June 2022

A LANGUAGE FOR SPECIFYING RATE LIMITS

JOEL TRUNICK
VISA

A LANGUAGE FOR SPECIFYING RATE LIMITS

VISA

INVENTOR:

JOEL TRUNICK

TECHNICAL FIELD

[0001] The present subject matter is, in general, related to digital fraud management, and particularly, to a method for encoding the rate-limiting rules into an understandable format for all users.

BACKGROUND

[0002] Rate limiting is an incredibly important defense for detecting automated attacks and fraudulent behavior. For example, limiting the number of log-in attempts, forgotten/forgetting passwords, and so on to something reasonable for a user. However, a fundamental issue with the rate limiting is that rate limiting restrictions or rules are usually specified by a layperson (for example, a product owner or a security official) and the specified rate limits are translated into a programming language, which is difficult for a layperson to understand. Since the implementation of the rate limiting rules are encoded in a difficult-to-understand format or a programming language, it will be challenging for the layperson to modify the rate limiting rules whenever it is required to modified.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, explain the disclosed principles. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same numbers are used throughout the figures to reference like features and components. Some embodiments of device or system and/or methods in accordance with embodiments of the present subject matter are now described, by way of example only, and with reference to the accompanying figures, in which:

[0004] **Fig. 1a** illustrates an exemplary architecture of an existing digital fraud management system.

[0005] **Fig. 1b** illustrates an exemplary architecture of the proposed digital fraud management system along with the logic rules for implementing embodiments consistent with the present disclosure.

[0006] **Fig. 2** illustrates a functional block diagram of a digital fraud management system for implementing embodiments consistent with the present disclosure.

[0007] **Figs. 3-5** show exemplary instances of Velocity Query Language (AQI) API for implementing embodiments consistent with the present disclosure.

[0008] **Fig. 6** illustrates a block diagram of an exemplary computer system for implementing embodiments consistent with the present disclosure.

[0009] The figures depict embodiments of the disclosure for purposes of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the disclosure described herein.

DESCRIPTION OF THE DISCLOSURE

[0010] In the present document, the word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any embodiment or implementation of the present subject matter described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments.

[0011] While the disclosure is susceptible to various modifications and alternative forms, specific embodiment thereof has been shown by way of example in the drawings and will be described in detail below. It should be understood, however that it is not intended to limit the disclosure to the particular forms disclosed, but on the contrary, the disclosure is to cover all modifications, equivalents, and alternative falling within the spirit and the scope of the disclosure.

[0012] The terms “comprises”, “comprising”, or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, device or method that comprises a list of components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or device or method. In other words, one or more elements in a device or system or apparatus proceeded by “comprises… a” does not, without more constraints, preclude the existence of other elements or additional elements in the device or system or apparatus.

[0013] The terms "an embodiment", "embodiment", "embodiments", "the embodiment", "the embodiments", "one or more embodiments", "some embodiments", and "one embodiment" mean "one or more (but not all) embodiments of the invention(s)" unless expressly specified otherwise.

[0014] The terms "including", "comprising", “having” and variations thereof mean "including but not limited to", unless expressly specified otherwise.

[0015] The present disclosure proposes a method and a system for developing a language which is specific for ‘rate limiting’ in digital transactions. Rate limiting is the process of controlling automated attacks and fraud by limiting the number of requests received by a server. Rate limitation improves the availability of the server-based applications by preventing resource starvation that is not even caused by a malicious attack. The proposed method is used to encode the rate limiting rules, for example, in English language format, which is understandable by all users (for example, product owners, software developers, or any other parties). In an embodiment, the proposed method may be used for encoding the rate limiting rules in any other languages, as per the needs and requirements of the users.

[0016] Fig. 1a illustrates an exemplary environment 100 of a Digital Fraud Management (DFM) system 203 which may be implemented for performing embodiments consistent with the present disclosure. In an embodiment, the exemplary environment 100 may include one or more components for developing a language that is specific for ‘rate limiting’ rules and restrictions. These components include, without limiting to, a client component 101 and a DFM component 103. The client component 101 may comprise client code (for example, client version 1, client version 2, as shown in Fig. 1a) and logic rules associated with the client versions. The DFM component 103 may comprise rules version (for example, rule version 1 corresponding to client version 1 and rule version 2 corresponding to client version 2) and related logic.

[0017] In an embodiment, consider a scenario when both the DFM system 203 and the client code are required to be deployed in lockstep, as the logic is present in one or more places, wherein one or more places include the DFM codebase and an upstream component. Further, if the rule versions were different from one another and were interconnected, then there was a chance of incompatibility, which causes failures in encoding the rate limiting rules. For example, if the ‘client version 2’ requests rules from ‘rule version 1’ from the DFM component

103, then the DFM system 203 may fail to encode the rules, wherein the ‘rule version 1’ is associated with different versions.

[0018] In an embodiment, the DFM system 203 may not encounter any problems associated with incompatibility due to the logic, if an English version of the rules are being sent as data to DFM (for example, DFM version 1 and DFM version 2 are active after the logic of English rules are sent to DFM as shown in Fig, 1b) and thereafter storing the data only in the upstream component. Here, the components may be released on their own timeframes and rolled back independently. For example, client version 2 may be able to request rules from DFM version 1 and similarly client version 1 may request rules from DFM version 2, as shown in Fig. 1b. Additionally, the client code may not require deployment in lockstep. Also, the client may send the data file (for example, expressed in the syntax data structure) to a system, which uses logic to translate it into an executable file and enables the components to be released or rolled back independently. As a result, the method proposed in the instant disclosure eliminates the requirement of using a specific version of the server to encode the rules and generates automated reports which are in an understandable format.

Processing the Pseudocode:

[0019] In an embodiment, a functional block diagram of a pseudocode data processing environment for handling client requests is illustrated in Fig. 2. According to the present invention, the method comprises receiving a request from client 201 by the DFM system 203, wherein the request may be a velocity request with English velocity rules. In general, velocity rules are related to analysis, which demands a more complicated analysis of a database. In other words, the rules force the fraud detection software to assess whether there has been excessive activity, which might lead to the conclusion that the transaction is fraudulent. Thereafter, DFM system 203 compares the velocity request to any similar and/or prior comparable velocity requests in look-up database 205 based on the language/text of English rules. If the English rules are not encountered before, then the English format is parsed into internal data structures and the obtained English format is stored for future requests.

[0020] In another embodiment, DFM system 203 analyzes the velocity request with the prior comparable velocity requests in look-up database 205 and if it is found that the English rules are already encountered, then the "compiled" internal data structure as stored previously is retrieved. Thereafter, the data from the current client request is used and applicable logic is applied using the internal data structure. Further, the system may update the velocity values,

say by updating an increment counter based on DFM rules. The increment counter is based on incrementing velocity, for example, incrementing the velocity based on the number of attempts is captured for pass attempts, as shown in Fig. 3. After the counter value is incremented, the value is added to the current amount and the counter is reset. Subsequently, the method comprises comparing current velocity values to determine a response and then returning to the appropriate response as defined by the English velocity rules. One such example is, if the current velocity value has been attempted more than 5 times in the previous 60m.

DFM rules:

[0021] In an embodiment, a number of DFM rules may require understanding by various groups including both non-technical groups as well as technical groups. The non-technical groups comprise, without limiting to, product owners (verticals), Geographic Information System (GIS) and/or cybersecurity, and risk operations. Further, the non-technical groups may specify the rules or read the rules (implementation). The technical groups comprise developers, Quality Assurance (QA) teams, and Penetration (Pen) Test teams. The non-technical groups may implement and/or test the rules and read the rules. DFM rules are difficult to understand, which are defined rules by utilizing wiki, and spreadsheets, and having to maintain in sync with code, and are error prone. Rule review sessions to interpret regulations are time demanding. For instance, DFM uses a Drools rule engine to evaluate the velocity and ensure that all stakeholders understand easily. The rule engine allows the business people/stakeholders to specify the rules themselves, as a result, they may create the rules without involving programmers. For example, while evaluating velocity limits based on ‘forgot password’ attempts for Pass Attempts (PAS) and one-line English substitutions are possible using the drools rule engine, as shown in Fig. 4.

[0022] JAVA Domain Specific Language (DSL) velocity variable rules (as given below) are data which are reusable rules. The reusable rules are written to interpret the obtained data and consistent limitations depending on before velocity request and after velocity request.

```
VelocityConstraint
        .named (_tracking (Value.deviceId))
        .countFlow("GenerateToken")
        .limit( _count( 5, _toHours( 3 ), _afterLimitHit( LOCKOUT, _lockDevice(
            _toHours( 2 )))))
```

```
.limit( _count( 10, _toHours( 12 ), _afterLimitHit( LOCKOUT, _lockDevice
    ( _toHours( 15 )
….
```

For example, the English Velocity Query Language (VQL) written according to velocity rules using the drools rule engine is illustrated below:

"Track user activity.
Counts flow 'LoginFailed'.
Count over 5 in 45m, action: LOCKOUT, lock account 90m.
Count over 10 in 9h, action: LOCKOUT, lock account 1d."

[0023] In an embodiment, user activity may be tracked to identify the uniqueness of a particular transaction, for example, “Track cardNumber per consumerId...” and when the transaction amount fails, i.e., “Amount over $1,000 USD in 60m, action: DENY…”. The count is reset once the OTP is passed, i.e., “Count over 5 in 30m reset by flow ‘OtpPassed’, …” and count all the flows when the decisions are allowed for specific results count, i.e., syntax: “Counts all flows with decisions of ALLOW”. Further, the reset count may vary based on time zone and using the same limits. For example, “Count over 5 since midnight Asia/Kolkata…” for time zone variation and “Track User & DeviceId…” for using the same limits.

[0024] The rule files function as a document which may be shared and written without the need for extra translation to understand the encoded file. The following are examples of VQL for tracking, counting condition, limiting conditions, along with the limits:

[0025] Velocity Query Language (VQL) example for tracking:

“Track User activity during Enrollment”

[0026] Velocity Query Language (VQL) example for Counting condition:

“Counts flow 'AuthFailed'”

[0027] Velocity Query Language (VQL) example for limiting condition:

“Limits all flows”

[0028] Velocity Query Language (VQL) example for limits:

“Count over 5 in 90m reset by flow 'AuthPassed', action: DENY”

"Count over 9 in 24h, action: DENY, lock device 24h"

[0029] Velocity Query Language (VQL) format:

VelocityConstraint := <Tracking> (Counting condition) (Limiting Condition) <Limit>+

Tracking := <Tracking Values> (Tracking During) "."

Limit := <Threshold> "," <Actions> "."

[0030] In an embodiment, tracking values includes specifies variables, say, single variable, combined variable, unique variable, and anded variable to be tracked. For example, in the case of a single variable, such as tracking user activity, the tracking user activity as well as user CardNumber activity may be required for a combined variable situation, as well as how many times that combination has occurred. In the case of unique variables, tracking values include CardNumber per user activity, which is the number of unique cards used by that specific user. Furthermore, the anded variable tracks both user activity and CardNumber activity. Limits and/or thresholds, for example, apply to both user activity values and CardNumber activity to avoid implementing multiple limitations.

[0031] In an embodiment, 'Tracking during' is performed based on all default events as well as a particular event specified during the specified event.

[0032] In an embodiment, 'Counting Condition' is based on counting, which may occur for all flows (say counts all default flows), a specific flow (for example, Counts flow 'Name'), and/or for a specific result (for example, Counts flow 'Name' for decisions of CHALLENGE, DENY condition). For 'OTP generate' counting condition, counts requests for flow 'Name', wherein "requests" is the default, a "gatekeeper" flow, before action is taken. Similarly, for Authentication Failed (AuthFailed) counting condition, counts results for flow 'Name', wherein the "results" is an ex post facto flow, which occurs after an action is performed.

[0033] In an embodiment, "Limiting Condition" where apply limits to all default flows (say, limits all flows) and apply limits to a single flow (say, Limits flow 'Name'). In an embodiment, "Limit Threshold" is based on a scalar limit (for example, count over 10 in 24h), by day limit (for example, count over 10 in 'Asia/Kolkata' based on the start of a day in India), by variable limiting (for example, Count over a threshold in 60m), by reset (Count over 10 in 24h reset by flow 'AuthPassed'), by count over 10% of total percentage count, and transaction amount exceeds above 1000 USD. Further, actions are defined as "Actions := <Result> (Action)+",

wherein Result/Decision: action: DENY which is required to be in a specific format, and actions are based on locking a device for 24h or locking an account for 24h or logging out from a session. Fig. 5 illustrates an example that specifies ALLOW or DENY action using a REST endpoint during tracking for event management, where overrides may allow an admin to set a particular value in the DFM system 201 to trigger specific behavior

Advantages of the present invention:

[0034] In an embodiment, the proposed invention helps in encoding the rules into business-readable formats as well as writeable velocity rules which are easily understood by all the parties/users.

[0035] In an embodiment, the present invention may validate the implemented product and reduces the need for documentation or rule review meetings.

[0036] In an embodiment, the present invention reduces the size of rule files by approximately 35% less compared to the original file size.

[0037] In an embodiment, the present invention may pass client code in the velocity rules from one service to another in electronic format and may require any specific version of the server.

General computer system:

[0038] Fig. 6 illustrates a block diagram of an exemplary computer system for implementing embodiments consistent with the present disclosure.

[0039] In an embodiment, the computer system 600 may be used to implement the system. The computer system 600 may include a central processing unit (“CPU” or “processor”) 602. The processor 602 may include at least one data processor developing a language specific rate limits bases on client 201 requests. The processor 602 may include specialized processing units such as, integrated system (bus) controllers, memory management control units, floating point units, graphics processing units, digital signal processing units, etc.

[0040] The processor 602 may be disposed in communication with one or more Input/Output (I/O) devices (612 and 613) via I/O interface 601. The I/O interface 601 employ communication protocols/methods such as, without limitation, audio, analog, digital, monoaural, radio corporation of America (RCA) connector, stereo, IEEE-1394 high speed serial bus, serial bus, universal serial bus (USB), infrared, personal system/2 (PS/2) port,

bayonet neill-concelman (BNC) connector, coaxial, component, composite, digital visual interface (DVI), high-definition multimedia interface (HDMI), radio frequency (RF) antennas, S-Video, video graphics array (VGA), IEEE 802.11b/g/n/x, Bluetooth, cellular e.g., code-division multiple access (CDMA), high-speed packet access (HSPA+), global system for mobile communications (GSM), long-term evolution (LTE), worldwide interoperability for microwave access (WiMax), or the like, etc.

[0041] Using the I/O interface 601, the computer system 600 may communicate with one or more I/O devices such as input devices 612 and output devices 613. For example, the input devices 612 may be an antenna, keyboard, mouse, joystick, (infrared) remote control, camera, card reader, fax machine, dongle, biometric reader, microphone, touch screen, touchpad, trackball, stylus, scanner, storage device, transceiver, video device/source, etc. The output devices 613 may be a printer, fax machine, video display (e.g., cathode ray tube (CRT), liquid crystal display (LCD), light-emitting diode (LED), plasma, plasma display panel (PDP), organic light-emitting diode display (OLED) or the like), audio speaker, etc.

[0042] In some embodiments, the processor 602 may be disposed in communication with a communication network 609 via a network interface 603. The network interface 603 may communicate with the communication network 609. The network interface 603 may employ connection protocols including, without limitation, direct connect, ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc. The communication network 609 may include, without limitation, a direct interconnection, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), the Internet, etc. Using the network interface 603 and the communication network 609, the computer system 600 may communicate with a database 614, which may be the enrolled templates database 613. The network interface 603 may employ connection protocols include, but not limited to, direct connect, ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc.

[0043] The communication network 609 includes, but is not limited to, a direct interconnection, a peer to peer (P2P) network, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), the Internet, Wi-Fi and such. The communication network 609 may either be a dedicated network or a shared network, which represents an association of the different types of networks that use a variety of protocols, for

example, hypertext transfer protocol (HTTP), transmission control protocol/internet protocol (TCP/IP), wireless application protocol (WAP), etc., to communicate with each other. Further, the communication network 609 may include a variety of network devices, including routers, bridges, servers, computing devices, storage devices, etc.

[0044] In some embodiments, the processor 602 may be disposed in communication with a memory 605 (e.g., RAM, ROM, etc. not shown in Fig. 6) via a storage interface 604. The storage interface 604 may connect to memory 605 including, without limitation, memory drives, removable disc drives, etc., employing connection protocols such as, serial advanced technology attachment (SATA), integrated drive electronics (IDE), IEEE-1394, universal serial bus (USB), fiber channel, small computer systems interface (SCSI), etc. The memory drives may further include a drum, magnetic disc drive, magneto-optical drive, optical drive, redundant array of independent discs (RAID), solid-state memory devices, solid-state drives, etc.

[0045] The memory 605 may store a collection of program or database components, including, without limitation, user interface 606, an operating system 607, etc. In some embodiments, computer system 600 may store user/application data, such as, the data, variables, records, etc., as described in this disclosure. Such databases may be implemented as fault-tolerant, relational, scalable, secure databases such as Oracle or Sybase.

[0046] The operating system 607 may facilitate resource management and operation of the computer system 600. Examples of operating systems include, without limitation, Apple™ Macintosh ™ OS X™, UNIX™, Unix-like system distributions (e.g., Berkeley Software Distribution (BSD), FreeBSD™, Net BSD™, Open BSD™, etc.), Linux distributions (e.g., Red Hat™, Ubuntu™, K-Ubuntu™, etc.), International Business Machines (IBM™) OS/2™, Microsoft Windows™ (XP™, Vista/7/8, etc.), Apple iOS™, Google Android™, Blackberry™ operating system (OS), or the like.

[0047] In some embodiments, the computer system 600 may implement web browser 608 stored program components. Web browser 608 may be a hypertext viewing application, such as Microsoft™ Internet Explorer™, Google Chrome™, Mozilla Firefox™, Apple™ Safari™, etc. Secure web browsing may be provided using secure hypertext transport protocol (HTTPS), secure sockets layer (SSL), transport layer security (TLS), etc. Web browsers 608 may utilize facilities such as AJAX, DHTML, Adobe™ Flash, Javascript, Application Programming

Interfaces (APIs), etc. In some embodiments, the computer system 600 may implement a mail server stored program component. The mail server may be an Internet mail server such as Microsoft Exchange, or the like. The mail server may utilize facilities such as ASP, ActiveX, ANSI C++/C#, Microsoft .NET, Common Gateway Interface (CGI) scripts, Java, JavaScript, PERL, PHP, Python, WebObjects, etc. The mail server may utilize communication protocols such as Internet Message Access Protocol (IMAP), Messaging Application Programming Interface (MAPI), Microsoft Exchange, Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP), or the like.

[0048] In some embodiments, the computer system 600 may implement a mail client stored program component. The mail client may be a mail viewing application, such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Mozilla Thunderbird, etc.

[0049] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer-readable storage medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer-readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term “computer-readable medium” should be understood to include tangible items and exclude carrier waves and transient signals, i.e., be non-transitory. Examples include Random Access Memory (RAM), Read-Only Memory (ROM), volatile memory, non-volatile memory, hard drives, Compact Disc (CD) ROMs, DVDs, flash drives, disks, and any other known physical storage media.

[0050] The described operations may be implemented as a method, system or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The described operations may be implemented as code maintained in a “non-transitory computer readable medium”, where a processor may read and execute the code from the computer readable medium. The processor is at least one of a microprocessors and a processor capable of processing and executing the queries. A non-transitory computer readable medium may include media such as magnetic storage medium (e.g., hard disk drives, floppy disks, tape, etc.), optical storage (CD-ROMs, DVDs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs,

PROMs, RAMs, DRAMs, SRAMs, Flash Memory, firmware, programmable logic, etc.), etc. Further, non-transitory computer-readable media may include all computer-readable media except for a transitory. The code implementing the described operations may further be implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.).

[0051] The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. These examples are presented herein for purposes of illustration, and not limitation. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope and spirit of the disclosed embodiments. Also, the words "comprising," "having," "containing," and "including," and other similar forms are intended to be equivalent in meaning and be open ended in that an item or items following any one of these words is not meant to be an exhaustive listing of such item or items or meant to be limited to only the listed item or items. It must also be noted that as used herein, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise.

[0052] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer readable storage medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term “computer readable medium” should be understood to include tangible items and exclude carrier waves and transient signals, i.e., are non-transitory. Examples include random access memory (RAM), read-only memory (ROM), volatile memory, non-volatile memory, hard drives, CD ROMs, DVDs, flash drives, disks, and any other known physical storage media.

[0053] Finally, the language used in the specification has been principally selected for readability and instructional purposes, and it may not have been selected to delineate or

circumscribe the inventive subject matter. Accordingly, the disclosure of the embodiments of the disclosure is intended to be illustrative, but not limiting, of the scope of the disclosure.

[0054] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

A LANGUAGE FOR SPECIFYING RATE LIMITS

ABSTRACT

The present disclosure relates to a method and system for developing a language, which is specific for ‘rate limiting’ in digital transactions. The present disclosure uses a specific language to encode ‘rate limiting’ rules and conditions, such that the encoded language is in a readily understandable format and covers many different use cases for ‘rate limiting’. As a result, the present disclosure eliminates requirement of specific version of server and makes it possible to pass rules from one service to another in electronic format. Further, the users may be able to generate automated reports which is in an understandable format.

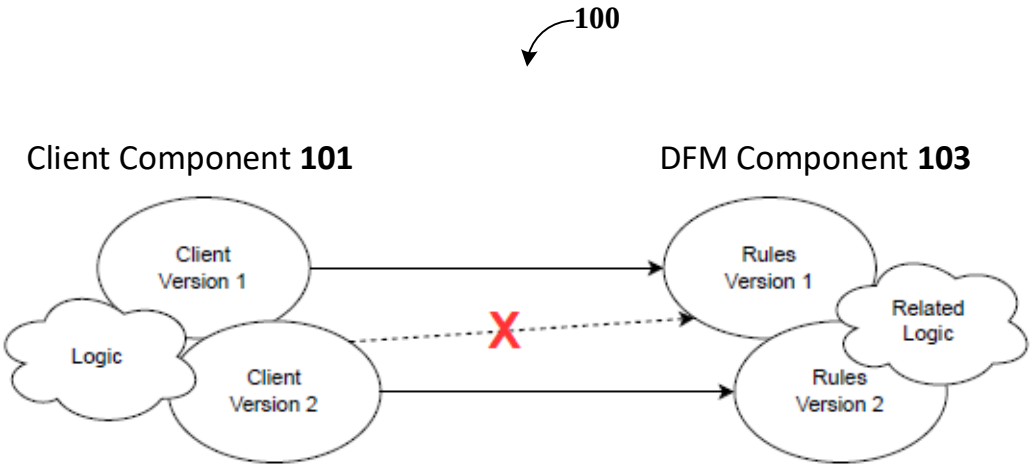
100
Client Component 101
DFM Component 103
Client Version 1
Logic
Client Version 2
Rules Version 1
Related Logic
Rules Version 2
X

Fig. 1a

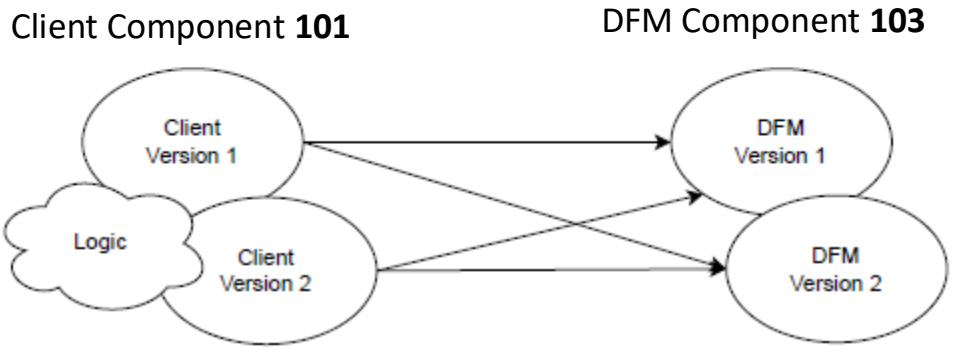
Client Component 101
DFM Component 103
Client Version 1
Logic
Client Version 2
DFM Version 1
DFM Version 2

Fig. 1b

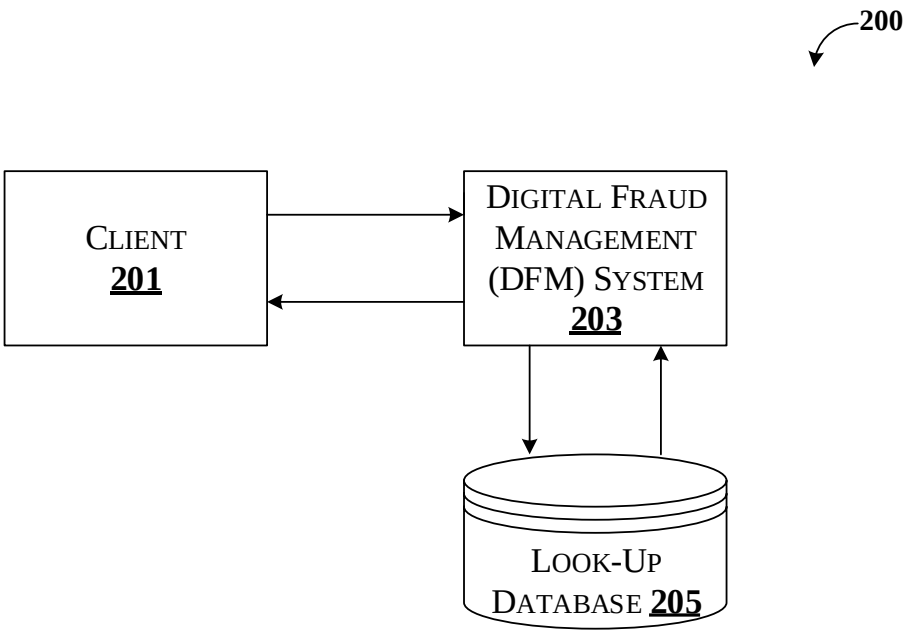

Fig. 2

```
rule "Capture number of attempts"
ruleflow-group "Velocity Capture"
    when
        DecisionRequest (event.eventType == EventType.FORGOT_PASSWORD, event.eventSubType == EventSubType.ATTEMPT)
        velocityServiceFact : VelocityServiceFact ()
        deviceData: DeviceData()
        eval(StringUtils.isNotBlank(deviceData.getDeviceId()))
    then
        VelocityVariable velocityVariable = new VelocityVariable("forgot_password_attempts",  deviceData.getDeviceId());
        velocityServiceFact.captureVelocity(velocityVariable, 0d);
end
```

Fig. 3

```
rule "ForgotPassword Attempts Velocity Limit check"
ruleflow-group "Lockout"
salience 400
    when
        decisionRequest : DecisionRequest (event.eventType == EventType.FORGOT_PASSWORD, event.eventSubType == null)
        decision : DecisionResponse ()
        riskActionFact : RiskActionFact ()
        velocityServiceFact : VelocityServiceFact ()
        deviceData: DeviceData(deviceId != null)
        eval(velocityServiceFact.getVelocityCount(42, TimeUnit.MINUTES, "forgot_password",  deviceData.getDeviceId()) >= 10)
        eval(!riskActionFact.isDeviceLocked(deviceData.getDeviceId()))
    then

        decision.setRiskDecision(RiskDecisionResultTypeEnum.LOCKOUT);
        decision.getDecisionDetail().setReason("DEVICE_ALREADY_LOCKED");
        decision.getDecisionDetail().setLockoutDuration(new Long(60));
        decision.setRiskAction(RiskActionResultTypeEnum.LOCK_DEVICE);
        decision.setRiskDecisionReason(decision.getRiskDecisionReason().concat("Rule : " + drools.getRule().getName() + "."));
        riskActionFact.lockDevice(deviceData.getDeviceId(),60);
end
```

Fig. 4

```
{
    "ProductCode": "ACME",
    "EventType": "Card",
    "Flow": "CardAdd",
    "Values": {
        "CardNumber": "3424234412312",
        "User": "342-13243-234-2341",
      },
    "Rules": [
        "Track CardNumber per User activity. Count over 3 in 24h, action: DENY.",
        "Track User activity. Limits flow 'CardPurchase'. Amount over 150 USD in 24h, action: ALLOW.",
    ]
}
```

Fig. 5

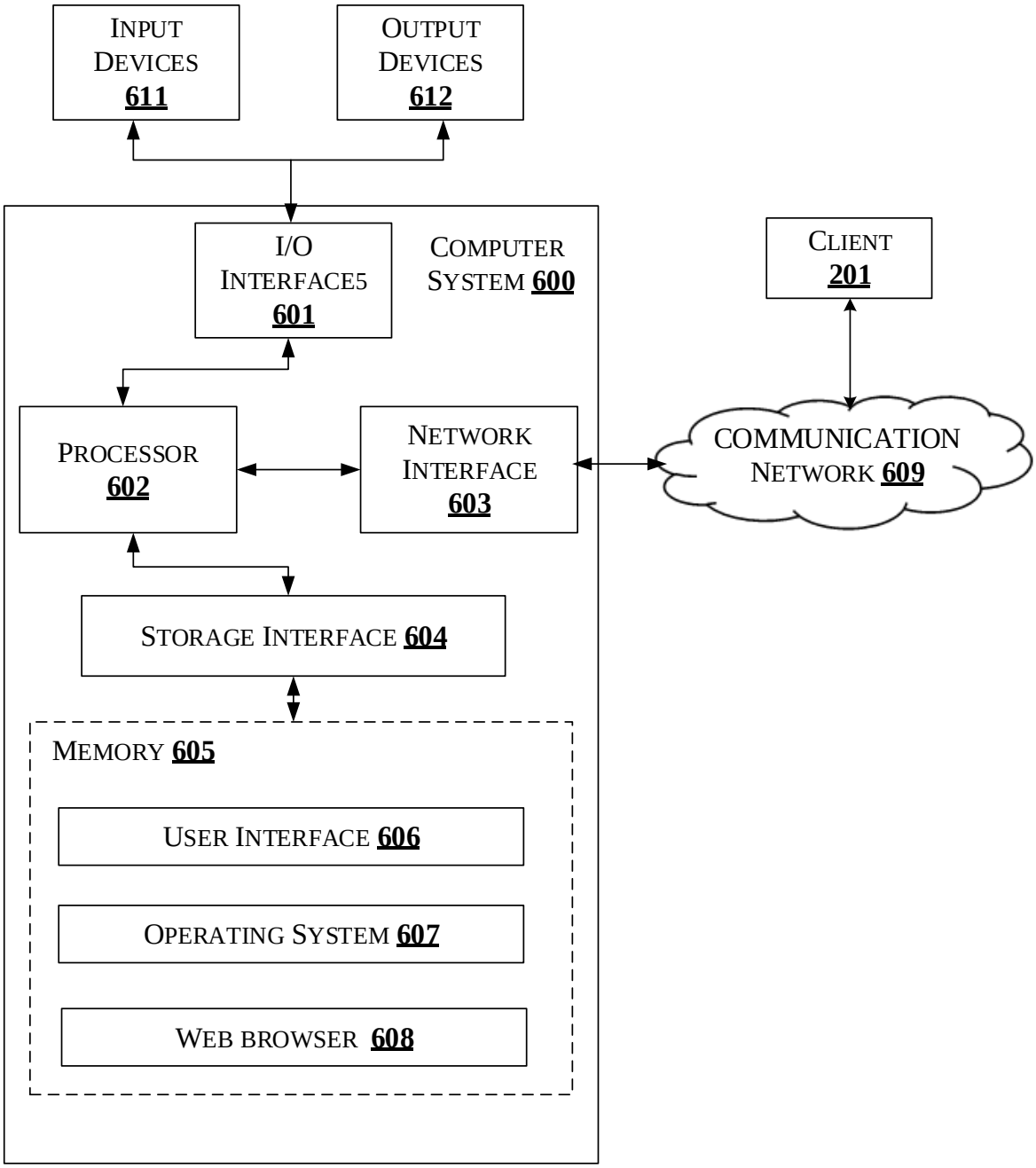
INPUT DEVICES 611
OUTPUT DEVICES 612
I/O INTERFACE5 601
COMPUTER SYSTEM 600
CLIENT 201
PROCESSOR 602
NETWORK INTERFACE 603
COMMUNICATION NETWORK 609
STORAGE INTERFACE 604
MEMORY 605
USER INTERFACE 606
OPERATING SYSTEM 607
WEB BROWSER 608

Fig. 6