**R-chain blockchain platform: common architecture and evolution**

**Attributes of corporate and intercorporate (B2B) systems**

To release the platform into the commercial exploration smoothly and without unexpected incidents, the platform should satisfy the following requirements:

**The platform should have an operator** – a legal person, responsible to the Members for platform functioning. The scenario, which involves an independence of every member, as well as the crucial role of members’ consensus, does not work there.

**The platform should be used by Member based on rather simple legal support**, the closest to an existing electronic workflow. Otherwise, the last question about your Platform may be asked by an ordinary corporate lawyer: “What will I take to a court?”. And, believe, he does not expect to get a description of mathematical algorithm of the multi-range blockchain network consensus in 15 pages as a response.

**The system should satisfy the requirements of regulators and “workflow customs” of the cryptography**, used for data security and insurance of legal significance of the actions. For example, when advertising your product to Russian clients, you will find that the words “we use GOST” raise a bunch of embarrassing questions and dramatically reduce a desire to use your Platform.

**Components of a Platform node, set up for Members, should satisfy a network segmentation, accepted in corporate systems**. The situation when a DMZ includes any commercial secret or digital signature keys will lead to collapse.

**The possibility of urgent changes in transaction states when the expected role model is disrupted**. There is no secret that everything can go wrong. Business can be highly mysterious, and the regulators and court decisions make things even more weird, and often post festum. Suddenly it can turn out that a transaction cannot be managed with any existing role model. In that case, a Platform should provide some mechanisms to bring a transaction into a required condition, provided that there is a proper consensus between members.

Some of you can be surprised that there are no words “transactions per second” and other performance ratings in that list. But our experience shows that the performance is not the most important condition of successful Platform implementation. The performance just needs to be “sufficient enough”. There is plenty of ways to create a data streams architecture, which allow to multiply the Platform capacity – from the data register to “lateral” turbo-streams. Your project will waste away far more likely if you do not pay attention to the details above.

**Common platform architecture**

*Architecture of program components*

Applications, used in our research projects in 2016-2017, were created based on the direct integration of dialogue interfaces with “technical” components of distributed platform – geth and swarm. However, having analyzed the possibilities and consequences of this approach, we came to conclusion that it is necessary to introduce an adapter of unified business objects - one more layer between the business-backend and “technical components”. This technique refers to standard program architecture templates, which still does not eliminate its effectiveness. As a result, we got the following program architecture of the Node of our Platform.

Busines processes mirror

Adapter DLT

DLT platform

Business-processes Database

Transaction formation

Distributed file system

Data synchronization

Cryptographic information protection means

State of business-processes

Queue of macro-operations on business -processes

Adapter DFS

Files request

Files lay-up

Local file store

Mobile app

Member's accounting system

Business app

WEB-app

According to the proposed architecture, Business app works not directly with abstractions of DLT-components, but with a certain “universal business process” (further – Process) – creates a Process, changes Process state. “**Business process mirror**” is called a set of components, which convert universal Process data structure and operations into the data and operations, used in DLT-client. “**Mirror**” also conducts a reverse conversion of the information, received from DLT-client, and filters Processes information, not related to the Node owner Member. Summing up, in every Node there is maintained a synchronous Database of relevant to this Node business processes. Besides, the format of the database is the most convenient to the Business app. Business app interacts with this Business processes Database, receiving information about Processes state and sending operations on the state changes.

*Universal business process*

There is a natural question which attributes should the **Universal Business Process** have, so these attributes provide a maximum use of blockchain-platform advantages on the one hand, and a maximum flexibility and functionality in Business app usage. There was additionally required an opportunity to implement selected attributes in the most common DLT-platforms, the functionality of which can significantly vary in different aspects (Ethereum/Quorum/Masterchain, Hyperledger Fabric, Corda, EOS, Waves). Bases on our experience and other projects, we made the following conclusions.

Universal Business Process should have the following attributes:

* Process parameters (type of Process, status, contextual Process attributes)
* Role list of Process members
* List of electronic documents related to Process
* Process transitions map

At the same time, the Platform should provide the following conditions of the Process spread in the net on the level of blockchain-component:

* Completeness and wholeness of the information
* Confidence of the electronic documents outside of the pool of Process members
* Control over the adherence to the Process transitions map
* Storage of the revision history of the Process states

In order to convert these opportunities, the “frame” smart contracts with the relevant sets of attributes and methods were developed.

We will shortly discuss some of the noted attributes and conditions – what, how and why.

**Process parameters** are of a relatively open nature, as they are transmitted directly through the smart contract. They are entirely public for some blockchain platforms (Ethereum/Masterchain), while for the others they can be closed by standard means of data privacy insurance (Quorum – private smart contracts, Hyperledger Fabric – channels and private data). Probably, the most important parameter of the Process “core” in terms of our implementation is a “type of Process”, as it carries not only a conceptual load, but also a functional load. Depending on the “type of a Process’, **DLT Adapter** selects a smart contract template for the current Process. What is it for? Obviously, there are numerous transaction types, as well as business process, providing their implementation. In many cases business processes differ only with their transition maps (in terms of a Platform) and can be implemented by means of only one smart contract, supporting the random transition map (see below). However, quite unique aspects can be related to a concrete business process:

* Appeal to the Oracle (for example, for the transactions with exchange rate)
* Appeal to other transactions or virtual accounts (for example, to the automatic reservation of the funds or checking cash)
* Action time control (for example, check of the deadline of documents submission on the letter of credit of calls of the guarantee)
* Etc.

It is clear, that it does not make sense to formalize this diversity of alternatives into the only one template of smart contracts. The development of the unique template for a concrete type of business process is significantly more effective way. It provides several options: a high flexibility; a possibility of a “strict” insertion of both necessary elements and critical examinations into the “core” of a blockchain-component, which eliminates any manipulations on the part of individual Members. Apart from this, the Platform will combine a commonality of interfaces for Apps-Processes interaction and a high modularity of their specific functionality.

The “core attributes” of out Processes also include a “status” and “remarks”. The first one is a short description of the Process state ("New", "Canceled", "Closed", "OnАpproval"), the second one is a “long” string of a more detailed “status” description. This long description is limited to approximately 1000 symbols (e.g., “Insufficient funds”), because there is another mean of transferring large amount information (especially confidential) – electronic documents, attached to the Process.

**Process transition map** describes whether a Member of a concrete role can change a Process state and how he can do this. The control over transition admissibility is performed by the “core” of the blockchain component and cannot be falsified by any “disenfranchised” member. Moreover, transition map can be used by Business app to detect any possible actions by the Node owner in the current Process status for the relevant management of dialogue components.

**Confident data transmission**. Electronic documents attached to a Process are used for confident data transmission. Business app posts necessary documents to the local file storage of the “Mirror” and refers to them in the Operation of the Process state change. Before transmission from the local storage to the DFS, the document is encrypted into the public keys of receivers’ Nodes – Process members. After transmission of the formed container to the DFS, its link or a hash-sum of the source document are transmitted to the Process smart contract. After receiving of the information about Process state change, file requisites are extracted into a business process database. After that a crypto container is extracted by its link from the DFS, it is then decrypted with the key of a receiver Node. Then there is an examination of the conformity of the document hash-sum with the one signed in the smart contract, and the document is sent to the local file storage and becomes available for a Business app. So, Business app works only with an open document version – “Mirror” is responsible for all other concerns about its safe transmission.

**Revision history of the Process states** – the sequence of “staff”, responsible for a certain operation of the Process state change. We store the following data for each member of the “staff”:

* status
* remark
* ID of the initiator of transaction state change

Revision history is recorded on the smart contract level and allows not only to track transmissions sequence for the audit purposes, but also allows Business app to correctly process action sequence, even in case of their simultaneous onset (for example, lagging, break in functioning etc.)

**Ensuring of legal value** – is a highly important question, discussed in the section “Attributes of corporate and intercorporate systems”. Initially we held the view that the legal value should be provided not by the means of a blockchain platform, but by use of the outside PKI, which has a regular support or a relevant trust level behind the Platform members. Roughly speaking, an electronic document, providing a legal value of your actions (pay document, contract etc.) and being attached to your document, should be signed on the base of PKI (GOST in Russia, SSL or PGP/GPG somewhere abroad). Business app will examine an “external” signature and perform a relevant action. Or it will not do it – depending on the result. Someone can say that it is wrong and “it is necessary to convince the lawyers of the legal value of blockchain transactions”. Be sure, we have experienced a lot and have always had the same result. Besides, in Russia Masterchain certification (<https://www.cryptopro.ru/news/2019/10/zaklyuchenie-fsb-na-skzi-masterchein-pervaya-sertifikatsiya-blokchein-resheniya>) provides some certain possibilities – so good hunting!

*Advantages of the universal business process*

Where does that get us?

* **Enhancement of the pool of potential developers of the decentralized business applications**

As the developer of a backend application is not required to work with any decentralized components in case of the proposed “mirror” architecture, it becomes possible to recruit a wide range of “common” developers, who are experienced in the necessary product areas. It can significantly reduce time frame and cost of development and increase its quality. In our projects, three out of four developers had never worked with blockchain systems before, and one more developer had worked with Corda, while the app was working via “Mirror” with Ethereum and then with Quorum. On the other hand, there are highly qualified blockchain developers who firstly are limited, and secondly are paid significantly higher than “common” ones of a similar qualification. They can perform some really complicated tasks related to the evolution of decentralized platforms instead of routine repeated replication of wrapping business-data into blockchain.

* **Apparent stability**

It is clear that any complicated program systems (and decentralized apps definitely refer to them) can go wrong. Taken into account a “youth” of many used decentralized components, there is an increasing risk of this “can go wrong” situation. No one would love to see a message like “*Your transaction is cancelled, because ...”*. If program layers are divided into a “Mirror” and a Business App, a “Mirror” will hide these errors from the end user, and he will interpret a general system functioning as correct. Still he can be confused only with relatively slow and unnecessary event, like a call from another member: “*We have agreed on 14, but now it is already 16 and you have still not done this..*.”. So, the technical support will have several minutes or hours to detect and correct an error. If the technical support does not forget its responsibility, the end user can never know about this terrible "*insufficient funds for gas*" or "ga*s required exceeds allowance or always failing transaction*". Many of those who has been involved in the demonstration of prototypes to important customers know the value of this gained time!

* **Possibility of the hidden modernizations of the blockchain basis**

Business App does not directly depend on the concrete implementation of blockchain basis, and it allows to change this basis without affecting the Business App. Labor costs for a Business App (especially with a developed dialogue interface) can comprise 75-95% of the total labor costs for a program complex. Of course, an intelligent leader will set up interface classes and take another stapes to create a modularity, but still there will be a bad aftertaste. Moreover, taken into account developers’ tendency to stay in the same place, there can arise another problem: “*The one who worked with it has already came away, but he had worked with agile*”. The approach “*It is better not to redo the thing which is still working*” was repeatedly proved by life. Thus, you can:

* + Replace a blockchain basis or DFS if you initially did an incorrect performance measurement, or were mistaken in the choice of blockchain platform, or decided to change any previous platform into another solution, or a new platform has been developed etc. So, a lot can happen in the time of rapid evolution of all the components of decentralized systems.
  + Develop complicated combined solutions as a blockchain basis. For example, two parallel blockchains Ethereum were used in one of the our research projects – this allowed to, on the one hand, create a channel for unlockable high-priority operations, having left package and unurgent operations in their own channels, and on the other hand – just to multiply capacity.
* **Possibility to change a legally significant cryptography**

In some degree, this point refers to the previous one. But according to our experience with International guarantee project, we decided to put it separate. In terms of wide interpretation of this experience, in case of cross-boundary application, and especially in many jurisdictions, it is advisable that your Platform could not only change its cryptography, but also work simultaneously with different cryptography with different members. A lot of interesting and technically complicated questions can arise there. For example, whether it is possible to insert some trusted nodes for re-encryption and signature insurance. But we will figure these questions out.

* **Potential possibility of standardization and gateway exchange on the Process states level**

This point seems to be fantastic on the current level of corporate decentralized apps. But necessity is mother of invention. Abstract objects creation and formalization – is an essential step in standardization and a necessary condition of a wide integration of blockchain nets. It is clear, that only a Universal Business Process (or its analogue) is not sufficient – complicated relations, and physical world processes, and requirements of business apps will certainly lead to the establishment of other abstract objects with their own features and possibilities.

*Disadvantages of the universal business process*

Firstly, I give an example: chain-code Hyperledger Fabric or Corda, opposite to so graceful Solidity, allow to implement almost infinitely complicated business process logic, and such approach disregards their functionality. And yes, they will be absolutely right. For those, who grumble, I will remind some well-known issues from software engineering:

* Where should you apply business logic: in the stored procedures in the database or in the program code of Business app?
* What is better – universal computer or special computer?
* Are you sure that a selected basis will maintain interoperability after the consequent updates? Will it function in 2 years?

The answer is quite simple.

* If you have a lot of money, time and available specialists in blockchain technologies
* If you are sure that you will never need to change a selected blockchain basis
* If you **really** need to make the most of your platform at 101%

Then you need a special computer, I mean Hyperledger Fabric or Corda adapted to a chain code and equipped with other hard attributes. If not – think for yourself.

*Node network monitoring*

Probably, it would be a revelation for some of you, still a well-designed monitoring is a core factor of successful software exploitation in the enterprise. This concept involves not only standard infrastructure server monitoring, but also a functional monitoring of software components. Your distributed Platform should not only function correctly, but also make mistakes correctly, providing a support service with enough appropriate information, which enable to detect, identify and resolve any failures as fast as possible. It would be even better, if a monitoring system has some proactive features – predict and prevent disruptions before they happen.

As noted before, current decentralized systems are constructed from components with high level of technological risk. Everything that can go wrong, will go wrong. That is why if you somewhen do not understand what happens to the Nodes of your network, and you are still going to work according to “users’ call” – it would be better to make room in the queue and not take users’ time. If an error in your Platform will be found by some user, but not by your support service – it is all over, you are tainted forever!

Summing up, your Platform should have a proactive monitoring system since the very beginning. We will shortly describe it:

* Special smart contract is set up in the Platform blockchain basis, which is responsible for a collection and dissemination of the monitoring information (further abbreviated as SCM)
* One, or better several, of the Nodes of the central monitoring (CM) are assigned to be responsible for delivering “sounding packaging” within the network, and a collection and dissemination of the monitoring information, coming from the other nodes. This functionality can be either core, or additional.
* For a blockchain basis, CM forms “sounding transactions” with a prescribed frequency, switching the relevant element of SCM to a new state – this can be just a time stamp.
* For a DFS, a “control file” is similarly formed and distributed. Link to this file is also recorded in the SCM.
* Each of the Nodes periodically calls to a SCM, extracts control data and a control file from the DFS and examines the validity of control stamps.
* Apart from that, each of the Nodes reports its state to the SCM. The state information includes:
  + A control time stamp
  + The last received value of a “sounding” stamp in the CM within the blockchain channel
  + The last received value of a “sounding” stamp in the CM within the DFS channel
  + A number of the last processed block of the blockchain channel (for an Ethereum group or an alternative score)
  + Presence of errors in operation queues of the “Mirror”
  + Presence of delay in operation queues of the “Mirror”, that are operations, which were not completed in time
  + A number of operations in operation queues of the “Mirror”
  + Presence of errors in the database in the “Mirror”
  + Control information from Business Apps

Having some certain set of values of monitoring indicators, “Mirror” automatically stops processing of the operation queues and blocks any potential undesired consequences. For example:

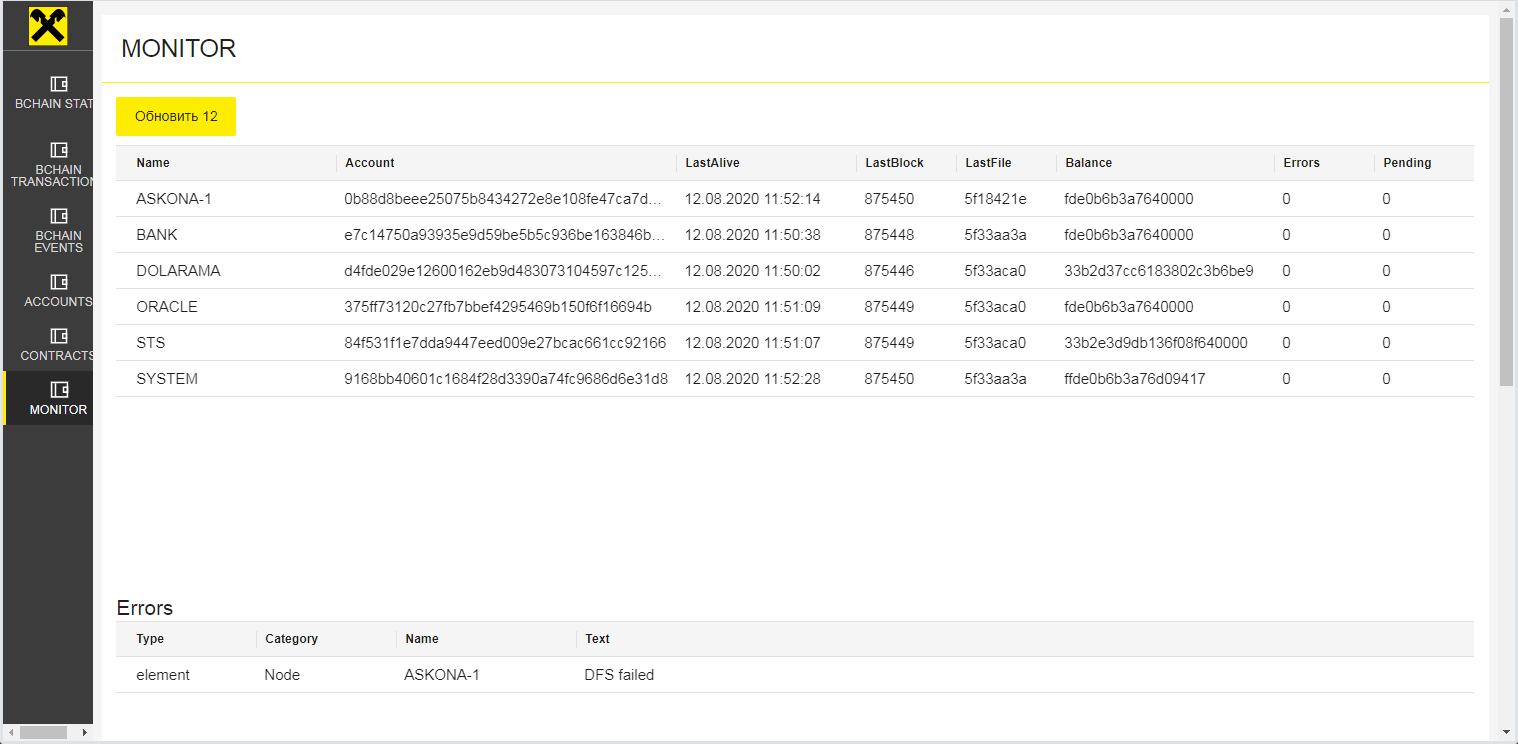
* In case of critical delay of the control stamp of the blockchain channel, which is interpreted as fallout of the Node from a blockchain network or a complete violation of its functioning
* In case of critical delay of the control stamp of the DFS channel, which is interpreted as fallout of the Node from a DFS network or a complete violation of its functioning
* In case of errors in the operation queues all the following operations, related to this business object (universal business process), are blocked

Special attention was paid to the processing of errors in database, used by “Mirror”. This database is used not only as a Business app interface, but also as a status stack for a state machine of “Mirror” operations queue. Experience has shown, that if there are specific errors during the data change in the database, there can happen an operation loop with a mass repeated sending of transactions etc. Once, due to a similar problem, we have processed a half-year volume of queue in Quorum in 2 days. So, in case of this type of errors, the “Mirror” completely turns off and waits for manual manipulations by a support service.

Nodes of Central Monitoring extract information about all the Nodes within network (including its own Node) from the SCM. This information is analyzed, which allows to promptly detect such dangerous or potentially dangerous states, as:

* Complete violation of functioning of a blockchain or DFS network
* Fallout of certain nodes from a blockchain or DFS network
* Delay in certain nodes during the processing of blockchain channels data
* Presence of errors in “Mirror” processing queues
* Presence of operation lagging and excessive growth of operations in the “Mirror”

A picture below shows an example of the primitive monitoring process within one of our networks:



Even more “high level” proactive monitoring schemes were developed and implemented, including business apps. But there we approach a “thin line” of intellectual property of certain customers.

We will not hide that monitoring traffic comprise a major part of total traffic in some of our blockchain networks. In relation to this, we got an idea to use a floating schedule of “sounding packaging” frequency – more often in the working hours and rarely in the off hours. But it is worth this, really.

In total, we advise you to monitor everything you can and what a capacity of your decentralized networks allows you to do. In other words, is not it always the last thing you find that offers the answer?

**Evolution of different technological components use**

We have reviewed the common conditions of deployment and functioning of corporate decentralized systems, and also the architecture principles, that we used during R-Chain Platform implementation. Now let’s consider why and how certain technical components were developing during the implementation of concrete projects.

**The first one was an International guarantee project (**[**https://www.kommersant.ru/doc/3797613**](https://www.kommersant.ru/doc/3797613)**), where we collaborated with our Belorussian colleagues since the March to December 2018.**

We have started with a configuration Ethereum - Ethereum Swarm – Crypto-Pro (DLT-DFS cryptography), which has proved its worth in previous research projects. Instead of previously used public testing PoA network Ethereum Rinkeby, we used a private network Ethereum PoA and a private network Ethereum Swarm. There were no technical problems in the beginning, however then we faced a “cryptographic” problem – one of our Belorussian members flatly refused to use proposed cryptographic instruments, referring to a local law about electronic workflow. We did not manage to find a rapid solution, but we got a consisted understanding of difficult, but important role of cryptography in the international projects success.

We have detected some failures in Ethereum Swarm work already during the testing of control transactions on a real network infrastructure (every member implemented a Node on his own resources) – file losses comprised 20%. We made a suggestion that these losses were related to problems, arising in the client Swarm during the parallel sending of several files. This suggestion was proved in general – we have empirically found that there should be 5 seconds pause between file sending in Swarm. We also revealed another problem during the transition to the runtime version, which had required a transit Swarm Node due to some features of applied network slicing in the Raiffeisen bank infrastructure. The problem was that Ethereum Swarm allowed 30% files loss during the work through a transit node. “Sliced” architecture and a well-designed monitoring system allowed to successfully conduct a real release of guarantee in a “hand priming pump” manner. But Ethereum Swarm’s fate was sealed. It should be noted Ethereum Swarm was selected as a technological DFS core primarily for its ability to work in topologies without a direct sender-receiver link. But its inability to work reliably in this mode made things worse.

It should be noted, that a private network based on the Ethereum pleased us with its ability to recover – project schedule supposed, that a released guarantee would be closed 3 months after its release, and some members closed their nodes during this period. Still, the network recovered its wholeness during a repeated release in just one day, and a closure operation went successfully.

**The next project was a creature of intra-group network for a companies’ group Ascona (https://www.kommersant.ru/doc/3985251) – September 2018 – till now.**

We have selected IPFS (InterPlanetary File System) as a technological core for DFS, taking into account International guarantee project experience. It successfully conducted a parallel file sending and did not require any additional manipulations. The only weak spot was its inability (specified!) to work with transit topologies. In case of networks with a large number of members, it is extremely hard to provide every member with access to all the other network members. On the other hand, all members can access each other via “base” Operator’s nodes. Consequently, the following mechanism of free file distribution was implemented:

* If a file is attached to a smart contract of some business process, there is formed a special event - DeliveryFile, containing a link to the file
* In the Operator’s infrastructure, one or several nodes capture DeliveryFile events and extract this file from IPFS. Due to IPFS structure features, the node, which received a file, becomes its distribution spot. Nodes of leading members can accept a similar function.
* If the member node wants to receive a file, but does not have access to a source node, it instead uses an Operator’s transit node to receive a file, if there is a direct link between these nodes.

Thus, Ascona project has started in the configuration Ethereum – IPFS – Crypto-pro.

Use of Crypto-Pro for file encryption and “external” signature of legally significant documents provided both a simplicity of juridical operations and an absence of claims from Information Security departments. This caused highly positive effect on the approval time frames as from the bank, as from the companies of Ascona group. In general, the project was developing on a regular basis, it completed a pilot stage and was ready for a release. And then...

And then an incident happened in two of our projects at one stroke – these projects were figuratively “others”, we were involved in them as experts. And here, with the analogical configurations, we got mega-forks of several thousands of blocks, with a loss of transactions in several networks. Having analyzed log files and investigated forums of blockchain communities, we came to a discouraging conclusion: it is highly risky to use Ethereum PoA (and sometimes even PoW) in compact networks with a small number of nides (corporate networks relates exactly to them). Moreover, we discovered a mysterious bug in our test network: a node was falling out from a network and then was never synchronizing with it again – even after a reinstallation of Ethereum and a full clean-up. To tell a long story short, it was evident that we need some alternative to a blockchain basis for a production network. And as quick as possible.

We have found a solution – Quorum – which is almost a blood brother to Ethereum. “Mirror” required a minimal number of modifications, while a Business App did not require any modifications at all.

To a date, we have found only advantages related to transition to Quorum:

* Utilized Raft-consensus eliminates any forks
* There is no empty block, which reduces a size of chain

Absence of forks allows to escape any pauses before a figurative finalization of transaction. This pause was mandatory due to a potential throwback of transactions and it took six cycles of blocks generation. And firstly, it obviously increases a Platform speed. Secondly, it resolves highly complicated problems, which arise if a fork exceeds an estimated pause and a state of affected “mirror” business objects does not satisfy their expected blockchain state.

The only disadvantage of Quorum is its trait to generate a mega-block of several megabytes when restarting after a long pause. This mega-block just loops Adapter DLT if you try to sort its content out. However, strictly speaking, a support service should not sleep for so long.

**Eventually,** after so much trouble we obtained a configuration **Quorum – IPFS – Crypto-Pro,** which we currently use in a Russian domestic market.

Someone can ask a fair question: “Have you never heard about Quorum before?”. Of course, we have heard both about Quorum, and Hyperledger Fabric, and EOS. And the author has even attended the very first workshop in Russian on Corda in the autumn of 2017. Probably, it was to find a bright answer for such kind of questions that a Hegel has invented his Dialectics for. A small team, conducting a research since 2016, was well experienced in Windows software development. As well, a public Ethereum (probably it was a test version) was the lowest threshold of entering blockchain. Still our goal was to conduct research on precisely a blockchain topic, but not to pock around various dockers which are essential to launch any “mature” Quorum or Hyperledger Fabric (it can be even impossible on some virtual Windows-based platforms). Consequently, our choice was clear. As far as our findings started to attract business-subdivisions of the Bank and its partners, it became possible to expand the team and define a role of team members, so everybody minds his own business. And, of course, we never got rid of our accumulated best practices and solutions, as far as they maintained their development capacity. Here is Dialectics and Evolution.

**Experience of investigation and exploitation of corporate platforms and their future development**

The author of the current article was lucky to participate in numerous blockchain projects held in Raiffeisen bank, in Fintech Association and somewhere else – as a developer or as an expert in decentralized platforms. Some of these projects were purely exploratory, the others ended up as pilot projects, and the rest have evolved into large industrial networks with several dozens of nodes. What are the main conclusions of this experience?

1. There is a certain diversity of blockchain platforms, significantly varying in their “consumer” properties:
   * “entrance threshold” and a simplicity of network deployment
   * capacity
   * functionality of smart contracts
   * time frames and cost of development

That is why we cannot say that any of the platforms can be absolutely dominating. Each platform has its potential pool of users, as well as a set of tasks where it can be used rationally and profitably. This relates both to Ethereum, and Quorum, and Hyperledger Fabric, and Corda. As they said, it is a poor mouse that has only one hole. The same thing is for programming languages – only those, who knows only one language, will be arguing **what** is better – “C++” or “Java”. While their more mature fellows, who have cut their teeth in programming, will argue **when** it is better to use “C++” or “Java”.

1. Although some DLT platforms (e.g., Hyperledger Fabric  and Corda) allow to transmit large elements of data (files, per se), it is the most likely that a blockchain basis with smart contract mechanisms will be separated from a file transmission functionality. This relates to several points:
   * Special decentralized file systems have been existing for a longer time, they are better established and they perform this task more effectively than any analogue DLT platforms. We performed some estimation experiments: according to results, files bigger than 2M can get “stuck in a channel” in case of Hyperledger Fabric and Corda. On the other hand, there are no problems with file transmission in case of IPFS, even if the files are as large as 100M. If your business case suggests transmission of any informal documents like pdf (contracts, agreements etc.), then 50M is an essential minimum that you should be ready for.
   * Such approach allows to quite effectively prepare a blockchain channel and a file transmission channel. This is highly relevant for mixed traffic systems (transactions + documents), especially since the operational priority of transactions is usually higher.
   * Cloud file storages (e.g., of S3 standard) are quite a good alternative to decentralized file systems. Although they do not meet the criteria of a pure “decentralization”, they fit the framework of distributed networks in terms of fail-safety. They even outperform DFS in terms of exchange speed and reliability. It is also possible to implement some hybrid solutions.
   * Looking ahead, to a possible integration of multiple corporate networks, especially based on different blockchain bases, we can see that assigning a special “file” channel will considerably simplify the whole process.
2. To date, there is a chronic lack of specialists in the field of decentralized platforms support service. To be more precise, we don not have them at all. To the best of our knowledge, in most projects a great part of support service is performed by developers and research engineers, who have created these projects, which does not look good. Suppose, it is explained by the “youth” of this area. There is a gradual process of generation of technical instructions, response templates, monitoring schemes, and other guidance materials, essential to manage a well-coordinated work of a support service. One problem is a lack of proper overviews in Russian language on particular blockchain platforms – we need to use just a word-of-mouth. But a support specialist in the enterprise is not a developer, he is focused on the other issues: monitoring, error diagnostics, failure-safety providing, system recovery after crash (you do not have a hope of avoiding crash!). Frankly speaking, a corporate project is more likely to die of a bad service support, rather than of a low-quality development. That is why it is highly important to seek for well qualified specialists experienced in a support service and enterprise system exploitation. This is one of the most significant factors in a long-lasting success of projects, which would continue to develop even after Founding Fathers have left.
3. Formalizing relations between Operator and network members is one of the vaguest juridical questions. This situation is aggravated by the fact that an Operator, on the one hand, is not an owner of a network and its resources. On the other hand, he has to support network functioning, even if his actions contradict with particular members’ interests. A balance between Operator’s rights and duties, his “tools of influence” on the network members, and his financial responsibility – all these are subjects of difficult discussion. Even the simplest question – how to provide a synchronous change of a critical software by all the network members – is a point of hot debate. There are emerging some examples of juridical formalization of the status of Operator and network members, based on the experience of already released platforms. This will speed up a decentralized networks implementation as a key element of corporate and inter-corporate relations.