterns change over time?*

4) Bitcoin Wallet: In Bitcoin Wallet, the most common anti-patterns are *Complex Class* and *Long Parameter List* (Figure 7). At 2018-07-06, there is an observable increase of *Complex Class* and *Class Data Shield Be Private*, and a decrease of *Long Parameter List*. The high peak of *Complex Class* and *Long Method* from 2012-01-01 to 2012-10-02, as mentioned in section III, are likely remnants of the removed divergence that occurred on 2011-10-03, and should be interpreted as such.

5) K9 Mail: In K9 Mail, *Complex Class* is the most common anti-pattern, and has a steady increase during the entire time-period (Figure 8). *Long Method* starts occurring at 2014-08-06 and continues to increase until 2018-11-03, where is starts decreasing again. *Long Parameter List* has a significant high peak from 2015-05-02 to 2016-02-02. *Lazy Class* has an observable decrease at 2017-11-22 to 2018-02-01, suddenly going from around 50 occurrences to only 4 for those two time-points.

6) Sweet Home 3D: In Sweet Home 3D, the most common anti-pattern is *Blob* (Figure 9). *Complex Class* and *Long Method* are the second two most common anti-patterns, and *Long Parameter List* being fourth most common and *Lazy Class* being fifth most common. None of the occurring anti-patterns in Sweet Home 3D make any drastic changes or has unusual abnormalities.

The three projects are observed to have significant differences in terms of the distribution of anti-patterns and how they change over time. Bitcoin Wallet and K9Mail both have *Complex Class* as the most common anti-pattern (Figures 7, 8), whereas Sweet Home 3D has *Blob* as the most common anti-pattern (Figure 9). Compared to Bitcoin Wallet and K9Mail, Sweet Home 3D has an unusually high amount of *Blob* in relation to the other anti-patterns. In Sweet Home 3D the distribution of anti-patterns remains stable within the time period, with only some minor changes. Specifically, the anti-pattern *Long Method* begins to surpass *Complex Class* as of 2014-04-01, but the degrees of change of the two anti-patterns throughout the whole time period are inconsiderable. The only notable change in Sweet Home 3D is the sharp increase of the *Blob* at 2013-04-01, which greatly contributes to the distribution shift towards the end of the time period.

[5]https://github.com/fabianfroding/apcrm/tree/master/Resources/raw-data

In contrast, Bitcoin Wallet and K9Mail have more dramatic degrees of change in certain anti-patterns. Respectively, the number of *Long Parameter List* in Bitcoin Wallet decreases drastically as of 2018-07-06, while more *Complex Class* begins to emerge, becoming the most occurring anti-pattern in the system. At the same time, a considerable amount of *Class Data Should Be Private* also starts to appear.

In K9Mail, however, *Complex Class* is consistently the leading anti-pattern, which is observed to also have the most rapid growth. Other remarkable increases are seen in *Long Method*, *Lazy Class* and *Long Parameter List*. There are sudden appearances and disappearances in large amounts of certain anti-patterns, namely the rise in the *Long Parameter List* at 2015-05-02 and the drop in the *Lazy Class* at 2017-11-22. This could be an issue of inaccurate classification in the anti-patterns detection tool, which is discussed further in section VI.

RQ2: *How does the occurrence of anti-patterns in specific role-stereotypes change over time?*

The Information Holder, Service Provider, and Controller-roles in Bitcoin Wallet has peaks of several anti-patterns in the period from January 2012 to October 2012. These are very likely remnants of the removed divergence from 2011-10-03 mentioned in section III, and should be interpreted as such.

A. Anti-patterns in Information Holder

1) Bitcoin Wallet: The Information Holder-role in Bitcoin Wallet has 10 occurring anti-patterns, with *Complex Class* and *Long Parameter List* being more common than the others (Figure 10). The *Class Data Should Be Private* has an observable increase at 2018-07-06 and 2019-01-18.

2) K9Mail: The Information Holder-role in K9Mail has a total of 13 occurring anti-patterns, with *Class Data Should Be Private*, *Complex Class*, *Long Method*, and *Long Parameter List* being more common than the others and also grows quite significantly as time passes (Figure 11). Furthermore, the mentioned anti-patterns has a noticeable fluctuation. The *Long Method* also drop from 15 down to just one occurrence at 2015-08-03 and 2015-11-06, and then back up again.

3) Sweet Home 3D: In SweetHome3D, the Information Holder has seven occurring anti-patterns. *Blob* is the most common, and has a high fluctuation, peaking at 8 occurrences at 2018-04-02 and 2019-01-08, and 0 occurrences at 2016-10-01 (see Figure 12). *Long Parameter List* and *Lazy Class* are also relatively high compared to the rest of the anti-patterns.

There are several anti-patterns that occurs in the Information Holder role in all three projects; *Base Class Should Be Abstract*, *Blob*, *Complex Class*, *Long Parameter List*, and *Refused Parent Bequest*. In all three projects, the Information Holder has high occurrences of *Complex Class* and *Long Parameter List*. The Information Holder in K9 Mail seems to have more anti-patterns both in frequency and types compared to Bitcoin Wallet and Sweet Home 3D.

B. Anti-patterns in Structurer

1) Bitcoin Wallet: The Structurer-role in Bitcoin Wallet only has two occurring anti-patterns; *Complex Class*, and *Long Parameter List* (Figure 13). Both anti-patterns start occurring around the mid of 2013 and continue to grow as time passes.

2) K9Mail: The Structurer-role in K9Mail has 11 occurring anti-patterns, with *Complex Class*, *Speculative Generality*, and *Long Method* being the most common (Figure 14). *Complex Class* has a quite fluctuating growth as time passes, and *Long Method* start occurring at 2014-08-06 and grows rapidly from there. *Speculative Generality* remains steady but increases greatly at 2013-08-01 and then back down at 2014-02-08, along with most of the other present anti-patterns.

3) Sweet Home 3D: The Structurer role in Sweet Home 3D has nine occurring anti-patterns, with *Complex Class* and *Long Method* being the most common (see Figure 15). *Base Class Should Be Abstract* and *Blob* is also quite common. The *Blob* is fluctuating greatly, varying from 5 to 0 occurrences, similarly to the behaviour of *Blob* in Information Holder in Sweet Home 3D.

The Structurer has two anti-patterns that occur in all of the projects; *Complex Class* and *Long Parameter List*. *Complex Class* grows moderately as time passes in Bitcoin Wallet, grows significantly and in a fluctuating manner in K9 Mail, while remaining quite the same in Sweet Home 3d. The *Long Parameter List* also behaves differently depending on the project; in Bitcoin Wallet, it moderately grows as time goes on, in K9 Mail it stays relatively low, and in Sweet Home 3D it increases in the beginning and stays the same for the rest of the time.

C. Anti-patterns in Service Provider

1) Bitcoin Wallet: The Service Provider-role in Bitcoin Wallet has 11 occurring anti-patterns, with *Long Parameter List* being significantly more common than the others (Figure 16). *Complex Class* is also quite more common the than other anti-patterns. From 2018-07-06, the *Long Parameter List* decreases significantly to the same frequency as the other anti-patterns.

2) K9Mail: The Service Provider-role in K9Mail has 12 occurring anti-patterns, with *Lazy Class* and *Complex Class* being significantly more common than the others (Figure 17). *Lazy Class* starts fluctuating greatly in the second half of the figure, ranging from 40 to just one occurrence. At 2017-02-03 and 2018-05-01, the *Lazy Class* has a sudden dip of more than 40 but quickly goes back up. *Complex Class* and *Lazy Class* start growing more notably in the second half of the figure.

3) Sweet Home 3D: Service Provider in Sweet Home 3D has eight occurring anti-patterns, with *Class Data Should Be Private*, *Complex Class*, *Lazy Class* and *Long Parameter List* being more common the the rest of the anti-patterns (see Figure 18). *Blob* is also present, having similar fluctuation as in the previous roles in Sweet Home 3D.

The Service Provider role has a number of anti-patterns that occurs in all three of the projects; *Base Class Should Be Abstract*, *Blob*, *Class Data Should Be Private*, *Complex Class*, *Lazy Class*, *Long Method*, *Long Parameter List*, and *Refused Parent Bequest*.

Blob stays relatively uncommon in Bitcoin Wallet and K9 Mail, while fluctuating greatly in Sweet Home 3D. Also, *Class Data Should Be Private* have low frequencies in Bitcoin Wallet and K9 Mail, while having a moderate frequency in Sweet Home 3D. *Complex Class* grows notably in K9 Mail as time passes, stays moderately high in Sweet Home 3D, and very low in Bitcoin Wallet. *Lazy Class* is the most common anti-pattern in the Service Provider in Sweet Home 3D. *Lazy Class* also rapidly increases in frequency about halfway through the versions in K9, and stays high except for a few massive individual drops (these are described in section IV.

D. Anti-patterns in Controller

1) Bitcoin Wallet: The Controller-role in Bitcoin Wallet has five occurring anti-patterns with very low frequency, with *Long Parameter List* being more common than the others (Figure 19).

2) K9Mail: The Controller-role in K9Mail has six occurring anti-patterns, with *Complex Class* being more common than the others (Figure 20). *Complex Class* grows until 2017-08-05, where it starts to decrease, dropping to just one occurrence at the last selected version.

3) Sweet Home 3D: The Controller role in Sweet Home 3D has seven occurring anti-patterns. *Complex Class*, *Blob*, *Long Parameter list*, and*Long method* are the most common (see Figure 21). All anti-patterns has a somewhat steady frequency compared to *Blob*, which, like the other *Blob*-occurrences in the other roles in Sweet Home 3D, fluctuates greatly.

The Controller role has the following anti-patterns occurring in all three projects; *Class Data Should Be Private*, *Complex Class*, *Long Method*, and *Long Parameter List*. The Controller in Bitcoin Wallet has very few anti-patterns in general (see Figure 19), most likely due to the role's low frequency of anti-patterns (see Figure 1). The *Complex Class* has a high frequency in K9 Mail and Sweet Home 3D, and seems to first grow as time passes and then decrease in K9 Mail (see Figure 20), while remaining somewhat the same throughout Sweet Home 3D (see Figure 21). The *Blob* and *Long Parameter List* occurs frequently in both K9 Mail and Sweet Home 3D. The *Blob* in Sweet Home 3D fluctuates greatly, ranging from 1 occurrence to up to 18 at other time-points.

E. Anti-patterns in Coordinator

1) Bitcoin Wallet: The Coordinator-role in Bitcoin Wallet only has two occurring anti-patterns at a very low frequency; *Complex* and *Long Parameter List* (Figure 22).

2) K9Mail: The Coordinator-role in K9Mail has nine occurring anti-patterns, with *Complex Class* being more common than the others and grows in a fluctuating manner (Figure 23). *Base Class Should Be Abstract* is also quite common, and starts growing more notably at 2015-11-06.

3) Sweet Home 3D: In Sweet Home 3D, the Coordinator role has six occurring anti-patterns, with *Blob* and *Long Method* being the most common (see Figure 24). Most of the occurring anti-patterns seems to fluctuate between 0-2 occurrences, while only *Blob* going up to five at certain points. None of them seem to have anything similar to a linear growth.

The Coordinator role only has two anti-patterns that occur in all three projects, which is *Complex Class* and *Long Parameter List*. However, the Coordinator in Bitcoin Wallet has a very low number and frequencies of anti-patterns in general and does not contain any noteworthy observations. The Coordinator in K9Mail has a fluctuating increase of *Complex Class* as time passes. The anti-patterns in the Coordinators in Sweet Home 3D fluctuates greatly, especially *Blob*, which is similar to the behaviour of *Blob* in the other roles in Sweet Home 3D.

F. Anti-patterns in Interfacer

1) Bitcoin Wallet: The Interfacer-role in Bitcoin Wallet has six occurring anti-patterns, with *Complex Class* and *Long Parameter List* being significantly more common than the others (Figure 25). At 2018-07-06 there is a drastic increase of *Complex Class* and a drastic decrease of *Long Parameter List*.

2) K9Mail: The Interfacer-role in K9Mail has 10 occurring anti-patterns, with *Complex Class* and being significantly more common than the others (Figure 26).

3) Sweet Home 3D: In Sweet Home 3D, the Interfacer role has eight occurring anti-patterns. *Long Method* distinguishes itself from the rest by having a much higher frequency (see Figure 27). All of the anti-patterns seems to remain steady with no notable changes or fluctuations.

The Interfacer role has the following anti-patterns in common in the three projects; *Base Class Should Be Abstract*, *Blob*, *Class Data Should Be Private*, *Complex Class*, *Long Parameter List*, and *Refused Parent Bequest*. Even though *Blob* occurs in all three projects, it's frequency is not particularly high in any of them. *Complex Class* increases in Bitcoin Wallet as time passes, has a steady frequency and than peaks and then decreases again back to it's previous frequency in K9 Mail, and stays somewhat the same in Sweet Home 3D. *Long Parameter List* rapidly stats occurring close to the beginning of Bitcoin Wallet, then becomes quite low towards the end, while fluctuating in a low frequency in K9 Mail, and stays very low in Sweet Home 3D.

RQ3: *Which role-stereotypes are more prone to change roles over time, and to which roles do they change?*

In Bitcoin Wallet, there have been 85 instances of classes changing roles throughout all selected versions (see Figure 28). Most changes are seen in Service Provider and Interfacer. Specifically, there are a total of 29 instances of classes switching from Service Provider to other roles and a total of 23 instances of classes switching from other roles to Service Provider. Furthermore, 20 classes changed from Interfacer to other roles, and conversely, 21 classes changed from other roles to Interfacer.

K9Mail underwent substantially more changes in role-stereotypes, concretely at 137 instances (see Figure 29). A majority amount of changes are from Service Provider, Coordinator and Interfacer to other roles, with 38, 22 and 36

instances respectively. Moreover, a majority of classes changes from other roles to Service Provider and Interfacer, with 41 and 28 instances respectively.

Sweet Home 3D, however, has only 62 occurrences of classes changing roles throughout the time period, with 14 instances derive from classes switching from Service Provider to other roles (see Figure 30). Additionally, 20 classes changed from other roles to Service Provider.

The number of classes that have changed role-stereotypes more than once are 17, 19 and 7 in Bitcoin Wallet, K9 Mail and Sweet Home 3D respectively (see Tables III, IV, V). The largest number of role changes is four, which can be seen in Bitcoin Wallet and K9 Mail. Sweet Home 3D, however, only consists of classes that have changed roles at most twice throughout the whole time period. Bitcoin Wallet contains a single class that have been assigned four different role-stereotypes, which is the largest number of roles a class was assigned in all three projects.

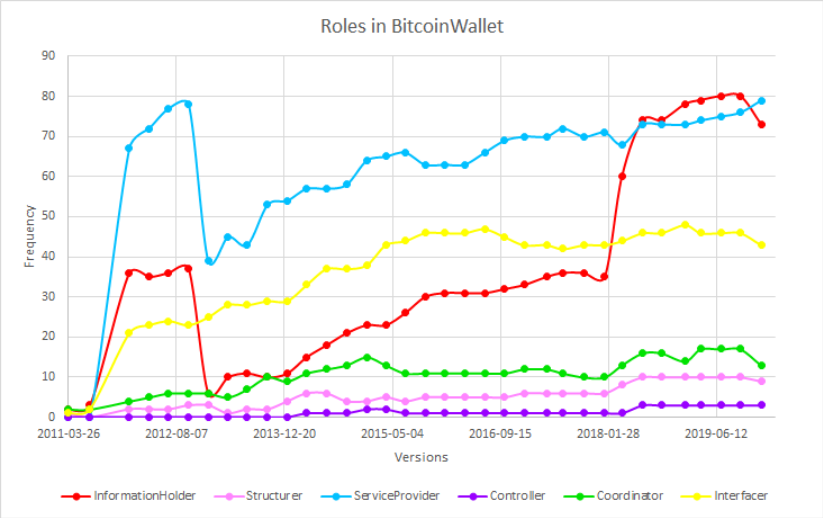

Fig. 1. Distribution of the roles in each version of BitcoinWallet

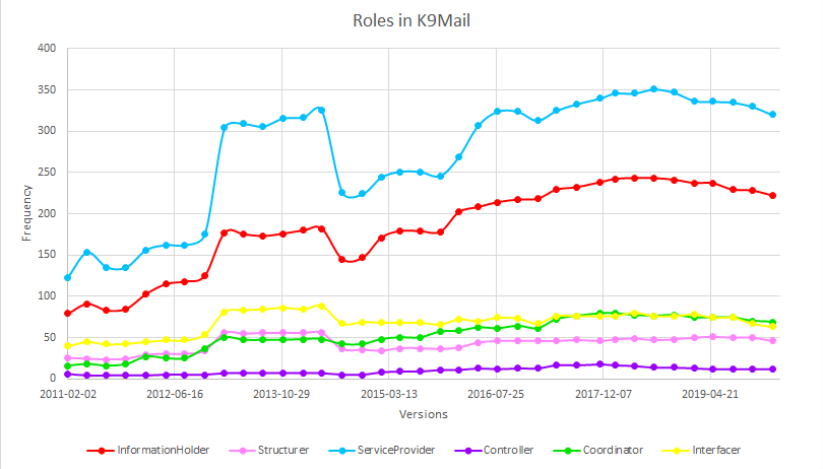

Fig. 2. Distribution of the roles in each version of K9Mail

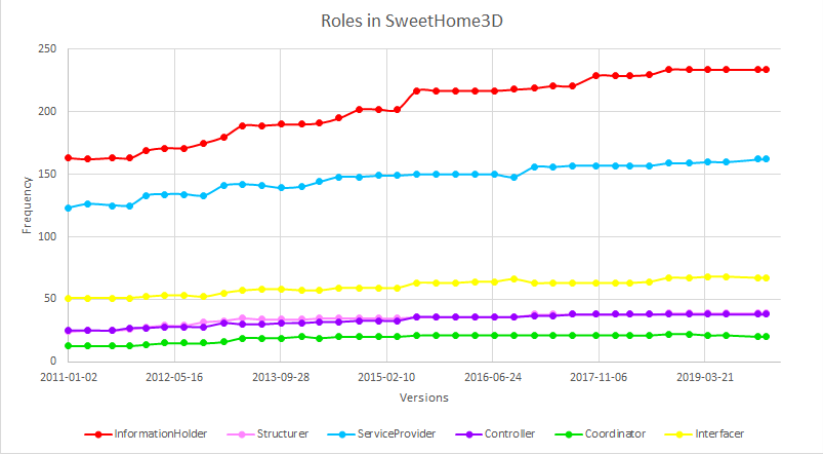

Fig. 3. Distribution of the roles in each version of SweetHome3D

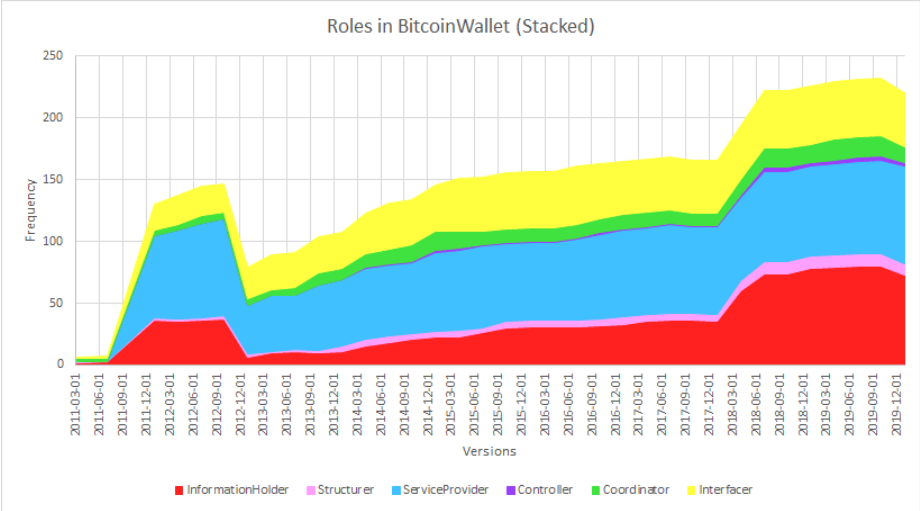

Fig. 4. Distribution of the roles in each version of BitcoinWallet (Stacked)

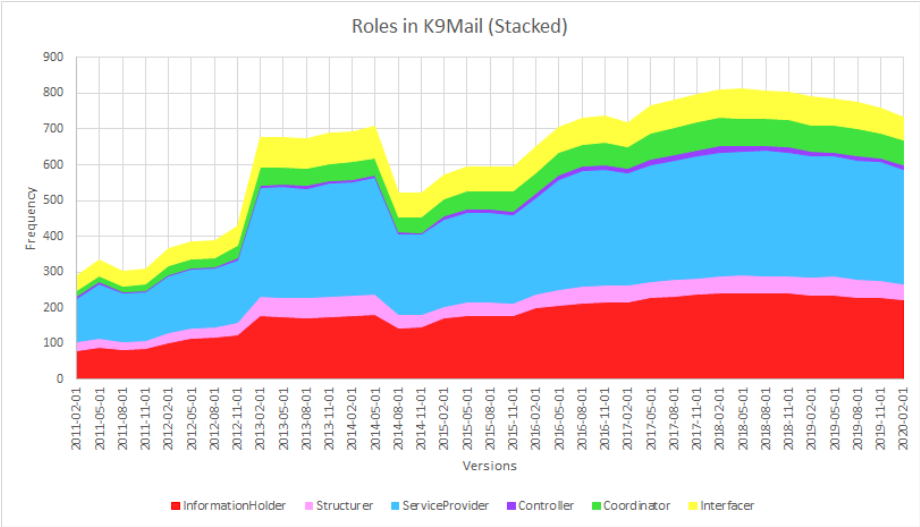

Fig. 5. Distribution of the roles in each version of K9Mail (Stacked)

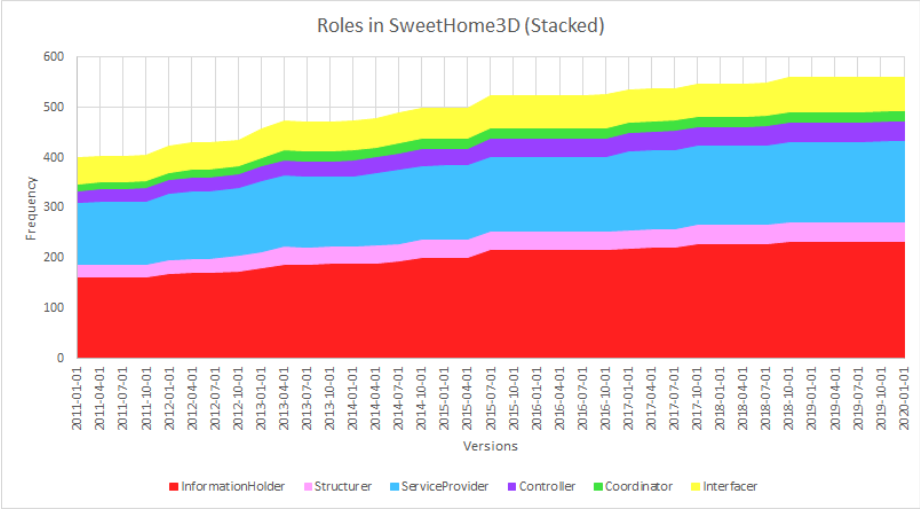

Fig. 6. Distribution of the roles in each version of SweetHome3D (Stacked)

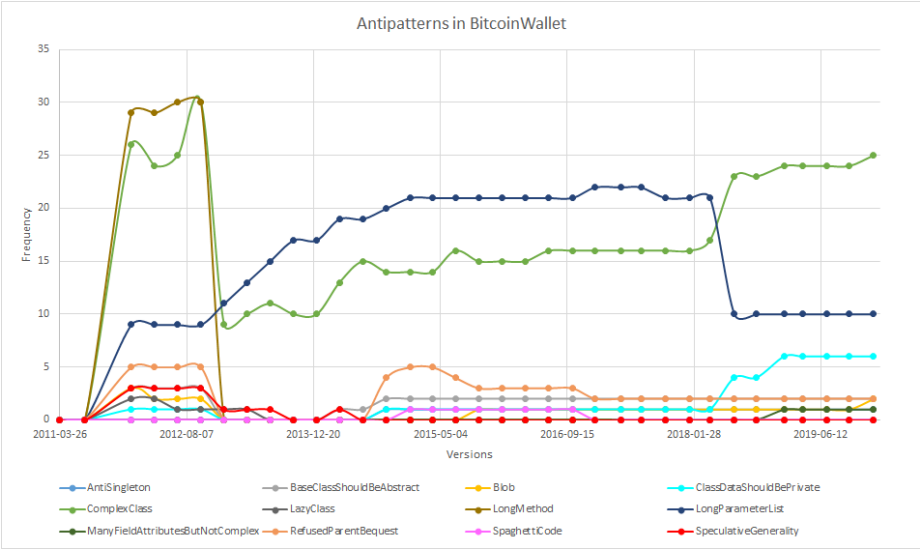

Fig. 7. Distribution of the anti-patterns in each version of BitcoinWallet

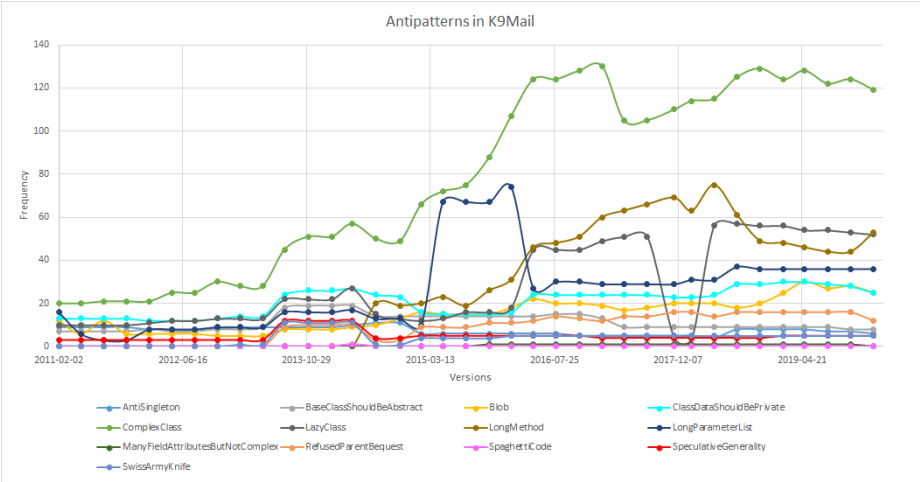

Fig. 8. Distribution of the anti-patterns in each version of K9Mail

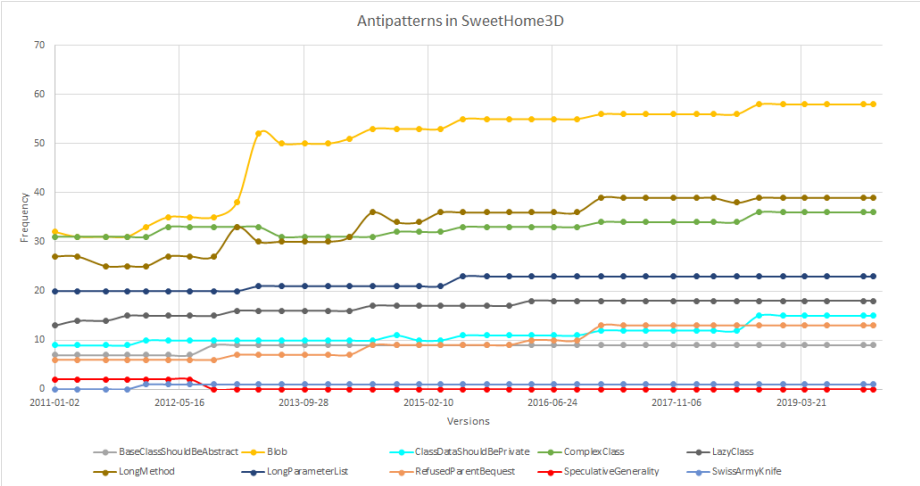

Fig. 9. Distribution of the anti-patterns in each version of SweetHome3D

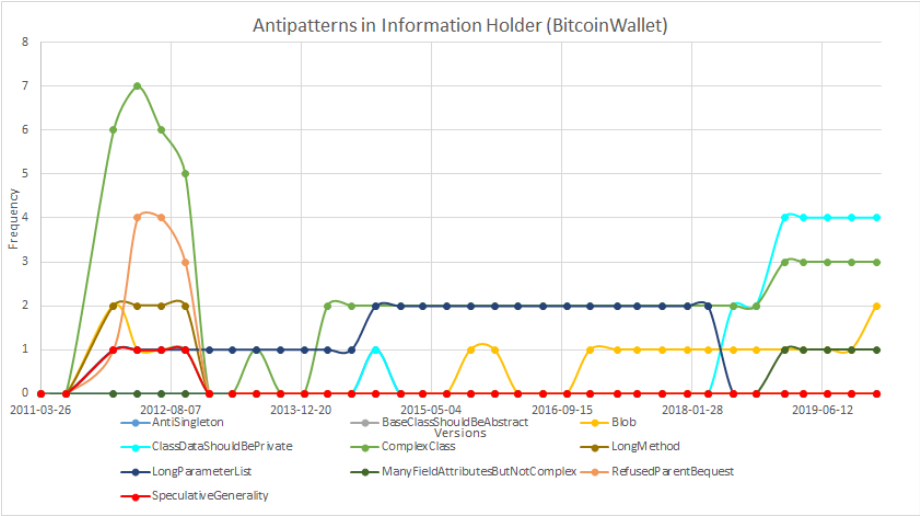

Fig. 10. Anti-patterns of Information Holder in BitcoinWallet

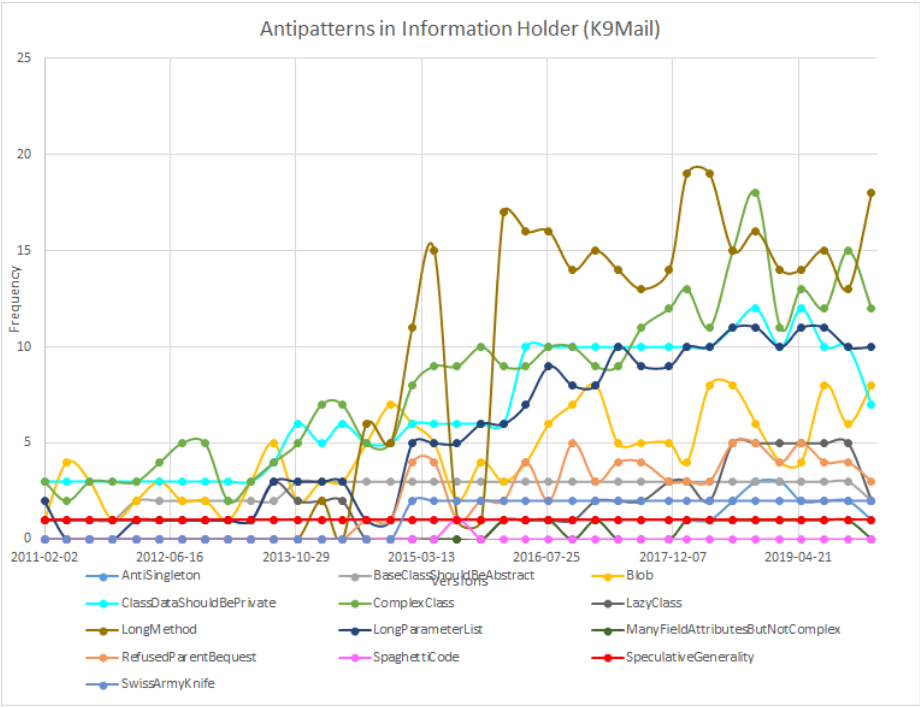

Fig. 11. Anti-patterns of Information Holder in K9Mail

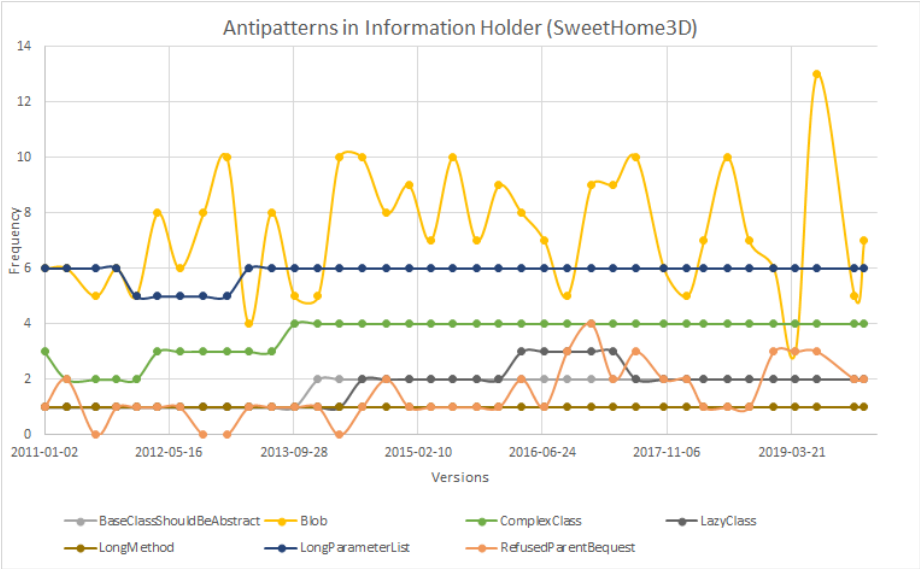

Fig. 12. Anti-patterns of Information Holder in SweetHome3D

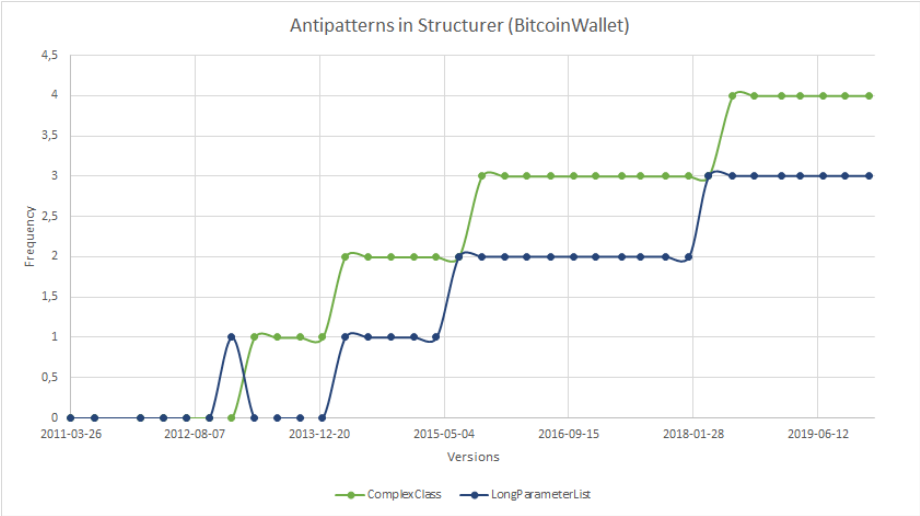

Fig. 13. Anti-patterns of Structurer in BitcoinWallet

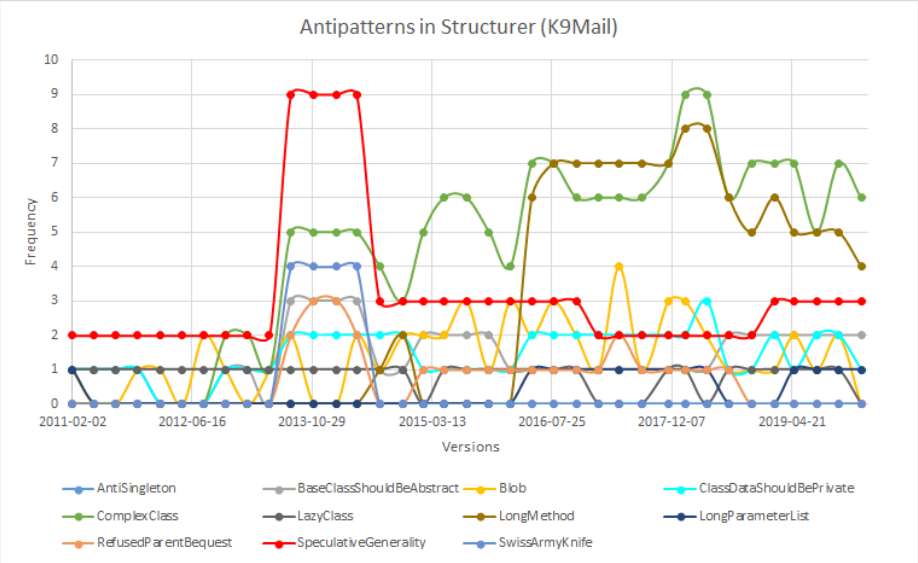

Fig. 14. Anti-patterns of Structurer in K9Mail

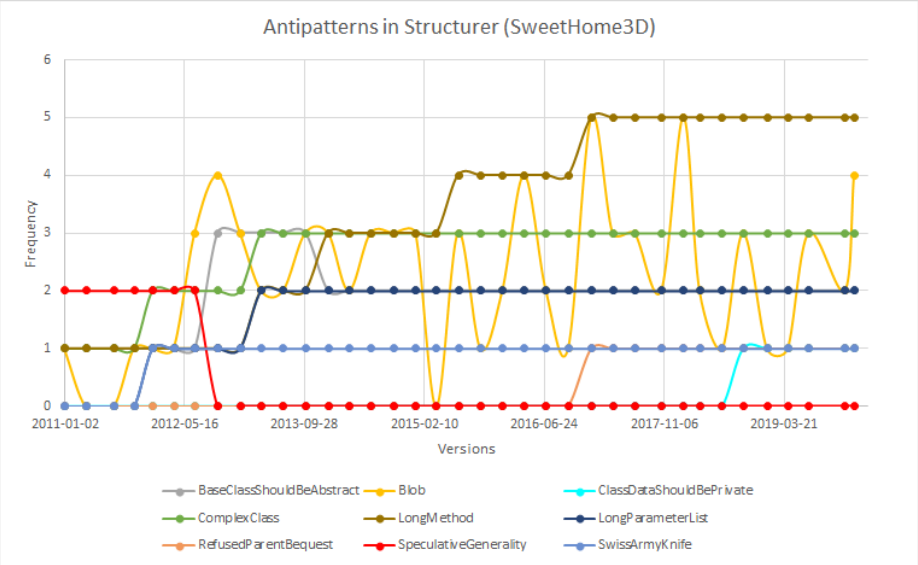

Fig. 15. Anti-patterns of Structurer in SweetHome3D

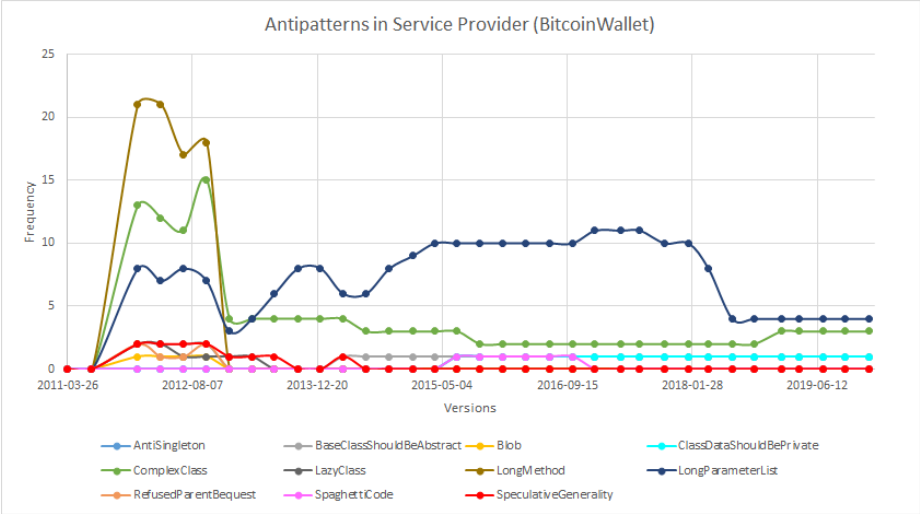

Fig. 16. Anti-patterns of Service Provider in BitcoinWallet

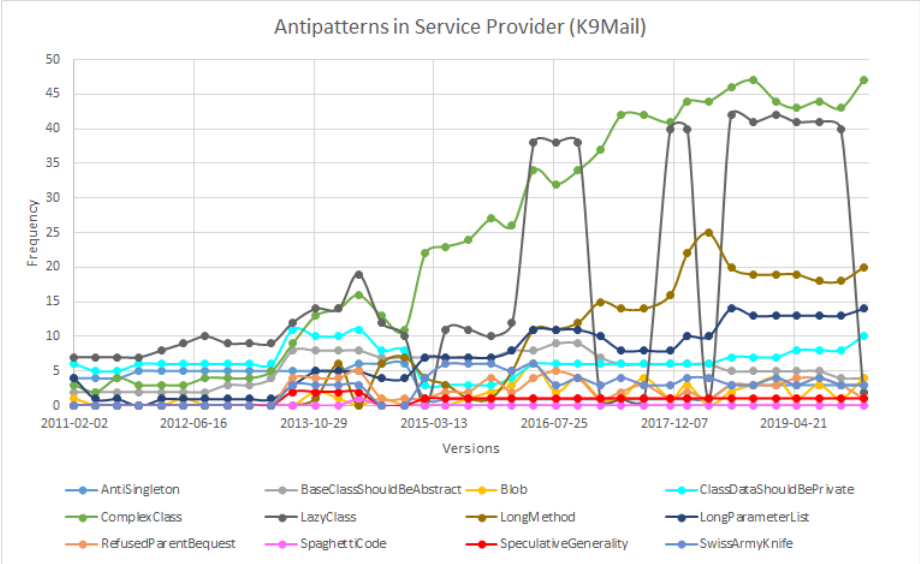

Fig. 17. Anti-patterns of Service Provider in K9Mail

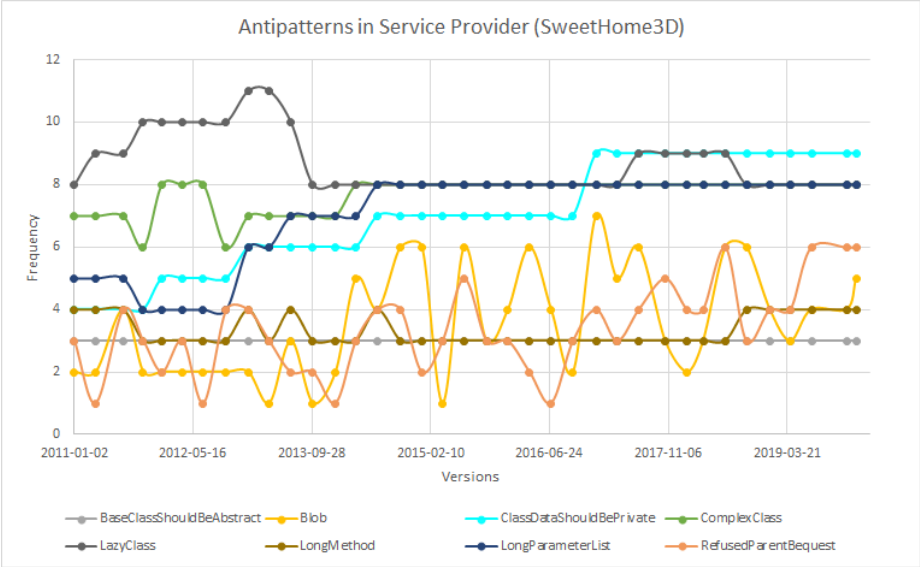

Fig. 18. Anti-patterns of Service Provider in SweetHome3D

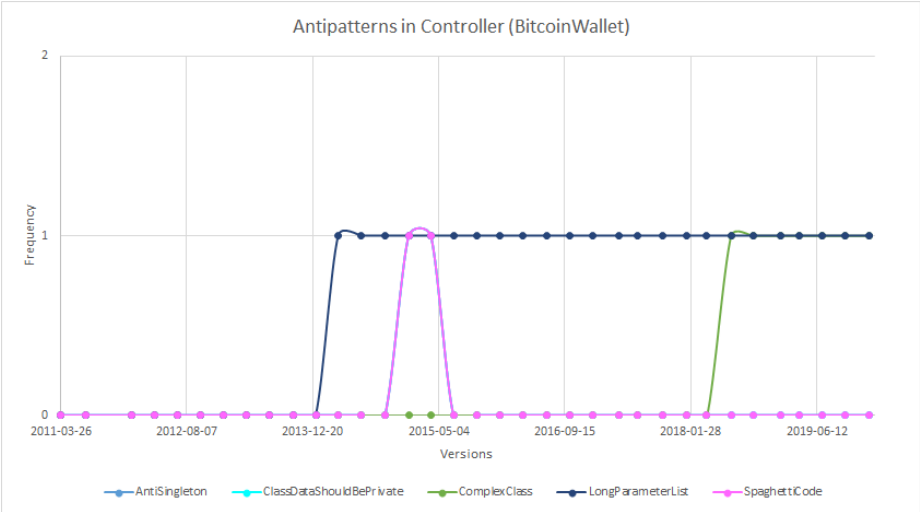

Fig. 19. Anti-patterns of Controller in BitcoinWallet

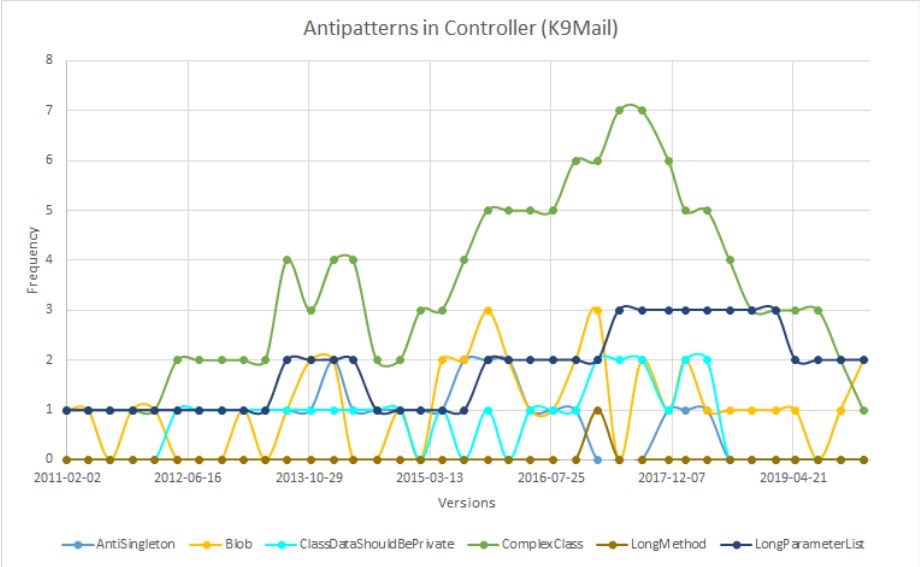

Fig. 20. Anti-patterns of Controller in K9Mail

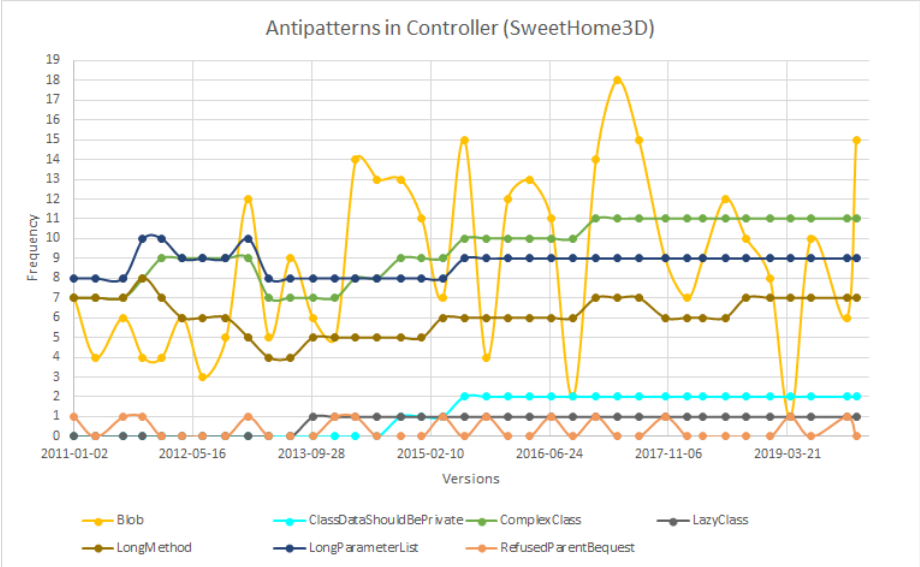

Fig. 21. Anti-patterns of Controller in SweetHome3D

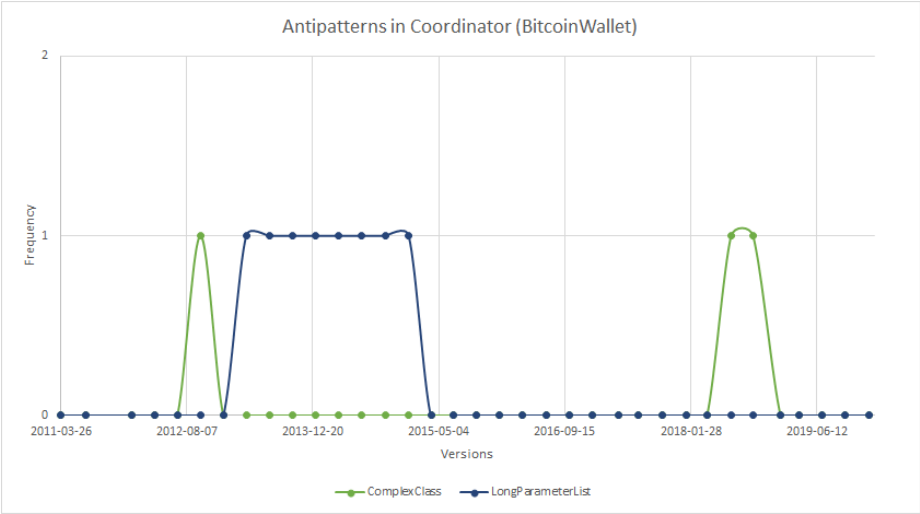

Fig. 22. Anti-patterns of Coordinator in BitcoinWallet

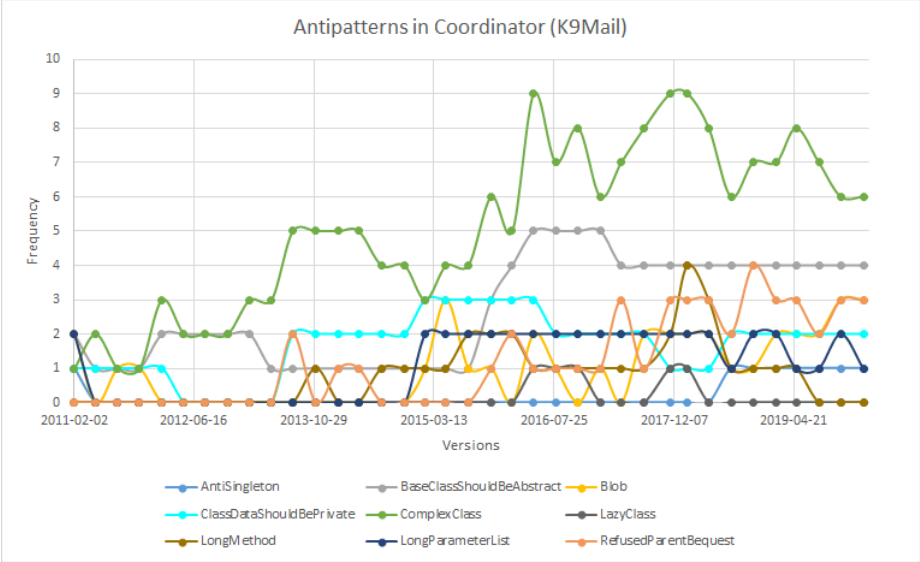

Fig. 23. Anti-patterns of Coordinator in K9Mail

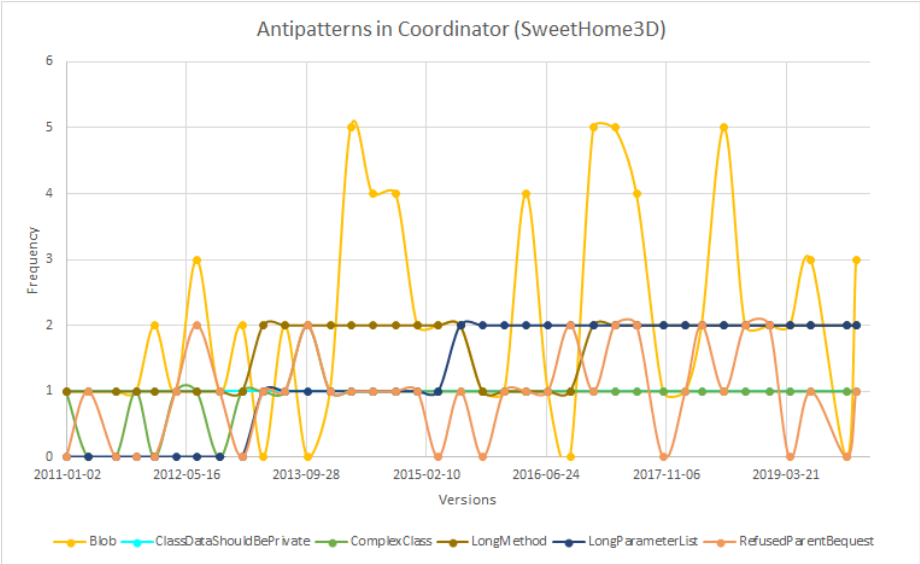

Fig. 24. Anti-patterns of Coordinator in SweetHome3D

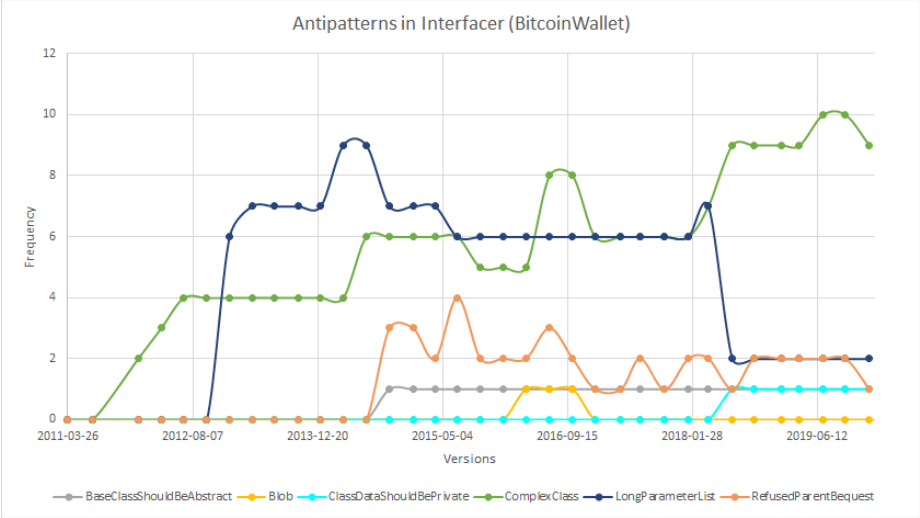

Fig. 25. Anti-patterns of Interfacer in BitcoinWallet

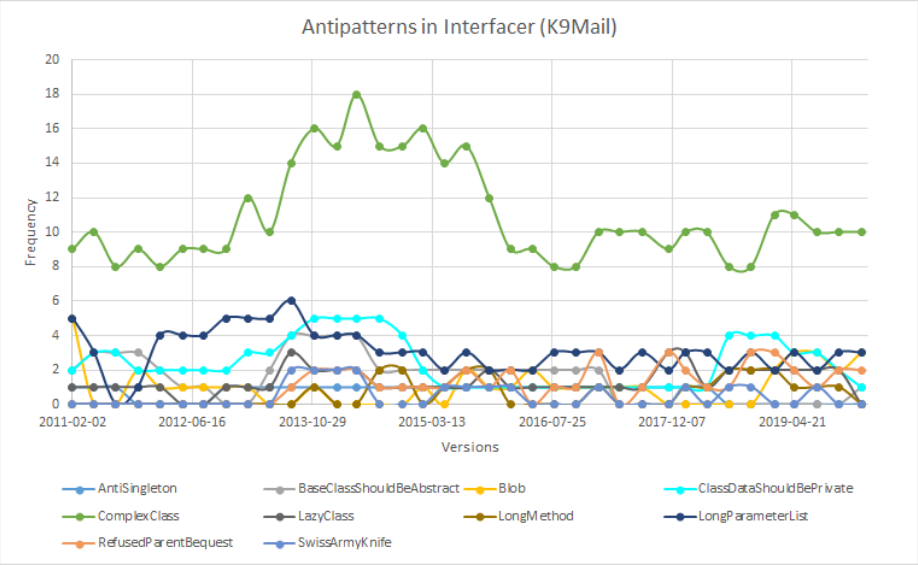

Fig. 26. Anti-patterns of Interfacer in K9Mail

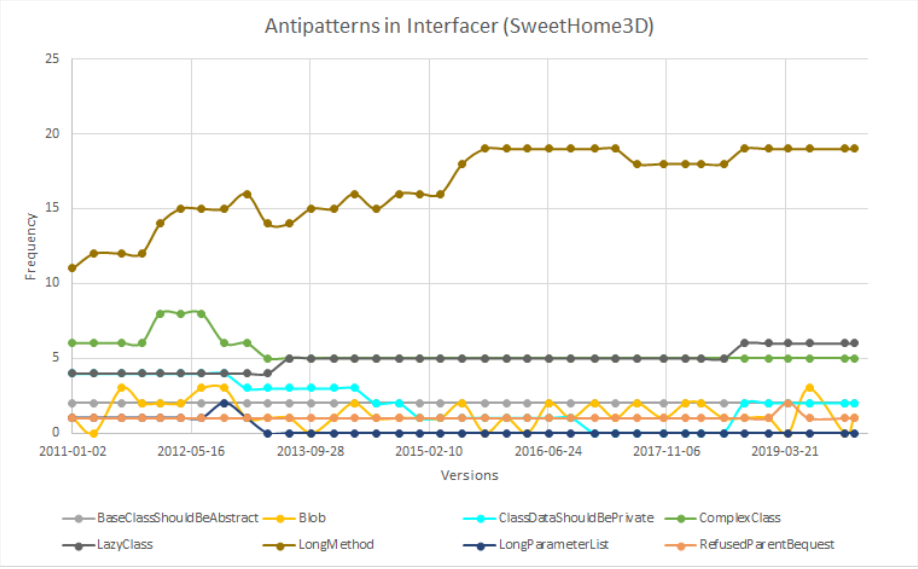

Fig. 27. Anti-patterns of Interfacer in SweetHome3D

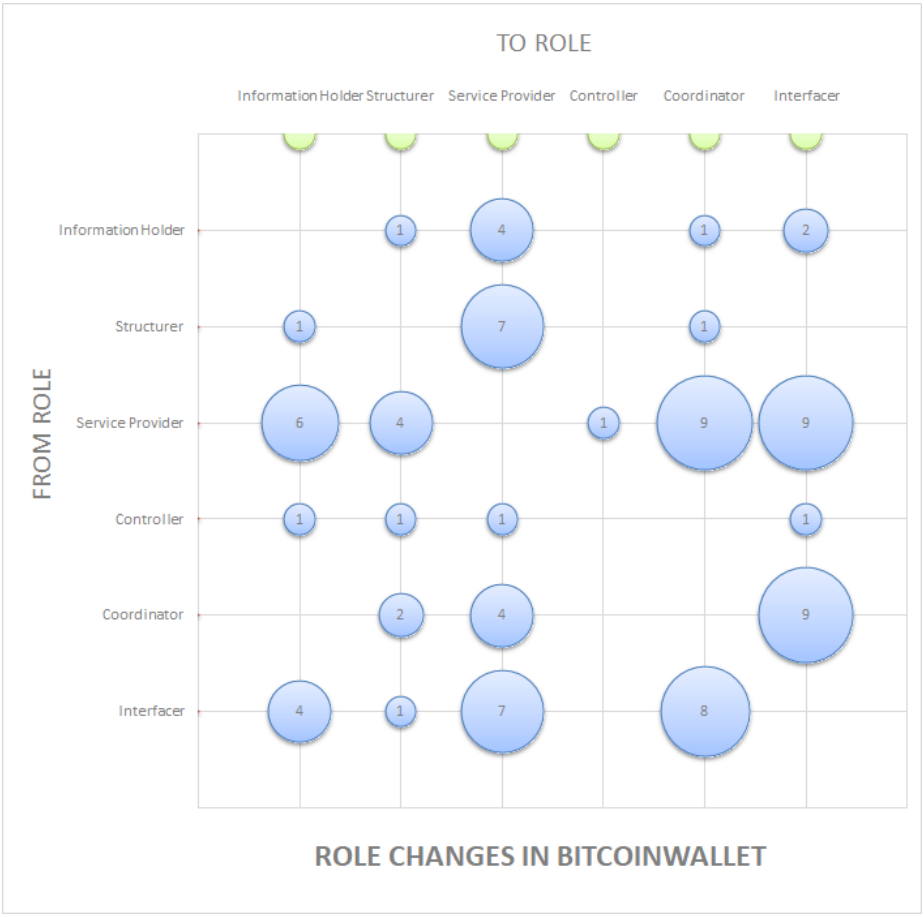

Fig. 28. Total changes in role-stereotypes in all selected versions of BitcoinWallet

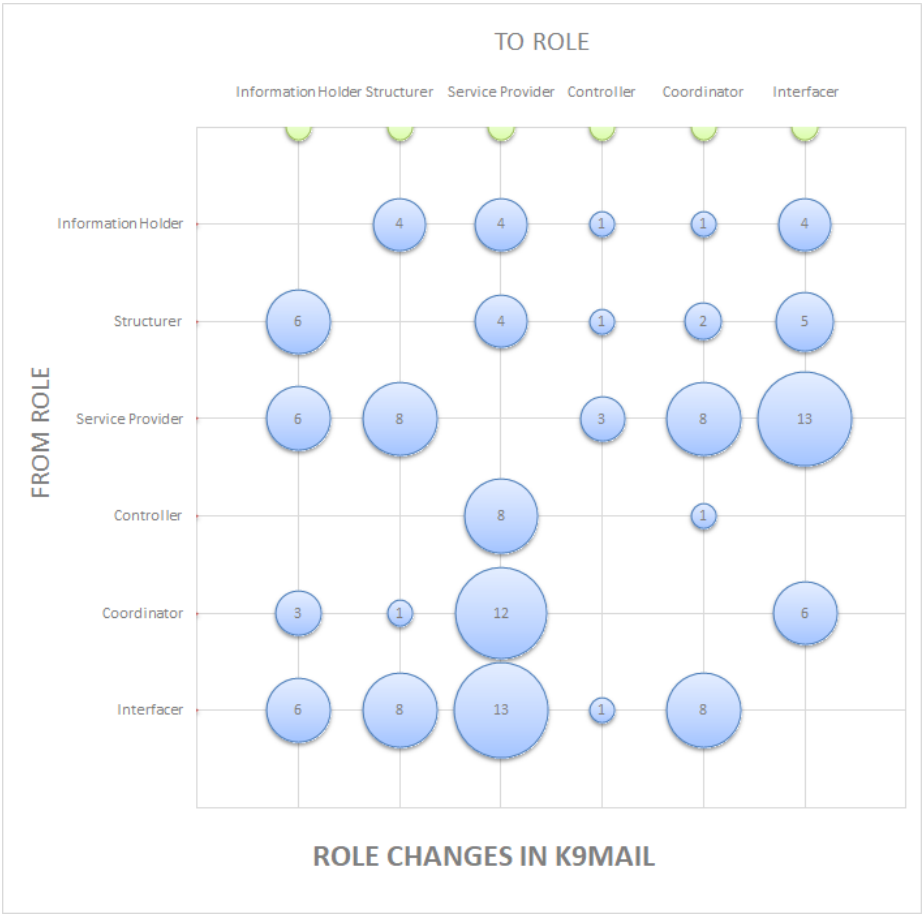

Fig. 29. Total changes in role-stereotypes in all selected versions of K9Mail

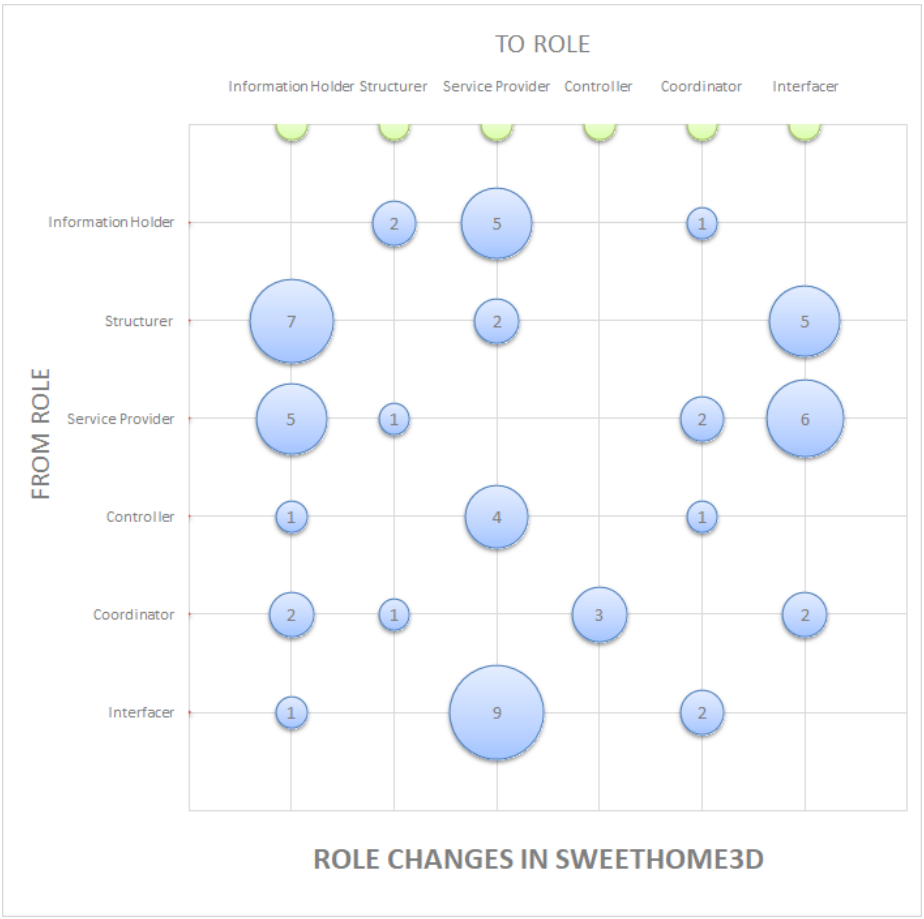

Fig. 30. Total changes in role-stereotypes in all selected versions of SweetHome3D

TABLE III
CLASSES THAT HAVE CHANGED ROLES MORE THAN ONCE IN BITCOIN WALLET.

Class	Number of roles	Number of role changes
\wallet\src\de\schildbach\wallet\ui\AbstractWalletActivity.java	4	4
\wallet\src\de\schildbach\wallet\camera\CameraManager.java	3	4
\wallet\src\de\schildbach\wallet\ui\PreferencesActivity.java	3	3
\wallet\src\de\schildbach\wallet\ui\WalletAddressesAdapter.java	2	3
\wallet\src\de\schildbach\wallet\WalletBalanceWidgetProvider.java	2	3
\wallet\src\de\schildbach\wallet\ui\AddressBookActivity.java	2	3
\wallet\src\de\schildbach\wallet\util\Formats.java	3	2
\wallet\src\de\schildbach\wallet\ui\MaybeMaintenanceFragment.java	3	2
\wallet\src\de\schildbach\wallet\util\WalletUtils.java	3	2
\wallet\src\de\schildbach\wallet\ui\CurrencyCalculatorLink.java	2	2
\wallet\src\de\schildbach\wallet\util\GenericUtils.java	2	2
\wallet\src\de\schildbach\wallet\service\BlockchainService.java	2	2
\wallet\src\de\schildbach\wallet\ui\BlockListFragment.BlockListAdapter.java	2	2
\wallet\src\de\schildbach\wallet\util\BitmapFragment.java	2	2
\wallet\src\de\schildbach\wallet\ui\RequestCoinsActivity.java	2	2
\wallet\src\de\schildbach\wallet\ui\send\RequestWalletBalanceTask.java	2	2
\wallet\src\de\schildbach\wallet\ui\PeerListFragment.ReverseDnsLoader.java	2	2

TABLE IV
CLASSES THAT HAVE CHANGED ROLES MORE THAN ONCE IN K9 MAIL.

Class	Number of roles	Number of role changes
\src\com\fsck\k9\activity\Accounts.java	3	4
\app\ui\src\main\java\com\fsck\k9\activity\FolderInfoHolder.java	3	4
\src\com\fsck\k9\activity\K9Activity.java	2	3
\app\ui\src\main\java\com\fsck\k9\activity\FolderList.FolderListAdapter.java	3	2
\src\com\fsck\k9\mail\store\WebDavStore.java	3	2
\src\com\fsck\k9\helper\Contacts.java	3	2
\k9mail\src\main\java\com\fsck\k9\activity\MessageReference.java	3	2
\src\com\fsck\k9\view\K9PullToRefreshListView.java	2	2
\src\com\fsck\k9\view\MessageTitleView.java	2	2
\k9mail-library\src\main\java\com\fsck\k9\mail\transport\WebDavTransport.java	2	2
\src\com\fsck\k9\mail\store\Pop3Store.Pop3Capabilities.java	2	2
\src\com\fsck\k9\mail\CertificateValidationException.java	2	2
\src\com\fsck\k9\activity\K9ListActivity.java	2	2
\src\com\fsck\k9\activity\FolderList.FolderListAdapter.java	2	2
\k9mail\src\main\java\com\fsck\k9\view\K9PullToRefreshListView.java	2	2
\k9mail\src\main\java\com\fsck\k9\ui\messageview\MessageCryptoPresenter.java	2	2
\k9mail\src\main\java\com\fsck\k9\activity\K9ActivityCommon.java	2	2
\src\com\fsck\k9\Account.java	2	2
\src\com\fsck\k9\EmailAddressAdapter.java	2	2

TABLE V
CLASSES THAT HAVE CHANGED ROLES MORE THAN ONCE IN SWEET HOME 3D.

Class	Number of roles	Number of role changes
\SweetHome3D\src\com\eteks\sweethome3d\tools\OperatingSystem.java	3	2
\SweetHome3D\src\com\eteks\sweethome3d\swing\ImportedFurnitureWizardStepsPanel.AbstractModelPreviewComponent.java	3	2
\SweetHome3D\src\com\eteks\sweethome3d\swing\ControllerAction.java	3	2
\SweetHome3D\src\com\eteks\sweethome3d\j3d\Ground3D.java	3	2
\SweetHome3D\src\com\eteks\sweethome3d\io\DefaultUserPreferences.java	3	2
\SweetHome3D\src\com\eteks\sweethome3d\io\DefaultHomeInputStream.HomeObjectInputStream.java	2	2
\SweetHome3D\src\com\eteks\sweethome3d\io\DefaultHomeOutputStream.HomeObjectOutputStream.java	2	2

V. Discussion

RQ1a: *How does the distribution of role-stereotypes change over time?*
In section IV we saw a high increase of Information Holders at the beginning of 2018 (2018-04-01) in Bitcoin Wallet. Since none of the other roles decreases at this time-point, this indicates that the developers added a significant amount of Information Holder-classes. We suspect that the developers attempted to extend the application by adding *domain models*. Domain models are OO classes that represents real-world objects or entities, and hold information related to these objects [18]. Considering that the responsibility of the Information Holder-role is to hold and provide information [2], it is likely that new domain models were added to the project, ergo the drastic increase of Information Holders in the beginning of 2018. However, inspection and evaluation of the source code is necessary to confirm this suspicion.

We also saw an increase of several roles in K9 Mail between 2013-2014. Similarly to what happened in Bitcoin Wallet, this increase did not occur in parallel with a decrease of any other roles, indicating an extension of the application. However, since this increase concerns all roles except the Controller, it becomes difficult to speculate what the source of this increase could have been. Perhaps the developers introduced third-party libraries and then removed them, since the involved roles decrease again in 2014.

In the results for Sweet Home 3D, we did not see any significant changes in the distribution of the roles. We can see a slow and steady increase of all roles respectively, indicating a stable and professional evolution of the software design.

All three projects share high frequencies of Information Holders and Service Providers. Based on the responsibilities of the roles defined by Wirfs-Brock *et al.* [2], we suspect that the demand for classes the hold and provide information, and compute tasks and perform work are in demand for these types of applications.

Based on Wirfs-Brock *et al.* [2] definition of the responsibility of the Interfacer-role as *"handling and transforming requests and information between different parts of a system"*, it seems likely that the variety and number of parts that a system has might increase the occurrence of Interfacer-classes. Based on this assumption, we can see that all three projects have more or less the same frequency of Interfacers (see Figures 7, 8, 9), averaging from 40 to 50 in frequency. This is interesting, assuming that the three projects differs in size and number of classes, that despite their difference in size, still have the same demand of classes that perform the work of the Interfacer-role.

To summarize, we saw one major change in the distribution of role-stereotypes in Bitcoin Wallet, namely a notable increase of Information Holders (**RQ1a**). Even though K9 Mail had a significant increase of several roles between 2013-2014, the distribution between the roles remained mostly the same during that time. Except for these two observations, and that we could not see any other major changes in the distribution of the roles. However, more research on a more broad variety of projects would help to further discuss the distribution of role-stereotypes over time.

RQ1b: *How does the distribution of anti-patterns change over time?* In section IV we observed some notable changes in the distribution of anti-patterns in the three projects. This is interesting since the distribution of roles generally did not have the same number of fluctuations. On a visual level, this may be interpreted as the changes in the distribution of anti-patterns not being related to the changes in the distribution of the roles. However, such claims would have to be statistically verified before making any such conclusions. In addition, this may also be explained by the anti-pattern detection being more sensitive to changes in the source-code compared to the sensitivity of the role-identification.

However, there are a few visual observations that strengthens the idea that the changes in the distribution of the anti-patterns can be explained by the changes in the distribution of roles.

For example, in section IV, we saw that Bitcoin Wallet had a notable increase of *Complex Class* at 2018-07-06. Looking at the increase of Information Holders in Bitcoin Wallet at 2018-04-01, these changes seem to occur close to each other. Chaudron *et al.* [8], found the that Information Holder-role was statistically associated with the *Complex Class*. This information heavily indicates that the increase of *Complex Class* in Bitcoin Wallet is related to the increase of Information Holders in the same project.

Furthermore, in the distribution of roles in K9 Mail, we saw an increase of several roles between 2013-2014. If we look closely at the same time-period in the distribution of anti-patterns in K9 Mail, we can see a small but notable increase of the majority of anti-patterns, indicating the the increase of the roles (or classes in general), increased the occurrences of a wide variety of anti-patterns.

Lastly, the distribution of anti-patterns in Sweet Home 3D remains most stable of all of the three projects. Comparing this to the almost non-existent changes in the distribution of roles in Sweet Home 3D, the two figures seem to match nicely on a visual level.

Generally, we can see similarities on a visual level between the changes in the distribution of anti-patterns and the changes in the distribution of roles (**RQ1b**).

RQ2: *How does the occurrence of anti-patterns in specific role-stereotypes change over time?*

A. Information Holder

In section IV, we saw that the frequency of anti-patterns in the Information Holder-role in K9 Mail increases significantly as time goes on, while they remain mostly the same in Bitcoin Wallet and Sweet Home 3D. Wirfs-Brock *et al.* [2] defined the responsibility of the Information Holder-role to hold and provide data or information. For such a low-complex responsibility, it seems strange that the Information Holders in

K9 Mail has such high and increasing frequency of it's anti-patterns. And since the number of Information Holders in K9 Mail does make any drastic increases, as seen in Figure 2, the increase of anti-patterns can not be explained by any increase of the role's frequency. Instead, we suspect that poor software design for Information Holders in K9 Mail are the cause of the increasing frequency of anti-patterns. We also noted that *Complex Class* and *Long Parameter List* have high frequencies in relation to other anti-patterns in the Information Holder-role in all three projects. Chaudron *et al.* [8], found the Information Holder role to be statistically associated with *Class Data Should Be Private*, *Complex Class* and *Long Parameter List*, which agrees with the results in our study, except for in Sweet Home 3D, where *Class Data Should Be Private* is absent.

B. Structurer

In section IV, we saw that the Structurer in Bitcoin Wallet had very few anti-patterns compared to the Structurer in K9 Mail and Sweet Home 3D. As defined by Wirfs-Brock *et al.* [2], the responsibility of the Structurer-role is to *"manage relationships between objects and information related to these relationships"*. Therefore, we believe that the Structurer-role is likely to be involved with Information Holder-classes, since the Information Holder can represent objects and provide information about that object. As such, we can see that the low number of Information Holders in Bitcoin Wallet (Figure 1), which range from 2 to 80 occurrences, is a possible reason for the low number of anti-patterns involved in the Structurer for Bitcoin Wallet, compared to the higher frequencies of the Information Holder in K9 Mail and Sweet Home 3D (Figure 2, 3), which range from 79-243, and 162-234 occurrences respectively. However, a more thorough investigation would need to be done regarding our assumption that the Structurer-role is heavily involved with dealing with Information Holders to support this suspicion.

Another notable observation is that, in K9 Mail, we can clearly see an increase of anti-patterns in the Structurer-role during 2013-2014, similar to the increase of anti-patterns in the Information Holder in K9 Mail during that time-period. On a visual level, we can see that this increase of anti-patterns is caused by the increase of several roles (or classes in general) during 2013-2014 for K9 Mail in Figure 2.

Furthermore, Chaudron *et al.* [8], found that the Structurer role had a statistically significant association to *Complex Class*, which agrees with the results presented in our study. *Swiss Army Knife* was also found to have an association to Structurer. However, *Swiss Army Knife* is not present in the Structurer of Bitcoin Wallet, and only occurs during the spike mentioned in section IV in the Structurer of K9 Mail, and generally has a low frequency in the Structurer of Sweet Home 3D. This may indicate that the statistical association between the Structurer-role and *Swiss Army Knife* may need to be re-evaluated on more broad sample-sizes.

C. Service Provider

Based on the results for Service Provider in section IV, we can see that the role does not seem to make any changes in the distribution of anti-patterns that are reflective to the changes in the distribution of the role-stereotypes. Wirfs-Brock *et al.* [2] describes the responsibility of the Service Provider-role to perform computational tasks and work, which, according to our interpretation, seems like a role that does not necessarily need to have a wide range of associations to other classes. Therefore, the changes in the distribution of anti-patterns in the Service Provider-role is not necessarily affected and/or reflected by the changes in the distribution of the roles.

One interesting observation is that the occurrences of *Lazy Class* are quite high in both K9 Mail and Sweet Home 3D. Chaudron *el al.* [8], found that Service Provider is associated with *Blob*, *Class Data Should Be Private* and *Complex Class* on a statistically significant level, but not *Lazy Class*. This may indicate that there is an unexplored association between Service Provider and *Lazy Class*, and that the statistical significance between the two may need to be re-evaluated on greater data samples.

D. Controller

In section IV, we observed a low amount of anti-patterns in the Controller-role in Bitcoin Wallet, compared to the amount of Controllers in K9 Mail and Sweet Home 3D. Wirfs-Brock *et al.* [2] defines the responsibility of the Controller as being responsible for making decisions and directing actions of other artefacts. Perhaps the type of work done by the classes in K9 Mail and Sweet Home 3D is more based on the responsibility of the Controller-role than the classes in Bitcoin Wallet, ergo the scarce variety of anti-patterns and their low frequency in Bitcoin Wallet. A far more likely explanation however, is the extremely low amount of Controller-classes in Bitcoin in general, as seen in Figure 7.

Chaudron *et al.* [8], found that the Controller role had an association to *Blob*, *Complex Class* and *Long Parameter List*, which matches well with the results for K9 Mail and Sweet Home 3D, while being more difficult to confirm for Bitcoin Wallet because of the generally low frequencies of anti-patterns.

Furthermore, since the frequency of the Controller-role in all three projects is very low and remains somewhat unchanging throughout the entire time-period (Figures 7, 8, 9), it becomes difficult to confirm if changes in the frequency of anti-patterns are related to the changes in the distribution of roles.

E. Coordinator

In section IV we saw the the anti-patterns in the Coordinator-role varies depending on the project. We only found one notable observation in this role; if we look closely, we can see an increase of *Complex Class* in the Coordinator in K9 Mail between 2013-2014 (2013-08-01 to 2014-05-02) that seems somewhat reflective to the changes in the distribution of roles in K9 Mail between 2013-2014 (2013-02-02 to 2014-05-02), where there is a slight increase in the number of

Coordinators (see Figures 23, 2). This may indicate that the additional Coordinators caused a small increase of *Complex Class* during that period. This also agrees well with the findings of Chaudron *et al.* [8], who found that the Coordinator is associated with *Complex Class*, which also matches well with the occurrences of *Complex Class* in the three projects. However, the increase of *Complex class* during the stated time-period may as well be caused by the increase of the other roles during that period, not necessarily the increase of the Coordinators. Chaudron *et al.* [8] also found that Coordinator is least involved in anti-patterns in general, which agrees with the results for Bitcoin Wallet and somewhat also for Sweet Home 3D, but not for K9 Mail (see Figures 1, 2, 3), which has a quite high frequency and number of anti-patterns in the Coordinator. But this can be explained by the quite high frequency of Coordinators in K9 Mail in general, compared to the low frequency of the role in Bitcoin Wallet and Sweet Home 3D.

Wirfs-Brock *et al.* [2] describes that the responsibility of the Coordinator is to *"delegate work to other artefacts by reacting to events"*. Considering that K9 Mail is an application that handles mail and is likely to have many events related to receiving and sending mail, it makes sense the the frequency of Coordinators is higher in K9 Mail compared to Bitcoin Wallet and Sweet Home 3D (as seen in Figures 7, 8, 9), and thus having a higher frequency of anti-patterns in this role.

F. Interfacer

Similarly to the Service Provider-role, the Interfacer does not seem have any notable changes in the frequency of anti-patterns that are reflective to the changes in the distribution of roles (compare Figures 7, 8, 9 to 25, 26, 27 respectively). This indicates that the anti-patterns in Interfacer change in frequency independently to the changes in the distribution of roles.

In addition, it seems that the changes of anti-patterns in the Interfacer-role for each project have more similarities to the anti-patterns of *other roles in the same project* rather then similarities to *the same role in other projects*, indicating the the design of the project plays a greater role in determining the occurrence of anti-patterns than depending on if a class is an Interfacer or not.

Furthermore, Chaudron *et al.* [8], found that the Interfacer role was associated with *Blob* and *Complex Class*, which agrees with the high frequency of *Complex Class* in Bitcoin Wallet and K9 Mail, and moderately high frequency in Sweet Home 3D.

G. Summary

The most interesting finding, according to our subjective opinion, is that the distribution of anti-patterns in specific roles seems to have more in common with the distribution of anti-patterns in *other roles in the same project*, rather than *the same roles in other projects*. One clear example of this is the *Blob* in the roles of Sweet Home 3D, which has a very fluctuating frequency in all roles except in Interfacer, while not having the same behaviour at all in any of the same roles in the other two projects. This is interesting because it may indicate that the design decisions of the developers may play a greater role in the occurrence of certain anti-patterns than which role-stereotype an OO class has.

Another interesting finding is that the changes in frequency of anti-patterns in certain roles seem to be reflective to the changes in the distribution of roles. For example, changes in the distribution of roles in Bitcoin Wallet and K9 Mail seem to influence the frequency of a variety of anti-patterns in Information Holders and Structurers, or vice versa. However, other roles such as Service Provider, Interfacer and Controller seems to have changes in frequency of anti-patterns that are unrelated to changes in the distribution of roles.

RQ3: *Which role-stereotypes are more prone to change roles over time, and to which roles do they change?*

From the interpretation of the presented data in Section IV, it is evident that the majority of classes classified as Service Provider are more prone to change roles over time in all of the selected projects, and conversely, most classes also tend to switch from other roles to Service Provider (**RQ3**).

An interesting finding is that there exists an interchanging relationships between certain pairs of role-stereotypes. This can be seen most evidently in Bitcoin Wallet, for example, 8 classes changed from Interfacer to Coordinator, and conversely, 9 classes changed from Coordinator to Intefacer. Accordingly, 7 classes changed from Interfacer to Service Provider, and 9 classes changed from Service Provider to Interfacer (Figure 28). More examples can be seen with other pairs of roles in all of the projects (Figures 28, 29 and 30). This could be an issue of inaccurate classification in the CRI-tool, as it may have classified a class as one role and the same class as another role at a later point in time. To confirm such speculation, we compare the confusion matrix table of the CRI-tool (Table I) with Figures 28, 29 and 30. Table I indicates that the classifier tends to misclassify Interfacer as Service Provider and vice versa. This is also the case for the roles Information Holder and Service Provider. Accordingly, in all projects, most changes in roles involve Interfacer changing from and to Service Provider as well as Information Holder changing from and to Service Provider (Figures 28, 29 and 30). We further investigate the interchanging relationship between two roles by checking whether the relationship occurs in the same class throughout the selected time period. Using the data from the CRI-tool, of which consist of mappings of Java classes and roles in different versions of all the selected project, we created timeline graphs that illustrate how classes change roles overtime in order to confirm the interchanging relationship (see Appendix B). As seen in the figures, when changing role-stereotypes, most classes tend to maintain their new roles throughout the whole time period.

Although there are some correlations between the misclassification of certain pairs of role-stereotypes and the number of changes in those pairs, we are not able to confirm whether

there exist interchanging relationships between the pairs or the relationships are caused by results of inaccurate classification without performing thorough code inspections. For example, if a class switch roles from Interfacer to Service Provider with only a few changes in the source code, or with changes that are insufficient to be labeled as Service Provider, we can suspect that the interchanging relationships are derived from inaccurate classification of role-stereotypes.

The same principle applies to classes that have changed role-stereotypes multiple times throughout the time period (Tables III, IV and V). We can see that the largest number of roles a class has been assigned is four, and thus, we are uncertain whether the changes come from the intention of the developers, or from the accuracy error of the CRI-tool. We only suspected that it is unlikely for a class to have been assigned four different role-stereotypes. Further code inspection is required to validate the suspicion.

Furthermore, we also suspect that if the CRI-tool had been trained on a more broad range of projects, it may have reduced the sensitivity of the classifier and consequently reduced the number of role changes.

VI. Threats to Validity

The *Internal validity* of our study concerns the non-existent roles, as can be seen in the figures illustrating the changes in specific classes in section B of the Appendix. This is because those classes were likely removed or migrated to different packages in the projects. Some anti-patterns in the projects, and in specific role-stereotypes in the projects, were non-existent. To resolve this, we removed those anti-patterns from the analysis.

The *External validity* of our study concerns the generalizability of the results. The study was conducted on three open-source software projects written in Java. Then, it would be natural to assume that the results only apply to software systems written in Java. However, since most OO-software systems share architectural characteristics and structures regardless of programming language [12] the result of the study can be generalized to benefit the majority of such systems.

The *Reliability* of the study concerns the accuracy of the different tools used.

Since only the pivot versions mentioned in section III had ground truths established for the role-stereotypes, it is unknown if the roles classified using the CRI-tool were actually accurate for the rest of the selected versions of each project. Ho-Quang *et al.* [11], confirmed the accuracy by comparing the ground truth to the results of the tool. The ground truth was established by manually labeling all classes in the projects. However, considering the amount of versions (110 in total) in our study, it is not realistic to conduct the same manual labelling for every class in each of the 110 versions of the projects. Instead, we relied on the CRI-tool's documented accuracy [10], [11], [12].

As for the detection of anti-patterns by the Ptidej tool, it was not possible to manually confirm that the detected anti-patterns were correct, mainly because of the same reason as with the CRI-tool, it was not feasible to perform manual checking of each class in each version of the projects. Instead, we relied on Ptidej's documented precision as mentioned in section III, which we deemed high enough to make the results trustworthy.

The APCRM tool used algorithms to produce data on roles, anti-patterns and the mapping between these two. To validate that the algorithms worked as intended, several executions were manually compared to the raw input data. All such comparisons showed that the results produced by the algorithms yielded the expected results.

VII. Related Work

There are several studies conducted for the purpose of finding code smells/design anti-patterns [14], [15]. For example, Moha *et al.* [14] introduced a method, namely DECOR, in order to establish a set of rules for the detection of anti-patterns. These rules are then applied as an embodiment of the DECOR method for the development of DETEX, a detection tool for software systems. As an improvement to DECOR, which was only able to identify a limited number of specific code smells, Moha *et al.* [15] presented a new method to automate the generation of newfound code smells by using detection algorithms from specifications derived during domain analysis.

There also exist multiple studies on the identification of roles in classes. For example, Nurwidyantoro *et al.* [10], [11], [12] used machine learning to automate the classification of the six role-stereotypes introduced by Wirfs-Brock *et al.* [2].

A study by Khomh *et al.* [16] investigated the relation between anti-patterns and change-proneness. The authors found that classes containing anti-patterns are more likely to require changes and contain faults than classes that are not involved in anti-patterns. However, they did not use the perspective of role-stereotypes, and did not provide any categorization-approach to indicate in what classes the anti-patterns are prone to emerge. Our study aims to close this gap so that anti-patterns can be predicted and prevented when making changes to software systems.

Chaudron *et al.* [8] studied the occurrence of anti-patterns in role-stereotypes found that there is a relationship between the two on a statistical significant level, and that some roles are more prone to anti-patterns than other roles. Thanks to the results of the study, it becomes relevant to look at how role-stereotypes evolve over time to understand the occurrence of anti-patterns. The study was conducted on three open-source software systems, namely Bitcoin Wallet, K9Mail and Sweet Home 3D. The study also covered relationships to design patterns, but our study only focus on anti-patterns. Our study also distinguishes itself by performing the same methods, but in a longitudinal approach.

Among the mentioned studies, only the study by Chaudron *et al.* [8] was dedicated for the findings of relations between design (anti)patterns and roles. However, the study did not consider how the relations evolve over multiple successive versions of a software system. Chatzigeorgiou *et al.* [17] investigated the evolution of anti-patterns/code smells in OO

systems, but did not include a any way to correlate the evolution of anti-patterns with role-stereotypes or specific responsibilities in classes. We believe that our is the first study to report on such matter.

VIII. Conclusion

In this study we saw, on a visual level, that the total distribution of anti-patterns seem to have some changes that can be explained by the changes in the distribution of roles (or vice versa). In addition, we found that the distribution of anti-patterns in specific roles also might be affected or reflective to the changes in the distribution of the roles. For example, the distribution of anti-patterns in Information Holder and Structurer seem to make changes that can be associated with the changes in the distribution of the roles, while the anti-patterns in Service Provider, Controller and Interfacer seem to make changes independently of the changes in the distribution of roles.

We also found that it seems that the changes of anti-patterns in specific roles have more in common with the changes in anti-patterns of *other roles in the same project* rather than with *the same role in other projects*, indicating that the design of the project plays a greater role in determining the occurrence of anti-patterns. For example, the frequency of anti-patterns in the roles in Sweet Home 3D have more similarities with each other than the frequency of anti-patterns in a specific role in Sweet Home 3D has with the same role in the other projects. One such similarity is the greatly varied frequency of *Blob* in most of Sweet Home 3D's roles, while it does not vary as much, or even occur at all, in the same roles in the other projects. We believe this may be related to the design decisions of the developers of the projects. However, this was based on visual observations of the charts, and would need further analysis on source-code level to make any conclusions.

Furthermore, we suspect a possible explanation for notable increases of Information Holders (as seen in Figure 1) in an OO system *might* be related to introducing new domain models, but this requires further investigation of the source code to make any further claims.

We discovered that classes assigned as Service Provider tend to change to other role-stereotypes, and vice versa, most classes are also likely to change from other roles to Service Provider. Furthermore, while analysing the total number of changes in each role, we uncovered that there seems to exist interchanging relationships when classes switch between certain pair of roles. Specifically, in most cases, the number of changes from *role A* to *role B* is almost equal to the number of changes from *role B* to *role A*. We suspected that the cause of this interchanging relationship comes from the accuracy error of the CRI tool. To confirm the suspicion, thorough code inspection is required.

Future Work

For future work, it would be interesting to investigate if the design decisions of developers play a greater role in determining the occurrence of certain anti-patterns than which role-stereotype an OO-class has, based on our findings in section V for RQ2. Chaudron *et al.* [8] found statistical correlations between certain role-stereotypes and anti-patterns, but our findings suggest that if a correlation exist between design-decisions of developers and certain anti-patterns, that correlation might be greater.

Another study is to investigate classes that change multiple role-stereotypes over time, since interesting elements can be found in the source code that can be analysed to determine whether to improve the CRI tool as the changes might be incorrect classification, or to examine why the classes go through so many changes. Thus, these elements can also answer the question to how the anatomy of features in classes relate to certain role-stereotypes.

It would also be interesting to look at how role-stereotypes change in the beginning of additional projects, considering that our study was only done on Bitcoin Wallet from the very start of the project, which was not the case for K9 Mail and Sweet Home 3D.

References

[1] R. Wirfs-Brock, B. Wilkerson. "Object-Oriented Design: A Responsibility-Driven Approach", OOPSLA 1989, 1989.

[2] R. J. Wirfs-Brock, A. McKean, "Object Design: Roles Responsibilities and Collaborations", Pearson Education, 2002.

[3] R. J. Wirfs-Brock, "Characterizing Classes", IEEE Software Vol. 23 No. 2, April 2006, pp. 9-11.

[4] L. Kuzniarz, M. Staron, C. Wohlin, "An empirical study on using stereotypes to improve understanding of UML models," Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004, pp. 14-23.

[5] E. Gamma, J. Vlissides, R. Johnson and R. Helm, "Design Patterns: Elements of Reusable Object-oriented software", 1994, pp. 14-16.

[6] M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts, E. Gamma, "Refactoring: Improving the Design of Existing Code", Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[7] U. A. Mannan, I. Ahmed, R. M. A. Almurshed, D. Dig, C. Jensen, "Understanding Code Smells in Andriod Applications", MOBILESoft '16: Proceedings of the International Conference on Mobile Software Engineering and Systems, 2016, pp. 1-12.

[8] M. R. V. Chaudron, "Studying the Occurrence of (Anti)Patterns through the Lens of Role-Stereotypes in Software Designs", Personal communication, 2020, pp. 1-10.

[9] A. Yamashita, L. Moonen, "Do developers care about code smells? An exploratory survey", 20th Working Conference on Reverse Engineering (WCRE), 2013, pp. 1-11.

[10] T. Ho-Quang, A. Nurwidyantoro, M. R. V. Chaudron, "Using Machine Learning for Automated Classification of Class Responsibility Stereotypes in Software Design", Department of Computer Science & Engineering, Chalmers University of Technology and University of Gothenburg, pp. 1-29.

[11] A. Nurwidyantoro, T. Ho-Quang, M. R. V. Chaudron, "Improving the Automated Classification of Role-Stereotypes by Machine Learning", EASE 2019, pp. 1-10.

[12] A. Nurwidyantoro, T. Ho-Quang, M. R. V. Chaudron, "Automated Classification of Class-Role Stereotypes via Machine Learning", EASE 2019, pp. 1-10.

[13] Y.-G. Guéhéneuc, "Ptidej: Promoting Patterns With Patterns", Proceedings of the 1st ECOOP workshop on Building a System using Patterns. Springer-Verlag, 2005, pp. 1-9.

[14] N. Moha, Y.-G. Guehéneuc, L. Duchien, A. Le Meur, "Decor: A Method for the Specification and Detection of Code and Design Smells", IEEE Transactions On Software Engineering, vol. 36, no. 1, Jan 2010, pp. 20-36.

[15] N. Moha, Y.-G. Guehéneuc, L. Duchien, A. Le Meur, A. Tiberghien, "From a Domain Analysis to the Specification and Detection of Code and Design Smells", Formal Aspects of Computing, vol. 22, no. 3, May 2010, pp. 345-361.

[16] F. Khomh, M. D. Penta, Y.-G. Guéhéneuc, G. Antoniol, “An exploratory study of the impact of antipatterns on class change- and fault-proneness”, Empirical Software Engineering 17, 243–275 (2012), Springer Science+Business Media, LLC 2011, pp. 1-33.
[17] A. Chatzigeorgiou and A. Manakos, ”Investigating the Evolution of Code Smells in Object-Oriented Systems,” 2010 Seventh International Conference on the Quality of Information and Communications Technology, Porto, 2010, pp. 106-115.
[18] S. Millet, N. Tune, ”Patterns, Principles, and Practices of Domain-Driven Design”, John Wiley Sons Inc, 2015, pp. 1-795.

APPENDIX

A. Bitcoin Wallet Graphs with Divergence Data

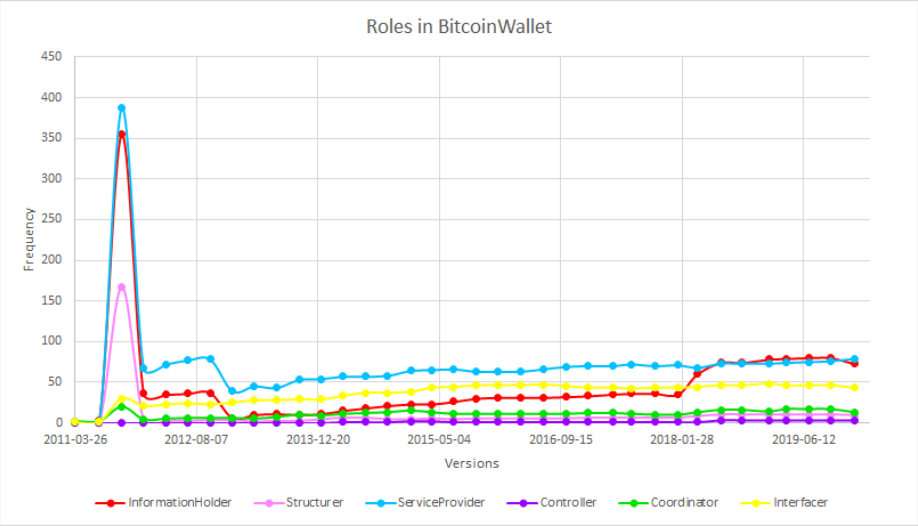

Fig. 31. Distribution of the roles in each version of Bitcoin Wallet (including version from 2011-10-03)

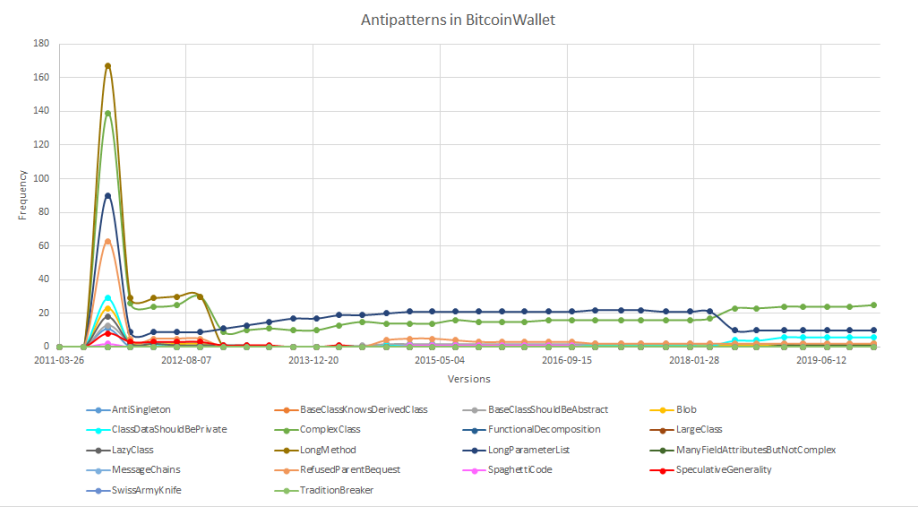

Fig. 32. Distribution of the anti-patterns in each version of Bitcoin Wallet (including version from 2011-10-03)

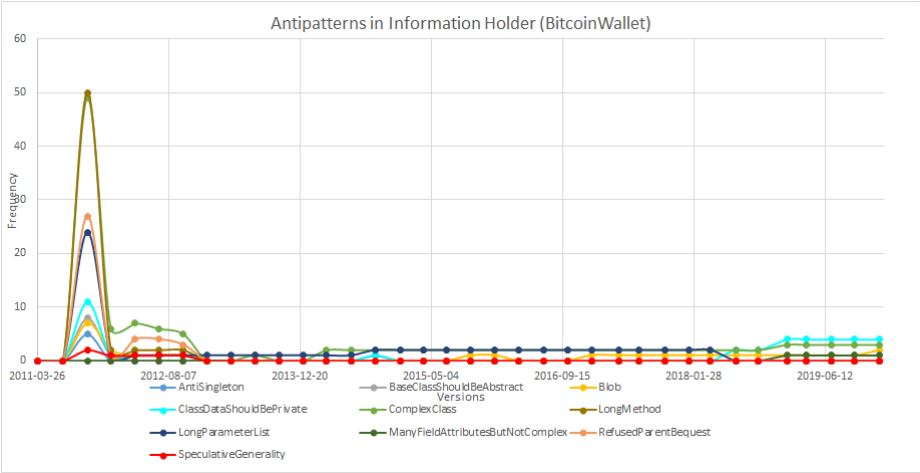

Fig. 33. Anti-patterns of Information Holder in Bitcoin Wallet (including version from 2011-10-03)

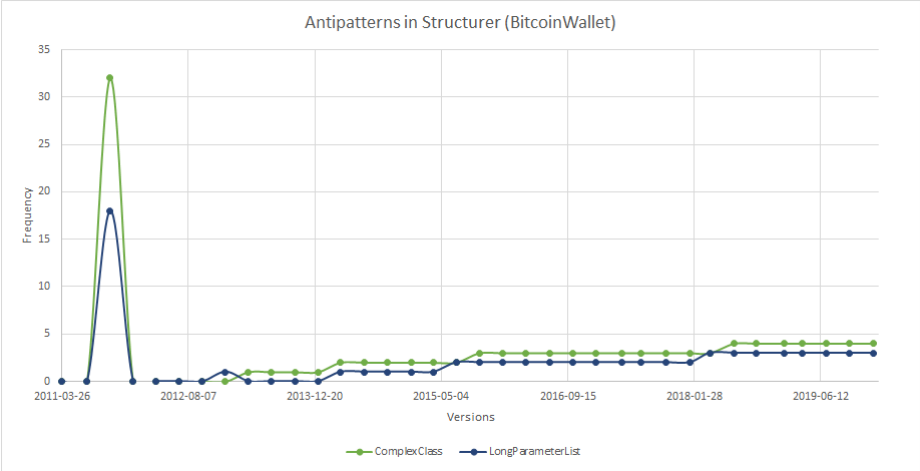

Fig. 34. Anti-patterns of Structurer in Bitcoin Wallet (including version from 2011-10-03)

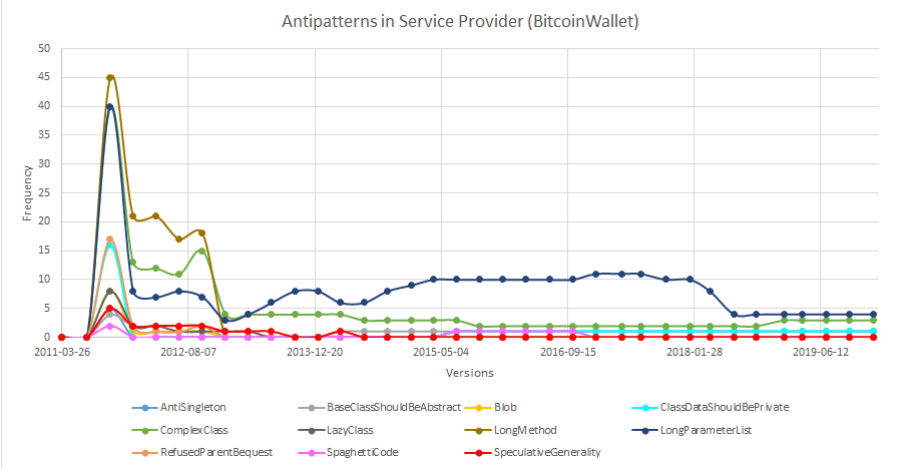

Fig. 35. Anti-patterns of Service Provider in Bitcoin Wallet (including version from 2011-10-03)

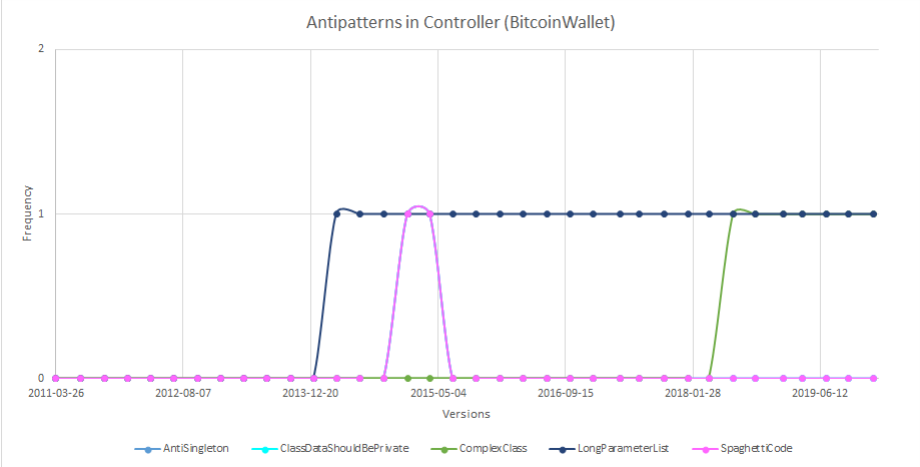

Fig. 36. Anti-patterns of Controller in Bitcoin Wallet (including version from 2011-10-03)

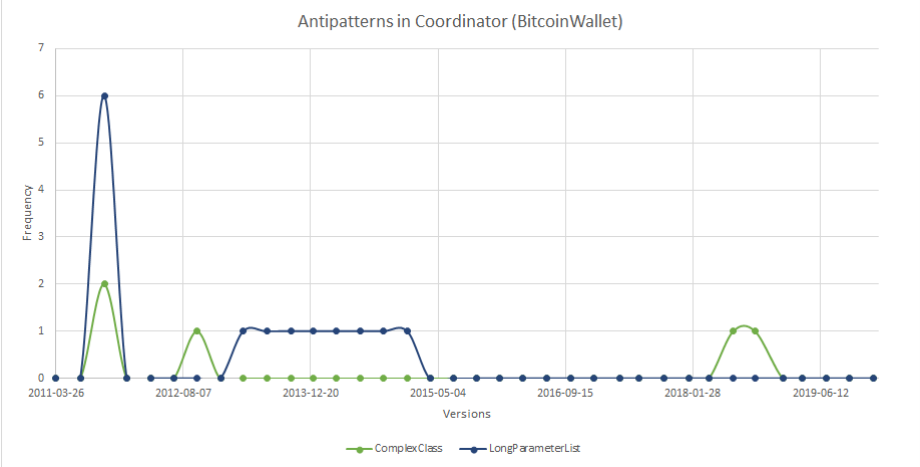

Fig. 37. Anti-patterns of Coordinator in Bitcoin Wallet (including version from 2011-10-03)

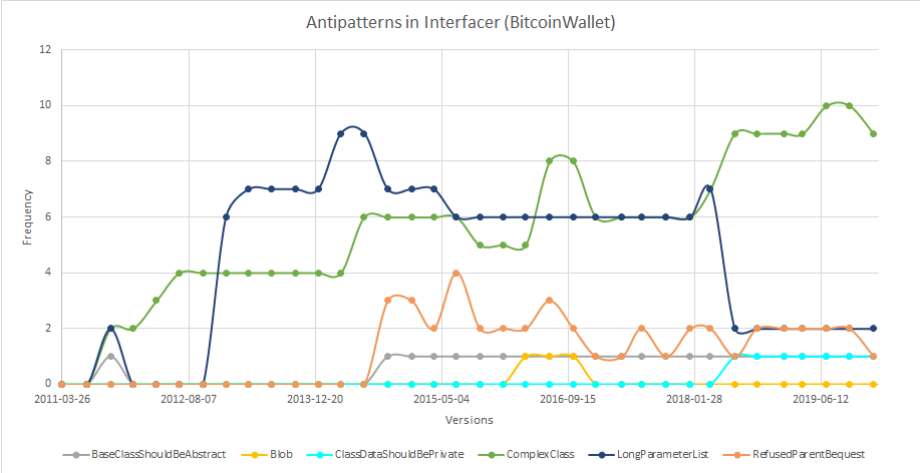

Fig. 38. Anti-patterns of Interfacer in Bitcoin Wallet (including version from 2011-10-03)

B. Timeline graphs of how individual classes change roles over time

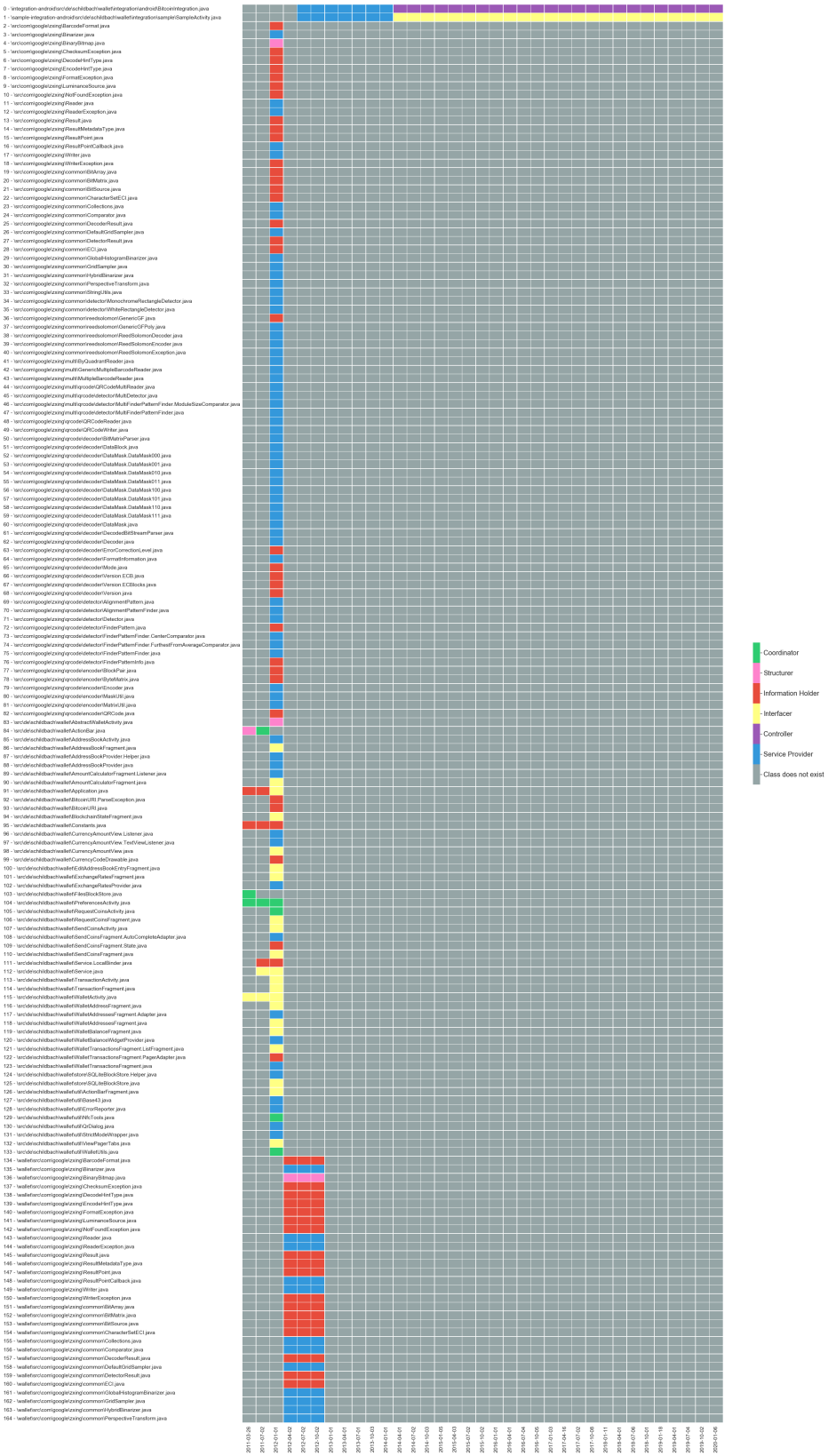

Fig. 39. How classes change role-stereotypes overtime in Bitcoin Wallet (First portion of classes)

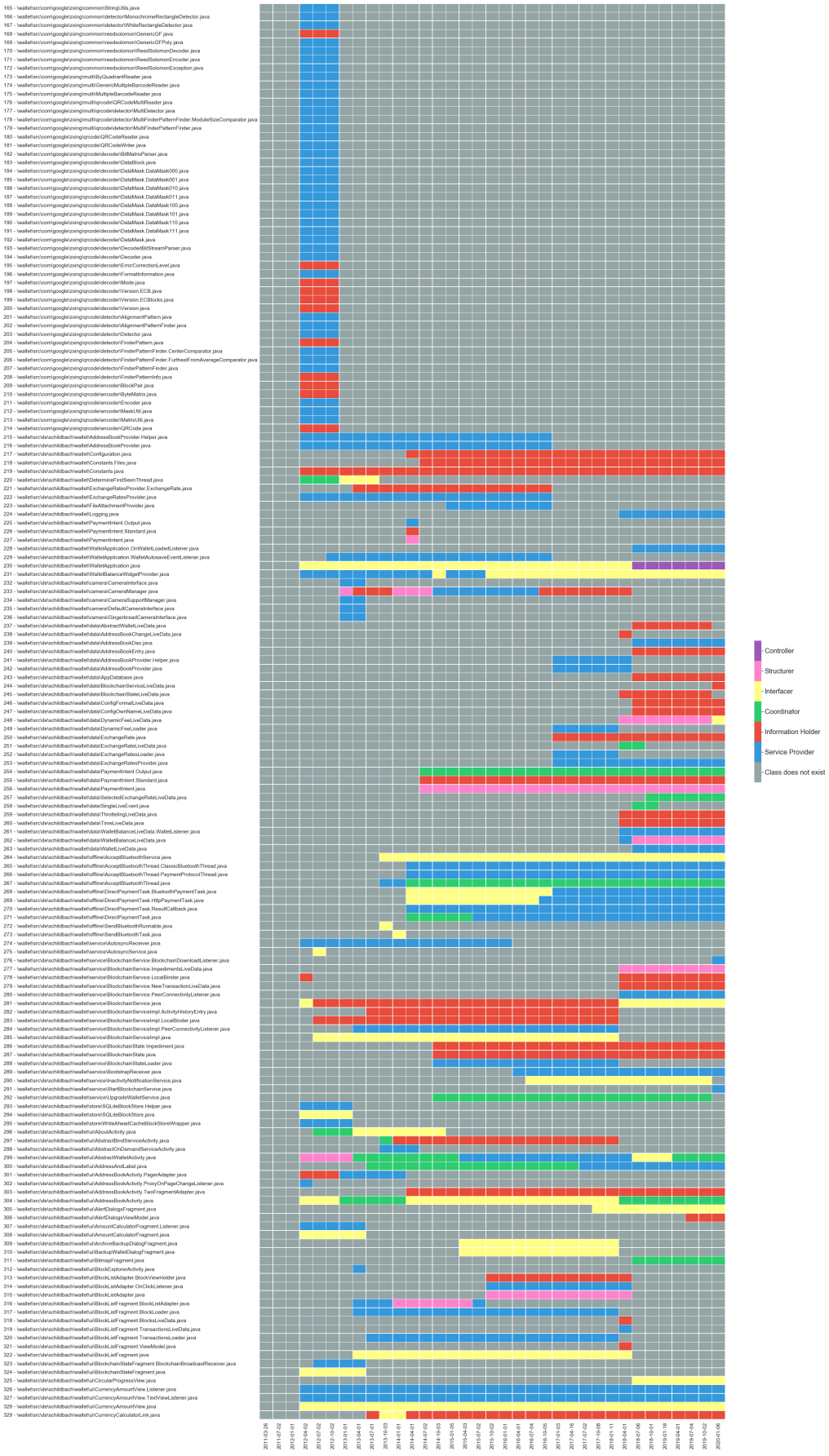

Fig. 40. How classes change role-stereotypes overtime in Bitcoin Wallet (Second portion of classes)

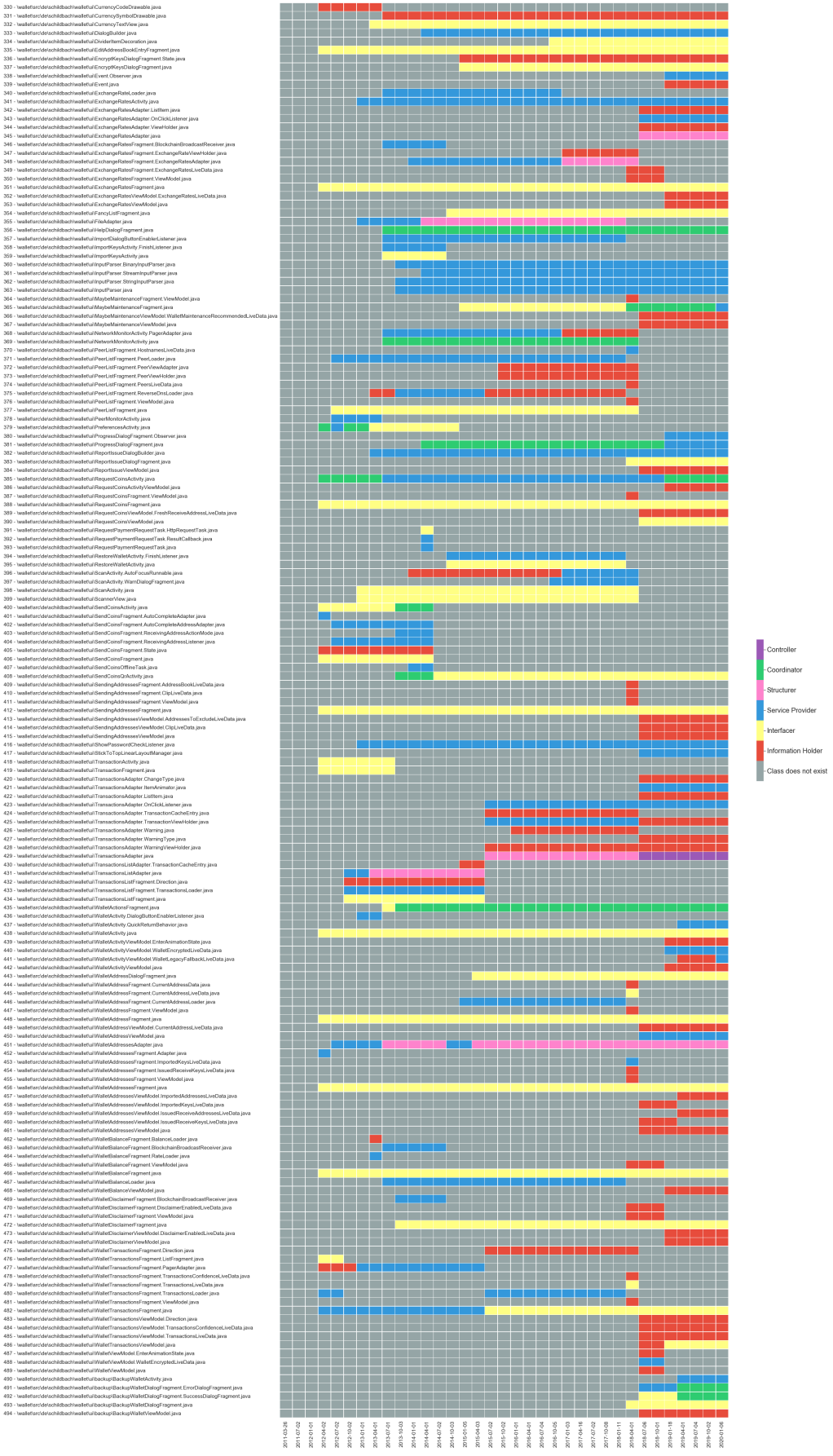

Fig. 41. How classes change role-stereotypes overtime in Bitcoin Wallet (Third portion of classes)

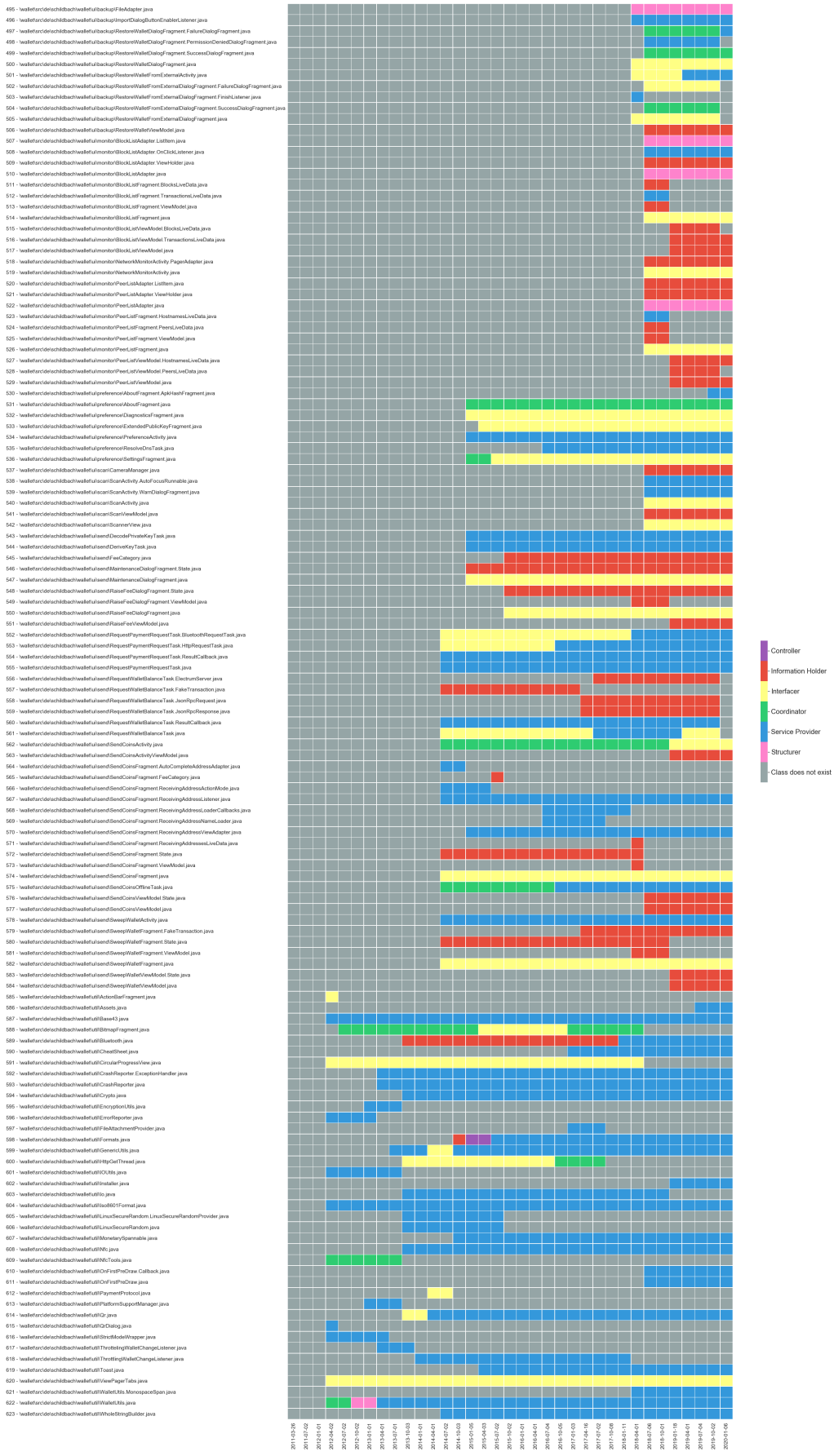

Fig. 42. How classes change role-stereotypes overtime in Bitcoin Wallet (Last portion of classes)

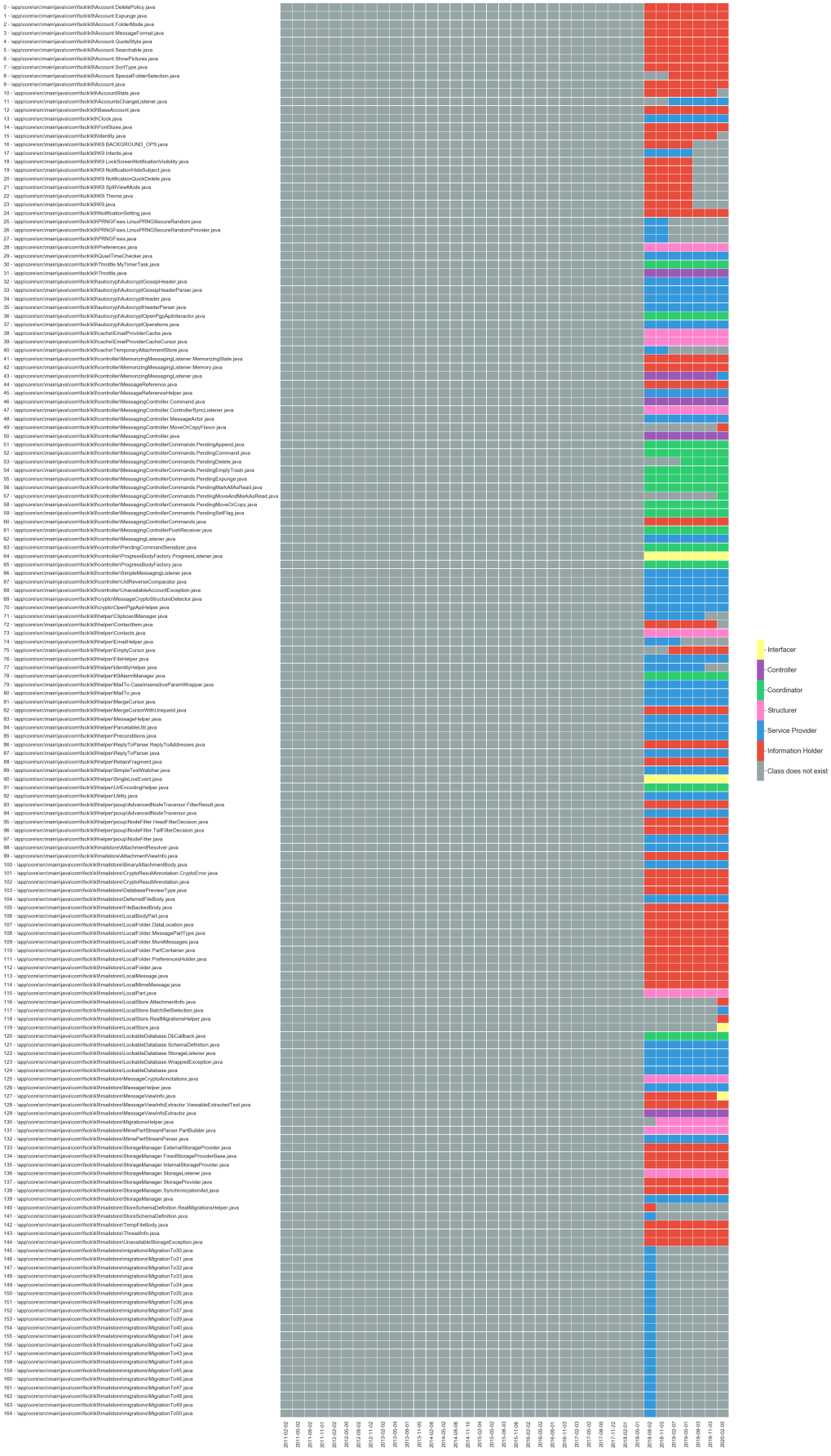

Fig. 43. How classes change role-stereotypes overtime in K9Mail (First portion of classes)

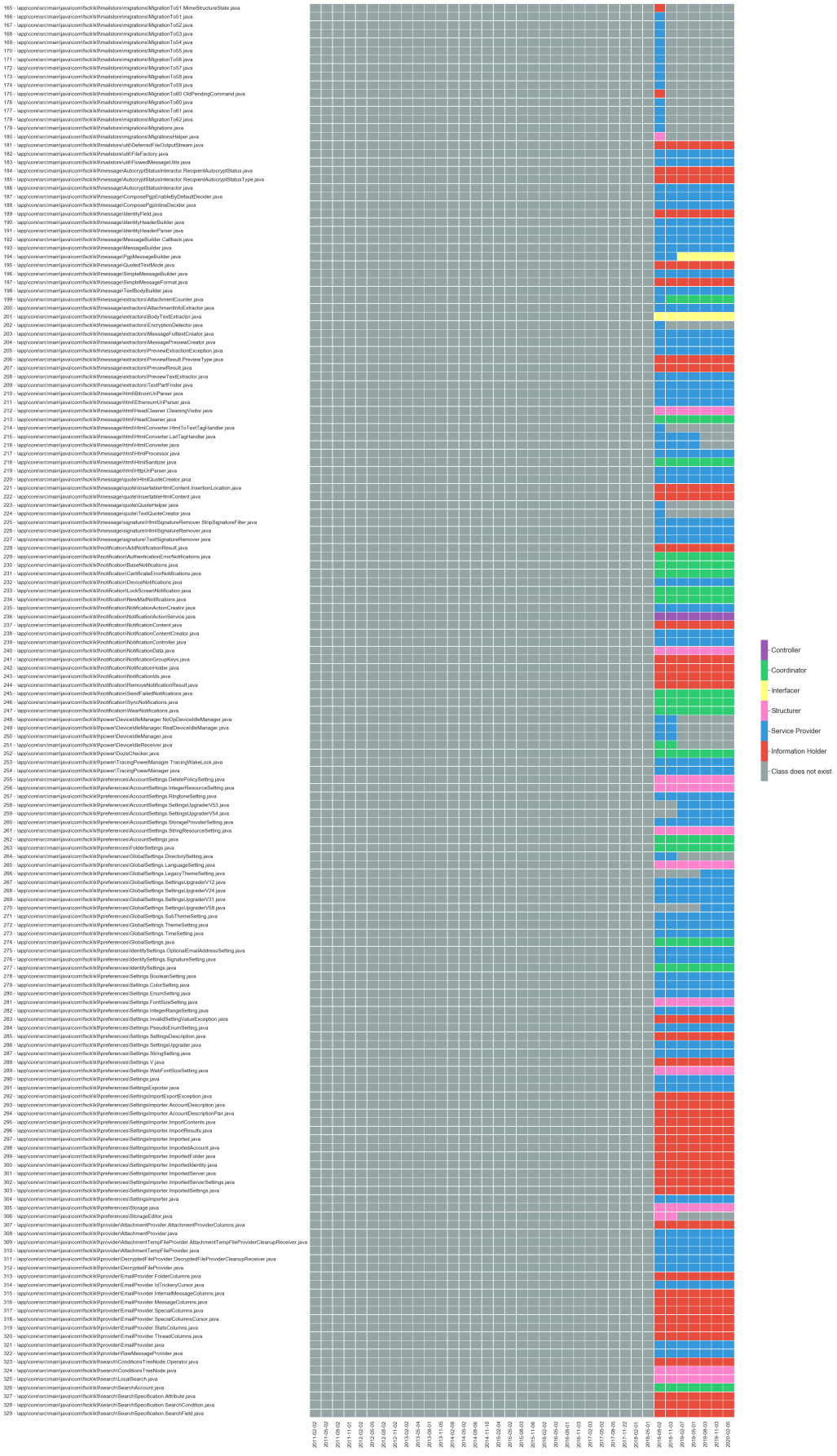

Fig. 44. How classes change role-stereotypes overtime in K9Mail (Second portion of classes)

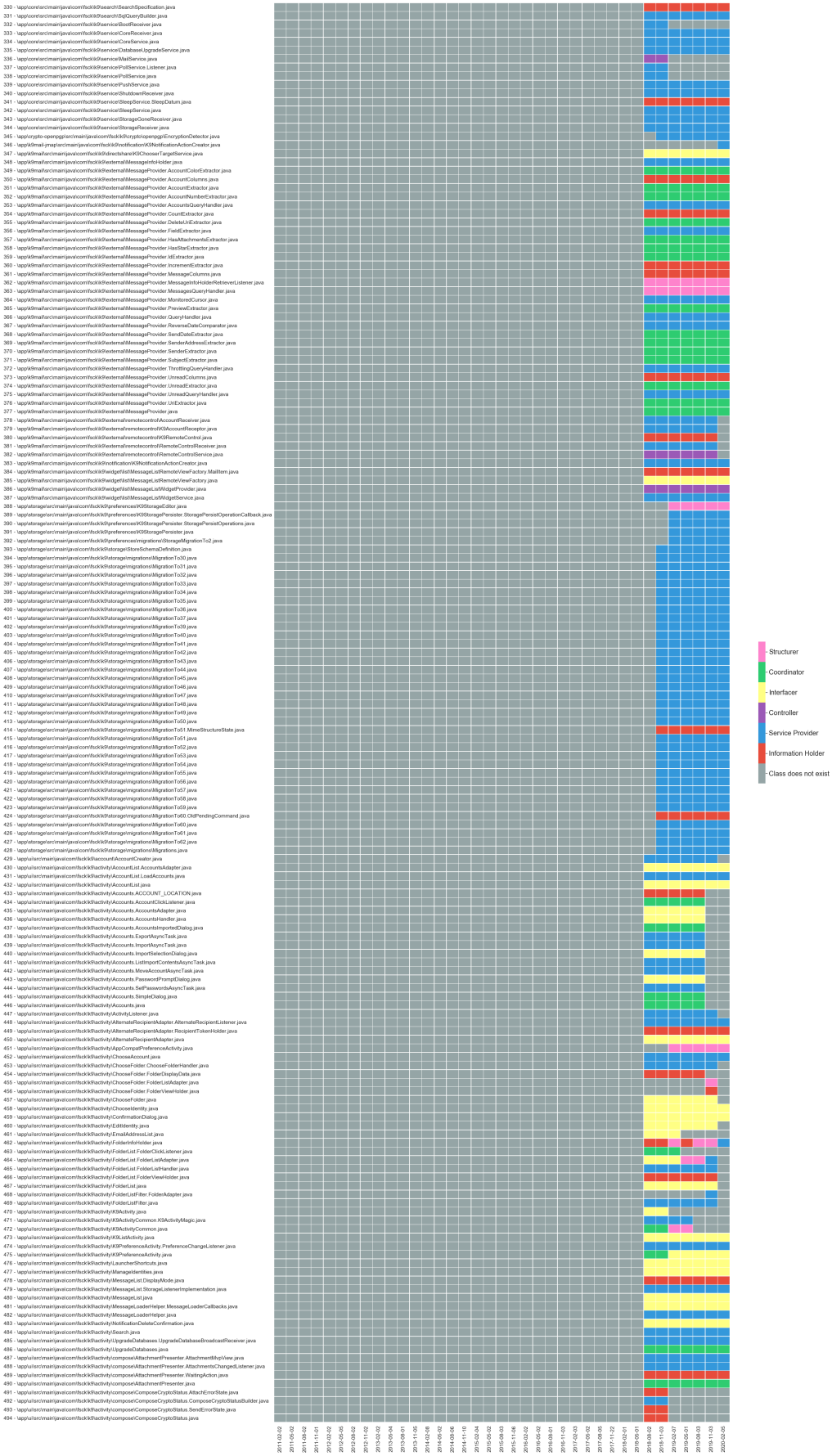

Fig. 45. How classes change role-stereotypes overtime in K9Mail (Third portion of classes)

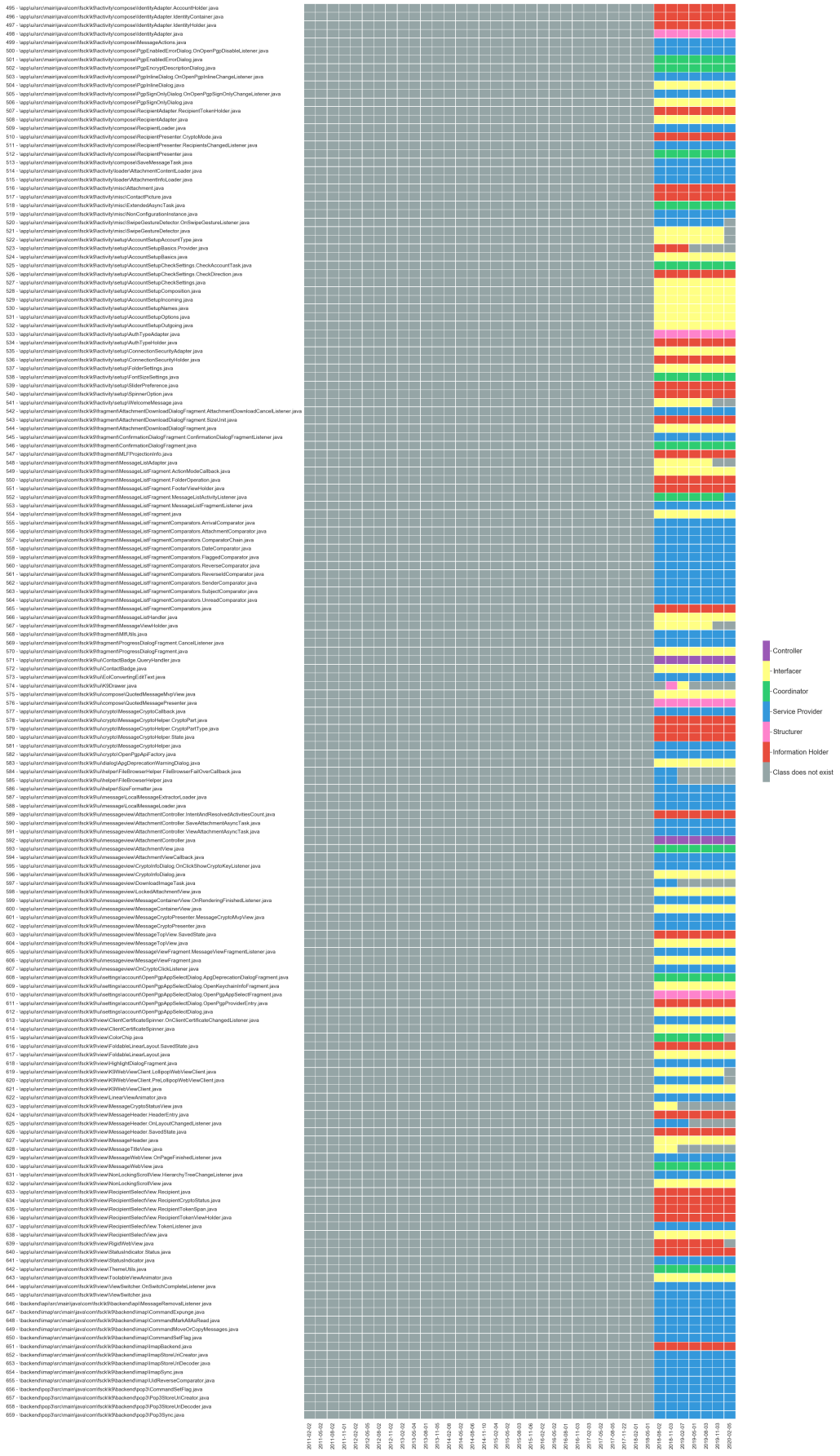

Fig. 46. How classes change role-stereotypes overtime in K9Mail (Fourth portion of classes)

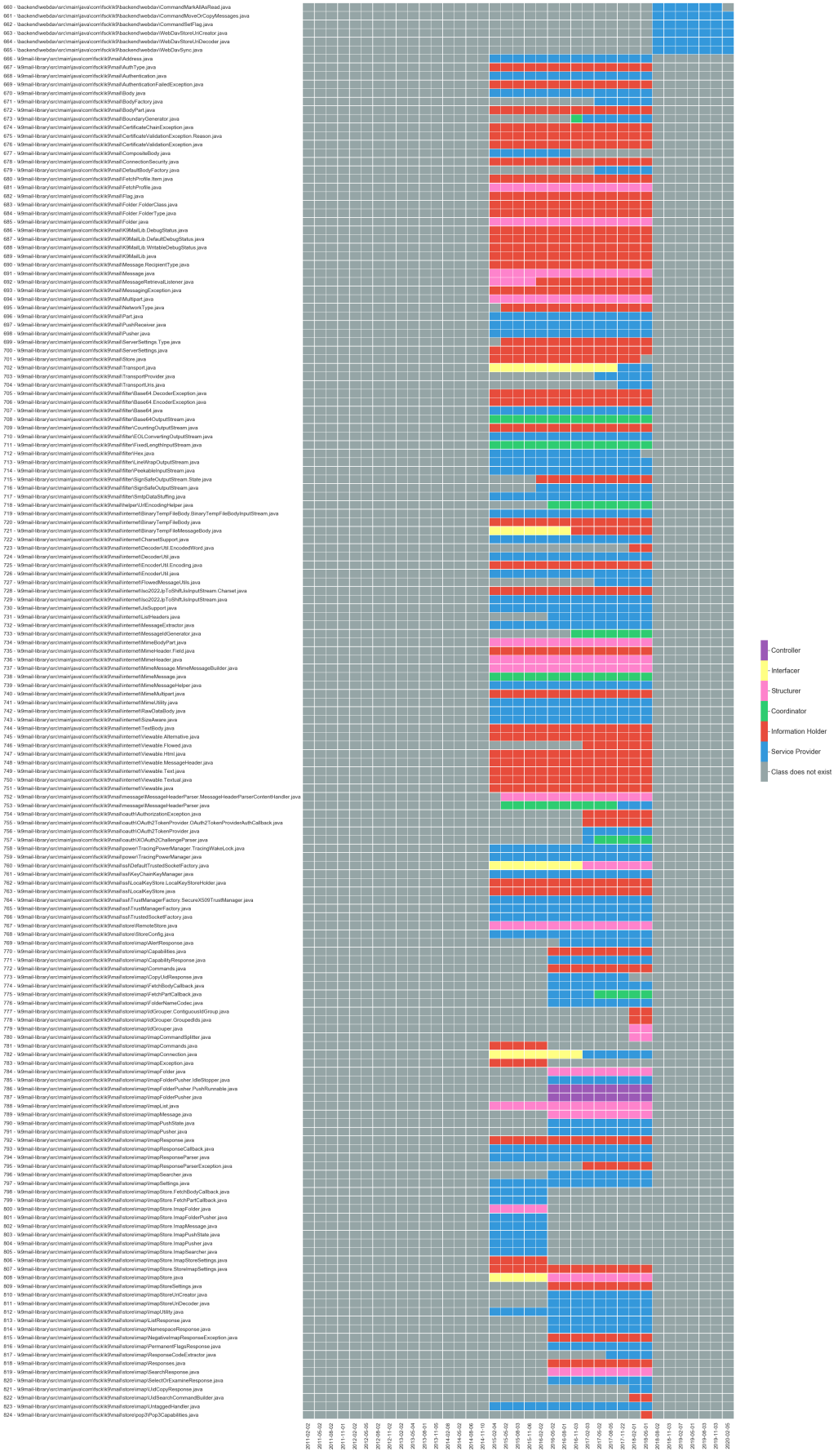

Fig. 47. How classes change role-stereotypes overtime in K9Mail (Fifth portion of classes)

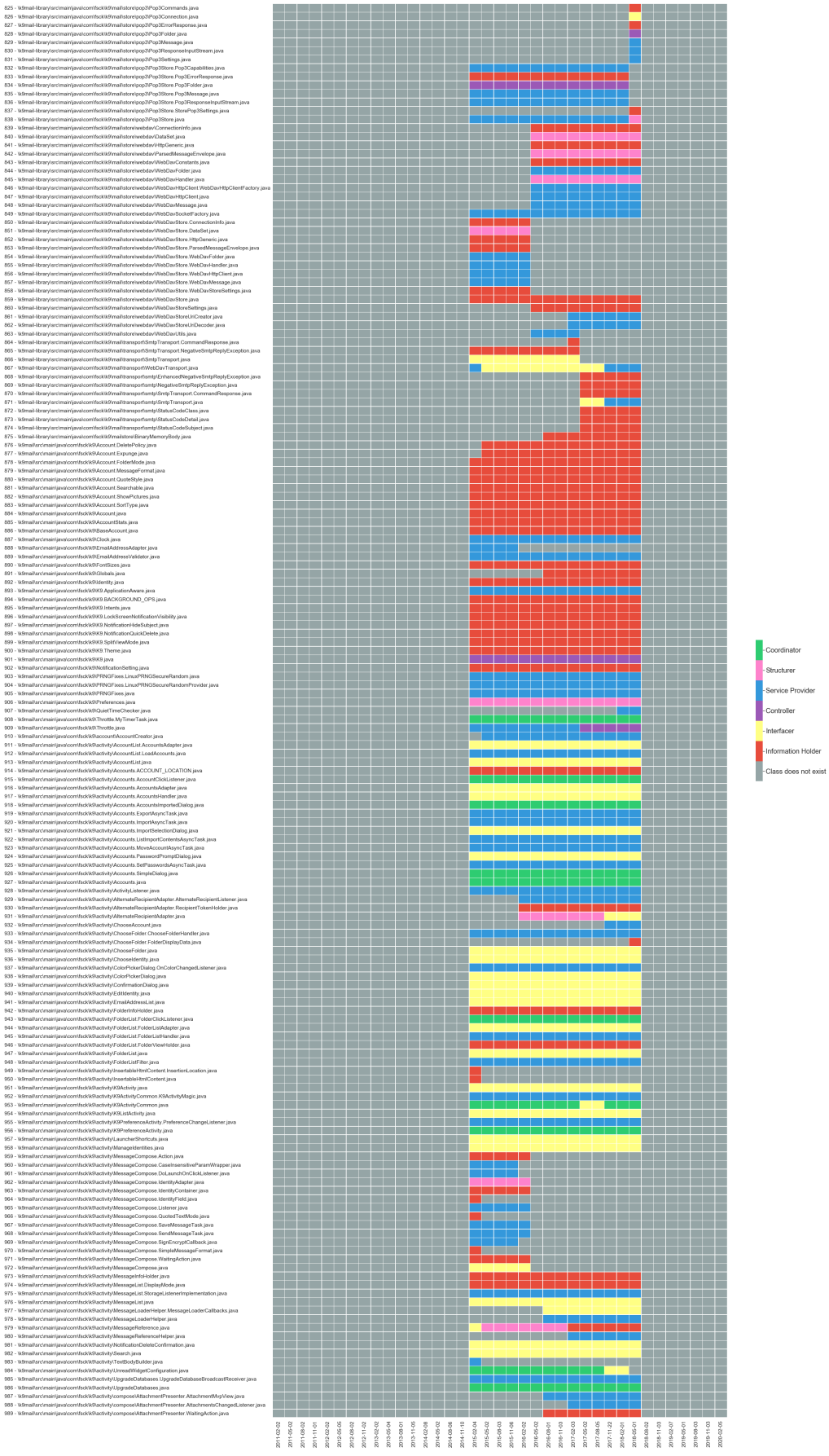

Fig. 48. How classes change role-stereotypes overtime in K9Mail (Sixth portion of classes)

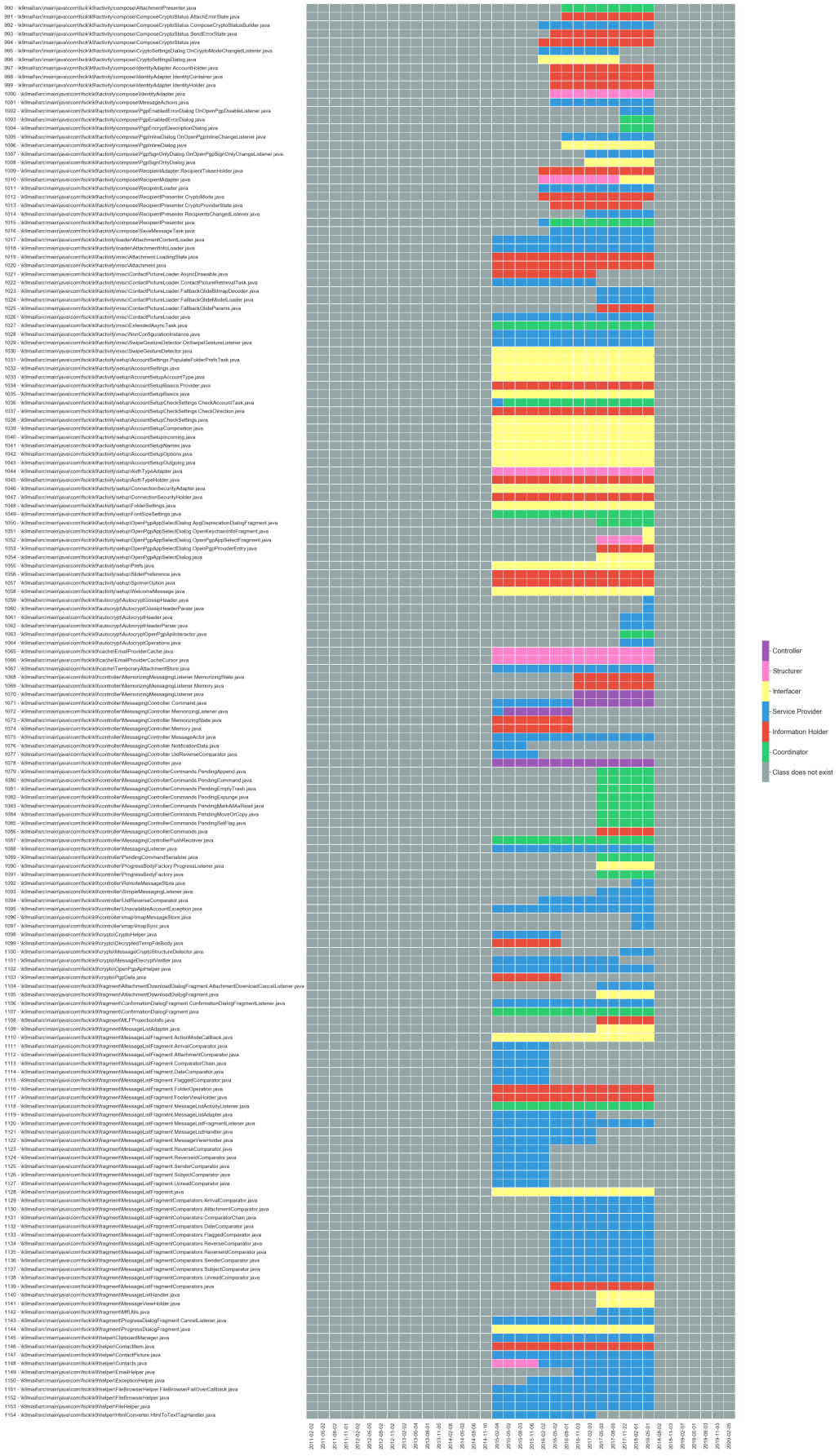

Fig. 49. How classes change role-stereotypes overtime in K9Mail (Seventh portion of classes)

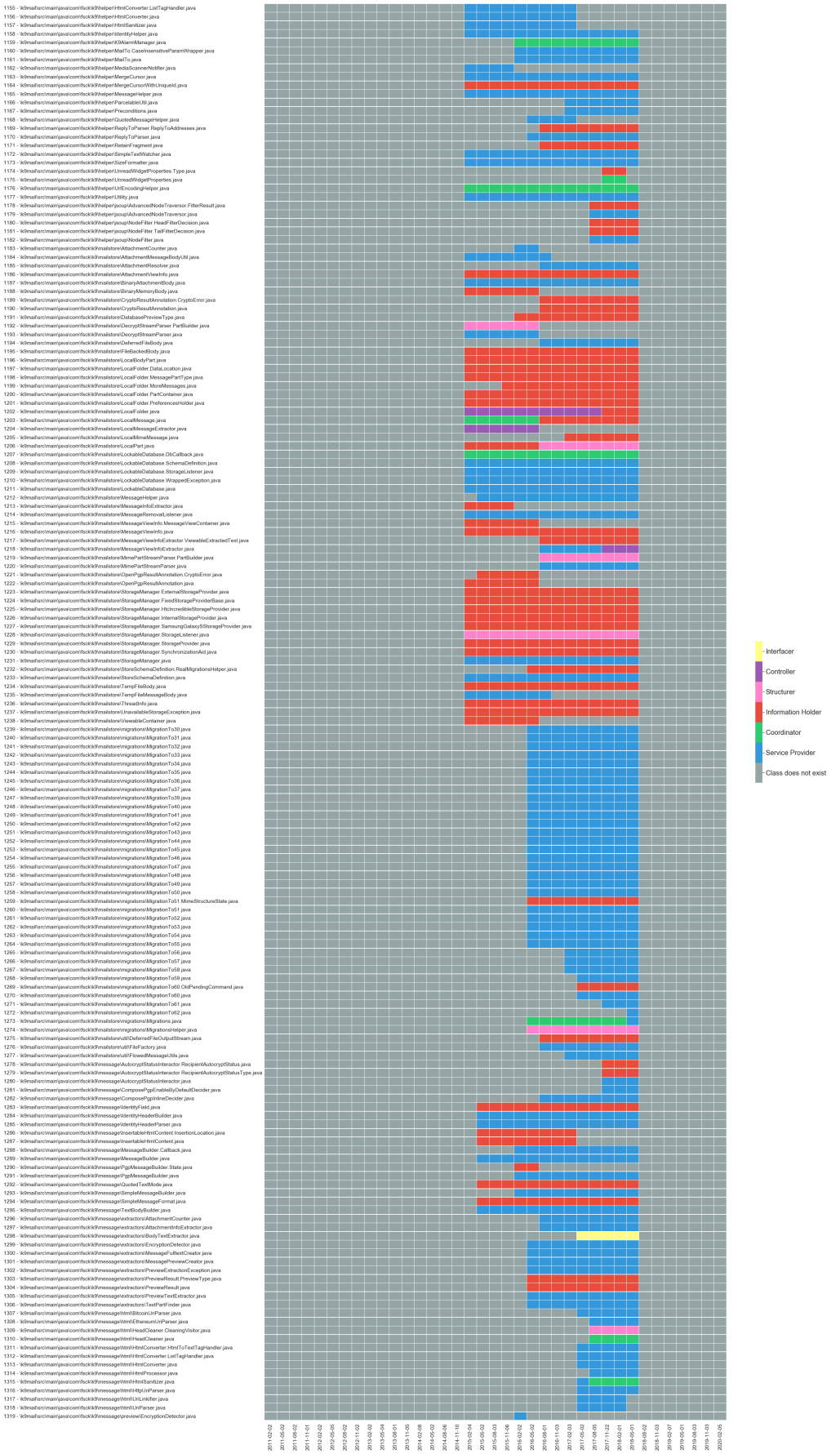

Fig. 50. How classes change role-stereotypes overtime in K9Mail (Eighth portion of classes)

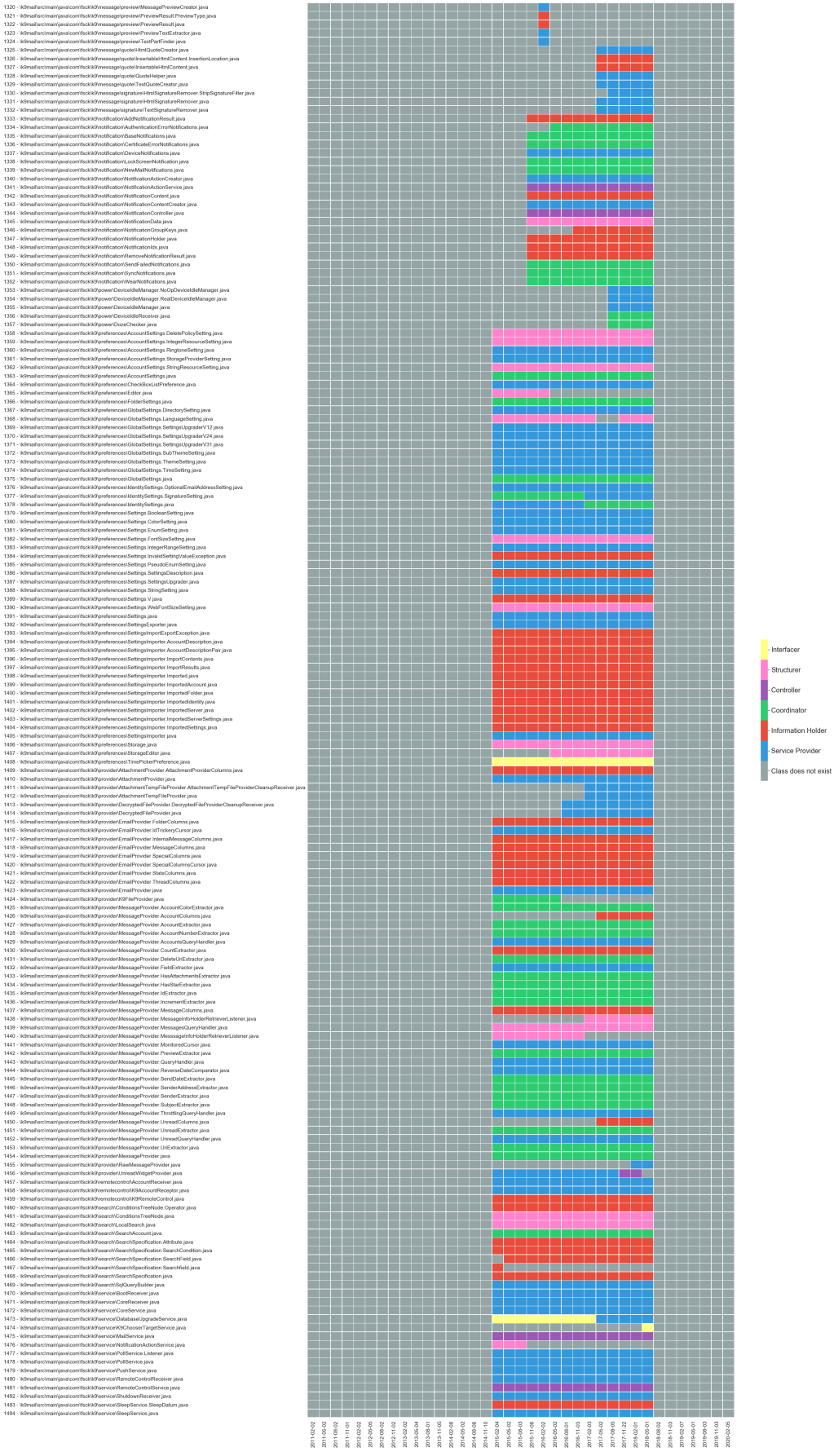

Fig. 51. How classes change role-stereotypes overtime in K9Mail (Ninth portion of classes)

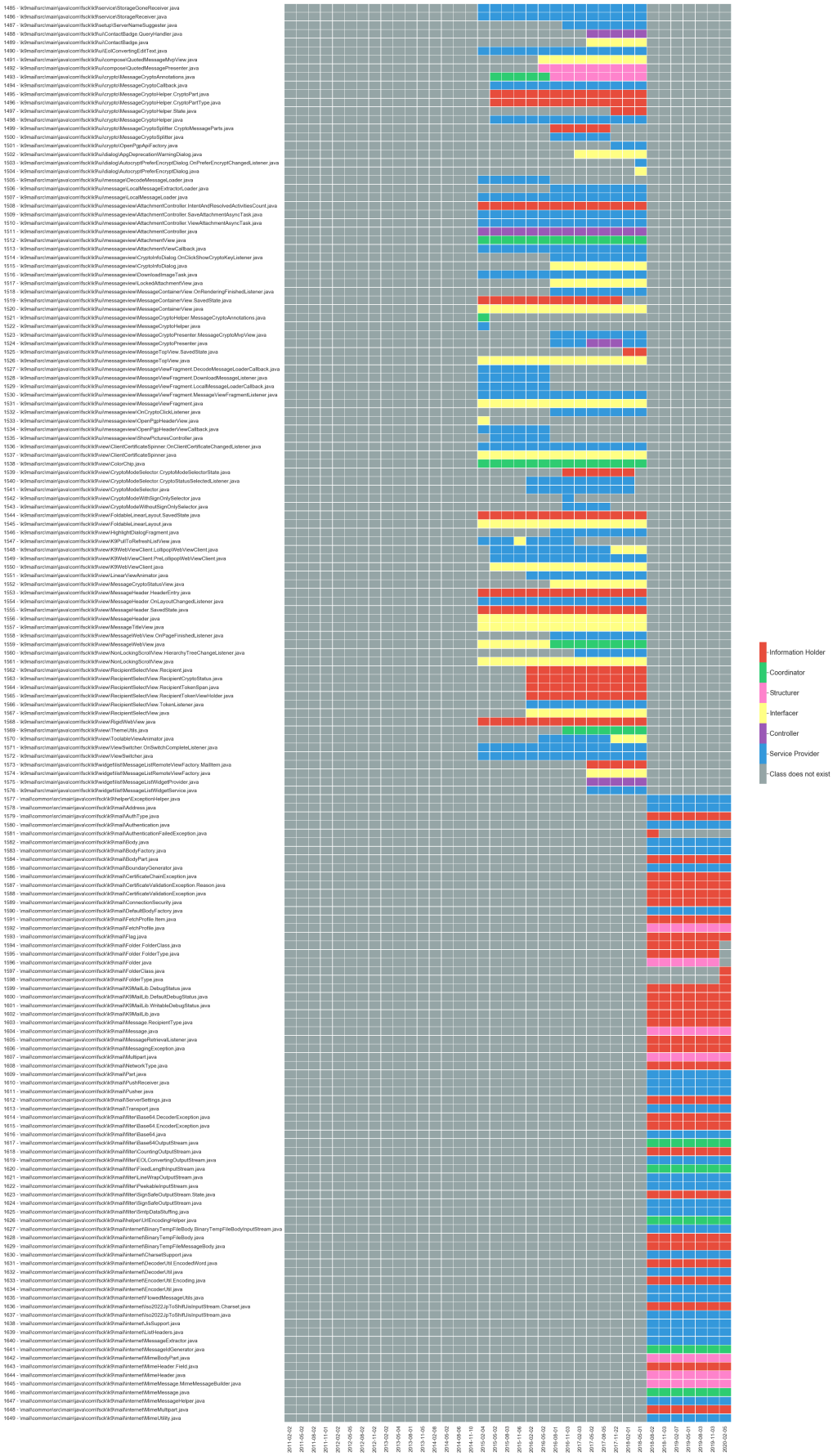

Fig. 52. How classes change role-stereotypes overtime in K9Mail (Tenth portion of classes)

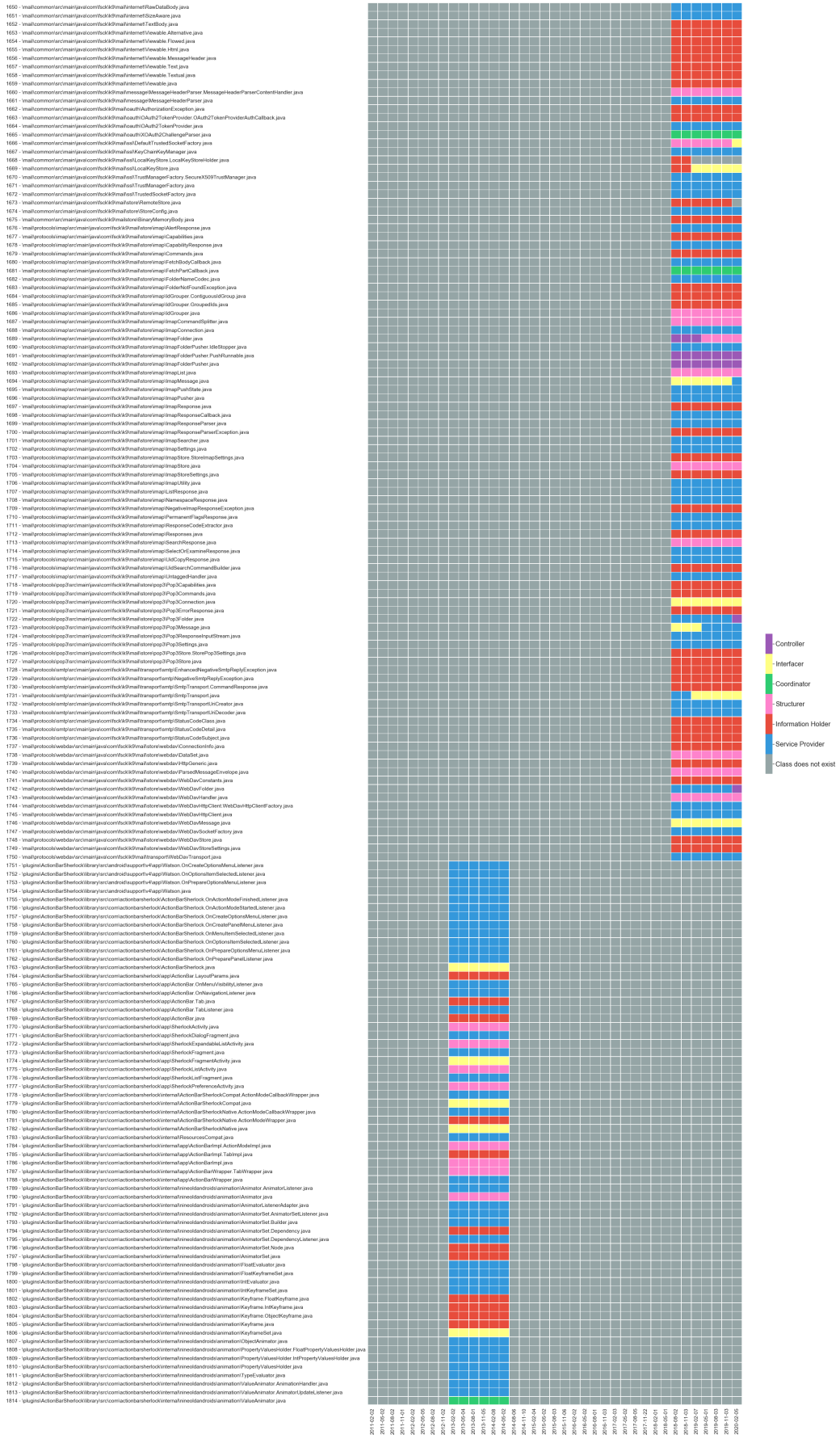

Fig. 53. How classes change role-stereotypes overtime in K9Mail (Eleventh portion of classes)

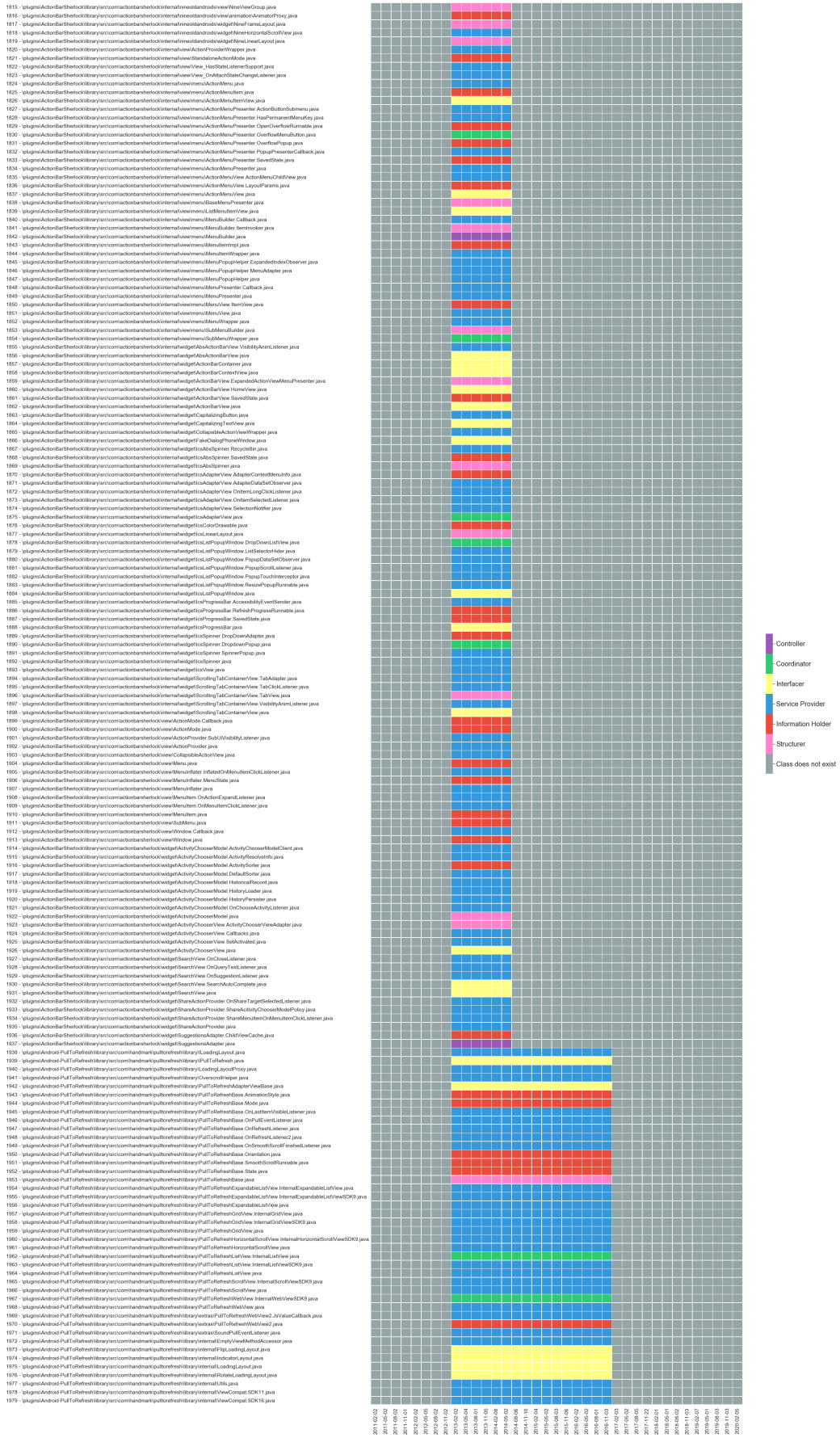

Fig. 54. How classes change role-stereotypes overtime in K9Mail (twelfth portion of classes)

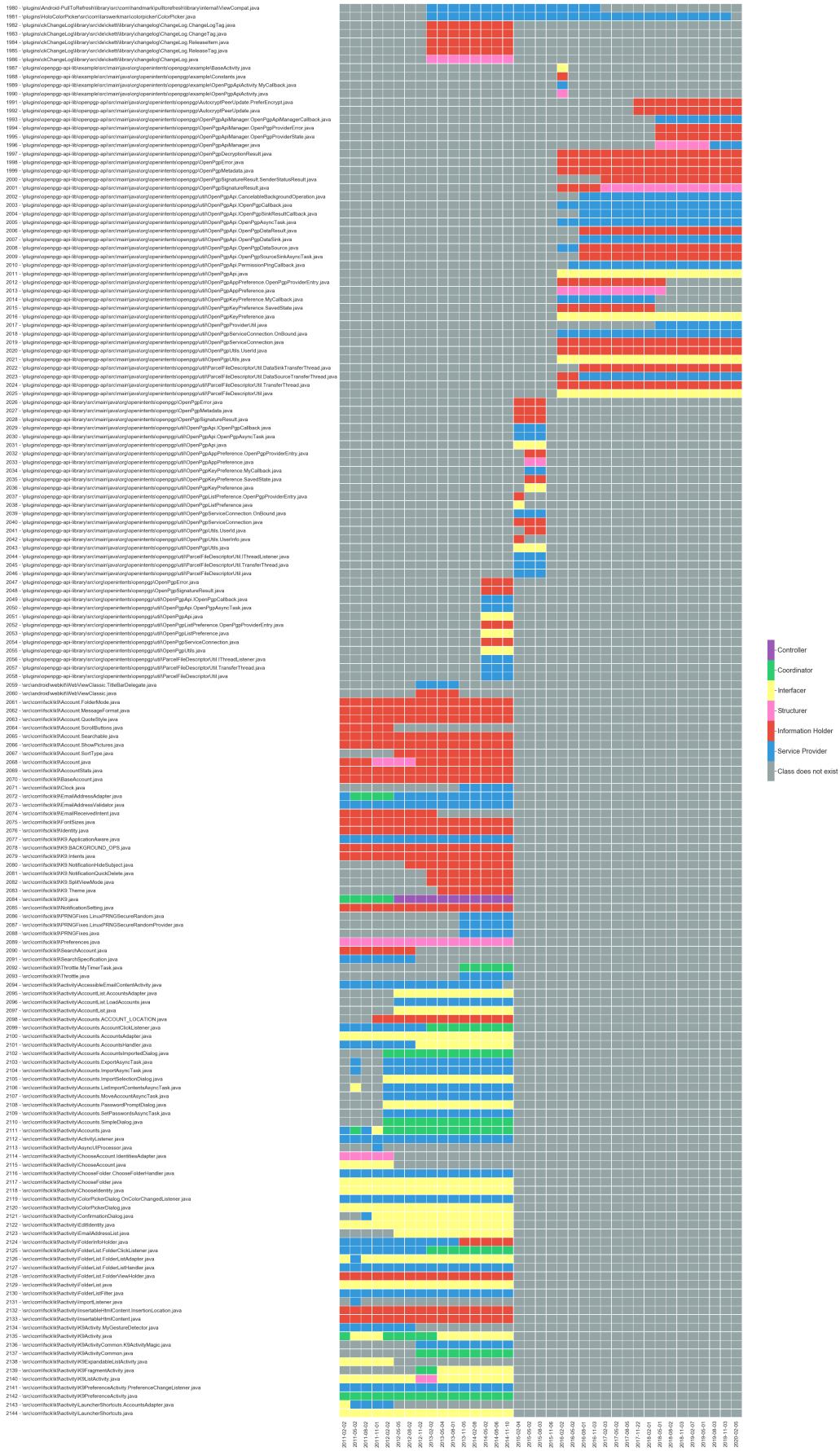

Fig. 55. How classes change role-stereotypes overtime in K9Mail (thirteenth portion of classes)

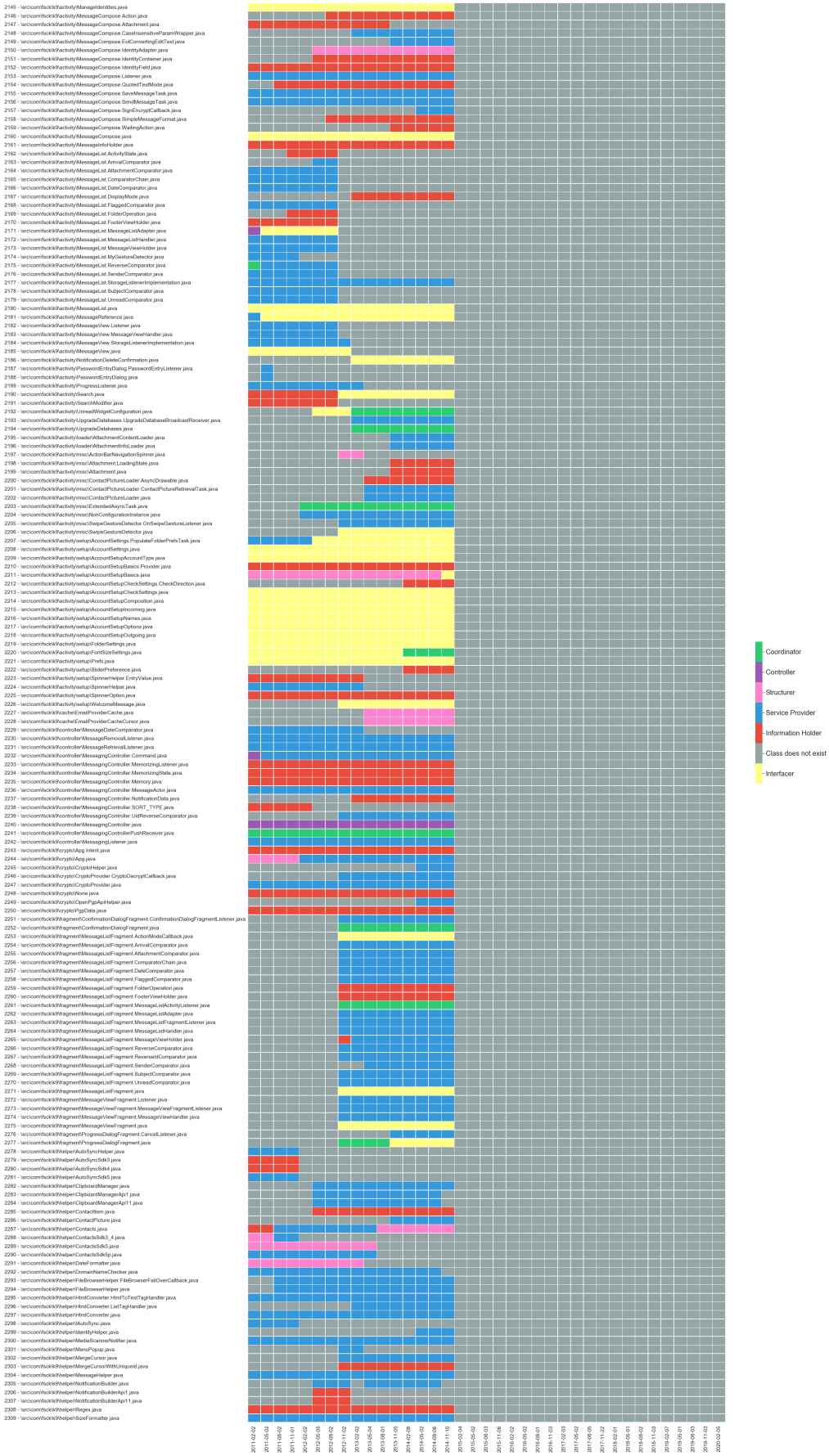

Fig. 56. How classes change role-stereotypes overtime in K9Mail (fourteenth portion of classes)

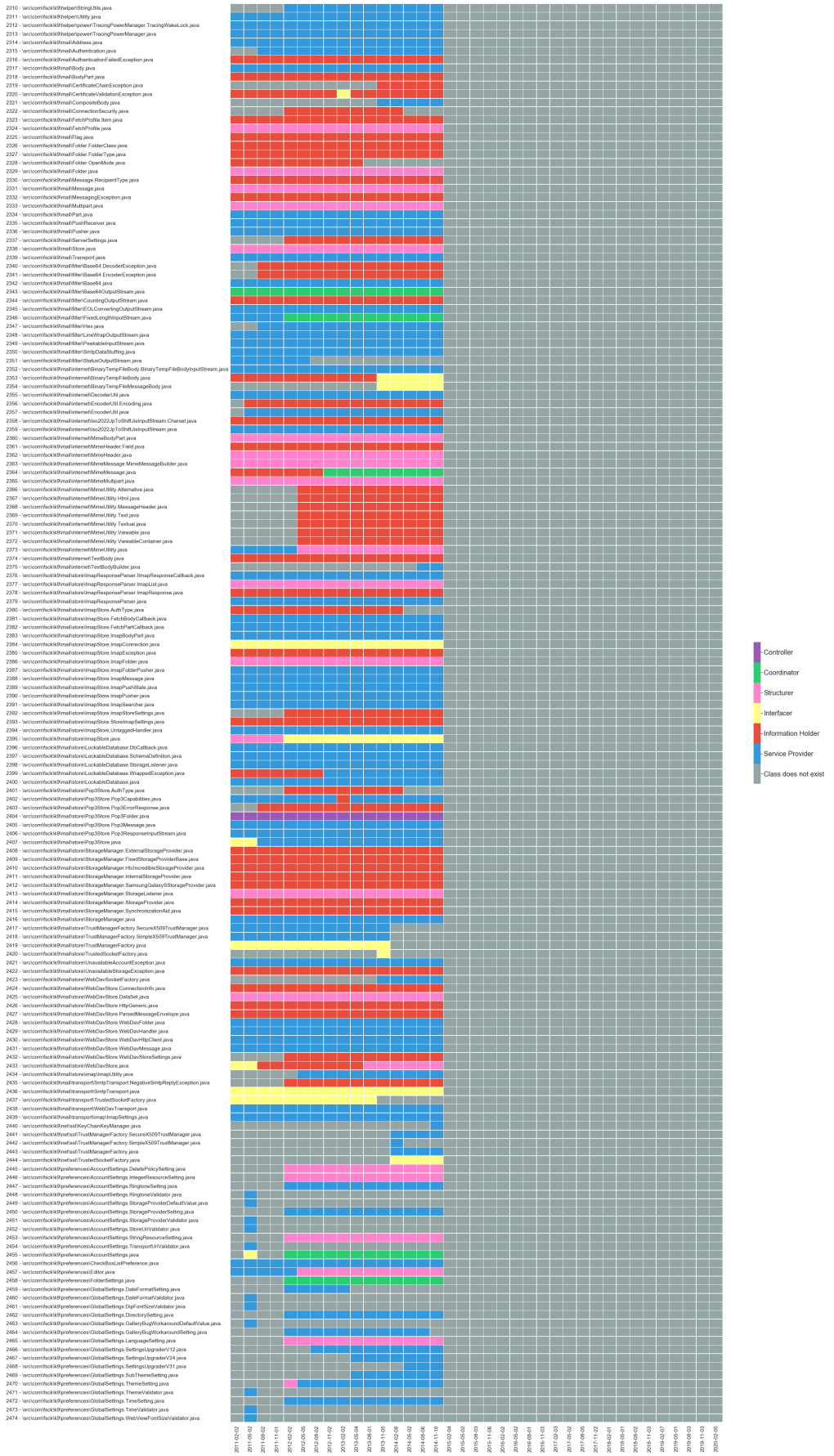

Fig. 57. How classes change role-stereotypes overtime in K9Mail (fifteenth portion of classes)

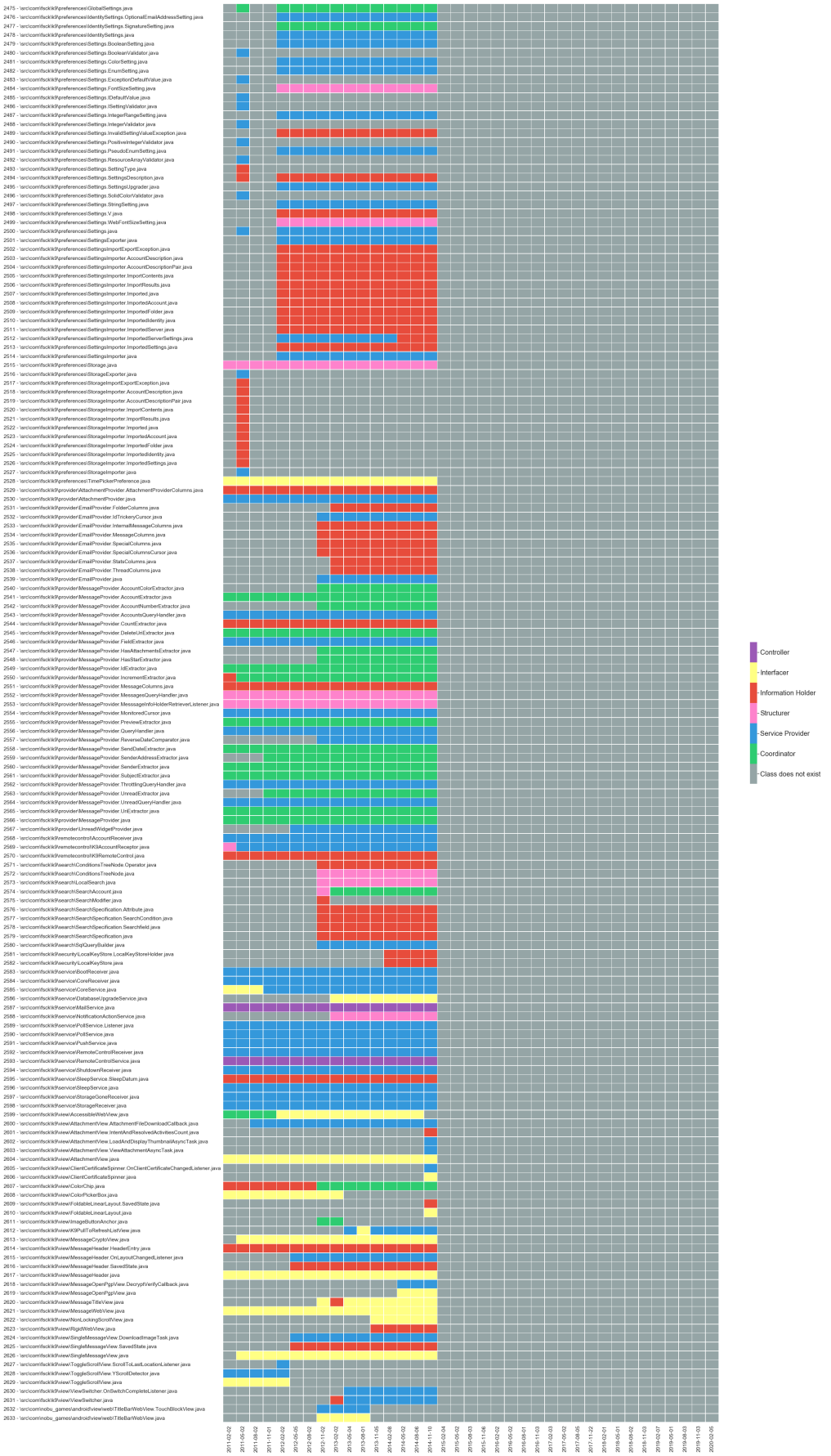

Fig. 58. How classes change role-stereotypes overtime in K9Mail (Last portion of classes)

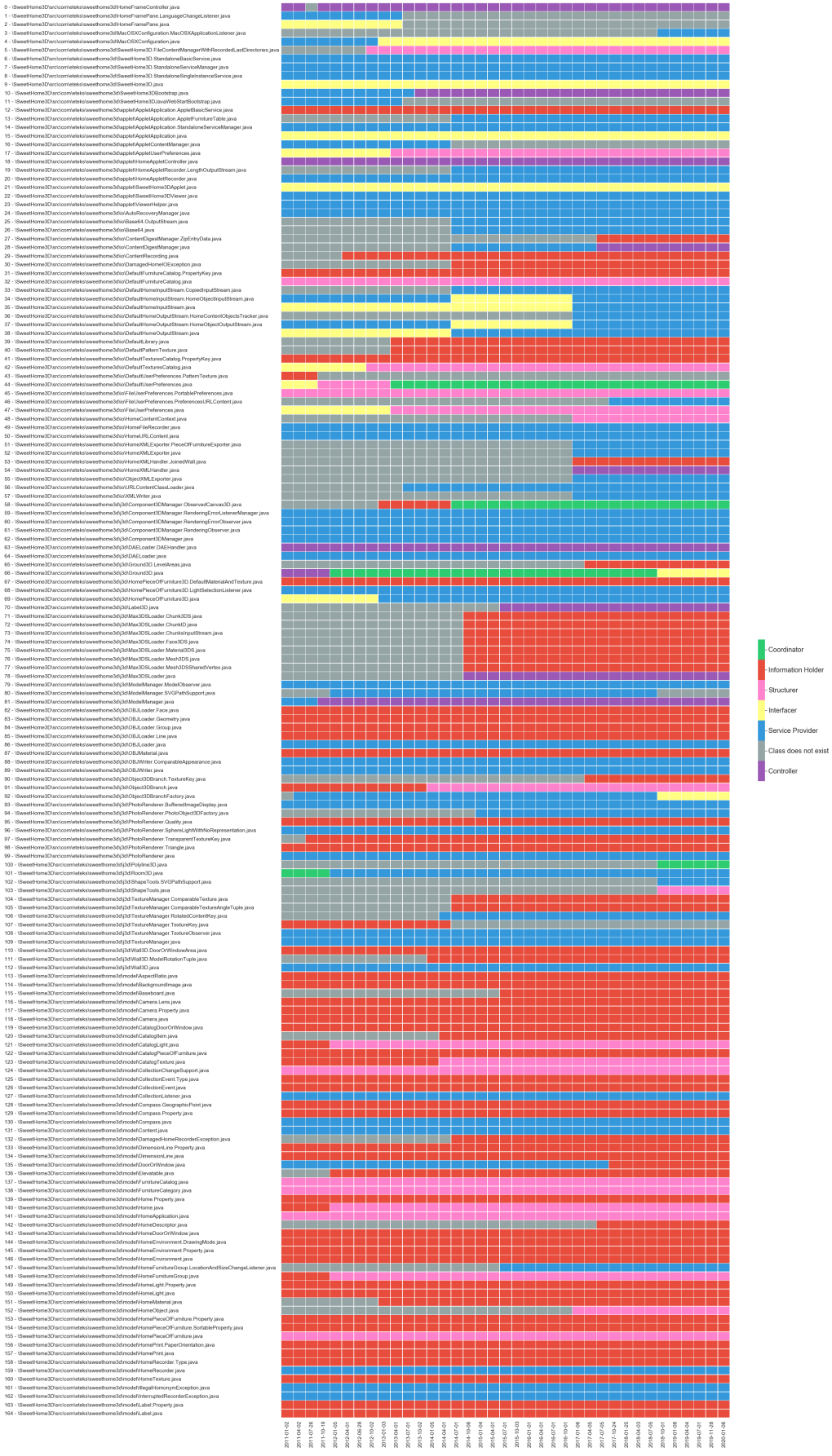

Fig. 59. How classes change role-stereotypes overtime in SweetHome3D (First portion of classes)

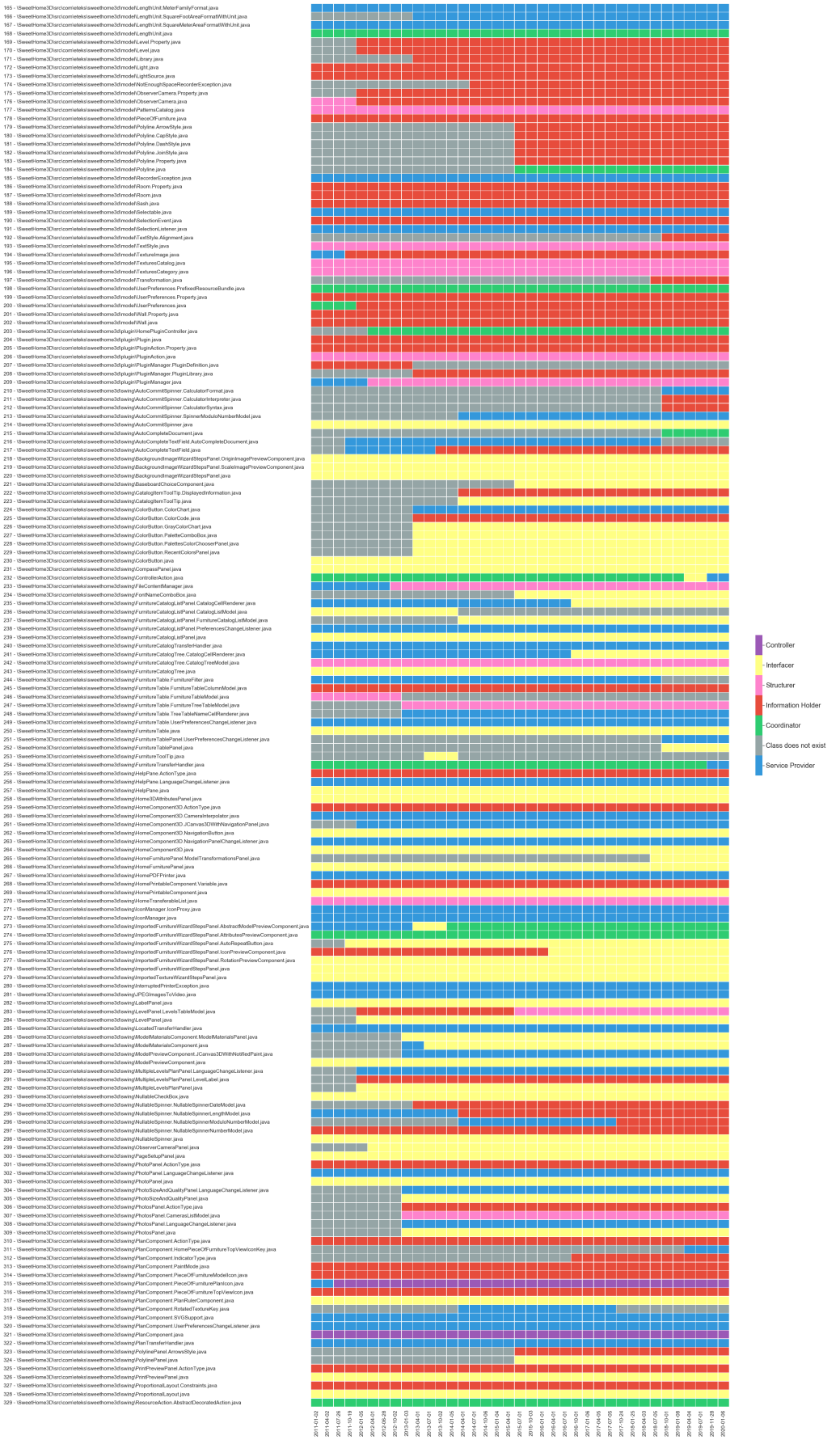

Fig. 60. How classes change role-stereotypes overtime in SweetHome3D (Second portion of classes)

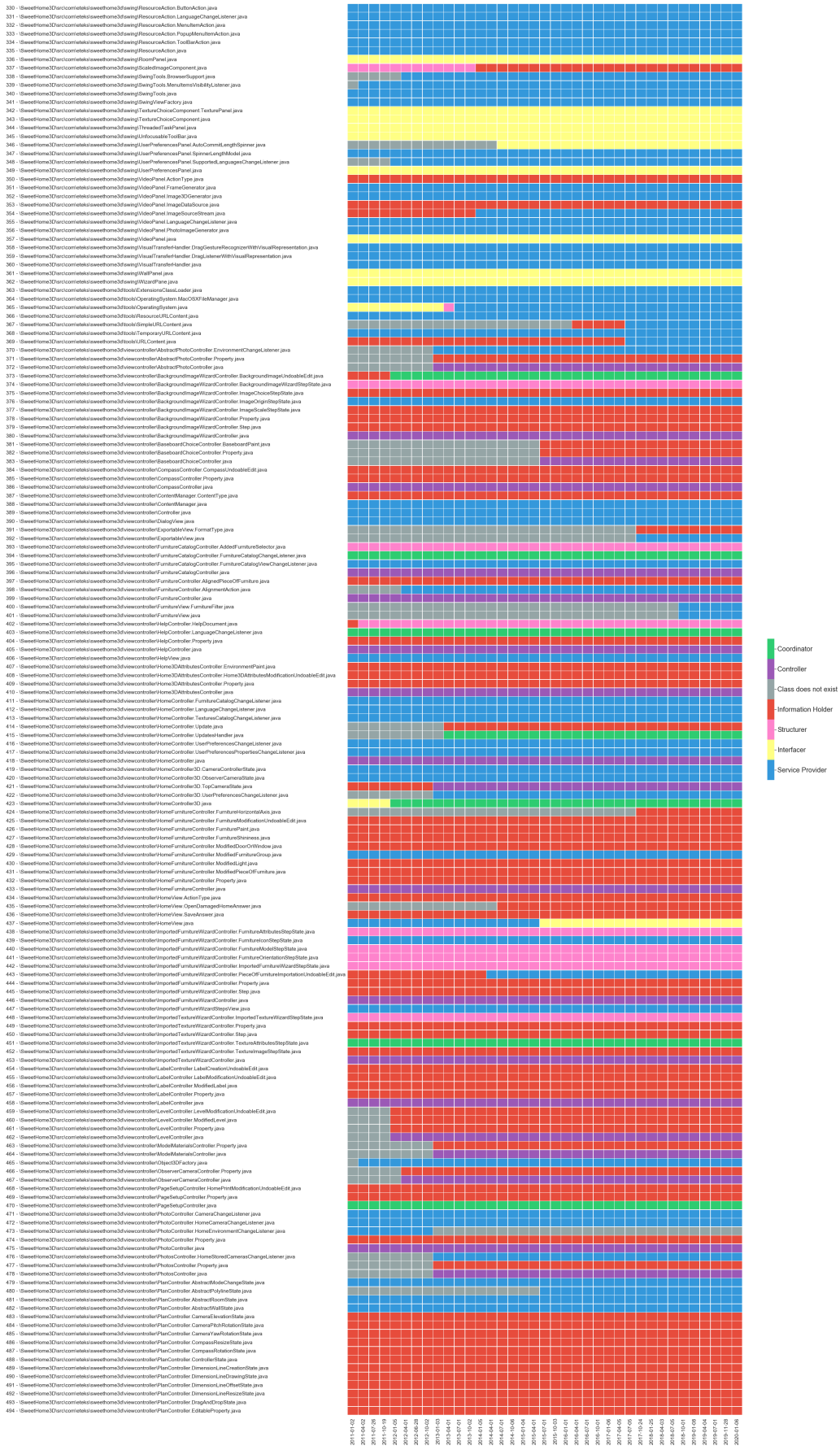

Fig. 61. How classes change role-stereotypes overtime in SweetHome3D (Third portion of classes)

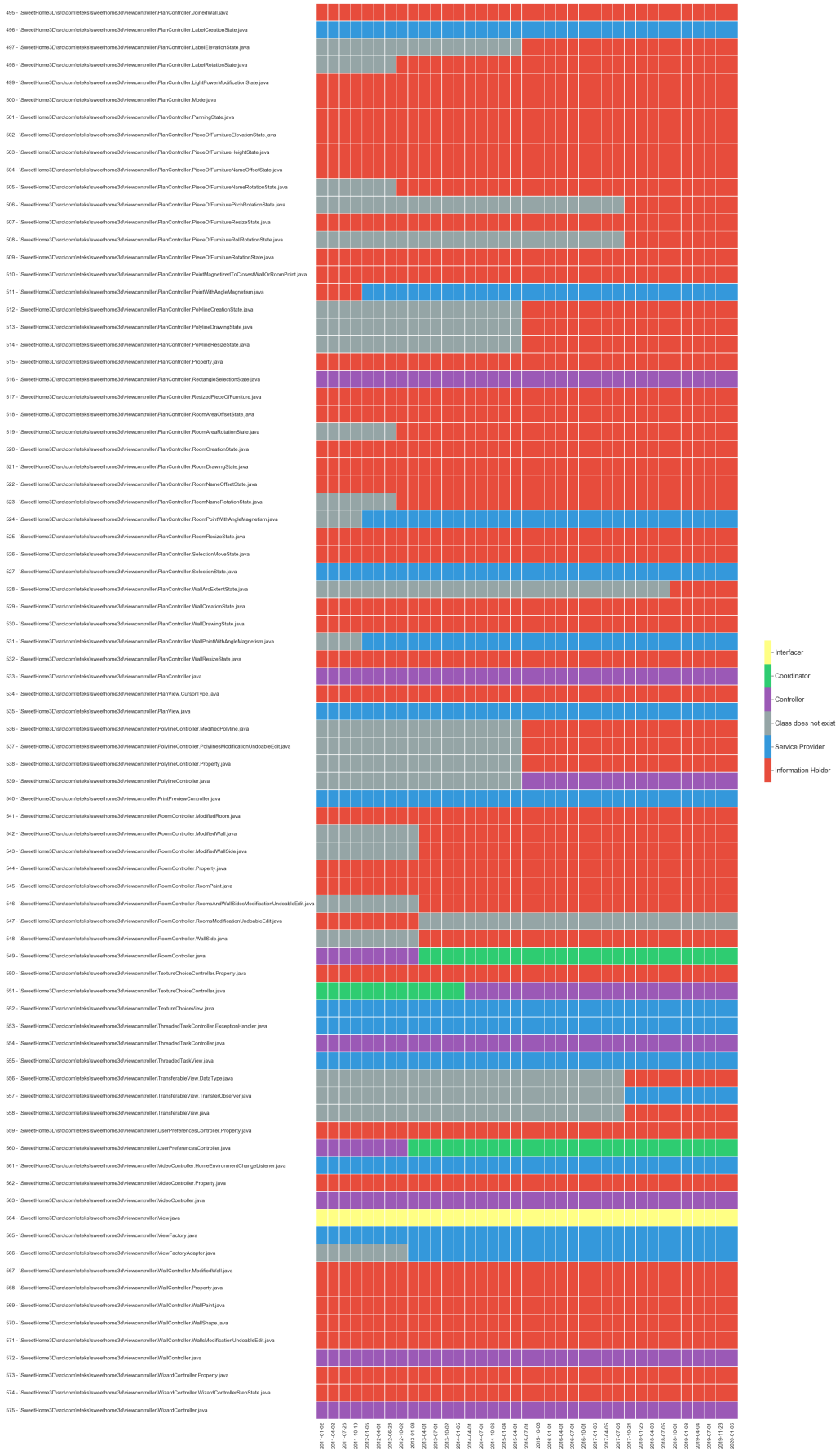

Fig. 62. How classes change role-stereotypes overtime in SweetHome3D (Last portion of classes)