# Runtime Evolution of Bitcoin's Consensus Rules

Jakob Svennevik Notland [1,1,1], Mariusz Nowostawski [2], and Jingyue Li [2]

[1]Norwegain University of Science and Technology
[2]Affiliation not available

October 31, 2023

# Runtime Evolution of Bitcoin's Consensus Rules

Jakob S. Notland, Mariusz Nowostawski, and Jingyue Li

**Abstract**—The runtime evolution of a system concerns the ability to make changes during runtime without disrupting the service. Blockchain systems need to provide continuous service and integrity. Similar challenges have been observed in centrally controlled distributed systems or mobile applications that handle runtime evolution, mainly by supporting compatible changes or running different versions concurrently. However, these solutions are not applicable in the case of blockchains, and new solutions are required. This study investigates Bitcoin consensus evolution by analysing over a decade of data from Bitcoin's development channels using Strauss' grounded theory approach and root cause analysis. The results show nine deployment features which form nine deployment techniques and ten lessons learned. Our results illustrate how different deployment techniques fit different contexts and pose different levels of consensus failure risks. Furthermore, we provide guidelines for risk minimisation during consensus rule deployment for blockchain in general and Bitcoin in particular.

**Index Terms**—Bitcoin, blockchain, consensus, grounded theory, root cause analysis, runtime evolution

## 1 Introduction

Deployment of consensus changes are the most important yet controversial [1], [2] and error-prone [3], [4], [5] activities in a blockchain. These changes redefine the fundamental behaviour of a blockchain, which can affect its security and the value of its currency. Trivial changes could cause disruption, which potentially results in suspended services [6], loss of mining revenue [4] and theft [7], [8].

Runtime evolution concerns the deployment of changes in a system while it is running. The concept is most relevant for critical systems that cannot afford to halt their services during an upgrade [9]. The critical system in the case of blockchains is a payment system which should constantly be operational. Typically, challenges in runtime evolution have been handled in distributed systems that can be centrally controlled. A single person can practically halt or roll back if any exceptions occur. Furthermore, distributed systems can handle different versions running concurrently [10].

In contrast, Bitcoin and other decentralised blockchains are autonomous systems that, by design, cannot be controlled centrally [11]. The design of blockchain enables immutability and constant uptime, which are principles conflicting with the ability to correct flaws. Blockchains also require all operational nodes to run compatible versions to be part of the same consensus agreement and consistently evaluate state changes. The differences between distributed and decentralised systems introduce new challenges in terms of technical and governmental aspects of runtime evolution.

This research is conducted to understand the implications of consensus rule changes, techniques for a safe transition and crisis management. There are two research questions, focusing on the current practice for consensus changes in blockchain and the lessons learned.

- RQ1: What techniques have been applied to deploy consensus changes?
- RQ2: What are the lessons learned from deploying consensus changes?

The motivation of this work is to collect knowledge on system evolution techniques from the Bitcoin financial system and to gather *unknown known* security requirements. *Unknown known* security requirements may have appeared as security incidents, but they are unknown to requirements engineers [12]. Therefore, our proposed techniques and security requirements may be well-known to a seasoned Bitcoin developer. However, any other blockchain engineer may have to shuffle through thousands of unstructured data samples to realise these security requirements. Rashid et al. suggest combining grounded theory (GT) analysis and incident fault trees to discover the root of these unknown known security requirements [12]. Similarly, we apply the Straussian GT [13] approach in combination with Ishikawa diagrams [14] for root cause analysis.

This study has been conducted as a qualitative analysis, covering 34 consensus rule changes over more than a decade of Bitcoin development and entails 1700 samples. Samples correspond to email threads, forum threads, Github issues/pulls, IRC days, formal and informal improvement proposals.

The results from RQ1 suggest nine features as building blocks for consensus evolution, e.g. a feature for the flag-day-like triggering of deployments. Additionally, we propose how the building blocks can be combined into nine deployment techniques, e.g. a blockchain community can use the Miner Activated Reduction Fork (MARF) technique to coordinate the super-majority deployment of backward-compatible changes.

The results demonstrate that deployment techniques pose a trade-off between changing the functionality of a blockchain and maintaining the consistency of the transaction data and the corresponding community cohesion. This means that blockchain engineers might have to choose between realising consensus changes and maintaining com-

- *J.S. Notland and J. Li are with the Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway. E-mail: jakob.notland@ntnu.no, jingyue.li@ntnu.no*
- *M. Nowostawski are with the Department of Computer Science, Norwegian University of Science and Technology, Gjøvik, Norway. E-mail: mariusz.nowostawski@ntnu.no*

*Manuscript received February 19, 2023; revised April 7, 2023.*

patibility with legacy implementations and consensus participation for all actors in the system. Sometimes participants in a blockchain community cannot agree on how to evolve functionality consistently. In these cases, the project might end up with forked chains such as Bitcoin Core (BTC) versus Bitcoin Cash (BCH).

Furthermore, with experience from the Bitcoin project, we create a systematisation of lessons learned to address RQ2. We identify ten lessons learned as follows to cover all the issues experienced during the evolution of Bitcoins consensus rules:

- Missing transformation assurance
- Improper reorganisation
- Improper human interference
- Too high thresholds
- Deploying 'irreversible' changes
- No prepared for forward compatibility
- Lacking knowledge regarding network dynamics
- Insufficient damage control
- Improper miner incentives to enforce new rules
- Insufficient incentives to review code

The main contributions of the study are:

- We propose unique features and deployment techniques to enable the safer evolution of consensus rules.
- We propose theories about the trade-offs between consistent consensus evolution and evolving functionality of the blockchains.
- We show which issues lead to consensus failure and suggest how to avoid and handle different crisis scenarios during deployment.

The remaining structure is as follows: Section 2 explains the background of blockchain and describes the nature of software and system evolution. Section 3 describes the research design and implementation. Section 4 presents the deployment features and techniques. Section 5 presents lessons learned. Section 6 discusses the results. Section 7 concludes the study and proposes future work. Throughout the article, we include inline quotes and related quotes in Appendix Section A to strengthen our claims.

## 2 Background

### 2.1 The Bitcoin consensus protocol

Bitcoin is "a peer-to-peer electronic cash system" [11] consisting of a chain of blocks containing transaction history, as illustrated by Figure 1. Any new block must abide by the consensus rules to be regarded as valid by the nodes in the network. Miners attempt different values for the nonce variable in a brute-force manner to produce a SHA-256 hash based on the entire block. The miners find a valid nonce for the block header when the resulting hash meets the required target difficulty. The difficulty indicates that the resulting hash must have a certain number of leading zeros. This mechanism is known as Proof-of-Work (PoW), first proposed to prevent email spam [15] and later applied for cryptocurrencies [16]. In the case of Bitcoin, PoW prevents Sybil attacks [17] and provides immutability of transaction data. The strength of these principles is preserved by the

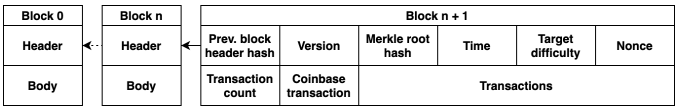


Fig. 1. A Bitcoin block that includes the header and the body.

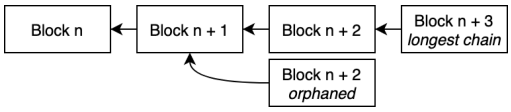


Fig. 2. Choosing the longest chain in the event of a collision.

difficulty adjustment algorithm [11], ensuring that the network will produce blocks with an average rate of around ten minutes. Miners must comply with further consensus rules. Generally, blocks and transactions must be in a valid format. Miners are incentivised to follow these rules by collecting fees and the block reward generated in a special coinbase transaction with no inputs from previous transactions.

Block collisions occur when two new valid blocks are produced at the same height at approximately the same time. Collisions are resolved by *the longest (valid) chain rule* as specified in Nakamoto's whitepaper: "The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it" [11]. The notion of a *valid* chain is important because validness is subjective from the implementation's point of view. Any collision should quickly resolve when another block is appended on top of one of the colliding blocks. This behaviour implies that blocks have a slight chance of being *orphaned* (discarded) as illustrated in Figure 2. The informal recommendation to prevent financial loss from orphaned blocks is to wait until the block containing the relevant transaction has at least six blocks built on top [18]. Collisions happen naturally, by an attack [19], or by inconsistent consensus validation [4].

### 2.2 Bitcoin consensus evolution

An inherent ideology within Bitcoin, especially BTC, indicates what changes are viable and which are controversial. One quote shown in Listing 1 from the creator(s) can be seen as a cornerstone of this ideology (other related quotes are in Section A.1).

```
The nature of Bitcoin is such that once version 0.1
was released, the core design was set in stone for
the rest of its lifetime.
```

Listing 1. Satoshi 2010-06-17 Forum, ID: 195

The statement from Nakamoto explains that the system itself, as well as the original specification [11] defines Bitcoin's fundamental behaviour. Moreover, it highlights the importance of non-disruptive changes, no matter how insignificant they seem. Our paper distinguishes between Bitcoin, Bitcoin Core (BTC) and Bitcoin Cash (BCH). Bitcoin is the idea of peer-to-peer electronic cash envisaged by Satoshi Nakamoto. BTC implements Bitcoin, preserves most compatibility with Nakamoto's original implementation, and contains the highest value [20] and consensus participation [21]. BCH is a minority fork that created an alternative chain by introducing a backward-incompatible consensus rule in 2017 [22].

The BTC and BCH communities have different approaches to consensus changes. Consensus rule changes in BTC should preferably allow backward compatibility such that legacy nodes can accept any new behaviour, keeping the network consistent (so-called soft forks). Backward-incompatible changes (so-called hard forks) are preferred in BCH and sometimes required in general if the fundamental implementation does not work as intended. Such a change could be essential to prevent exploits or allow for the adaption and survival of the system. As shown in Listing 2, the Bitcoin community is skeptical of consensus changes and making them a habit because bugged or ill-intended code may be deployed and disrupt the integrity and stability of the system (other quotes are in Section A.2).

```
5. Testing. I don't have time to personally test
every PULL request, but if a pull involves more
than trivial code changes I'm not going to pull it
unless it has been thoroughly tested. We had a very
good rule at a company I used to work for--
programmers were NOT allowed to be the only ones to
test their own code. Help finding money and/or
people for a dedicated "core bitcoin quality
assurance team" is welcome. More unit tests and
automated testing is also certainly welcome.

If this was open source blogging software I'd be
much less uptight about testing and code review and
bugs. But it's not, it is software for handling
money.
```

Listing 2. 2011-08-10 Gavin Andresen Email: bitcoin-dev

In this article, we refer to conflicting blocks as *chain splits*. Chain splits can be *temporary*; less than six blocks will be orphaned, *persistent*; six or more blocks are orphaned, or *permanent*; both chains are expanded independently for all foreseeable future. An accidental chain split caused by inconsistent validation among the nodes will be referred to as a *consensus failure*. The longest chain rule is the most fundamental factor in deciding whether a rule change is successfully adopted in Bitcoin. The longest chain rule implies that the majority of miners can apply a network-wide backward-compatible rule change. Relying on the majority is the preferred technique to deploy changes in BTC.

In Bitcoin, one may rely on the majority (>50%) of block-producing nodes ($n$) to perform a backward-compatible consensus change. The reason for this to work is that the Nakamoto consensus model has a Byzantine Fault Tolerance (BFT) [23] threshold of 50% ($f$). Other consensus models might have different fault tolerance, such as 33% or 20% [24]. This paper uses the term *super-majority* (SM) to generalize the minimal threshold requirement for different deployment techniques and to describe the required threshold where faulty nodes are equal to or less than the tolerated threshold. A super-majority of abiding nodes ($h$) is denoted as $h>(n\text{-}f)$.

Consensus changes in BTC's history have mainly been deployed with techniques such as IsSuperMajority (ISM) [25] and the more established miner-activated soft fork (MASF) [26]. Other well-known techniques are the user-activated soft fork (UASF) [27] or BCH's preferred user-activated hard fork (UAHF) [22].

### 2.3 Software evolution

Software evolution is a field entailing processes and models for changing software. Within this domain, there is the subfield of runtime evolution [9]. Relevant to blockchain and this study is mainly the challenge of avoiding service outages while changing the running system. In the case of blockchain, this also relies on consistency in the network, combined with the strict requirements of partition tolerance. Hence, minimizing the impact of a change on the consistency, availability, and partition tolerance of the system (CAP) [28] is challenging.

To distinguish and characterise consensus changes, we initially categorise consensus changes by different types of maintenance [9]:

- *Adaptive maintenance:* Change the deployed software to adapt to a changing environment.
- *Perfective maintenance:* Change the deployed software to perfect existing functionality by improving the user experience or performance.
- *Preventive maintenance:* Change the deployed software to avoid faults before they occur.
- *Corrective maintenance:* Change the deployed software to correct a fault discovered.

### 2.4 Evolution and maintenance in distributed systems

Before the decentralised networks, there were, and still are, distributed networks dominating sectors that blockchains have been envisaged to handle, such as finance [29], [30], logistics [31], [32], and healthcare [33], [34]. Researchers in these areas use several satisfactory frameworks for maintenance where updates are deployed with central control and rely on either being compatible [35] or running different versions in parallel [36]. In these cases, changes have been deployed with fast reboot, rolling upgrade, and big flip [10]. However, we argue that these methods are not directly applicable when deploying consensus rule changes on a blockchain. Firstly, known techniques are hard to coordinate without central administration. Secondly, a consensus rule change in a blockchain conflicts with legacy rules as it changes the set of valid actions. Some techniques may even drastically affect the stability of a blockchain, such as the total hash power in the network. Thirdly, running different versions in parallel is problematic because it may lead to consensus failures.

## 3 Research Design and Implementation

### 3.1 Research motivation and research questions

Blockchain systems must evolve to meet current and future requirements. Although there have been studies on different kinds of consensus code changes in blockchains [37], there continues to be a considerable lack of understanding of *how* these changes should be deployed safely, and *how* to handle failure modes. A flawed deployment can result in financial loss for any invested participant or loss of faith in the system, undermining the value of the contained cryptocurrency.

Throughout the history of Bitcoin, there have been cases of suspended services [6], lost mining revenue [4], and theft [7], [8]. One example shown in Listing 3 illustrates that consensus changes are possible reasons for critical incidents in the blockchain (Other related quotes are in Section A.3). Thus, timing and correctness are essential to minimize these

incidents' damage. Therefore, we investigate how consensus changes can be deployed (RQ1) and what lessons can be learned from past failures (RQ2).

```
Every time that you open up the door to changing
the rules, you are opening yourselves up to attack
```

Listing 3. 2020-08-03 Eric Lombrozo [38]

### 3.2 Research method

To answer RQ1, we used a qualitative and inductive approach to achieve an insightful and holistic view of consensus changes in blockchains. We chose Strauss' approach of grounded theory (GT) as it is more applicable to studies with predefined research questions [39]. The GT approach is an iterative and recursive approach where the researchers must go back and forth until they achieve theoretical saturation, i.e., when new samples stop contributing to the developing theories. The observations done throughout the study are covert [40], where a researcher can get the most authentic experience of how the actors conduct a process. Although covert observations can be ethically questionable, it is crucial to consider that the public archives of Bitcoin were created for this purpose and to provide transparency and accountability.

To answer RQ2, root cause analysis has been utilized to address lessons learned by drawing an Ishikawa diagram [14]. The approach is similar to "Discovering unknown known security requirements" [12] as it also uses concepts from GT and root cause analysis with incident fault trees.

### 3.3 Data collection and filtering

Considering the scope, the study started with purposive sampling [40] of data archives, specifically Bitcoin Core's development channels. Samples were discovered in these archives by purposive sampling and filtering relevant to consensus rule changes. The selected samples were efficiently sorted using flexible coding [41]. Further, we applied snowball sampling and triangulation [13] to avoid limitations from the initial samples. A graph of the research method implemented is outlined in Figure 3.

#### *3.3.1 Purposive sampling*

The initial step in collecting data related to the consensus changes in Bitcoin was to identify the consensus-changing events in Bitcoin. An exhaustive list of all the consensus changes throughout Bitcoin Core's history is listed in the Bitcoin Wiki [42] and in Table 1, highlighting eventual issues and maintenance types. Further inspection reveals an overview of the main events and information such as the time, block height, version number, and deployment techniques. Development channels were identified as the most fruitful sources, and their data was extracted to initiate the purposive selection of samples and the filtering process.

The channels that were selected as initial data were the Bitcoin improvement proposals (BIPs) [43], the bitcoin-dev emails [44], the Development & Technical Discussion topic in the Bitcoin forum [45], the code repository (pull requests [46] and issues [47]), and IRC channels (#bitcoin-dev and #bitcoin-core-dev [48]). The total sample count is 1700 and is available online [49]. One sample corresponds to one proposal, one thread (email, forum and GitHub), or one day of IRC messages. Figures 9 and 10 in Appendix Section A depict the distribution of samples over archives and events.

However, the sample boundary was fluid, so samples from other domains were considered whenever they appeared. These could be domains such as other Bitcoin channels, announcements, news articles, magazines, videos, or other cryptocurrencies. The additional sources primarily strengthened the theories rather than expanding them, which shows the relevance of the development channels initially selected by purposive sampling.

#### *3.3.2 Data filtering*

After acquiring an overview of consensus change events and corresponding BIP specifications, filtering was applied to the initial data. The first data considered were emails from the developers' mailing list. With a relatively compact overview of all the threads made, it was considered viable to traverse through the titles to purposefully select relevant samples.

After the email samples were included, the next step was the analysis of the forum texts. However, the forum source proved large and challenging to sort through. Therefore, a search was conducted in two stages: first, all the threads leading up to and surrounding the dates of each consensus incident were checked for relevance and purposively sampled. Second, the forum was filtered with the help of a search tool targeting the Bitcoin forum [50]. This filtering approach was also applied to the GitHub search. The relevant BIPs could be found as they directly correlate to consensus changes and deployment techniques.

We used the search strings in Table 1 to filter the Bitcoin forum and GitHub. We based our initial search on Bitmex's overview of consensus changes [51], which did not contain the consensus changes after 2017. We used unique identifiers to search for samples related to consensus changes: Title, BIP number, changed code, and block height. The consensus-changing events after 2017 were found later by snowball sampling.

Regarding IRC channels, the initial plan was to focus on the meetings conducted in Bitcoin's history and limit the number of samples collected in the initial phase. However, after getting familiar with information in Bitcoin meetings, it became clear that there were only a few meetings (four in total) before the developers decided on weekly meetings in the fall of 2015. Therefore, a search was conducted by applying a few search strings that could indicate conversations about the issues of consensus changes that have taken place. These strings were "fork," "chain split," "stuck," and "reorg". Using `grep` [52], the files were traversed all together, showing each related sentence. The whole log from each sample was collected whenever indicating some value or relevance.

The search strings used for IRC samples are fewer and more general than those used for the other archives. That is because almost as many IRC samples are available as there are days since the first conversations from 2010. Additionally, a single IRC sample often contains discussions on many different topics. Therefore, the search strings in Table 1 would provide too many samples, making them infeasible

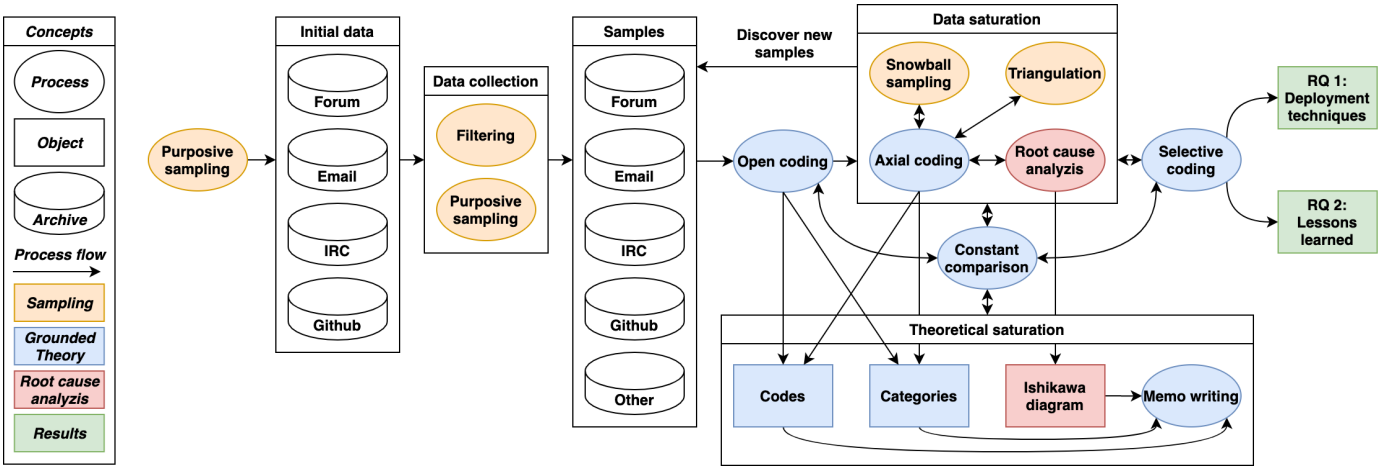


Fig. 3. Research method implementation. The oval shapes indicate processes, the lines show the process flow, the rectangular shapes indicate data objects and the cylinders indicate archives. Different colours highlight whether the concepts relate to data sampling, grounded theory, or root cause analysis.

TABLE 1
Overview of consensus changes in Bitcoin Core and applied search strings. **Bold** = blockchain specific issue.

| V | Consensus change | Maintenance type | Issues | Search strings |
|---|---|---|---|---|
| BTC 0.1.6 | Time based locking nLockTime | Corrective | No | nLockTime |
| BTC 0.3.5 | CVE-2010-5137 & CVE-2010-5141 | Corrective | No | CVE-2010-5137, CVE-2010-5141, 0.3.5 |
| BTC 0.3.6 | Disable/enable opcodes | Adaptive & preventive | Upgrading issues (bad dependency) | OP_CHECKSIG, OP_NOP, 0.3.6 |
| BTC 0.3.7 | Separate scriptSig & scriptPubKey | Corrective | No | scriptSig, scriptPubKey, 0.3.7 |
| BTC 0.3.10 | Output-value-overflow CVE-2010-5139 | Corrective | **Slow adoption & long split** | overflow bug, 184, CVE-2010-5139, 0.3.10, 74638, 74 638 |
| BTC 0.3.12 | 20 000-signature operation limit & 1MB blocksize | Corrective | **"Irreversible" change, source of controversy** | 0.3.12, MAX_BLOCK_SIZE, MAX_BLOCK_SIGOPS, 79400 & 79 400 |
| BTC 0.6.0 | BIP30: Duplicate transactions CVE-2012-1909 | Corrective | No | bip30, bip 30, duplicate transactions |
| BTC 0.6.0 | BIP16: Pay-to-script-hash | Perfective | **Stuck Nodes, slow adoption, conflicting proposals,** new bug | bip12, bip 12, bip16, bip 16, bip17, bip 17, bip18, bip 18, OP_EVAL, P2SH, pay-to-script-hash, 0.6, 0.6.0 |
| BTC 0.7.0 | BIP34: Include block height in coinbase | Corrective | No | bip34, bip 34, 0.7, 0.7.0, 227835, 227 835 |
| BTC 0.8.0 | BIP50: Migrate from Berkeley DB to LevelDB | Corrective & perfective | **Accidental fork, long split, financial loss, double spend** | bip50, bip 50, leveldb, berkeley db, 225430, 225 430, max 4 500, reduce blocksize, lock limit & 0.8 |
| BTC 0.8.1 | BIP50: Database lock limit & Max TxID limit | Corrective | No | (same as BTC 0.8.0) |
| BTC 0.8.1 | BIP50: Relax database lock limit & max TxID limit | Corrective | No | (same as BTC 0.8.0) |
| BTC 0.9.2 | BIP42: 21 million supply | Preventive | No | bip42, bip 42, 21 million, 21 000 000, 13,440,000, 13440000, 13 440 000 & 0.9.0 |
| BTC 0.10.0 | BIP66: Strict DER signature | Corrective | **Short splits, false signaling** | bip66, bip 66, strict der signatures, 0.10.0, 0.9.5 |
| BTC 0.11.0 | BIP65: Check lock time verify | Perfective | No | bip65, bip 65, check lock-time verify, CHECKLOCKTIMEVERIFY, CLTV , 0.10.4 & 0.11.2 |
| BTC 0.12.1 | BIP68, BIP112, BIP113: Check sequence verify | Perfective | No | bip68, bip 68, bip112, bip 112, bip113, bip 113, Check sequence verify, Relative lock-time, CHECKSEQUENCEVERIFY, CSV, 0.11.3, 0.12.1 |
| BTC 0.13.1 | BIP141. BIP143. BIP147: Segregated Witness | Perfective, corrective, adaptive & preventive | **High risk for chain split, slow adoption, conflicting proposals, potential network parition &** complicated change | bip141, bip 141, bip143, bip 143, bip147, bip 147, segwit, 0.13, bip91, bip 91, bip148, bip 148, segregated witness, 481824 & 481 824 |
| BTC 0.14.0 | CVE-2018-17144 | Perfective | new bug | - |
| BTC 0.16.3 | Fix CVE-2018-17144 | Preventive | No | - |
| BTC 0.21.1 | BIP341, BIP342, BIP343: Taproot | Perfective & adaptive | No | - |

to sort out. We focused on search strings indicating how developers assessed concerns on consensus failures during deployment.

The size of the resulting data set of IRC samples indicates that the search strings were accurate and broad enough to catch relevant samples. For instance, it revealed samples explaining how the deployment techniques were initially implemented and further developed to avoid failure. Additionally, the logs around important dates were purposively sampled and inspected. Finding relevant samples also became simpler from the autumn of 2015 as the weekly meetings could be collected. A challenge with the IRC logs has been that the logs are somewhat dispersed between different and inconsistent archives [53], [54], [55], [56]. Therefore, it became clear that all the different archives had to be considered when searching.

### 3.4 Data analysis

Although Strauss' approach [13] was promising to answer the research questions, it became apparent that the grounded theory approaches describe little on how modern data analysis tools should be utilized most effectively when performing analysis with a large number of samples. To cope with this, we applied flexible coding proposed in [41], which explains ways that large sets of data can be collected and coded through qualitative data analysis software (QDAS) such as Atlas.ti [57], NVIVO [58], or MaxQDA [59].

#### 3.4.1 Open coding

The indexing approach was applied as a specific form of open coding to analyse large datasets using QDAS according to the guidelines on flexible coding [41]. The purpose is to get an overview of the initial data and effectively define and evaluate codes and categories. In practice, the data was initially indexed on a sample basis to highlight the essence of each sample. The process of indexing was constantly evaluated as new codes and categories emerged. Memos were created to understand the correlation between codes, categories, and events. One specific code was used to highlight notable quotes that considerably impacted the results [41]. This code always overlaps with some other code and is labelled "aha" to signal the aha experience that these quotes represent. Quotes labelled with this code were constantly revisited and were candidates to present and support the content of this paper.

#### 3.4.2 Axial coding

The axial coding phase was conducted by revising, combining, and splitting codes and categories. Some codes were combined with decreasing levels of granularity, and others were split to increase granularity and enhance insights. This stage also developed patterns and relations within the codes and categories. as shown in Figure 4.

The open and axial coding processes also revealed issues highlighted in Table 1. A few issues in Table 1 are related to well-known general software engineering practices, such as introducing bugs caused by insufficient reviewing and testing. Other issues are blockchain-specific and should be analysed in depth. These issues include chain split, network partitioning, adoption, controversial changes, stuck nodes, conflicting proposals, and malicious behaviour.

#### 3.4.3 Selective coding

The process of selective coding would focus on saturating the main categories of the data, their correlation, and the root causes of blockchain-specific issues. We realize the causalities of where deployment issues are rooted and where they surface. The lessons learned were revised to reflect this, and the measurements to address these issues are summarized.

#### 3.4.4 Constant comparison and theoretical saturation

The codes, categories and emerging theories were constantly compared by controlling whether they made sense regarding the research questions and the objective domain. It was assumed that the samples collected in the initial search could lead to an adequate theory in this paper. However, there was always a possibility that the current data set was too narrow. Therefore, the techniques of data saturation, snowball sampling [60], and data triangulation [13] were systematically applied.

The data saturation process led to additional samples that were not found in the initial data collection phase. The findings could, for instance, be missing links from one of the resources used in the data collection, or it could be an article from the Bitcoin Project's website [61], a blog post, or a video. These new samples were systematically collected by applying the snowball sampling technique. The essence of the snowball sampling technique is to include samples found by references in the collected samples.

Data triangulation was applied between the different resources to see the same phenomena from different perspectives. It was possible that when something important happened in one place, there would probably be more to read about from other samples. These techniques gave a rich data set with multiple perspectives. For instance, the IRC chat had inconsistencies on the 2015 event (version 0.10.0), where the samples of 2015-06-03, 2015-06-04, and 2015-06-05 were missing. Many other samples also revealed how this specific consensus failure was caused by custom and lazy validation and spy-mining [62].

#### 3.4.5 Root cause analysis

The appliance of Ishikawa diagrams [14] as seen in Section B and Figure 11 in Appendix Section A further enhanced the analysis for RQ2 (lessons learned). The identified root causes were classified within the categories of human errors [63] to gain further insight. The different errors were applied as codes during the GT analysis as seen in Section 5. Complementing the grounded theory approach with root cause analysis gave higher confidence in covering relevant issues and gaining in-depth understanding.

## 4 Results of RQ1 (Deployment Techniques)

The data analysis revealed a chain of events, shown in the timeline in Figure 5, which summarizes 34 consensus-changing events, including 24 Bitcoin Core changes and 10 Bitcoin Cash changes. The figure indicates some issues where the deployment was performed by emergency, caused a chain split, or other problems. These issues are further assessed in Section 5 on lessons learned.

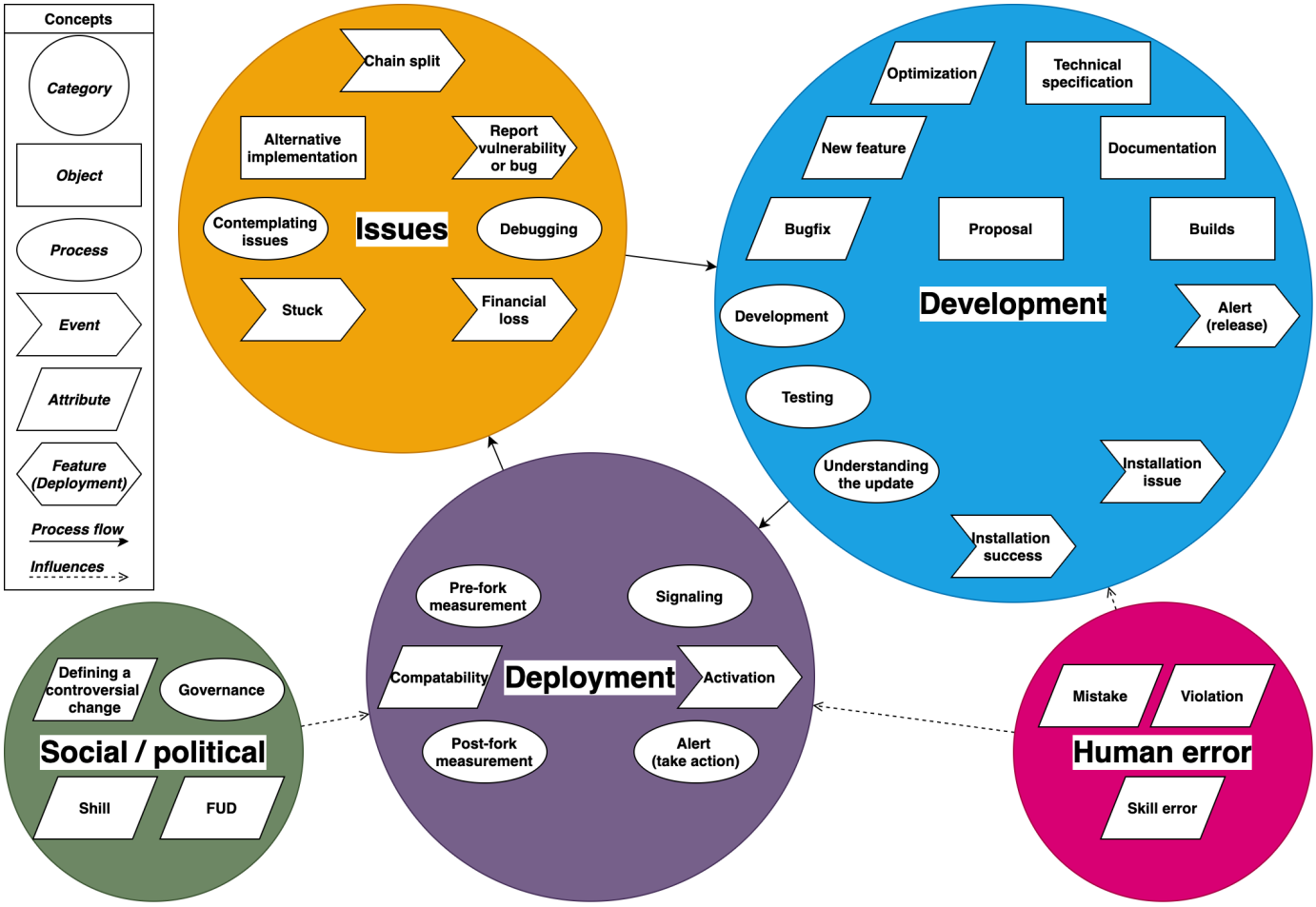


Fig. 4. Analytical codes and categories. The relation between the categories indicates a pattern in the cycle of consensus evolution.

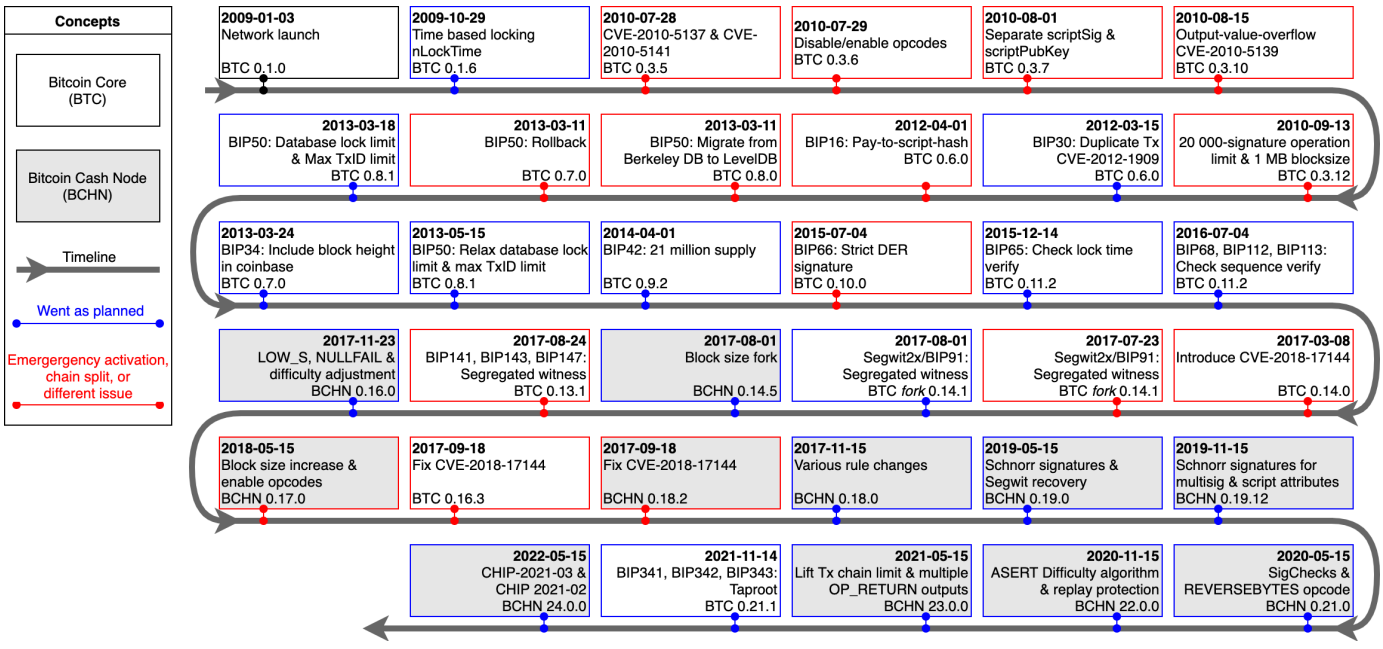


Fig. 5. The figure shows the timeline of consensus changes in Bitcoin Core and Bitcoin Cash. The date and order of these changes are based on either the flag day/block for the activation, the date where the changes were activated based on signal thresholds, or when these versions were released. The red boxes indicate some issues where the deployment was performed by emergency, caused a chain split, or other issues.

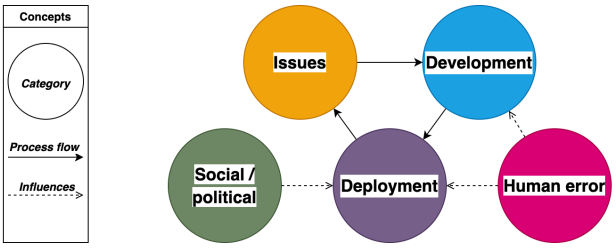


Fig. 6. The figure presents a minimal overview of analytical categories highlighting the pattern in the development cycle: An issue sparks the development processes, which eventually is realised by the deployment processes. The human error category could negatively impact both development and deployment, as further discussed in Section 5. The social/political aspects decide whether a consensus change fork will be deployed and which deployment techniques are utilised.

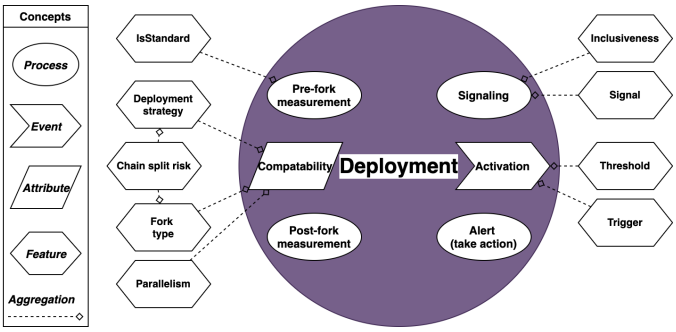


Fig. 7. The figure shows the feature-code relations. The features were derived from low-level analytical codes and showed consensus evolution's essential technical building blocks.

The categories discovered and applied through grounded theory also revealed the process of defining, implementing, and deploying consensus rules, as shown in Figure 6. In Bitcoin's case, these changes are usually motivated by some *issues* which prevent the implementation from providing the full service envisaged in Nakamoto's white paper or code. The discovery of an issue leads to *development* before moving on to *deployment*. The nine *features* for deployment were derived from the codes applied during analysis. Figure 7 illustrates how the deployment features were derived. These features are used in nine combinations to define different *deployment techniques* that answer RQ1.

## 4.1 Deployment features

The features for deployment are **Deployment strategy, Fork type, Chain split risk, Parallel, Standard, Signal, Inclusive, Threshold** and **Trigger**.

### 4.1.1 Deployment strategy

The **deployment strategy** feature defines whether nodes 1) depend on each other to coordinate the timing of an upgrade, i.e., miner-activated strategy. 2) The upgrade is forced regardless of miners' promised support, i.e., a user-activated strategy. 3) Deployment must be forced due to an imminent issue, i.e., emergency-activated strategy.

### 4.1.2 Fork type

Common terminology [64] describing different consensus rule changes in a blockchain distinguish between hard and soft forks. However, these concepts fail to provide an accurate description when considering the low-level details of a rule change. For instance, a hard fork has been established as a term for rule changes that results in a permanent chain split. However, this can also happen in a soft fork if a minority deploys the change. Therefore, Zamyatin et al's terminology [37] was adopted to accurately distinguish relevant **fork types:**

- *Expanding*: Changes that make previously illegal actions legal (commonly referred to as a hard fork).
- *Reducing*: Changes that restrict the set of valid actions (commonly referred to as a soft fork).
- *Bilateral*: Changes that deem all previous legal actions illegal and expand the rule set (commonly referred to as a hard fork).

Whenever deploying rule changes to a blockchain, one must consider the compatibility between new and old versions by understanding the **fork type** of the implementation. The main difference is that a reducing fork will be backward-compatible, allowing it to be enforced by the network with a super-majority of supporting hash power. Therefore, a reducing fork can be desirable as miners can keep the network consistent without relying on the whole network to perform the deployment. As Listing 4 indicates, BTC developers usually look for ways to implement changes as reducing forks since they have desirable compatibility attributes and are easier to digest for the network and the community (Other related quotes are in Section A.4).

```
I belief we shold flesh out luke-jr's idea for
cleanly deploying segregated witness in bitcoin as
a soft fork and see what that looks like.
```

Listing 4. Gmaxwell 2015-11-04 IRC: #bitcoin-dev

### 4.1.3 Chain split risk

The different fork types and deployment strategies imply different levels of **chain split risk**. This feature indicates how likely a prolonged chain split is. Applying additional deployment features can sustain chain splits' potential risk, length, and impact.

### 4.1.4 Parallel

Another compatibility issue is whether performing several deployments in **parallel** is possible. Deployments can be conducted in parallel if the rule changes are isolated and the deployment attributes are independent. Listing 5 shows an example that Bitcoin adopting parallel deployments after realising that non-parallelism could become a problem (Other related quotes are in Section A.5).

```
BIP 34 introduced a mechanism for doing
soft-forking (...). As it relies on comparing
version numbers as integers however, it only
supports one single change being rolled out at
once, requiring coordination between proposals, and
does not allow for permanent rejection: as long as
one soft fork is not fully rolled out, no future
one can be scheduled.
```

Listing 5. Pieter Wuille et al. 2015-10-04 Github, BIP9

#### 4.1.5 Standard

In addition to the consensus rules, nodes can utilize relay policies to specify what transactions they will include in their blocks and whether they are relayed to other nodes. This can be seen as softly enforced rules that can be applied at any rate before activating new consensus rules, allowing individual miners to avoid unwanted or experimental behaviour. A deployment conducted by gradually changing the policies before or after the rule deployment will be recognized by the **standard** feature.

#### 4.1.6 Signal

A **signal** is a feature used to signal the intention to upgrade and enforce new consensus rules in Bitcoin that are usually represented by the version bits in the block header or a string in the coinbase transaction message. Other implementations of on-chain signals observed are multi-signature commitments to the chain as implemented in Dash [65]. The signals make deployment more predictable and allow the measurement of total hash power support on the network. Listing 6 is one example describing the first approach to signalling to deploy pay-to-script-hash (Other related quotes are in Section A.6).

```
when 50% of the last N coinbases contain "I support
FOOFEATURE", it's enabled
```

Listing 6. Luke-jr 2011-10-02 IRC: #bitcoin-dev

On-chain signalling provides confidence that a significant portion of miners will behave according to the new rules. However, a signal can also come in other forms, such as a verbal agreement. This happened once in Bitcoin's history during the deployment of BIP 30, as illustrated in Listing 7 (Other related quotes are in Section A.7).

```
<gavinandresen> luke-jr: you're a mining pool
operator, would you be willing to coordinate with
the other big pools to get this fixed quickly[?]
```

Listing 7. gavinandresen 2012-02-17 IRC: #bitcoin-dev

#### 4.1.7 Inclusive

The signals are most helpful on-chain, where they can be interpreted by validating nodes to coordinate an upgrade. They can also be used to exclude blocks mined by non-signalling nodes to persuade them to upgrade and avoid unwanted behaviour. This behaviour is defined by the **inclusive** feature. An inclusive fork will continue to append blocks from miners that do not intend to validate by the new rules. In contrast, an exclusive fork will stop accepting blocks from miners who do not show intention to validate by new rules. This restriction can be lifted after a certain time or if the deployment fails.

#### 4.1.8 Threshold

Activation and enforcement of consensus rule changes can happen in stages, which can be controlled by the **threshold** feature. One deployment technique may implement several thresholds. For instance, the first threshold enforces rules for all signalling nodes. The second threshold enforces the rules for all nodes. Having several thresholds is a trade-off. On the one hand, the first threshold incentivises miners to stay true to their intention of validating by the new rules. On the other hand, every threshold is a potential trigger for consensus failure. This is possibly one of the reasons that Bitcoin ceased using two-stage activation by ISM (IsSuperMajority) [25]. The activation thresholds are most relevant for miner-activated strategies because they rely on coordination with other nodes. The thresholds should be at least the super-majority (>50% in Bitcoin) to preserve consistency, ensuring enforcement on the longest chain.

#### 4.1.9 Trigger

When the threshold is reached, the **trigger** feature will enforce activation. The activation is triggered dynamically, statically, or instantly. Using a rolling window to decide the timing for miner-activated strategies dynamically can be desirable. A rolling window trigger will determine the amount of support based on a number of recent blocks. The most primitive trigger can be based on a static flag day (FD) or block height (BH), as used for user-activated strategies and demonstrated by Nakamoto, as shown in Listing 8 (Other related quotes are in Section A.8). Instant triggers are utilized in the urgency of an emergency and are adopted as soon as they are rolled out to prevent or resolve exploits or consensus failures. The static and instant triggers have a higher risk as they do not guarantee that a super-majority of miners will prevent a chain split when the rules activate.

```
if (blocknumber > 115000)
   maxblocksize = largerlimit
It can start being in versions way ahead, so by the
time it reaches that block number and goes into
effect, the older versions that don't have it are
already obsolete.
```

Listing 8. Satoshi 2010-10-03 Forum, ID: 1347

### 4.2 Deployment techniques

Our first contribution is to define nine possible consensus change deployment techniques. These techniques have different qualities indicated by the inherited chain split risk. Our findings do not necessarily indicate that one technique is better than the other. Instead, some techniques are more viable than others depending on the context. Therefore we suggest the theory that choosing a deployment technique is a trade-off between the functionality and consistency of the blockchain and its community. This is a spectrum. On the one hand, if the whole community supports a fork, then it can be deployed without causing a chain split. On the other hand, if a fork does not have unanimous support, then different techniques can be used to persuade the network to move together or split the chain if the new functionality is more important than maintaining consistency.

Furthermore, deployment techniques preserve consistency and predictability for the parties involved in deployment. We summarize nine deployment techniques, shown in Table 3. The techniques are defined as a combination of two deployment features: 1) The fork type (*expanding*, *reducing*, or *bilateral*) and 2) the deployment strategy (*miner*, *user*, or *emergency*). For each deployment technique, the optimal combination to reduce the risk and impact of a chain split is shown using the remaining deployment features.

An overview of Bitcoin's evolution over time and the deployment features utilized are depicted in Table 2. This section presents the nine deployment techniques (Table 3) and

three special cases. The features in Table 3 are highlighted as **essential**, *useful*, or insignificant (-) for the corresponding technique. The only feature with consistent behaviour across all the deployment techniques is parallel because it can always be useful to allow several deployments in flight simultaneously.

#### 4.2.1 Miner-activated reduction fork (MARF)

The MARF deployment technique is the only technique with a low chain split risk when deployed using all the available deployment features. Like all miner-activated techniques, it should rely on coordination between miners in the network. The coordination is achieved by relying on signals from other miners to reach at least super-majority support. However, a higher threshold is desirable to reduce the chain split risk. A dynamic trigger ensures that the threshold is reached before triggering the activation.

#### 4.2.2 Miner-activated expansion fork (MAEF)

Expansion forks will cause legacy nodes to deviate from patched nodes when the behaviour of new rules appears in blocks. Thus the chain split risk is high, and the network will only stay consistent with 100% adoption. A super-majority threshold can reduce the impact of a split by keeping patched nodes together on the new chain from the time of activation. With less than super-majority adoption, the patched nodes will continue to follow the legacy chain as new blocks violating the legacy rules will be discarded due to the longest chain rule. This can cause frequent chain splits depending on the adoption percentage, as shown in Figure 8. The issue of low adoption can also be avoided by exclusiveness to enforce the discarding of all legacy blocks, causing patched nodes to follow their own path of the *valid* longest chain.

#### 4.2.3 Miner-activated bilateral fork (MABF)

Bilateral forks carry the property that patched nodes will never create valid blocks according to legacy nodes and vice versa. Therefore, it is only possible to avoid a chain split with 100% adoption. Choosing any activation threshold less than 100% carries less utility in limiting the impact of a chain split, and there can only be one split. However, it can be helpful to demand a certain amount of support to ensure that the patched nodes can provide sufficient security and reliable service for the patched network.

#### 4.2.4 User-activated reduction fork (UARF)

When moving over to the domain of user-activated forks, the deployment has a different objective. In contrast to keeping the network consistent, it is more important that the fork activate regardless. Therefore, such a fork does not require any threshold and should activate by a static trigger. Furthermore, the exclusion is essential since the updated consensus rules cannot be expected to reach a super-majority. Exclusion ensures that the deployment may only cause a single chain split. However, including legacy nodes will cause chain splits every time the new rules are violated. The standard feature can be essential for UARF to discriminate against upcoming rule-breaking transactions. As shown in Listing 9, feather-forking can be a viable technique to persuade other nodes to upgrade by actively attempting to orphan legacy blocks (Other related quotes are in Section A.9).

```
A feather-fork is when a miner refuses to mine on
any chain that includes a transaction it doesn't
like in the most recent several blocks.
```

Listing 9. Socrates1024 2013-10-17 Forum, ID: 312668

#### 4.2.5 User-activated expansion fork (UAEF)

UAEF requires all nodes to upgrade to avoid a chain split and is mainly applied when expecting full network adoption with high confidence. This property was observed as the preferable deployment technique for consensus changes in both BCH and Ethereum. In these cases, the forks have usually held high or unanimous support from the community. Changes are implemented in different node distributions and are expected to be adopted by the time of activation. Exclusion must be applied if there is any doubt of super-majority adoption. Otherwise, the upgraded nodes might follow the legacy chain even after adoption, as it might be the longest valid chain, as illustrated in Figure 8. However, if there is doubt, implementing a bilateral fork will be more beneficial in avoiding influence from legacy nodes, just like when BCH forked off BTC by UABF [22].

#### 4.2.6 User-activated bilateral fork (UABF)

The most outstanding example of a UABF is the activation of BCH. The fork became bilateral by demanding that the first block produced after activation was larger than 1 MB. Hence, legacy nodes would never accept the patched chain, and patched nodes would never accept the legacy chain. Bilateral forks ensure that a permanent chain split will commence and there will be two different cryptocurrencies.

#### 4.2.7 Emergency-activated reduction fork (EARF)

Emergency-activated deployment strategies are required when the implementation does not work as specified, and a consensus failure has already occurred or might occur. One of the earliest cases, when the implementation did not work as intended, was seen in BTC 0.3.10 with the overflow bug where a seemingly valid transaction could be created to generate additional bitcoins. An example of a consensus failure was when BTC 0.8.0 deployed a new database, and it caused a chain split. The third case, a potential exploit, can be illustrated by the inflation bug in version 0.14.0, which was discovered before being exploited. All of these deployments were EARFs. It is useful for EARF deployments to be inclusive to allow unpatched nodes to reorganize and generate blocks on the valid chain originating from the reduction fork when it becomes the longest.

An interesting observation in BTC 0.3.10 and 0.8.0 is that miners performed a rollback of blocks, deviating from the longest valid chain rule and Bitcoin's immutability property to reach consensus. First, the overflow bug was so severe that the consensus rules had to be changed such that the chain containing the malicious transaction would be rejected. During the consensus failure caused by the database deployment (BTC 0.8.0), there was a need to downgrade nodes even though the new chain was the longest and valid according to the specification. That was because it was the

TABLE 2
Deployed rule changes in BTC and BCH. Star(*) = Forked repository. Triggers: BH = Block height, FD = Flagday, and RW = Rolling window.

| V | Consensus change | Deployment Strategy | Fork Type | Chain split risk | Parallel | Standard | Signal | Inclusive | Threshold | Trigger |
|---|---|---|---|---|---|---|---|---|---|---|
| BTC 0.1.6 | Time based locking nLockTime | UA | RF | Medium | Yes | No | None | Inclusive | None | BH |
| BTC 0.3.5 | CVE-2010-5137 & CVE-2010-5141 | EA | RF | Medium | Yes | No | None | Inclusive | None | Instant |
| BTC 0.3.6 | Disable/enable opcodes | EA | EF & RF *hybrid* | High | Yes | No | None | Inclusive | None | Instant |
| BTC 0.3.7 | Separate scriptSig & scriptPubKey | EA | EF & RF *hybrid* | High | Yes | No | None | Inclusive | None | Instant |
| BTC 0.3.10 | Output-value-overflow CVE-2010-5139 | EA | RF | Medium | Yes | No | None | Inclusive | None | Instant |
| BTC 0.3.12 | 20 000-signature operation limit & 1 MB blocksize | UA | RF | Medium | Yes | No | None | Inclusive | None | BH |
| BTC 0.6.0 | BIP30: Duplicate transactions CVE-2012-1909 | MA | RF | Low | Yes | No | Offchain agreement | Inclusive | >50% | FD |
| BTC 0.6.0 | BIP16: Pay-to-script-hash | MA | RF | Low | Yes | No | Coinbase vote | Inclusive | 55% | FD |
| BTC 0.7.0 | BIP34: Include block height in coinbase | MA | RF | Low | No | No | VersionBits | Exclusive | 75%/95% | RW |
| BTC 0.8.0 | BIP50: Migrate from Berkeley DB to LevelDB | UA | EF *non-deterministic* | High | Yes | No | None | Inclusive | None | Instant |
| BTC 0.7.0 | BIP50: Rollback | EA | RF | Medium | Yes | No | None | Inclusive | None | Instant |
| BTC 0.8.1 | BIP50: Database lock limit & Max TxID limit | UA | RF *temporarily* | Medium | Yes | No | None | Inclusive | None | Instant |
| BTC 0.8.1 | BIP50: Relax database lock limit & Max TxID limit | UA | EF | High | Yes | No | None | Inclusive | None | FD |
| BTC 0.9.2 | BIP42: 21 million supply | UA | RF | Medium | Yes | No | None | Inclusive | None | Instant |
| BTC 0.10.0 | BIP66: Strict DER signature | MA | RF | Low | No | Yes | VersionBits | Exclusive | 75%/95% | RW |
| BTC 0.11.2 | BIP65: Check lock time verify | MA | RF | Low | No | No | VersionBits | Exclusive | 75%/95% | RW |
| BTC 0.12.1 | BIP68, BIP112, BIP113: Check sequence verify | MA | RF | Low | Yes | No | VersionBits | Inclusive | 95% | RW |
| BTC 0.13.1 | BIP141, BIP143, BIP146: Segregated Witness | MA | RF | Low | Yes | Yes | VersionBits service bits | Inclusive | 95% | RW |
| BTC 0.14.0 | CVE-2018-17144 | UA | EF | High | Yes | No | None | Inclusive | None | Instant |
| BTC* 0.14.0 | BIP148: Segregated Witness | UA | RF | Medium | Yes | Yes | VersionBits | Exclusive *temporarily* | None | FD |
| BTC* 0.14.1 | SegWit2x/BIP91: Segregated Witness | MA | RF | Low | Yes | Yes | VersionBits | Inclusive | 80% | RW |
| BCHN 0.14.5 | Block size fork | UA | BF | High | Yes | No | None | Exclusive | None | FD |
| BCHN 0.16.0 | LOW_S & NULLFAIL & difficulty adjustment | UA | EF & RF *hybrid* | High | Yes | Yes | None | Inclusive | None | FD |
| BCHN 0.17.0 | Block size increase & enable opcodes | UA | EF | High | Yes | No | None | Inclusive | None | FD |
| BCHN 0.18.0 | Various rule changes | UA | EF & RF *hybrid* | High | Yes | No | None | Inclusive | None | FD |
| BCHN 0.18.2 | Fix CVE-2018-17144 | EA | RF | Medium | Yes | No | None | Inclusive | None | Instant |
| BTC 0.16.3 | Fix CVE-2018-17144 | EA | RF | Medium | Yes | No | None | Inclusive | None | Instant |
| BCHN 0.19.0 | Schnorr signatures & SegWit recovery | UA | EF | High | Yes | No | None | Inclusive | None | FD |
| BCHN 0.19.12 | Schnorr signatures for multisig & script attributes | UA | EF & RF *hybrid* | High | Yes | No | None | Inclusive | None | FD |
| BCHN 0.21.0 | SigChecks & Reversebytes opcode | UA | EF | High | Yes | No | None | Inclusive | None | FD |
| BCHN 22.0.0 | ASERT Difficulty algorithm | UA | BF | High | Yes | No | None | Inclusive | None | FD |
| BCHN 23.0.0 | Lift transaction chain limit & Multiple OP_RETURN outputs | UA | EF | High | Yes | No | None | Inclusive | None | FD |
| BTC 0.21.1 | BIP341, BIP342, BIP343: Taproot | MA | RF | Low | Yes | Yes | VersionBits | Inclusive | 90% | RW |
| BCHN 24.0.0 | CHIP-2021-03 CHIP 2021-02 | UA | EF | High | Yes | No | None | Inclusive | None | FD |

TABLE 3
Deployment techniques. The notations indicate whether the features are required to reduce the chain split risk and the impact of a chain split: **Bold: Essential**, *Italic: Useful*, -: Insignificant. The abbrevations are: Miner-activated (MA), user-activated (UA), emergency-activated (EA), reduction fork (RF), expansion fork (EF), bilateral fork (BF), and super-majority (SM)

| Deployment strategy | Fork type | Chain split risk | Parallel | Standard | Signal | Inclusive | Threshold | Trigger | Example consenus rule change |
|---|---|---|---|---|---|---|---|---|---|
| MA | RF | Low | *Yes* | **Yes** | **Yes** | *Exclusive* | **SM** | **Dynamic** | Reduce max blocksize |
| MA | EF | High | *Yes* | *Yes* | **Yes** | *Exclusive* | **100%** | **Dynamic** | Increase max blocksize |
| MA | BF | High | *Yes* | *Yes* | **Yes** | - | **100%** | **Dynamic** | Increase max blocksize & set min blocksize >legacy blocksize |
| UA | RF | Medium | *Yes* | **Yes** | *Yes* | **Exclusive** | **None** | **Static** | Reduce max blocksize |
| UA | EF | High | *Yes* | *Yes* | *Yes* | **Exclusive** | **None** | **Static** | Increase max blocksize |
| UA | BF | High | *Yes* | *Yes* | *Yes* | - | **None** | **Static** | Increase max blocksize & set min blocksize >legacy blocksize |
| EA | RF | Medium | *Yes* | - | - | *Inclusive* | **None** | **Instant** | Reduce max blocksize |
| EA | EF | High | *Yes* | - | - | **Exclusive** | **None** | **Instant** | Increase max blocksize |
| EA | BF | High | *Yes* | - | - | - | **None** | **Instant** | Increase max blocksize & set min blocksize >legacy blocksize |

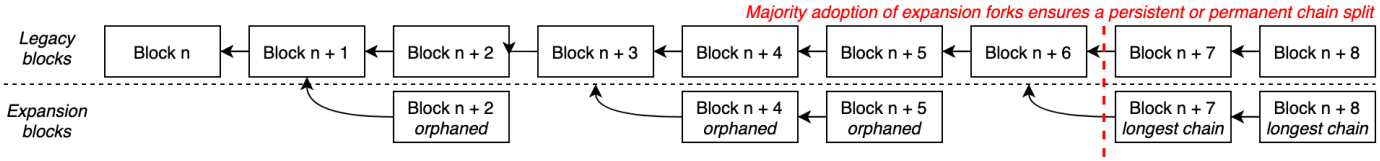


Fig. 8. Inclusive expansion forks can cause frequent chain splits before gaining super-majority adoption.

most conservative approach to keep compatibility with old nodes and because many merchants and users were likely to follow the legacy chain. Listing 10 shows that it was not obvious to downgrade and deviate from the longest chain rule. Furthermore, Listing 11 highlights that users and services depended on the legacy chain (Other related quotes are in Section A.10).

```
<Luke-Jr> gavinandresen: sipa: jgarzik: can we get
a consensus on recommendation for miners to
downgrade?
(...)
<gavinandresen> the 0.8 fork is longer, yes? So
majority hashpower is 0.8....
<Luke-Jr> gavinandresen: but 0.8 fork is not
compatible
```

Listing 10. Luke-Jr & gavinandresen 2013-03-12 IRC: #bitcoin-dev

```
Doesn't matter which chain is longer if a majority
of the people aren't on it. Breaking changes need
to be given lots of warning to be effective. Trying
to force everyone to use 0.8 would have only made
the situation worse. From the chat discussion, I
don't think mtgox was using 0.8. So trading at the
largest exchange would be halted until it could be
upgraded. If that doesn't sound disastrous, I'm not
sure what does.
```

Listing 11. nevafuse 2013-03-13 Forum, ID: 152470

There is no point in signalling or waiting for a certain threshold in the urgency of emergency activation. If the flaw is exploited, the triggering will happen naturally as soon as the bug in question triggers a consensus failure. The exclusion happens naturally because legacy nodes violating the rules of the EARF will be orphaned. In addition, gradual soft enforcement by standard relay policies becomes unnecessary since the rule changes of an emergency fork require instant consensus enforcement.

#### 4.2.8 *Emergency-activated expansion fork (EAEF)*

Deploying EAEF alone can be risky as it might not become widely adopted on a network basis. Miners might be reluctant to deploy a hasty and radical expansion of the consensus rules. Anything less than 100% adoption could cause a permanent chain split if never fully adopted. This encourages miners rather perform an emergency-activated reduction fork if feasible, as it is the safer alternative.

Figure 8 illustrate the problem that patched nodes might keep jumping back to the legacy blockchain as long as that is the longest. This will eventually be resolved as soon as the super-majority of miners work on the expanded blocks, and that chain will become the longest. However, legacy nodes will still work on the legacy chain as they do not see the expansion blocks as valid. The BIP50 consensus failure caused by BTC 0.8.0 might have looked somewhat like the EAEF figure before making a persistent chain split, although that cannot be assessed without access to the orphaned blocks.

#### 4.2.9 *Emergency-activated bilateral fork (EABF)*

This deployment technique has not been observed in any known upgrade. However, one could imagine the BCH fork being deployed with EABF as a reaction to revert the SegWit deployment. In that case, the patched chain would have to roll back to a block before the first SegWit-block was created and create a conflicting block.

#### 4.2.10 *Special cases*

In addition to the nine deployment techniques, there are three special cases. These cases fit into more than one of the defined techniques:

- Temporary reduction forks (related to MARF, UARF, and EARF)

```
2056 // Special short-term limits to avoid 10,000 BDB lock limit:
2057 if (GetBlockTime() >= 1363867200 && // start enforcing 21 March
         2013, noon GMT
2058     GetBlockTime() < 1368576000) // stop enforcing 15 May 2013
             00:00:00
2059 {
2060     // Rule is: #unique txids referenced <= 4,500
2061     // ... to prevent 10,000 BDB lock exhaustion on old clients
2062     set<uint256> setTxIn;
2063     for (size_t i = 0; i < vtx.size(); i++)
2064     {
2065         setTxIn.insert(vtx[i].GetHash());
2066         if (i == 0) continue; // skip coinbase txin
2067         BOOST_FOREACH(const CTxIn& txin, vtx[i].vin)
2068             setTxIn.insert(txin.prevout.hash);
2069     }
2070     size_t nTxids = setTxIn.size();
2071     if (nTxids > 4500)
2072         return error("CheckBlock() : 15 May maxlocks violation");
2073 }
```

Listing 12. Code snippet illustrating a temporary reduction fork from BTC 0.8.1 [66]

- Hybrid deployment (related to all deployment techniques)
- Non-deterministic forks (relate to all deployment techniques)

In addition to ordinary activation at time `T`, a temporary reduction fork has a *predefined deactivation time*. BTC 0.8.1 demonstrated a temporary reduction fork shown in Listing 12. The code defines a temporal reduction that was activated on 2013-03-21 and deactivated on 2013-05-15 (lines 2057 & 2058). The code for the temporary reduction counts transaction IDs (TxIDs) (lines 2062-2069) and enforces the limitation (lines 2071 & 2072). The limit of 4,500 TxIDs was assumed low enough to avoid reaching the database lock limit of 10,000.

Temporarily reduction forks can also be illustrated by an example from Bitcoin's legacy: The 1 MB limit was initially applied as a reduction fork. However, expanding that limit would not require an expansion fork if the limit had a predefined deactivation time. Then the community would have years to find a solution or delay the issue by another temporary reduction fork before the end time. Legacy nodes can still accept all blocks created under the reduction fork, while patched nodes will know the start and end-time.

A hybrid deployment is another special case where different fork types are deployed together. This technique inherits the attributes of the most disruptive fork type regarding chain split risk. That is in the following order: BF>EF>RF. Hybrid deployment with combinations of expansion and reduction forks has become a relatively common practice in BCH, which performed hybrid deployments in BCHN 0.16.0, BCHN, 0.18.0, and BCHN 0.19.12. Combining several forks into one deployment is practical because it limits the number of deployments where the network is exposed.

The non-deterministic forks are best explained by the example of BIP50 and the upgrades deployed with BTC 0.8.0 and BTC 0.8.1. The implementations contained a MAX_BLOCK_SIZE of 1 MB. However, this rule was often overrun by the default database locks setting in pre-0.8.0 nodes that were too small to handle certain large blocks containing many transactions. The problem would surface long before the consensus failure because blocks used too many locks led to reorganisation. This caused many nodes to run custom configurations. Listing 13 shows the problem surfacing one year before the consensus failure and that some miners had to set custom lock limits (Other related quotes are in Section A.11).

```
<TD> EXCEPTION: 11DbException
<TD> Db::put: Cannot allocate memory
<TD> bitcoin in ProcessMessage()
<TD> ProcessMessage(block, 5798 bytes) FAILED
<TD> received block 00000000000001c0a13e
<TD> REORGANIZE
(...)
<DrHaribo> sturles said he got out of memory errors
without being out of memory, and that adding locks
and lockers fixed it
```

Listing 13. sipa, TD & DrHaribo 2012-03-10 IRC: #bitcoin-dev

Furthermore, the Berkeley Database would behave inconsistently depending on the underlying hardware. The result is that chain splits and stuck nodes appear non-deterministic. Listing 14 describes how nodes running identical code would result in a non-deterministic fork depending on how the blockchain is stored on disk (Other related quotes are in Section A.11).

```
(...) contents of each node's blkindex.dat database
is not identical, and the number of locks required
depends on the exact arrangement of the
blkindex.dat on disk (locks are acquired per-page).
```

Listing 14. Gavin Andresen 2013-03-20 BIP50

When the database was changed in BTC 0.8.0, the new implementation would handle the locks differently and always be able to handle the edge-case blocks. The legacy nodes with custom lock limits would also handle these blocks. On the contrary, a non-deterministic set of the legacy node implementations would regard these blocks as invalid, causing a chain split. The fork was non-deterministic because of the inconsistent compatibility to blocks among legacy nodes running the same protocol.

## 5 Results of RQ2 (Lessons Learned)

All consensus rule changes in a blockchain can be a liability as they increase the attack surface. The deployment process itself can disrupt the community as conflicts arise. Lessons learned from Bitcoin deployments are synthesised to minimise the risk of future deployments in any blockchain. The GT and root cause analysis derive these lessons as seen in the Ishikawa diagram in Figure 11 in Appendix Section B.

The human error categories [63] were used as codes in the GT analysis to classify the issues discovered in the root cause analysis. These are 1) Skill-based errors, i.e., execution failure: Slips and lapses. 2) Mistakes, i.e., planning failures: Rule-based (RB) mistakes and knowledge-based (KB) mistakes. 3) Violations: Routine violations, e.g., laziness and 4) exceptional violations, e.g., sabotage. Table 4 shows the lessons derived, the impacted deployment features, their corresponding error categories, and the affected Bitcoin versions.

### 5.1 Missing transformation assurance

The most dangerous forks are those deployed by accident. They occur either because existing consensus rules are exploitable or new rules are deployed by accident. Accidental

TABLE 4
Lessons learned, impacted features, corresponding error categories and impacted Bitcoin versions.

| ID | Lesson | Impacted features | Error categories | Negatvely affected versions |
|---|---|---|---|---|
| L1 | Missing transformation assurance | Fork type | RB Mistake | BTC 0.8.0, BTC 0.14.0, BCHN 0.17.0 |
| L2 | Improper reorganisation | - | Lapse, KB Mistake | BTC 0.3.10, Before BTC 0.8.0, BTC 0.8.0 |
| L3 | Improper human interference | Trigger | Lapse, RB mistake, KB mistake | BTC 0.6.0 |
| L4 | Too high thresholds | Threshold, inclusive & deployment strategy | RB Mistake | BTC 0.13.1 |
| L5 | Deplyoing 'irreversible' changes | Fork type | KB Mistake | BTC 0.3.12 |
| L6 | Not prepared for forward compatibility | Fork type, standard & parallelism | KB Mistake | BTC 0.7.0, BTC 0.10.0, BTC 0.11.2 |
| L7 | Lacking knowledge regarding network dynamics | Signal | KB Mistake | - |
| L8 | Insufficient damage control | - | KB Mistake | BTC 0.8.0 |
| L9 | Improper miner incentives to enforce new rules | - | Routine violation, Exceptional violation | BTC 0.10.0 |
| L10 | Insufficient incentices to review the code | - | Routine violation | BTC 0.8.0, BTC 0.14.0, BCHN 0.17.0 |

forks are not safely deployed using the deployment features and will have a high risk of a chain split. This error is seen as an RB mistake because developers misclassify the fork-type feature of the given code change. As proposed in Listing 15, the most obvious remedy is to perform extensive testing and review, although further assurance is required (Other related quotes are in Section A.12).

```
The review process is definetly a good idea, I dont
know if it provides as much security as people
assume it does. One thing that slip past one person
may as well slip past ten people or whatever.
```

Listing 15. 2020-06-23 Luke-Jr [67]

The lack of transformation assurance in Bitcoin has caused an accidental chain split on one occasion (BTC 0.8.0) and allowed a serious bug to enter the code (BTC 0.14.0). However, Bitcoin has never had an accidental chain split caused by compatibility issues due to cross-node implementation, although the split in Ethereum's Berlin UAEF [68] demonstrates this. **To avoid bad code from entering deployment and accidental chain splits, techniques to provide assurances [69] of consensus rule transformation in blockchain must become widely adopted and further developed.**

Having several implementations can both cause and detect invalid transformations. Although BTC mainly relies on a single implementation, many cryptocurrencies, such as BCH, use multiple different implementations, all of which should follow the same consensus rules. Running testing on a test network with different implementations increases the chance of discovering transformation issues before deployment. However, as pointed out in Listing 16, having different implementations increase the risk of causing transformation issues (Other related quotes are in Section A.13).

```
Diversity is good and may help discover issues. But
as Gavin was saying and as I like to point out: The
most dangerous kind of failure in bitcoin isn't an
implementation bug- any blockchain validation
inconsistencies in widely deployed implementations
are significantly worse than pretty much anything
other than a full private key leak or remote root
exploit... and are even harder to avoid.
```

Listing 16. Gmaxwell 2012-10-28 Forum, ID: 120836

### 5.2 Improper reorganisation

In case of an accidental split, nodes must be prepared to handle reorganisation to coordinate everyone to work on the same chain. Some nodes have been forced to redownload the whole blockchain. However, the original slow initial block download (IBD) [70] made it troublesome (BTC 0.3.10). Moreover, the database lock limit caused stuck nodes during reorganisations before BTC 0.8.0 (see explanation in Section 4.2.10). Some nodes would also wipe the existing mempool on reboot, making it harder to detect double-spend attempts (BTC 0.8.0). **Measurements should be taken to keep the current state of valid blocks and pending transactions when performing an emergency fork, enabling a swift recovery.** The error leading to slow reorganisation could be a lapse in the case where node operators, in a weak moment, delete the whole blockchain on reboot and patch. It can also be a KB mistake where developers defining the code for reorganisations did not have the knowledge and experience to handle them properly. Listing 17 shows the issue of quickly reorganising the blockchain during the BTC 0.3.10 EARF (Other related quotes are in Section A.14).

```
knightmb, do you still have any of your monster
network available to turn on to help build the new
valid chain?
```

Listing 17. Insti 2010-08-15 Forum, ID: 823

### 5.3 Improper human interference

Bitcoin's early history shows improper handling of deployment features. This happened in the pay-to-script-hash upgrade. The BTC developers manually set and moved the flag day trigger depending on whether the threshold was reached (BTC 0.6.0). The threshold was not met in time for the first flag day, and nodes had to update to change the new flag day. Some nodes did not catch this in time and lost track of the correct chain as an invalid pay-to-script-hash transaction was mined after the first flag day. The error can be seen as an RB mistake from the developers' side, which had false expectations for node operators. It could also be a KB mistake from the node operators' side if they were unaware of the changed flag day or a lapse in case they forgot to update in time. The error in the pay-to-script-hash

deployment demonstrates that **dynamic thresholds must be incorporated into the software, not changed manually.**

Deployment features should not be changed during deployment because each change acts as a fork by itself and is a liability. In addition, **developers should not alter ongoing deployment without giving time to review changes.** That can be severe as it can allow the inclusion of flawed or ill-intended changes at the last minute.

### 5.4 Too high thresholds

High thresholds are crucial to onboard hash power during deployment. The threshold feature is relevant in combination with the deployment strategy feature since miner-activated strategies utilize thresholds. The Segwit deployment (BTC 0.13.1) showed that high thresholds, such as 95%, are troublesome because it allows a >5% minority veto as shown in Listing 18 (Other related quotes are in Section A.15). The error can be seen as an RB mistake because SegWit was falsely considered non-controversial. The slow adoption of SegWit engaged using the less safe user-activated strategy to overthrow non-signalling nodes using the inclusive feature to exclude the opponents.

```
Activation is dependent on near unanimous hashrate
signalling which may be impractical and is also
subject to veto by a small minority of
non-signalling hashrate.
```

Listing 18. Shaolinfry 2017-04-06 Email: bitcoin-dev

BTC demonstrated some changes to avoid issues with high thresholds in the most recent Taproot upgrade (BTC 0.21.1). **They changed the activation threshold to 90% to reduce the decision-making time and intended to use user-activated deployment if the miner-activated deployment failed. Another method to cope with high thresholds is gradually decreasing the threshold towards the lower limit of the super-majority.** Dash's dynamic activation thresholds utilize this technique where the initial limit is 80% and is gradually reduced to 60% [71]. However, there is a trade-off that lower thresholds are more likely to disrupt consensus.

### 5.5 Deploying 'irreversible' changes

Changes conducted with reduction forks should be applied carefully, as they might only be reverted if the nodes are willing to adopt future expansion forks. Nakamoto could probably never have imagined the fuzz caused by his 1 MB block size reduction created as a remedy for denial-of-service (BTC 0.3.12). So far, this reduction is nearly irreversible in practice for the expansion-reluctant BTC community. **Therefore, it can be valuable to consider the fork type of temporary reduction forks when there is any doubt whether such a reduction fork should be permanent.** The error is classified as a KB mistake because the developer did not foresee the future challenges of expanding the consensus rules. Listing 19, shows frustration for the 'irreversible' block size limit since the early days of Bitcoin (Other related quotes are in Section A.16.

```
I'm very uncomfortable with this block size limit
rule. This is a "protocol-rule" (not a
"client-rule"), what makes it almost impossible to
change once you have enough different softwares
running the protocol. Take SMTP as an example...
it's unchangeable.
```

Listing 19. Caveden 2010-11-20 Forum, ID: 1347

### 5.6 Not prepared for forward compatibility

Nakamoto implemented support for Bitcoin to be forward-compatible. As shown in Listing 20, he created domains of undefined behaviour by initially defining block versions, transaction versions, and later OP_NOP opcodes (BTC 0.3.6) (Other related quotes are in Section A.17). This facilitates specifying future changes as reduction forks, which are safer. Most of the planned reduction forks in Bitcoin have depended on forward compatibility. **To ensure forward compatibility, a blockchain should be defined with domains of undefined functionality, such as version numbers and empty opcodes.**

```
(...) OP_EVAL == OP_NOP1 can be safely rolled out
as soon as 50% of the miners upgraded
```

Listing 20. Sipa 2011-10-02 IRC: #bitcoin-dev

Forward compatibility was further adopted when SegWit was created. The developers defined a 4-byte nVersion field to allow future changes to the script specification to be created as reduction forks (BTC 0.13.1, BTC 0.21.1). The error of not preparing for forward compatibility can be seen as a KB mistake, as developers might not be aware of future compatibility issues. However, some people in the BCH community and many other blockchain projects (e.g., Ethereum and Dash) do not value compatibility between versions. They instead perform less safe expansion forks if that makes the end product more elegant. This can be seen as choosing functionality over consistency.

**The standard feature can be used to facilitate forward compatibility.** This was done for all consensus rules dealing with malleability (BTC 0.10.0, BTC 0.13.1, BTC* 0.14.0, BTC* 0.14.1, BCHN 0.16.0, and BTC 0.21.1). All the rule changes related to malleability were already softly enforced by standardness. The standard nodes would minimize the success of malleability attacks before activating the consensus change by not including or relaying those transactions.

**Additionally, the parallel feature is relevant to forward compatibility as it makes it possible to perform several deployments simultaneously or sequentially.** This was not the case for the first established deployment method ISM used in BTC 0.7.0, BTC 0.10.0, and BTC 0.11.2. These versions were deployed without forward compatibility and would not allow other deployments in parallel. Furthermore, this technique permanently consumed versionBits. So, they could never be used in a reduction fork again.

### 5.7 Lacking knowledge regarding network dynamics

Some rule changes may require nodes to broadcast additional information to other nodes in the network. The worst-case outcome of this behaviour could be that the network would create partitions of nodes that could only validate blocks made within that partition. **Therefore, changes in the peer-to-peer network must be handled to enable compatibility with legacy nodes and avoid network partition.**

A potential error of network partitioning would be a KB mistake because of the lack of knowledge regarding

network dynamics. For instance, the extension blocks introduced for SegWit contain signature data the legacy nodes would not recognise or relay. As seen in Listing 21 the peer-to-peer network relied on the signal feature by using a service bit for a node to signal the ability to provide the witness data (BTC 0.13.1) (Other related quotes are in Section A.18).

```
To ensure that the network is not partitioned and
that segwit blocks are being passed to segwit
enabled nodes, a Core 0.13.1 node will use its
outgoing connection slots to connect to as many
nodes with the NODE_WITNESS service bit as possible
(...)
```

Listing 21. achow101 2016-11-26 Forum, ID: 1682183

### 5.8 Insufficient damage control

Forks are necessary for the evolution of blockchains. As history has proven and Murphy's law will ensure, consensus failures will occur in the future. **End-users and miners should take measurements to perform damage control.** Past failures to perform these measurements would be a KB mistake because the actors did not know or did not have experience with chain splits.

These measurements would be to detect a chain split and suspend transactions or increase the number of confirmations required. In the past, merchants have been subject to double-spend attacks, and pool funds have been drained by miners working on the chain that eventually orphaned (BTC 0.8.0, Listing 22, other related quotes are in Section A.19). In Bitcoin, there are mechanisms for detecting chain splits. Additionally, one can run nodes with different versions to monitor that they stay on the same chain. Listing 23 discusses some ways to perform damage control in case of a consensus failure. The simplest solution in case of a consensus failure is to stop accepting and processing transactions (Listing 24).

```
I've lost way too much money in the last 24 hours
```

Listing 22. Eleuthria 2013-03-12 IRC: #bitcoin-dev

```
So I think the only way Mallory gets free beer from
you with segwit soft-fork is if:
- you're running out of date software and you're
ignoring warnings to upgrade (block versions have
bumped)
- you've turned off standardness checks
- you're accepting low-confirmation transactions
- you're not using any double-spend detection
service
```

Listing 23. Erisian 2015-12-18 Email: bitcoin-dev

```
If you're unsure, please stop processing
transactions.
```

Listing 24. Pieter Wuille 2013-03-12 Email: bitcoin-dev

Replay attacks can be performed by re-broadcasting transactions from one chain to another in the event of a permanent chain split. To prevent this attack, one of the chains should implement replay protection [8]. BCH implements replay protection for all planned consensus changes [72].

Some damage control can be prepared up front, e.g., by incorporating a kill switch mechanism [73] to activate an emergency rollback. However, this kill-switch mechanism increases the risk of centralisation and foreign interference if a single person or a closed community holds it.

### 5.9 Improper miner incentives to enforce new rules

Even though miners give a signal for an upgrade in blocks, this does not guarantee that these miners will enforce the new rules. Simple-payment-verification (SPV) mining has become popular because less validation gives an advantage in the block race. The incentive mechanism in Bitcoin rewards the first valid block, and the tradeoff between the risk of not being first and the risk of being invalid may favour being first, as it was seen in BTC 0.10.0 [4] (Listing 25, other related quotes are in Section A.20). The grace time between the time of reaching the first threshold and the time of activation added through BIP9 [26] was likely included because of this incident to give miners some time to ensure proper validation in time for activation.

```
If there is a cost to verifying transactions in a
received block, then there is an incentive to *not
verify transactions*. However, this is balanced by
the a risk of mining atop an invalid block.
```

Listing 25. nathan 2015-07-11 Email: bitcoin-dev

This error is caused by routine or exceptional violations where miners generate blocks without performing validation. **Measurements should be taken to incentivise validation.** For instance, Ethereum's slashing mechanism [74] discourages reckless behaviour. Alternatively, Dash incentivises validation by requiring collateral for master nodes [75] and giving them extra rewards.

### 5.10 Insufficient incentives to review the code

Another incentive issue concerns reviewing code. Most actors in Bitcoin benefit from having bug-free code deployed in the network to secure the currency's value. However, testing and reviewing can be tricky, costly, and tedious. The average Bitcoin participant (e.g., end-users and miners) would not even have the skills to perform that task. The stakes might be high for anyone pushing code that affects the network badly, as it may harshly influence their reputation. At the same time, there needs to be more incentive to encourage spending substantial time and resources on secure development and code review. The lack of incentives could make developers lazy and errors are made by routine violations.

Many of the critical bugs contained in Bitcoin have been fixed since its conception, and new ones arise as developers make mistakes (BTC 0.8.0, BTC 0.14.0, and BCHN 0.17.0). However, these mistakes are not for developers to bear alone but for those who naively adopt flawed code. A project directly incentivising its development is Dash, where 10% of block rewards are allocated to development [76]. The takeaway for this lesson is that **blockchain communities should allocate incentives to review code and minimise the chance of bugs being accepted into production.**

## 6 Discussion

### 6.1 Comparison with related work

The available literature on the evolution of Bitcoin mainly concerns socio-technical aspects of governance [77], [78] and

their economics [79]. For instance, [77] addresses "how and when code development practices combine into a pattern of self-organizing." Deployment of consensus rules is slightly mentioned in the paper as a practice that can result in competing infrastructures. Our research builds upon their work by showing how to tame the evolution of the infrastructure.

Kiffer et al. [8] evaluate the event of the Decentralised Autonomous Organization (DAO) hack in Ethereum and replay attacks. The case of Ethereum's chain split is relevant for damage control. This paper provides a bigger picture by showing how chain splits appear and how to resolve them.

Abortable and adaptable consensus [80], [81], [82] are similar because they look at how and when the network should switch to an arbitrary consensus algorithm. The main goal for switching the consensus protocol is to gain performance when needed and increase fault tolerance when the network fails. However, abortable and adaptable consensus does not discuss much about how the switching mechanism should work to perform deployment in a decentralised environment. Our paper encompasses any consensus change and entails changing the consensus algorithm.

The similarities between open source software and blockchain evolution are seen in the decision-making on a code repository level [83], [84]. Anyone is free to propose a change, and it is up to the repository's maintainers whether they want to include that change. However, the decision is not only based on the maintainers' preference in blockchain but also on the opinion of the community, miners and possibly developers of other implementations of the same protocol. Even when a change is included in a code repository, it does not mean the network will adopt it.

The adoption rate of a new network protocol spans several years, does not require a specific threshold for adoption, and can tolerate different versions running in parallel [85]. In contrast, blockchain relies on abandoning old consensus rules by activating new ones. BTC and TLS are similar in that they value backward compatibility to allow nodes running old protocols to be part of the network during and after an upgrade.

The most significant difference when comparing blockchain deployment techniques to techniques in distributed systems, such as fast reboot, rolling upgrade, and big flip [10], is that blockchain is decentralised, and the network must reach a consensus before changes can be activated. The system reaches consensus through unique deployment features, such as standardness, signals, thresholds, and inclusiveness.

### 6.2 Implications

To our knowledge, this research is the first to present a holistic overview of deployment techniques for runtime evolution in blockchains. The deployment processes of consensus rule changes in the blockchain are vital as they can cause and remedy consensus failures. By generalizing the adoption logic of blockchain, this paper can contribute to the field of self-adaptive systems where processes for runtime evolution of new domain logic require further exploration [86].

Our findings apply to any decentralised blockchain. Regardless of how practitioners perform consensus rule evolution today, their method is a variant of those covered by our paper. Our work can provide guidance on how to strengthen the existing practices of consensus evolution to avoid failures. Further, we summarise deployment techniques and best practices for different scenarios, such as planned consensus deployment or emergency deployment.

Practitioners can utilize the contributions of this paper to perform consensus rule changes predictably and safely. Our results present a comprehensive overview of measurements to avoid and handle consensus failure. The lessons learned in Bitcoin are valuable to prevent history from repeating itself. These lessons can strengthen the security of blockchains, hinder direct financial loss, and preserve blockchains' value as cryptocurrencies.

Interestingly, BTC is not necessarily a trendsetter in the space yet. Their conservative approach favouring miner-activated reduction forks (MARFs) and predictable consensus support by using signals and thresholds is unique. Most other blockchains are willing to drastically change the consensus rules by expansion forks and make older nodes obsolete by the use of user-activated expansion forks (UAEFs) and a flag day to set the activation time. UAEF deployment can be reasonably safe to avoid chain splits as long as the network and community act unanimously. However, as blockchains mature and communities become content, they may prefer more conservative and predictable approaches to consensus evolution, similar to BTC.

### 6.3 Threats to validity

The sheer amount of data and the limited number of researchers dedicated to this project may raise questions about missing data or analysis. Regular cross-author discussions have evaluated the analysis and results to address this. All the samples used for the analysis are also available at [49]. We have gathered a holistic picture of Bitcoin evolution and other blockchains outside the initial domain by conducting rigorous data collection through snowball sampling and triangulation. The results are also strengthened by combining GT with root cause analysis.

This study seeks to avoid bias by looking at other viable projects with similar attributes, such as BCH, Ethereum and Dash. Different actors have different concerns, and results are represented by diverse perspectives provided by those who have worked on Bitcoin over the last decade. We included additional quotes in Appendix Section A to show different people from different times supporting our claims. The thoroughness applied in this study gives confidence that the results apply to different blockchain architectures.

## 7 Conclusion and Future Work

Safe deployment of consensus rules in blockchains is vital to hinder failures causing financial losses for miners and end-users. The paper demonstrates an extensive study using the grounded theory approach, flexible coding, and root cause analysis to address these issues. This study specifies nine deployment techniques for blockchain with nine different features. Additionally, the study shows how contention may arise during consensus rule changes in Bitcoin, resulting in ten lessons learned. The findings bring novel insights to promote a safe evolution of blockchains.

Decision-making and governance of the consensus rules were intentionally left out of scope for this study to focus solely on the technical approaches and the implications of consensus change. However, the decision-making progress in blockchain communities is exciting and differentiates itself from typical open-source projects. Therefore, these aspects should be explored further.

The greatest challenge in blockchain evolution is compatibility and transformation assurance [87]. Like adaptive systems, blockchain systems would benefit from identifying whether a change is a fork and what type of fork it will be. One future work is to study transformation assurance to reduce the risk of deployment failures significantly. Another future work is to handle deployments in a multichain environment. Various research indicates that these environments will rely on middleware to relay actions across chains. We believe that this middleware must be responsible for listening to and triggering changes based on the deployment techniques used in the attached chains.

## ACKNOWLEDGMENTS

This work is jointly supported by the National Key Research and Development Program of China (No.2019YFE0105500) and the Research Council of Norway (No.309494).

## REFERENCES

[1] A. V. Wirdum. (2020) The battle for p2sh: The untold story of the first bitcoin war. Bitcoin Magazine. [Online]. Available: https://bitcoinmagazine.com/technical/the-battle-for-p2sh-the-untold-story-of-the-first-bitcoin-war

[2] ——. (2017) The long road to segwit: How bitcoin's biggest protocol upgrade became reality. Bitcoin Magazine. [Online]. Available: https://bitcoinmagazine.com/technical/the-long-road-to-segwit-how-bitcoins-biggest-protocol-upgrade-became-reality

[3] G. Andresen. (2014) March 2013 chain fork post-mortem. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki

[4] Bitcoin.org. (2015) Some miners generating invalid blocks. Bitcoin.org. [Online]. Available: https://bitcoin.org/en/alert/2015-07-04-spv-mining

[5] B. Core. (2018) Cve-2018-17144 full disclosure. Bitcoin Core. [Online]. Available: https://bitcoincore.org/en/2018/09/20/notice/

[6] C. Harper. (2021) Open ethereum clients encounter 'consensus error' after berlin hard fork; coinbase pauses eth withdrawals. CoinDesk. [Online]. Available: https://www.coindesk.com/tech/2021/04/15/open-ethereum-clients-encounter-consensus-error-after-berlin-hard-fork-coinbase-pauses-eth-withdrawals/

[7] Bitcoin.org. (2013) A successful double spend us$10000 against okpay this morning. Simple Machines Forum. [Online]. Available: https://bitcointalk.org/index.php?topic=152348

[8] L. Kiffer, D. Levin, and A. Mislove, "Stick a fork in it: analysing the ethereum network partition," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. New York, NY, USA: ACM, 2017, pp. 94–100. [Online]. Available: https://doi.org/10.1145/3152434.3152449

[9] T. Mens, "Introduction and roadmap: History and challenges of software evolution," in *Software evolution*. Berlin, Heidelberg: Springer, 2008, pp. 1–11.

[10] E. A. Brewer, "Lessons from giant-scale services," *IEEE Internet computing*, vol. 5, no. 4, pp. 46–55, 2001.

[11] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. Bitcoin.org. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[12] A. Rashid, S. A. A. Naqvi, R. Ramdhany, M. Edwards, R. Chitchyan, and M. A. Babar, "Discovering "unknown known" security requirements," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 866–876. [Online]. Available: https://doi.org/10.1145/2884781.2884785

[13] J. Corbin and A. Strauss, *Basics of qualitative research: Techniques and procedures for developing grounded theory*. London, England, UK: Sage publications, 2014.

[14] K. Ishikawa, *Guide to quality control*. Hong Kong: Asian Productivity Organization, 1982.

[15] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology — CRYPTO' 92*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 139–147.

[16] M. Jakobsson and A. Juels, *Proofs of Work and Bread Pudding Protocols(Extended Abstract)*. Boston, MA: Springer US, 1999, pp. 258–272. [Online]. Available: https://doi.org/10.1007/978-0-387-35568-9_18

[17] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[18] M. Rosenfeld, "analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, vol. abs/1402.2009, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1402.2009

[19] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 51% attack," *Applied Sciences*, vol. 9, no. 9, p. 1788, 2019.

[20] Coinmarketcap. (2023) Today's cryptocurrency prices by market cap. Coinmarketcap. [Online]. Available: https://coinmarketcap.com/

[21] bitinfocharts. (2023) Bitcoin, bitcoin cash hashrate historical chart. bitinfocharts. [Online]. Available: https://bitinfocharts.com/comparison/hashrate-btc-bch.html#3y

[22] bitcoincash.org. (2017) Uahf technical specification. bitcoincash.org. [Online]. Available: https://github.com/bitcoincashorg/bitcoincash.org/blob/master/spec/uahf-technical-spec.md

[23] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, jul 1982. [Online]. Available: https://doi.org/10.1145/357172.357176

[24] H. Knudsen, J. S. Notland, P. H. Haro, T. B. Raeder, and J. Li, *Consensus in Blockchain Systems with Low Network Throughput: A Systematic Mapping Study*. New York, NY, USA: Association for Computing Machinery, 2021, p. 15–23. [Online]. Available: https://doi.org/10.1145/3475992.3475995

[25] P. Todd. (2014) Op_checklocktimeverify. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki

[26] P. Wuille, P. Todd, G. Maxwell, and R. Russel. (2015) Version bits with timeout and delay. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki

[27] S. Fry. (2017) Mandatory activation of segwit deployment. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0148.mediawiki

[28] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.

[29] A. Tapscott and D. Tapscott, "How blockchain is changing finance," *Harvard Business Review*, vol. 1, no. 9, pp. 2–5, 2017.

[30] P. Treleaven, R. G. Brown, and D. Yang, "Blockchain technology in finance," *Computer*, vol. 50, no. 9, pp. 14–17, 2017.

[31] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.

[32] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *proceedings of the 50th Hawaii international conference on system sciences*. Manoa Hawaii, USA: ScholarSpace, 2017, pp. 4182–4191.

[33] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom)*. Munich, Germany: IEEE, 2016, pp. 1–3.

[34] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 135, pp. 62–75, 2019.

[35] J. Kramer and J. Magee, "The evolving philosophers problem: Dynamic change management," *IEEE Transactions on software engineering*, vol. 16, no. 11, pp. 1293–1306, 1990.

[36] S. Ajmani, B. Liskov, and L. Shrira, "Modular software upgrades for distributed systems," in *ECOOP 2006 – Object-Oriented Pro-*

*gramming*, D. Thomas, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 452–476.
[37] A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W. J. Knottenbelt, "A wild velvet fork appears! inclusive blockchain protocol changes in practice," in *International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2018, pp. 31–42.
[38] E. Lombrozo. (2020) Bitcoin magazine's independence day: Activating taproot with eric lombrozo and luke dashjr. Bitcoin Magazine. [Online]. Available: https://www.youtube.com/watch?v=yQZb0RDyFCQ
[39] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: A critical review and guidelines," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 120—-131. [Online]. Available: https://doi.org/10.1145/2884781.2884833
[40] B. J. Oates, "Researching information systems and computing," 2006.
[41] N. M. Deterding and M. C. Waters, "Flexible coding of in-depth interviews: A twenty-first-century approach," *Sociological methods & research*, vol. 50, no. 2, pp. 708–739, 2021.
[42] B. Wiki. (2022) Consensus versions. Bitcoin Wiki. [Online]. Available: https://en.bitcoin.it/wiki/Consensus_versions
[43] B. Core. (2021) Bitcoin core integration/staging tree. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bips
[44] L. Foundation. (2022) Bitcoin-dev email list. Linux Foundation. [Online]. Available: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/
[45] S. M. Forum. (2021) Bitcoin forum. Simple Machines Forum. [Online]. Available: https://bitcointalk.org/
[46] B. Core. (2021) Pull requests. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bitcoin/pulls
[47] ——. (2021) Issues. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bitcoin/issues
[48] B. wiki. (2020) Irc channels. Bitcoin Core. [Online]. Available: https://en.bitcoin.it/IRC_channels
[49] J. S. Notland. (2022) Exported samples. Norwegian University of Technology and Science. [Online]. Available: https://folk.ntnu.no/jakobsn/Runtime%20Evolution%20of%20Bitcoin's%20Consensus%20Rules/exported%20samples.zip
[50] Ninjastic. (2021) Ninjastic.space. Ninjastic.space. [Online]. Available: https://ninjastic.space
[51] B. Research. (2017) A complete history of bitcoin's consensus forks. BitMex. [Online]. Available: https://blog.bitmex.com/bitcoins-consensus-forks/
[52] T. F. S. Foundation. (2020) Gnu grep. the Free Software Foundation. [Online]. Available: https://www.gnu.org/software/grep/
[53] Erisian. (2021) bitcoin-core-dev. Erisian Consulting. [Online]. Available: https://www.erisian.com.au/bitcoin-core-dev/
[54] ——. (2021) bitcoin-dev. Erisian Consulting. [Online]. Available: https://www.erisian.com.au/bitcoin-dev/
[55] BuildingBitcoin. (2019) bitcoin-dev. buildingbitcoin. [Online]. Available: https://buildingbitcoin.org/bitcoin-dev/
[56] Erisian. (2018) bitcoin-dev. Erisian Consulting. [Online]. Available: http://azure.erisian.com.au/~aj/tmp/irc/
[57] Atlas.ti. (2022) All-in-one research software. ATLAS.ti Scientific Software Development GmbH. [Online]. Available: https://atlasti.com/
[58] NVIVO. (2022) Qualitative data analysis software. QSR International. [Online]. Available: https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home
[59] MAXQDA. (2022) Organize. code. analyse. present. VERBI GmbH. [Online]. Available: https://www.maxqda.com/
[60] J. Penrod, D. B. Preston, R. E. Cain, and M. T. Starks, "A discussion of chain referral as a method of sampling hard-to-reach populations," *Journal of Transcultural nursing*, vol. 14, no. 2, pp. 100–107, 2003.
[61] Bitcoin.org. (2022) Bitcoin is an innovative payment network and a new kind of money. Bitcoin.org. [Online]. Available: https://bitcoin.org/
[62] B. Research. (2017) Empty block data by mining pool. BitMex. [Online]. Available: https://blog.bitmex.com/empty-block-data-by-mining-pool/
[63] J. Reason, *Human error*. New York, NY, USA: Cambridge university press, 1990.
[64] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton, New Jersey, USA: Princeton University Press, 2016.
[65] Dash. (2018) Introducing long living masternode quorums. Dash Core. [Online]. Available: https://www.dash.org/blog/introducing-long-living-masternode-quorums/
[66] G. Andresen. (2013) Before 15 may, limit created block size to 500k. Bitcoin Core. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/v0.8.1/src/main.cpp
[67] Luke-jr. (2020) Segwit/psbt vulnerability (cve-2020-14199) with luke dashjr — bitdevsla. BitDevsLA. [Online]. Available: https://www.youtube.com/watch?v=CojixIMgg3c
[68] C. Harper. (2021) Open ethereum clients encounter 'consensus error' after berlin hard fork; coinbase pauses eth withdrawals. CoinDesk. [Online]. Available: https://www.coindesk.com/tech/2021/04/15/open-ethereum-clients-encounter-consensus-error-after-berlin-hard-fork-coinbase-pauses-eth-withdrawals/
[69] R. d. Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo *et al.*, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Berlin, Heidelberg: Springer, 2017, pp. 3–30.
[70] Bitcoin.org. (2022) Running a full node. Bitcoin.org. [Online]. Available: https://bitcoin.org/en/full-node#initial-block-downloadibd
[71] PastaPastaPasta. (2020) Dash core 0.16.0.1 release announcement. Dash Core. [Online]. Available: https://github.com/dashpay/dash/releases/tag/v0.16.0.1
[72] B. C. Node. (2017) Buip-hf digest for replay protected signature verification across hard forks. Bitcoin-ABC. [Online]. Available: https://gitlab.com/bitcoin-cash-node/bchn-sw/bitcoincash-upgrade-specifications/-/blob/master/spec/replay-protected-sighash.md
[73] D. Core. (2021) sporks. Dash Core. [Online]. Available: https://docs.dash.org/en/stable/introduction/features.html#sporks
[74] Ethereum.org. (2022) Proof-of-stake and security. Ethereum. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/#pos-and-security
[75] I. Dash Core Group. (2022) Understanding masternodes. Dash Core. [Online]. Available: https://docs.dash.org/en/stable/masternodes/understanding.html
[76] I. Dash Core Group, Inc. (2021) Understanding dash governance. Dash Core. [Online]. Available: https://docs.dash.org/en/stable/governance/understanding.html
[77] J. V. Andersen and C. I. Bogusz, "Patterns of self-organising in the bitcoin online community: Code forking as organising in digital infrastructure." in *ICIS*. Seoul, South Korea: AIS, 2017. [Online]. Available: https://pure.itu.dk/portal//files/83566552/ICIS_revision_2017.pdf
[78] P. De Filippi and B. Loveluck, "The invisible politics of bitcoin: governance crisis of a decentralised infrastructure," *Internet Policy Review*, vol. 5, no. 4, 2016. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2852691
[79] J. A. Kroll, I. C. Davey, and E. W. Felten, "The economics of bitcoin mining, or bitcoin in the presence of adversaries," in *Proceedings of WEIS*, no. 11, Washington, DC. 37th and O Streets, Rafik B. Hariri Building, Washington, DC 20057, United States: ISE, 2013. [Online]. Available: http://www.infosecon.net/workshop/downloads/2013/pdf/The_Economics_of_Bitcoin_Mining,_or_Bitcoin_in_the_Presence_of_Adversaries.pdf
[80] W. Chen, "Abortable consensus and its application to probabilistic atomic broadcast," Technical Report MSR-TR-2006-135, Tech. Rep., 2007.
[81] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *ACM Trans. Comput. Syst.*, vol. 32, no. 4, jan 2015. [Online]. Available: https://doi.org/10.1145/2658994
[82] J.-P. Bahsoun, R. Guerraoui, and A. Shoker, "Making bft protocols really adaptive," in *2015 IEEE International Parallel and Distributed Processing Symposium*. Hyderabad, India: IEEE, 2015, pp. 904–913.
[83] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. Van Deursen, "Communication in open source software development mailing lists," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. San Francisco, CA, USA: IEEE, 2013, pp. 277–286.
[84] P. N. Sharma, B. T. R. Savarimuthu, and N. Stanger, "Extracting rationale for open source software development decisions—a study

of python email archives," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. Madrid, ES: IEEE, 2021, pp. 1008–1019.

[85] R. Holz, J. Hiller, J. Amann, A. Razaghpanah, T. Jost, N. Vallina-Rodriguez, and O. Hohlfeld, "Tracking the deployment of tls 1.3 on the web: A story of experimentation and centralization," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 3, pp. 3–15, 2020.

[86] R. de Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, D. Weyns, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cukic, R. Desmarais, S. Dustdar, G. Engels, K. Geihs, K. M. Göschka, A. Gorla, V. Grassi, P. Inverardi, G. Karsai, J. Kramer, A. Lopes, J. Magee, S. Malek, S. Mankovskii, R. Mirandola, J. Mylopoulos, O. Nierstrasz, M. Pezzè, C. Prehofer, W. Schäfer, R. Schlichting, D. B. Smith, J. P. Sousa, L. Tahvildari, K. Wong, and J. Wuttke, *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–32. [Online]. Available: https://doi.org/10.1007/978-3-642-35813-5_1

[87] R. d. Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo *et al.*, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Berlin, Heidelberg: Springer, 2017, pp. 3–30.

[88] T. Ohno, *Toyota production system: beyond large-scale production*. crc Press, 1988.

# APPENDIX A
# SUPPLEMENTARY QUOTES

This appendix lists additional quotes to support those already presented in the main article. The first quote in each section shows the corresponding quote from the article, while the remaining quotes supplement further evidence of our claims. At the end, Figures 9 and 10 show the sample distribution among events and sources.

## A.1 Core design was set in stone for its lifetime

> The nature of Bitcoin is such that once version 0.1 was released, the core design was set in stone for the rest of its lifetime.

Listing 26. Satoshi 2010-06-17 Forum, ID: 195

> 2986 2011-09-13 22:12:26 <gavinandresen> ... and that's one of the problems with scheduling a block-chain split, it opens up debate to fix a zillion things.

Listing 27. gavinandresen 2011-09-13 Forum, ID: 195

> >I don't see what the big deal is with changing the maximum block size.
>
> >Why all the fuss ? It's not like blocks which are currently on average about half the current maximum size will suddenly and inexplicably jump in size overnight just because it's possible.
>
> hard fork

Listing 28. samson & HCLivess 2015-06-01 Forum, ID: 1347

> This would fork the blockchain. More useful changes have been kept-out-until-later for that reason. ;)

Listing 29. Luke-jr 2011-08-22 Forum, ID: 45211

> <sipa> the reason against incorporating bip142 is people yelling "see! sehwit needs new address types! everyone has to upgrade! not backward compatible!"

Listing 30. sipa 2016-03-10 IRC: #bitcoin-core-dev

## A.2 Cautious about updates and worry about flaws

> 5. Testing. I don't have time to personally test every PULL request, but if a pull involves more than trivial code changes I'm not going to pull it unless it has been thoroughly tested. We had a very good rule at a company I used to work for-- programmers were NOT allowed to be the only ones to test their own code. Help finding money and/or people for a dedicated "core bitcoin quality assurance team" is welcome. More unit tests and automated testing is also certainly welcome.
>
> If this was open source blogging software I'd be much less uptight about testing and code review and bugs. But it's not, it is software for handling money.

Listing 31. 2011-08-10 Gavin Andresen Email: bitcoin-dev

> ? The wallet encryption bug was embarrassing and stressful, and chewed up a lot of my time over the past couple of weeks. Bugs happen, but I've been spending time thinking about what I can do differently to make it less likely major bugs slip into releases.
>
> Finding the money to hire some professional QA people to help create test plans and then execute them (the test plans, not the QA people) is one possible answer. If you have experience finding funding for open source projects (or know somebody who does) I'd like to talk with you-- I would much rather spend my time writing code and thinking about technical issues instead of trying to figure out if advertising or sponsorship or a Donate menu entry in the client is a reasonable way to get more testing resources for the project.

Listing 32. 2011-11-22 Gavin Andresen Email: bitcoin-dev

> <luke-jr> cjdelisle: the review cycle for *protocol* changes is
> <luke-jr> the dev cycle is already far too slow
> <rjk2> same problem as most projects - lack of many dedicated testers :(

Listing 33. luke-jr & rjk2 2011-12-27 IRC: #bitcoin-dev

> <luke-jr> IMO, deploying OP_EVAL on miners only gets gavinandresen what he wants, and gets the delay camp what we want.
> <BlueMatt> also, the "we have had plenty of eyes on the old engine, dont add OP_EVAL", Id like to see a group formed to pay for a professional code analysis of OP_EVAL
> <BlueMatt> well funds concerns, but Id like to see all of bitcoin reviewed on each release

Listing 34. luke-jr & BlueMatt 2011-12-28 IRC: #bitcoin-dev

## A.3 Rule changes are a liability

> Every time that you open up the door to changing the rules, you are opening yourselves up to attack

Listing 35. 2020-08-03 Eric Lombrozo [38]

> The amount of code and the amount of changes in SegWit makes this a very dangerous change in (of?) Bitcoin. I counted 10 core concepts in Bitcoin being changed with it. We don't yet know how they will interact. We can?t.
>
> You are asking people to create everyone-can-spend transactions that would mean a loss of funds to everyone that used it if we do find a major flaw and need to rollback.

Listing 36. Tom Zander 2016-10-16 Email: bitcoin-dev

> Then you don't really know a lot about large institutions, do you ?
>
> I would bet that many banks run software written in 1999 or even before, you know why ? Because when huge risks and costs are involved, you don't change something that works properly unless there is a serious reason. Hell, some large companies still run COBOL code written 30-40 years ago !
>
> Changing software versions is a significant risk each time it is done.

Listing 37. ShadowOfHarbringer 2012-03-12 Email: bitcoin-dev

## A.4 Prefer backward-compatible reducing forks

> I belief we shold flesh out luke-jr's idea for cleanly deploying segregated witness in bitcoin as a soft fork and see what that looks like.

Listing 38. Gmaxwell 2015-11-04 IRC: #bitcoin-dev

```
<devrandom> gmaxwell: I'm not sure that the
argument of valid->invalid vs invalid->valid makes
sense. In either case, the clients that did not
upgrade will be on the wrong fork until they
upgrade.
<gmaxwell> Not true.
<gmaxwell> They'll switch to the right fork once it
has a longer chain, which" if a super majority of
the miners have upgraded in advance" will always be
true.
<gmaxwell> This isn't true for invalid->valid.
<gmaxwell> There could even be some crazy rule on
the upgraded nodes: "New txn will be permitted and
the rules will begin being enforced at the first
block number after 12345 when the prior 1000 blocks
have not had the opcode, prior to that point I will
not mine any txn with that opcode"
<gmaxwell> And then you _intentionally_ inject txn
that use the opcode.. and the network will activate
only after almost all the miners are smart enough
to reject it
<gmaxwell> and everyone can agree on when it
activated (because its an objective fact of the
chain) so no splits.
Xunie has joined
<gmaxwell> This would avoid the risk that you don't
quite get a super majority of the miners and you
end up with old clients on a seperate fork for days
at a time.
```

Listing 39. devrandom & gmaxwell 2011-10-03 IRC: #bitcoin-dev

```
<jgarzik> gavinandresen: so, dumb question... is
this multisig stuff a breaking change, that makes
blocks incompatible with older clients?
<imsaguy2> They were talking about a db version
change that would
<gavinandresen> jgarzik: no, not my current
proposal. It uses only existing opcodes
<imsaguy2> one way
piotrp has quit (Quit: leaving)
<gavinandresen> ... the db4.7-4.8 is a different
issue
<jgarzik> gavinandresen: That's about the only
thing I would push back on, in a major way. I
really think breaking changes ("if (block > 200000)
new_behavior()") should be avoided absent a
catastrophic problem such as sha256 is broken.
<jgarzik> gavinandresen: Adding new standard
transactions is a good thing in general. I like the
testnet roll-out that people currently do
tynx has quit (Quit: Leaving)
<gavinandresen> jgarzik: yes, this doesn't feel
like a good reason to schedule a block chain split.
<jgarzik> gavinandresen: agreed
<gavinandresen> (although it it is SO tempting to
make it perfect.....)
<jgarzik> gavinandresen: yes ;-) ;-)
<jgarzik> gavinandresen: I have ideas about that,
too... IMNSHO people need an outlet for breaking
changes. It's tempting to either (a) work on a
bitcoin2, which is current bitcoin + rational
breaking changes like new hash algo or new
protocol, or (b) someone maintain a list of
"changes community would like to see, if and only
if there is a planned block chain split"
<gavinandresen> jgarzik: I was thinking about how
to handle "when we DO decide to fork the
blockchain" patches/changes, too... didn't come up
with any solution I really like.
<jgarzik> gavinandresen: from an existential and PR
standpoint, I think major blockchain forks are
tough no matter how high the technical
justification, because, in theory, blockchhain
forks are like US Constitutional Conventions:
_anything_ can be changed, in theory, including the
basic rules like the 21M limit. Blockchain forks
are our equivalent of Federal Reserve policy
changes. It is the "nuclear option."
<jgarzik> gavinandresen: Thus, I prefer extremely
ugly hacks, or simply saying "no" (or "put it in
btc2") than blockchain forks that are incompatible
with older clients
<jgarzik> blockchain forks that are compatible with
older clients, I am OK with. The sendmany was a
good example of that... clients wouldn't relay for
a while, but older clients supported it just fine.
```

Listing 40. jgarzik, imsaguy2 & gavinandresen 2011-08-24 IRC: #bitcoin-dev

## A.5 Parallell deployments

```
BIP 34 introduced a mechanism for doing
soft-forking (...). As it relies on comparing
version numbers as integers however, it only
supports one single change being rolled out at
once, requiring coordination between proposals, and
does not allow for permanent rejection: as long as
one soft fork is not fully rolled out, no future
one can be scheduled.

In addition, BIP 34 made the integer comparison
(nVersion >= 2) a consensus rule after its 95%
threshold was reached, removing 231+2 values from
the set of valid version numbers (all negative
numbers, as nVersion is interpreted as a signed
integer, as well as 0 and 1). This indicates
another downside this approach: every upgrade
permanently restricts the set of allowed nVersion
field values. This approach was later reused in BIP
66 and BIP 65, which further removed nVersions 2
and 3 as valid options. As will be shown further,
this is unnecessary.
```

Listing 41. Pieter Wuille et al. 2015-10-04 BIP9

```
I do not see how this helps much; the reversibility
is a selling point, but at a far from zero cost.
We'll be moving on 62 once 66 is actually deployed
(one flaw in the the legacy softfork deployment
mechanism is that only one change can be in flight
at a time)
```

Listing 42. gmaxwell 2015-04-21 Forum, ID: 1033396

```
// Start enforcing the DERSIG (BIP66) rules, for
block.nVersion=3 blocks, when 75% of the network
has upgraded:
if (block.nVersion >= 3 && IsSuperMajority(3,
pindex->pprev,
Params().EnforceBlockUpgradeMajority())) {
   flags |= SCRIPT_VERIFY_DERSIG;
}
```

Listing 43. Pieter Wuille 2015-02-01 BIP66

```
// Start enforcing CHECKLOCKTIMEVERIFY, (BIP65) for
block.nVersion=4
// blocks, when 75% of the network has upgraded:
if (block.nVersion >= 4 && IsSuperMajority(4,
pindex->pprev,
chainparams.GetConsensus().nMajorityEnforceBlockUpgrade,
chainparams.GetConsensus())) {
   flags |= SCRIPT_VERIFY_CHECKLOCKTIMEVERIFY;
}
```

Listing 44. Peter Todd 2015-10-08 BIP65

## A.6 Signaling

```
when 50% of the last N coinbases contain "I support
FOOFEATURE", it's enabled
```

Listing 45. Luke-jr 2011-10-02 IRC: #bitcoin-dev

```
Perhaps as a safeguard:
(3) Before applying the new rule, require 50% of
the last Y blocks contain a
   coinbase with a "I am upgraded" code
(4) Until the new rule is active, include an "I am
upgraded" code in every
   block; after it's active, this can be turned off
```

Listing 46. Luke-jr 2011-10-03 Email: bitcoin-dev

```
First of all, that's an expensive beer!

Second of all, any consensus rule change risks
non-full-validating or non-upgraded nodes seeing
invalid confirmations...but assuming a large
supermajority (i.e. > 95%) of hashing power is
behind the new rule, it is extremely unlikely that
very many invalid confirmations will ever be seen
by anyone. The number of confirmations you require
depends on your use case security
requirements...and especially during a new rule
activation, it is probably not a good idea for
non-validating nodes or non-upgraded nodes to
accept coins with low confirmation counts unless
the risk is accounted for in the use case (i.e. a
web hosting provider that can shut the user out if
fraud is later detected).

Third of all, as long as the rule change activation
is signaled in blocks, even old nodes will be able
to detect that something is fishy and warn users to
be more cautious (i.e. wait more confirmations or
immediately upgrade or connect to a different node
that has upgraded, etc...)

I honestly don't see an issue here - unless you're
already violating fundamental security assumptions
that would make you vulnerable to exploitation even
without rule changes.
```

Listing 47. 2015-12-18 Eric Lomrozo Email: bitcoin-dev

### A.7 Verbal agreement

```
<gavinandresen> luke-jr: you're a mining pool
operator, would you be willing to coordinate with
the other big pools to get this fixed quickly[?]
<luke-jr> gavinandresen: looks good from what I
see, though I commented where I think it could be
improved inline
<luke-jr> oh, you mean me talk to the other poolops
for it?
<luke-jr> sure I guess
```

Listing 48. gavinandresen 2012-02-17 IRC: #bitcoin-dev

```
<sipa> gavinandresen: i've got replies for around
55% percent support for BIP30, and deepbit intends
to implement before march 15
(...)
<gavinandresen> sipa: Great, March 15 it is, then
```

Listing 49. sipa & gavinandresen 2012-03-03 IRC: #bitcoin-dev

```
<Luke-Jr> gavinandresen: sipa: jgarzik: can we get
a consensus on recommendation for miners to
downgrade? (...)
<Eleuthria> I can single handedly put 0.7 back to
the majority hash power
<Eleuthria> I just need confirmation that thats
what should be done (...)
<sipa> Eleuthria: imho, that is was you should do,
but we should have consensus first (...)
<jgarzik> sipa, Eleuthria: ACK on preferring 0.7
chain, for the moment (...)
<gavinandresen> Eleuthria: if you can cleanly get
us back on the 0.7 chain, ACK from here, too (...)
<Eleuthria> alright (...)
<doublec> ok, rolling back to 0.7.2
```

Listing 50. Developers: Luke-Jr, sipa, jgarzik, and gavinandresen. Pool operators: Eleuthria and doublec 2013-03-12 IRC: #bitcoin-dev

### A.8 Trigger

```
if (blocknumber > 115000)
   maxblocksize = largerlimit
It can start being in versions way ahead, so by the
time it reaches that block number and goes into
effect, the older versions that don't have it are
already obsolete.
```

Listing 51. Satoshi 2010-10-03 Forum, ID: 1347

```
1 // Deployment of SegWit (BIP141, BIP143, and BIP147)
2 consensus.vDeployments[Consensus::DEPLOYMENT_SEGWIT]
3 .bit = 1;
4 consensus.vDeployments[Consensus::DEPLOYMENT_SEGWIT]
5 .nStartTime = 1479168000; // November 15th, 2016.
6 consensus.vDeployments[Consensus::DEPLOYMENT_SEGWIT]
7 .nTimeout = 1510704000; // November 15th, 2017.
```

Listing 52. Pieter Wuille 2016-10-17 BIP141, BIP143 & BIP147

```
1 // Start enforcing BIP68 (sequence locks) and BIP112
        (CHECKSEQUENCEVERIFY) using versionbits logic.
2 int nLockTimeFlags = 0;
3 if (VersionBitsState(pindex->pprev, chainparams.GetConsensus(),
        Consensus::DEPLOYMENT_CSV, versionbitscache)
        ==THRESHOLD_ACTIVE) {
4     flags |= SCRIPT_VERIFY_CHECKSEQUENCEVERIFY;
5     nLockTimeFlags |= LOCKTIME_VERIFY_SEQUENCE;
6 }
```

Listing 53. btcdrak 2016-03-18 BIP66 & BIP112

### A.9 Feather fork/block discouragement

```
A feather-fork is when a miner refuses to mine on
any chain that includes a transaction it doesn't
like in the most recent several blocks.
```

Listing 54. Socrates1024 2013-10-17 Forum, ID: 312668

```
<gavinandresen> roconnor: fixing that bug safely is
non-trivial.... the right answer is for new
clients/miners to 'discourage' (refuse to
relay/mine) blocks/transactions with weird version
numbers. I've got a 'discourage blocks' patch
sitting on my machine that I haven't had time to
clean up into a pull request yet.
```

Listing 55. gavinandresen 2012-01-02 IRC: #bitcoin-dev

```
<gmaxwell> BlueMatt: wrt reorg, I think we should
discourage blocks with duplicate coinbases in the
short term.
```

Listing 56. gmaxwell 2012-02-06 IRC: #bitcoin-dev

### A.10 Emergency activated reduction fork

```
<Luke-Jr> gavinandresen: sipa: jgarzik: can we get
a consensus on recommendation for miners to
downgrade?
(...)
<gavinandresen> the 0.8 fork is longer, yes? So
majority hashpower is 0.8....
<Luke-Jr> gavinandresen: but 0.8 fork is not
compatible
```

Listing 57. Luke-Jr & gavinandresen 2013-03-12 IRC: #bitcoin-dev

Doesn't matter which chain is longer if a majority of the people aren't on it. Breaking changes need to be given lots of warning to be effective. Trying to force everyone to use 0.8 would have only made the situation worse. From the chat discussion, I don't think mtgox was using 0.8. So trading at the largest exchange would be halted until it could be upgraded. If that doesn't sound disastrous, I'm not sure what does.

Listing 58. nevafuse 2013-03-13 Forum, ID: 152470

Please upgrade to 0.3.6 ASAP! We fixed an implementation bug where it was possible that bogus transactions could be displayed as accepted. Do not accept Bitcoin transactions as payment until you upgrade to version 0.3.6! If you can't upgrade to 0.3.6 right away, it's best to shut down your Bitcoin node until you do.

Listing 59. Satoshi 2010-07-29 Forum, ID: 626 bitcoin-dev

*** WARNING *** We are investigating a problem. DO NOT TRUST ANY TRANSACTIONS THAT HAPPENED AFTER 15.08.2010 17:05 UTC (block 74638) until the issue is resolved.

Listing 60. Satoshi 2010-08-15 Email: bitcoin-dev

### A.11 Non-deterministic forks

<TD> EXCEPTION: 11DbException
<TD> Db::put: Cannot allocate memory
<TD> bitcoin in ProcessMessage()
<TD> ProcessMessage(block, 5798 bytes) FAILED
<TD> received block 00000000000001c0a13e
<TD> REORGANIZE
(...)
<DrHaribo> sturles said he got out of memory errors without being out of memory, and that adding locks and lockers fixed it

Listing 61. sipa, TD & DrHaribo 2012-03-10 IRC: #bitcoin-dev

(...) contents of each node's blkindex.dat database is not identical, and the number of locks required depends on the exact arrangement of the blkindex.dat on disk (locks are acquired per-page).

Listing 62. Gavin Andresen 2013-03-20 BIP50

<gmaxwell> JyZyXEL: versions prior to 0.8 couldn't follow along with new blocks in a somewhat non-determinstic way that depended on the fine structure of the node's local database.
<gmaxwell> JyZyXEL: there are still unmodified 0.7 nodes that are trucking along, though the recent trigger block seems to have gotten most of them.
<gmaxwell> JyZyXEL: the non-determinism of it was why it was absolutely essential to fix it via a hardfork.

Listing 63. 2013-09-03 gmaxwell IRC: #bitcoin-dev

### A.12 Missing transformation assurance

The review process is definetly a good idea, I dont know if it provides as much security as people assume it does. One thing that slip past one person may as well slip past ten people or whatever.

Listing 64. 2020-06-23 Luke-Jr [67]

<gavinandresen> I'm probably reading the code wrong, but I think OP_EVAL wouldn't cause a blockchain split!
<sipa> how so? each client that doesn't support it would fail verifying such a transaction
<sipa> so the first time an OP_EVAL txout is spent, it would cause a blockchain split for those clients
<gavinandresen> Nope. They'd look like anybody-can-spend transactions to old clients, assuming we use OP_NOP1 for OP_EVAL

Listing 65. gavinandresen & sipa 2011-10-02 IRC: #bitcoin-dev

More generally, this OP_EVAL is a very large change that clearly hasn't been vetted nearly enough. It took me all of 70 minutes of looking to find this bug. You guys are not ready to implement this. OP_EVAL turns a fundamentally Turing-incomplete langauge with clear termination conditions into what I believe an "in-principle" Turing complete language that is only held in check by hacks (which haven't even been implemented properly).
You guys need to stop what you are doing and really understand Bitcoin. In particular you should make a proper specification of the existing scripting language, ideally by creating a formal model of the scripting language. Prove upper bound on the space and time usage of scripts. Decide what bounds you want to maintain, and only then start defining OP_EVAL, proving that it preserves whatever properties you want your scripting system to have. OP_IF, OP_CODESEPARATOR, OP_EVAL all have the possibility of interacting complicated ways and you can't just hack the scripting language arbitrarily.

The problem with poor review of other clients/branches can be addressed by largely revamping the download on Bitcoin.org. IMO, Bitcoin.org needs to list 3 categories of clients: "well-tested for production server deployments" (currently bitcoind 0.4.x), "no major problems, safe for everyday users" (Bitcoin-Qt + bitcoind 0.5.x), "testing, please help if you can" (Bitcoin-Qt + bitcoind 0.6.x beta/rc, and MultiBit 0.2.0), and finally "experimental, it compiled, maybe it works" (Bitcoin-Qt + bitcoind "next"); giving users these options enables them to more easily provide input.

Listing 66. gavinandresen 2013-03-16 Github, Issue: 729

<gavinandresen> It wasn't caught by the unit test for a couple of amusing-in-retrospect reasons....

Listing 67. gavinandresen 2011-12-27 IRC: #bitcoin-dev

Before releasing 0.6, I would like to have an "intelligent,
bitcoin-specific fuzzing tool" that automatically finds this type of
bug that we can run before every release. If anybody already has one,
please speak up!

Listing 68. 2011-12-25 Gavin Andresen Email: bitcoin-dev

### A.13 Different versions - client diversity

Diversity is good and may help discover issues. But as Gavin was saying and as I like to point out: The most dangerous kind of failure in bitcoin isn't an implementation bug- any blockchain validation inconsistencies in widely deployed implementations are significantly worse than pretty much anything other than a full private key leak or remote root exploit... and are even harder to avoid.

Listing 69. Gmaxwell 2012-10-28 Forum, ID: 120836

```
Like Olipro, got a lot of people doing custom
builds out there -- in fact, I must use a custom
build on several machines.
```

Listing 70. jgarzik 2010-07-29 Forum, ID: 626

```
If you got 22DbRunRecoveryException and you've used
someone else's build before, you may need to delete
(or move the files somewhere else)
database/log.000000*
```

Listing 71. Satoshi 2010-07-29 Forum, ID: 626

### A.14 Improper reorganisation

```
knightmb, do you still have any of your monster
network available to turn on to help build the new
valid chain?
```

Listing 72. Insti 2010-08-15 Forum, ID: 823

```
Patch is uploaded to SVN rev 132!
For now, recommended steps:
1) Shut down.
2) Download knightmb's blk files. (replace your
blk0001.dat and blkindex.dat files)
3) Upgrade.
4) It should start out with less than 74000 blocks.
Let it redownload the rest.

If you don't want to use knightmb's files, you
could just delete your blk*.dat files, but it's
going to be a lot of load on the network if
everyone is downloading the whole block index at
once.

I'll build releases shortly.
```

Listing 73. Satoshi 2010-08-15 Forum, ID: 823 bitcoin-dev

```
Question about fallout: I had a transaction that I
submitted after the bad block, using the bad block
chain.

What is the status of that transaction?
From what I can tell, my (updated) sending client
wallet shows the deducted amount.

Will it get reincorporated into the fixed chain,
and will the recipient be able to spend it?

Right, it will get reincorporated into the fixed
chain. The transaction won't disappear, it'll still
be visible on both sides, but the confirmation
count will jump back to 0 and start counting up
again.

It's only if you generated a block in the bad chain
after block 74638 that the 50 BTC from that will
disappear. Any blocks in the bad chain wouldn't
have matured yet.
```

Listing 74. Ground Loop & Satoshi 2010-08-16 Forum, ID: 823

```
Most people running clients are not reading this
message thread. So... Silly questions:

1) How will this continue to affect version 3.8.1
(pre-catastrophe) clients with bad block chain?
2) How will this affect clients that upgrade to
3.8.10 but don't remove their block chain files?

1) Once more than 50% of the node power is upgraded
and the good chain overtakes the bad, the 0.3.10
nodes will make it hard for any bad transactions to
get any confirmations.
2) If you didn't remove your blk*.dat files, you're
not helping to contribute to that 50%, and you'll
still show bad transactions until the good chain
overtakes the bad chain.
```

Listing 75. Gold Rush & Satoshi 2010-08-16 Forum, ID: 823

```
Un-upgraded nodes have the correct chain most of
the time, but they are still trying to include the
overflow transaction in every block, so they're
continually trying to fork and generate invalid
blocks. If an old version node is restarted, its
transaction pool is emptied, so it may generate
valid blocks for a while until the transaction gets
broadcast again. 0.3.9 and lower nodes still must
upgrade.
```

Listing 76. Satoshi 2010-08-16 Forum, ID: 823

```
11:24:46 <ArtForz> blockexplorer doesnt deal with
reorgs nicely without manual intervention
```

Listing 77. ArtForz 2011-05-26 IRC: bitcoin-dev

```
<jgarzik> thankfully, that block chain reorg was
not longer than coin maturation
```

Listing 78. jgarzik 2011-04-05 IRC: bitcoin-dev

### A.15 Too high thresholds

```
Activation is dependent on near unanimous hashrate
signalling which may be impractical and is also
subject to veto by a small minority of
non-signalling hashrate.
```

Listing 79. Shaolinfry 2017-04-06 Email: bitcoin-dev

```
- the 95% threshold allows small minorities to veto
proposed changes, lead to stagnation (viz. current
standoffs)
```

Listing 80. Sancho Panza 2017-04-03 Email: bitcoin-dev

```
Possibility? Seems like a probability to me.
Bitcoin.com and ViaBTC are sustaining ~10% of the
hashing power meaning they can veto the upgrade and
they seem ideologically bent on on-chain,
block-size scaling.
```

Listing 81. Yefi 2016-11-15 Forum, ID: 1682183

### A.16 Deploying 'irreversible' changes

```
I'm very uncomfortable with this block size limit
rule. This is a "protocol-rule" (not a
"client-rule"), what makes it almost impossible to
change once you have enough different softwares
running the protocol. Take SMTP as an example...
it's unchangeable.
```

Listing 82. Caveden 2010-11-20 Forum, ID: 1347

```
Why so many OP codes are disabled? Is it possible
that the official client accept them again in the
near future? (at least INVERT, OR, AND, XOR and
arithmetic ones)
```

Listing 83. Jackjack 2011-08-15 Forum, ID: 37157

```
Mark Boldyrev: Back in 2010, there was a bug found
in Core which allowed denial-of-service attacks due
to the software crashing on some machines while
executing a script - see CVE-2010-537. I believe
the removed ("disabled") opcodes should be
re-introduced along with a standardized behavior
definition.
```

```
For example, when execution of an opcode results in
an arithmetic error, such as OP_DIV with a zero
divisor, the script should exit and fail. The
string splice opcodes should also check their
arguments for correctness, etc. These opcodes would
enhance the flexibility of scripts and allow
sophisticated native smart contracts to be created.

AFAICT, re-enabling these old OP-codes would
require a hardfork.

If we had SegWit enabled, we could via a soft fork
allocate new OP-codes for the same functionality
(by introducing a new version of Script). I believe
the Elements alpha project has been experimenting
with re-enabling old OP-codes:
https://elementsproject.org/elements/opcodes/
```

Listing 84. Mark Boldyrev & Hampus Sjøberg 2017-05-19 Email: bitcoin-dev

```
<ForceDestroyer> Did you read the thread? It's a
bit convoluted if you want the counter- and
counter-counter-arguments further down, but the
first post has the rough idea
<ArtForz> ForceDestroyer: increasing the block size
limit *when it's needed* won't be really
controversial, so it's only a matter of changing a
#define and getting everyone to update
<Diablo-D3> Im trying to read the thread, but the
amount of dumb shit in it is amazing
<Diablo-D3> ArtForz: WELLL
<Diablo-D3> we should do it early
<Diablo-D3> just so we arent stuck with assholes
who wont upgrade
```

Listing 85. ForceDestroyer, ArtForz & Diablo-D3 2011-04-23 Email: bitcoin-dev

```
What was the reason for disabling these opcodes in
the first place? I can
understand wanting to prevent excessive
signature-verification, or limitation
of arithmetic to a limited amount of bits, but
completely disabling all
arithmetic beyond addition and subtraction, and all
bitwise operations seems
very limiting to me. Thus, if we agree to do a
future incompatible update,
i would vote to re-enable these, and maybe allow
arithmetic up to 520 or
1024 bits numbers.
```

Listing 86. Pieter Wuille 2011-08-24 Email: bitcoin-dev

```
<[Tycho]> But someday additional ops should be
enabled anyway...
<gavinandresen> [Tycho]: agreed. someday....
<luke-jr> the critical block and secure string
changes, I wasn't so sure about
<luke-jr> they seem more like refactors with a
potential for damage
<luke-jr> [Tycho]: probably the best way to handle
it is to do a single block chain fork, changing as
many things as possible
```

Listing 87. Tycho, gavinandresen & luke-jr 2011-12-02 IRC: bitcoin-dev

```
<roconnor> worst case is that we are stuck with
OP_EVAL forever, which means miners never get the
opportinity to implement static analysis of script
code ever again.
```

Listing 88. Roconnor 2011-12-28 IRC: bitcoin-dev

```
While we could right now make all these rules
non-standard, and
schedule a soft fork in a year or so to make them
illegal, it would
mean removing potential functionality that can only
be re-enabled
through a hard fork. This is significantly harder,
so we should think
about it very well in advance.
```

Listing 89. Pieter Wuille 2014-02-19 Email: bitcoin-dev

### A.17 Not prepared for forward-compatibility

```
This is another problem that only exists because of
the desire to soft fork. If "script 2.0" is a hard
fork upgrade, you no longer need weird hacks like
scripts-which-are-not-scripts.
```

Listing 90. Mike Hearn 2014-11-04 Email: bitcoin-dev

```
Ya joking? A scripting system inside a scripting
system. Hacks on hacks on hacks will lead to a
messier protocol than FTP is now. Well, it seems
good at first glance. But fast-tracking this into
the block-chain is probably not a wise idea.
There's no rush so it might be prudent to think of
this as something for 2 years time or later.
Bitcoin is not exploding tomorrow, so there's no
big loss from holding off on momentous changes like
these. https://en.bitcoin.it/wiki/BIP_0001That's a
good place to start. Re-enabling parts of the old
scripting system in a controlled manner is a good
idea. Adding new operations- not so much *right*
now.
```

Listing 91. genjix 2011-10-02 Email: bitcoin-dev

### A.18 Lacking knowledge regarding network dynamics

```
To ensure that the network is not partitioned and
that segwit blocks are being passed to segwit
enabled nodes, a Core 0.13.1 node will use its
outgoing connection slots to connect to as many
nodes with the NODE_WITNESS service bit as possible
(...)
```

Listing 92. achow101 2016-11-26 Forum, ID: 1682183

```
In addition to defining witness structures and
requiring commitments in future blocks (BIP141 -
Consensus segwit BIP), new mechanisms must be
defined to allow peers to advertise support for
segregated witness and to relay the witness
structures and request them from other peers
without breaking compatibility with older nodes.
```

Listing 93. Eric Lombrozo & Pieter Wuille 2016-01-08 BIP50

```
<BlueMatt> handling connections to othernodes on
the p2p network and then handling block chain
reorgs is hard
```

Listing 94. BlueMatt 2011-08-13 IRC: #bitcoin-dev

### A.19 Insufficient damage control

```
I've lost way too much money in the last 24 hours
```

Listing 95. Eleuthria 2013-03-12 IRC: #bitcoin-dev

```
So I think the only way Mallory gets free beer from
you with segwit soft-fork is if:
- you're running out of date software and you're
ignoring warnings to upgrade (block versions have
bumped)
- you've turned off standardness checks
- you're accepting low-confirmation transactions
```

```
- you're not using any double-spend detection
service
```

Listing 96. Erisian 2015-12-18 Email: bitcoin-dev

```
DO NOT TRUST ANY TRANSACTIONS THAT HAPPENED AFTER
15.08.2010 17:05 UTC (block 74638). We are
investigating a problem.
```

Listing 97. Satoshi 2010-08-15 Forum, ID: 823

```
If you're unsure, please stop processing
transactions.
```

Listing 98. 2013-03-12 Pieter Wuille Email: bitcoin-dev

```
For now I have stopped generating on my nodes.
```

Listing 99. 2010-08-15 aceat64 Forum, ID: 822 bitcoin-dev

```
<slush> jgarzik: does your pool have any checks
against double spending?
```

Listing 100. slush 2011-02-02 IRC: #bitcoin-dev

```
<moa7> or appoint an independent 3rd party to audit
your blocks for irregulaties is another option
(...)
<[Tycho]> moa7, I already started a pledge of 50
BTC to make fork monitoring site (that was like
~1500 USD), but no one cared. Also I proposed a
change in miners code that will add something like
anti-fork functionality at client side, independent
from any pool, but no one cared.
```

Listing 101. moa7 & Tycho 2011-07-08 IRC: #bitcoin-dev

```
I'm afraid the community is just too big and
distributed now to expect much in the way of
voluntary quick action on anything, especially
generation which I'm sure many have on automatic
and largely unmoderated.
```

Listing 102. kencausey 2010-08-15 Forum, ID: 823

```
1. Where are we at with network health? What
metrics should we be using? Is there work to be
done?
And meta-issue: can somebody volunteer to be the
Bitcoin Network Health Inspector to keep track of
this?
```

Listing 103. 2011-08-10 Gavin Andresen Email: bitcoin-dev

### A.20 Improper miner incentives to enforce new rules

```
If there is a cost to verifying transactions in a
received block, then there is an incentive to *not
verify transactions*. However, this is balanced by
the a risk of mining atop an invalid block.
```

Listing 104. nathan 2015-07-11 Email: bitcoin-dev

```
<sipa> just require the vereion bit to be set to 1
in the first block that has the rule activated
<CodeShark> I'm not too concerned about that, I
suppos
<sipa> so your second bit is purely informational
<CodeShark> yeah, in this example it would be
<sipa> there is no point in that, i think
<sipa> there is no reason why miners would not
incorrectly set that bit if they are already
incorrectly setting the other
<CodeShark> in the end, what matters is not really
whether or not miners acknowledge the version
change...what matters is whether they enforce the
new rule
<sipa> yes, and you can't measure that in a softfork
<sipa> in a hardfork you can require the forking
block to be explicitly incompatible with the old
rules
<CodeShark> with BIP66, imagine what would have
happened if miners would have been able to continue
mining version 2 blocks after the rule change...
<sipa> yes, that's why i think forcing a fork is
good
<CodeShark> there were two things at play - 1)
whether miners were enforcing the version rule, 2)
whether miners were enforcing BIP66
<sipa> oh
<sipa> you can't force a fork
<sipa> i was wrong
<sipa> not in any useful way
<CodeShark> so then this boils down to a problem of
miner incentives
<sipa> only informationally
```

Listing 105. sipa & CodeShark 2015-09-129 Email: bitcoin-dev

```
Satoshi Nakamoto's original Bitcoin implementation
provided the nSequence number field in each input
to allow replacement of transactions containing
that input within the mempool. When receiving
replacements, nodes were supposed to replace
transactions whose inputs had lower sequence
numbers with transactions that had higher sequence
numbers.

In that implementation, replacement transactions
did not have to pay additional fees, so there was
no direct incentive for miners to include the
replacement and no built-in rate limiting that
prevented overuse of relay node bandwidth. Nakamoto
removed replacement from Bitcoin version 0.3.12,
leaving only the comment, "Disable replacement
feature for now"
```

Listing 106. Dave A. Harding & Peter Todd 2015-12-04 BIP125

## APPENDIX B
## ROOT CAUSE ANALYSIS

The appliance of Ishikawa diagrams [14] as seen in Figure 11 further enhanced the analysis for RQ2 (lessons learned). The identified root causes were classified within the categories of human errors [63] to gain further insight. The different errors were applied as codes during grounded theory analysis as seen in Section 5. The complementing of the grounded theory approach with root cause analysis gave higher confidence in covering relevant issues and gaining an in-depth understanding.

We based the diagram's main paths on blockchains' roles (validators, users/merchants, developers and miners). The first stage of each consensus change issue was based on a surfacing problem, such as "Introduce bugs or vulnerabilities". After that, we based the following steps on "the five whys" to understand the root cause of the surfacing issue: "By repeating *why* five times, the nature of the problem as well as its solution becomes clear" [88]. Sometimes the issue could be understood without going five steps deep, and we did not go further. Additionally, some root causes are similar for different surfacing issues. For instance, "Introduce bugs or vulnerabilities" and "Not prepared for consensus failure" could be caused by the same reason: "Do not understand the implications of a change". We highlighted similar causes by colour-coding the diagram. From the complete diagram, we derived which lessons learned were relevant for consensus changes in particular.

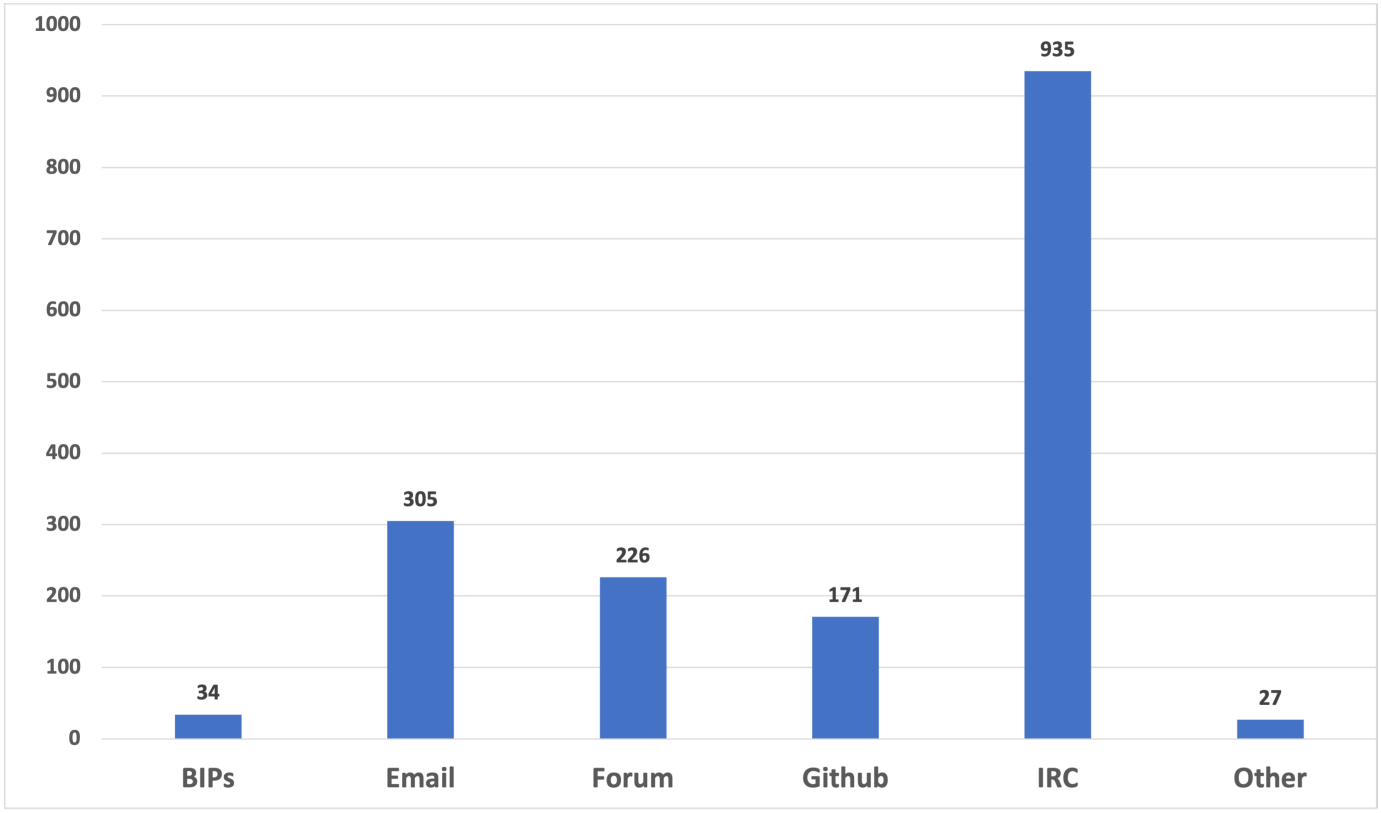


Fig. 9. Sample distribution among different sources.

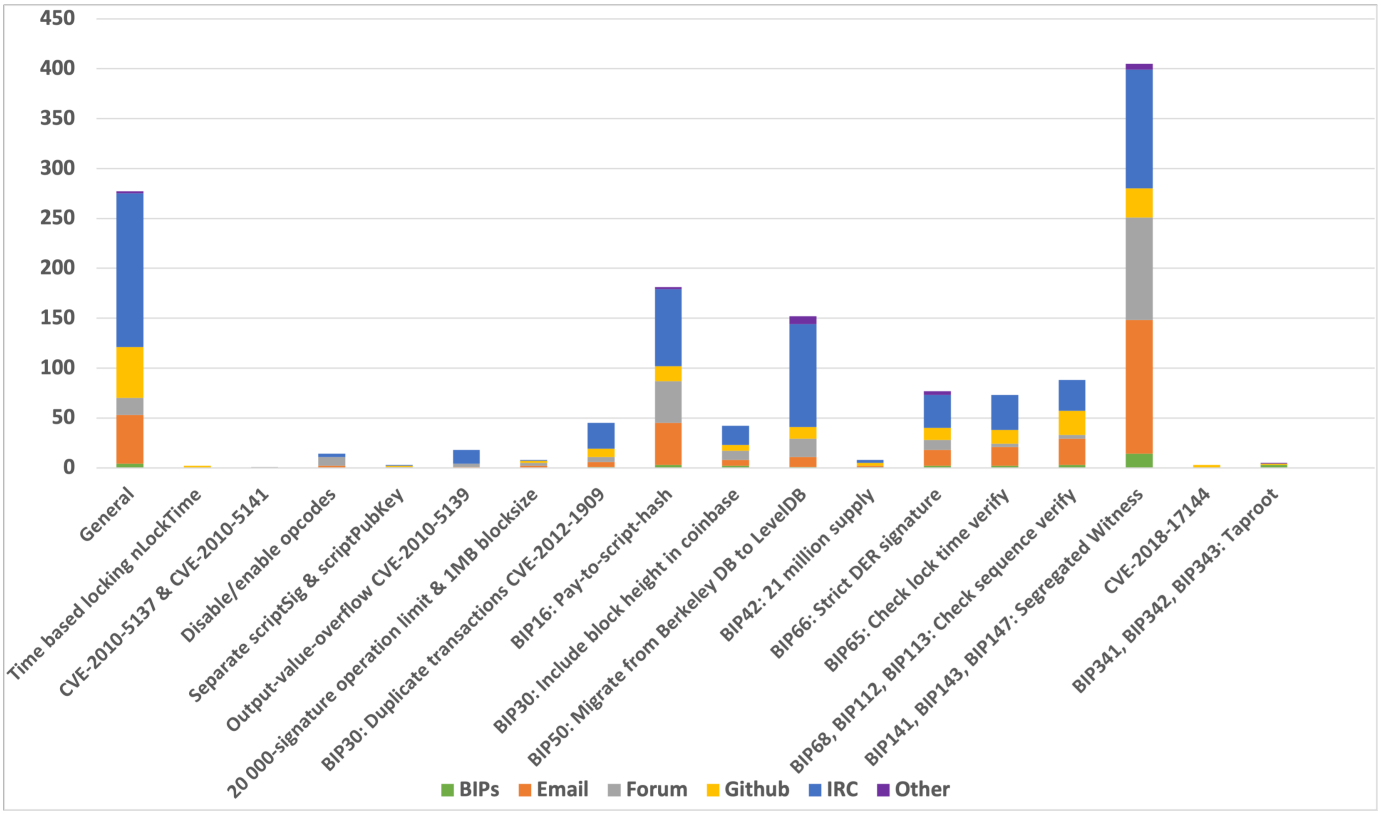


Fig. 10. Sample distribution among different consensus changes. The sum of samples across the events is larger than that across sources because some samples are related to multiple events.

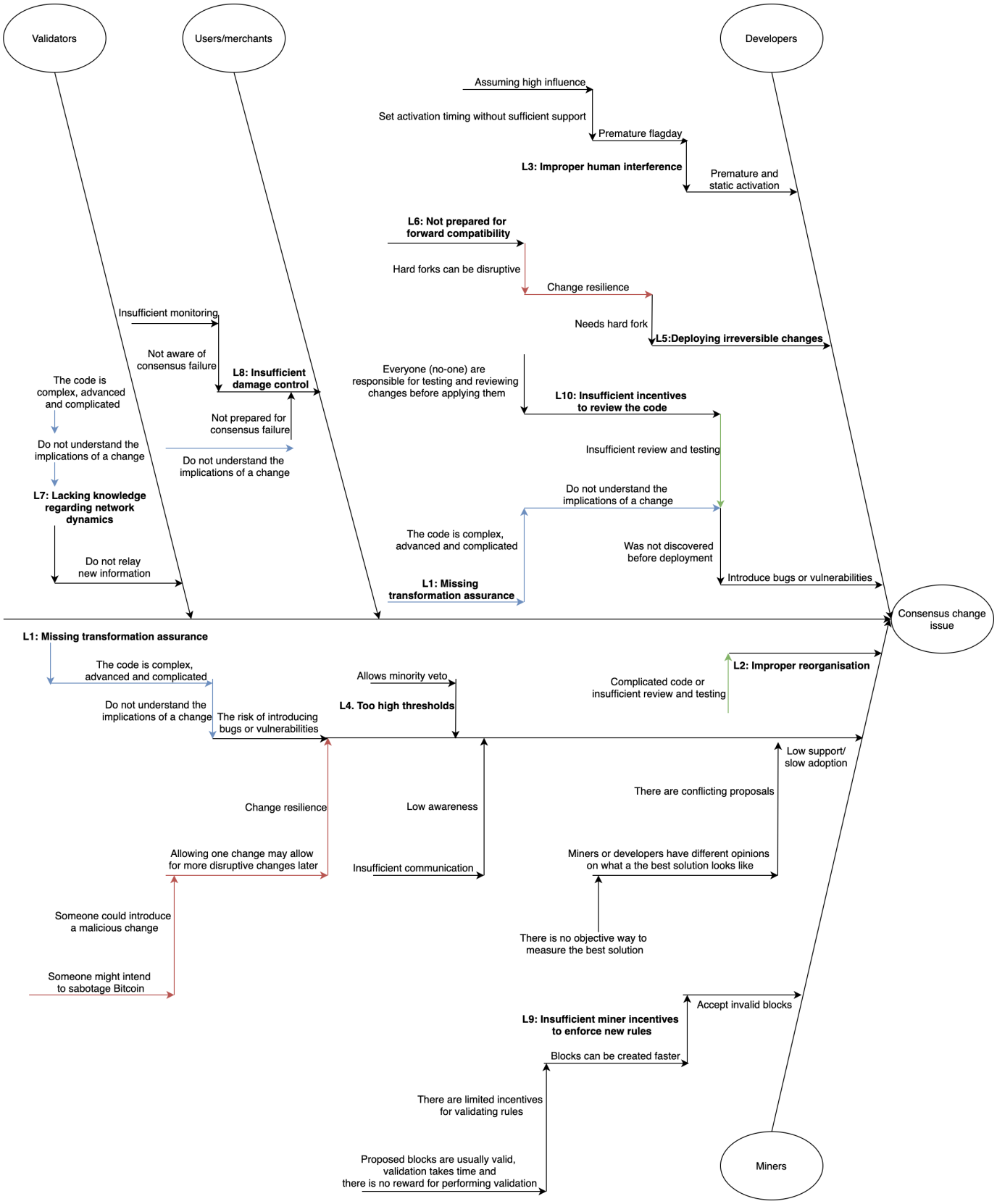


Fig. 11. The figure shows the Ishikawa diagram used for root cause analysis and reveals lessons learned.