©NIJSES Publications

Open Access

NEWPORT INTERNATIONAL JOURNAL OF SCIENTIFIC AND EXPERIMENTAL SCIENCES (NIJSES)

VOLUME 3 ISSUE 2 2023

Workflow Models in Automated Business Environments

[1]Ezeamasobi Chibuzor, [2]H.C Inyiama, [3]Godspower I. Akawuku,

[1,2,3]Nnamdi Azikiwe Unversity, Awka (NAU).

Chibuzoredith@gmail.com, hc.inyiama@unizik.edu.ng, gi.akawuku@unizik.edu.ng

ABSTRACT

In recent times, Organizations are moving toward broader workflow automation across business operations and IT processes, particularly in this era of cloud services. This helps to speed up processes and improve communication. Workflow automation optimizes processes by replacing manual tasks with software that executes all or part of a process. Today, this is usually done through workflow automation software that consists of low-code, drag-and-drop features and adoption-friendly User interfaces. Nevertheless, the fundamental workflow models and principles are not undermined to achieve organizational objectives, specifically for business startups hence the focus of this paper.

Keywords: workflow, automation, business process, control flow

INTRODUCTION

The Workflow Management Coalition defined a workflow as: "*the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal.*" [1]. According to Alfresco Enterprise Documentation (2012), workflow can be defined as "*a "sequence of connected tasks applied to a document or other item of content. Each task can be performed by a person, a group, or automatically e.g., a document that you needed reviewing and approving by a number of people*" [2]. The main purpose of workflows is to automate business processes – particularly those processes involving combination human and machine-based activities [3]. Some key workflow concepts, defined in the WMC Glossary of Terms, are listed below:

1. **Business Process**: A sequence of activities which collectively pursue a common business objective or policy goal within an organization or between organizations.
2. **Activity**: A discrete process step which may consist of one or more tasks. An activity may be performed manually (requiring human intervention) or automatically (should computer automation be supported).
3. **Instance**: A single execution of a process or an activity within a process. An instance also includes any data associated with the process or activity.

4. **Workflow Participant**: A resource involved in the workflow instance which carries out the work to execute the process. This work is typically categorized as one or more work items which are allocated to the workflow participant via the work list.
5. **Work Item**: A single task to be processed by a workflow participant. A work item is represented in the context of an activity.
6. **Work List** – A set of work items which are assigned to a particular workflow participant. In cases where work lists are shared, a group of workflow participants are responsible for executing the set of work items on the work list [4-5].

Early motivation for workflows stemmed from the "*paperless office*" vision. Whilst the idea of completely eliminating paperwork within office environments never succeeded – in fact, paperwork has since proliferated – isolated office tasks are increasingly becoming automated, hence improving office efficiency. Efficiency is an important factor, particularly in today's global economy (characterized by global competition and rapidly changing technology), where an organization's level of responsiveness is imperative in order to retain a competitive advantage. Inefficient processes impact an organization's ability to react to the demands of business environments [6].

Although workflows can be manually organized, most workflows are normally organized within the context of a computer-based system. In this case, a Workflow Management System (WfMS) is commonly used to provide procedural automation of the business process. A WfMS can be defined as "*a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic.*" [7] defined WfMS as "*a piece of software that provides an infrastructure to setup, execute, and monitor scientific work flows*". An important function of WfMS during the workflow execution, or *enactment*, is the coordination of operation of individual components that constitute the workflow – the process also often referred to as *orchestration.*

A WfMS coordinates and controls the execution of workflows through the use of software. It automates business processes by interpreting process definitions, interacting with workflow participants and invoking external software applications. Its primary responsibilities include: Monitoring and reporting on the performance of processes and the users involved in their execution, enforcing deadlines, ensuring security, and Authenticating users [8].

[9-10], discussed the specific WfMS examples. An advantage of these systems is their capacity to separate workflow logic from the logic of other software applications (which are used to execute individual activities in the workflow). This ability allows application programs to operate independently from the WfMS, resulting in simplified enterprise integration.

In the past, a WfMS would mainly focus on automating isolated office procedures. Nowadays, they are shifting towards managing inter-organizational information flows to achieve automation across the entire business process. This shift introduces coordination problems since multiple locations and multiple interests now exist. But this shift is necessary, especially when you consider the current trend of organizations outsourcing more and more of their work, there is a need for the seamless integration of heterogeneous workflow systems across enterprises [11-12].

WORKFLOW FRAMEWORKS

The Workflow Management Coalition (WfMC) developed the Workflow Reference Model to promote inter-operability between workflow technologies. This model identifies the major components and interfaces associated with workflow systems. Figure 2.1 illustrates the model.

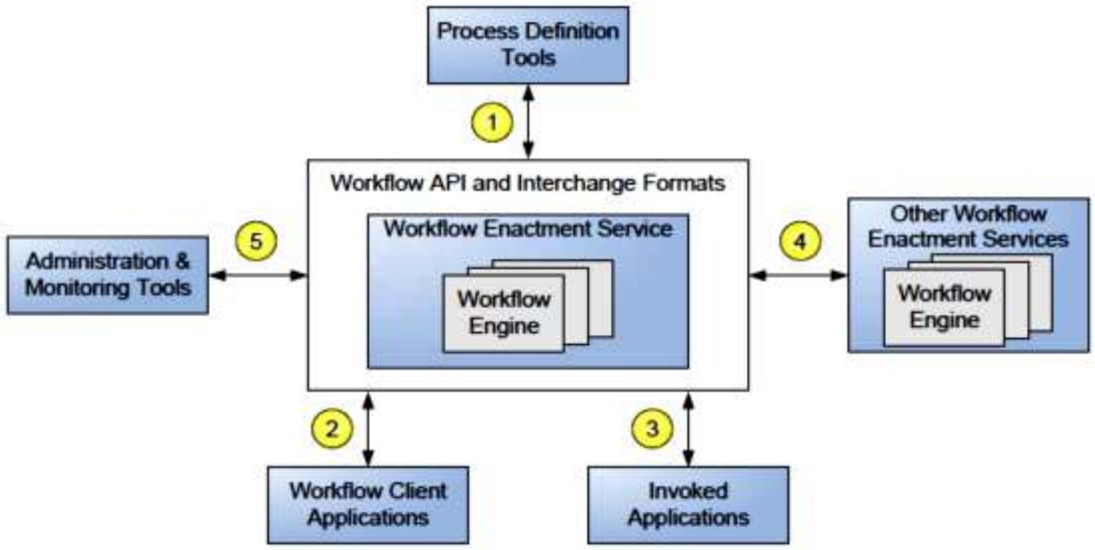
Process Definition Tools
1
Workflow API and Interchange Formats
Workflow Enactment Service
Workflow Engine
5
Administration & Monitoring Tools
4
Other Workflow Enactment Services
Workflow Engine
2
3
Workflow Client Applications
Invoked Applications

Figure 1: Workflow Reference Model: http://www.wfmc.org/standards/docs/tc003v11.pdf

The workflow enactment service, shown at the center of the figure above, provides the run-time environment where process instantiation occurs, using one or more workflow management engines. External resources interact with this service via the workflow programming interface (WAPI). Five individual interfaces are specified in the model:

1. Process Definition Tools: used at build-time to define the workflow process.
2. Workflow Client Applications: allows workflow engines to interact with work-list handlers.
3. Invoked Applications: allows workflow engines to interact with user applications (e.g. mainframe legacy systems).
4. Other Workflow Enactment Services: allows workflow engines provided by different vendors to interoperate.
5. Administration and Monitoring Tools: allows interaction between management/ monitoring applications and the workflow engines [13-15].

According to [16]) "the Workflow Reference Model is useful for understanding the relationships between workflow engines, its users and other software systems". But for considering the analysis and design requirements for a workflow system, the framework illustrated in Figure 2.2 is better suited. [17- 19]. illustrated this framework to emphasize five related views or perspectives.

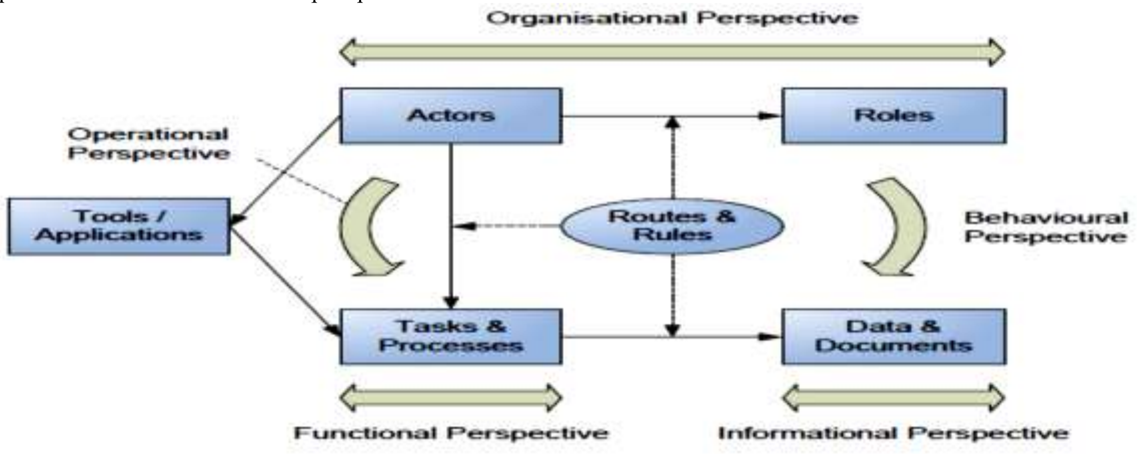

Figure 2: Workflow Perspectives

[20], enumerate the following list to explain each of these perspectives:

1. **The functional perspective** – What does the workflow do? Workflow is specified by decomposing high level functions into smaller tasks that can be allocated to users or software agents.
2. **The behavioral perspective** – When are the activities and tasks executed? Uses a process model that defines the time precedence of individual activities, events and triggers, and the pre- and post-conditions for activities.
3. **The informational perspective** – What data is consumed and produced? Describes the business data, documents and electronic forms that are sent between agents, as well as the files and databases that store persistent application data.
4. **The operational perspective** – How is a workflow activity implemented? Specifies the workflow tools and applications that perform the workflow activities.
5. **The organizational perspective** – Who performs what tasks and with what tools? Defines the organizational hierarchy, the roles, the security and access authorizations, teams and work groups, and individual users and software applications. A role is a collection of tasks and responsibilities that can be assigned to an agent at run-time.

Workflow Patterns

Workflow patterns representing comprehensive workflow functionality are identified by [21]. [22] described a pattern as "*the abstraction from a concrete form which keeps recurring in specific non arbitrary contexts*".
According to [23], the purpose of the workflow patterns is to "*address business requirements in an imperative workflow style expression ... removed from specific workflow languages*". Van der Aalst et al. (2003) argued that although most workflow languages (and workflow management systems for that matter) support the basic workflow constructs, the interpretation of these constructs across different languages is not uniform. Furthermore, it is often unclear how languages even support the more complex workflow constructs. According to [24], Tavaxy is a cloud-based workflow system that implements a pattern-based approach for enabling interoperability between Galaxy and Taverna, two workflow engines popular in the bioinformatics domain.
Workflow patterns are relevant to this thesis as they define workflows on a level of abstraction that is removed from specific implementation tools and workflow languages. This is necessary for the architecture model discussed in Chapter 4 as our research aims to promote inter-operability across different organizations. Hence the architecture model cannot be constrained in respect to workflow definitions, or any other component for that matter. The following subsections describe the workflow patterns, quoted from [24]. [25], provide implementations of these workflow patterns in Orc as illustrated by [26] as a process calculus for orchestrating wide-area computations.

Basic Control Flow Patterns

This is one of the categories of patterns proposed by [27] in their paper Workflow Patterns. These include:
Pattern 1 (Sequence): This is activity in a workflow process is enabled after the completion of another activity in the same process."
Pattern 2 (Parallel Split): This is a point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order. Also known as AND-split, parallel routing and fork
Pattern 3 (Synchronization): This is a point in the workflow process where multiple parallel sub-processes/activities converge into one single thread of control, thus synchronizing multiple threads; also known as AND-join.
Pattern 4 (Exclusive Choice): A point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen; also known as XOR-split, switch and decision.
Pattern 5 (Simple Merge): This is a point in the workflow process where two or more alternative branches come together without synchronization. It assumes that no branches are executed in parallel. Also known as XOR-join, asynchronous join and merge.

Advanced Branching and Synchronization Patterns

According to [28], this category has the following sub-patterns;
Pattern 6 (Multi-Choice): A point in the workflow process where, based on a decision or workflow control data, a number of branches are chosen." A multi-choice is a non-exclusive choice; also known as OR-split.
Pattern 7 (Synchronizing Merge): A point in the workflow process where multiple paths converge into one single thread. If more than one path is taken, synchronization of the active threads needs to take place. If only one path is taken, the alternative branches should re-converge without synchronization: also known as synchronizing join.
Pattern 8 (Multi-Merge): A point in a workflow process where two or more branches re-converge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for activation of every incoming branch.

Pattern 9 (Discriminator): The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity." The completion of all remaining branches is ignored. For example, a search engine may use multiple backend databases, but only the results from the first database to finish executing the search are displayed. The results from the remaining databases are ignored.

Structural Patterns

As described by [29], structural patterns exist in the following forms:

Pattern 10 (Arbitrary cycles): A point in a workflow process where one or more activities can be done repeatedly; also known as loop, iteration and cycle.

Pattern 11 (Implicit termination): A given sub-process should be terminated when there is nothing else to be done. In other words, there are no active activities in the workflow and no other activity can be made active (and at the same time the workflow is not in deadlock).

Patterns Involving Multiple Instances

[30], defined patterns involving multiple instances as the category of patterns with the sub patterns below:

Pattern 12 (Multiple instances without synchronization): Multiple instances of an activity can be created (within a single workflow instance). In order words, new threads of control can be spawned, executing independently of one another.

Pattern 13 (Multiple instances with a priori design time knowledge): The number of instances of a given activity for a given (workflow) instance is known at design time. Once all instances are completed, some other activity needs to be started.

Pattern 14 (Multiple Instances with a Priori Runtime Knowledge): The number of instances of a given activity for a given (workflow instance) varies ... but is known at some stage during runtime, before the instances of that activity must be created. Once all instances are completed, some other activity needs to be started.

Pattern 15 (Multiple Instances without a Priori Runtime Knowledge): The number of instances of a given activity for a given (workflow instance) is not known during design time, nor is it known at any stage during runtime, before the instances of that activity must be created. Once all instances are completed, some other activity needs to be started.

State-Based Patterns

State-based patterns have the listed sub patterns as illustrated by [31]. **Pattern 16 (Deferred Choice)**: A point in the workflow process where one of several branches is chosen (based on information which is not necessarily available when this point is reached). In contrast to the XOR-split, the choice is not made explicitly ... but several alternatives are offered to the environment. Once the environment activates one of the branches, the other alternative branches are withdrawn.

Pattern 17 (Interleaved Parallel Routing): A set of activities is executed in an arbitrary order: Each activity in the set is executed, the order is decided at runtime and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time).

Pattern 18 (Milestone): An activity ... is only enabled if a certain milestone has been reached which did not expire yet.

Cancellation Patterns

[32], enumerates the following sub patterns under this category:

Pattern 19 (Cancel Activity): An enabled activity is disabled, i.e. a thread waiting for the executing of an activity is removed.

Pattern 20 (Cancel Case): A case (i.e. workflow instance) is removed completely. Even if parts of the process are instantiated multiple times, all descendants are removed.

WORKFLOW IMPLEMENTATION MODELS

According to [33], three basic architectures exist for implementing workflows:

Production Architecture: Commonly implemented using workflow folders. Depending on the implementation, a workflow folder can relate to a particular workflow instance, and this folder (containing all documents related to the workflow instance) is circulated in turn to each workflow participant involved in the workflow. Most existing production architectures consist of a single workflow engine, providing services to users in a client-server architecture fashion.

Messaging-based Architecture: Extend messaging systems (such as internal email systems, web services, etc.) with workflow capabilities (e.g. adding electronic forms, logging and work-list generation capabilities).

Document-centric Architecture: Add workflow capabilities to document management systems. For example, a workflow participant notifies another participant of an activity to be performed. The notified participant checks out the relevant documents from the database, performs the activity, and finally returns the documents to the database for the next participant.

Workflow Folders

This implementation model uses special folders that possess workflow information. These folders act as workflow-aware containers, and depending on the implementation, may be reactive to changes in the workflow state. [34] mentioned that X-Folders are an example of where scripts are applied to folders. A change in the document's state (e.g. the document status changing to final) triggers the folder's script into executing some task (e.g. moving the document to another folder).

[35] cited another example as the POLITeam project. This project allocates a folder to each workflow. The folder is then circulated amongst the workflow participants, operating on top of a groupware base system. When the folder is received, the participant extracts the required documents from the folder, processes them in accordance to the workflow activity, returns them to the folder, and forwards the folder to the next participant in the workflow.

The POLITeam project example lacks adequate support for inter-organizational workflows. It is better suited for internal workflows within a single organization (centralized administration authority). Its reliance on an underlying (possibly proprietary) groupware base system hinders inter-operability efforts, unreasonably forcing all partner organizations in the workflow to deploy the same workflow management system. In contrast, the X-Folders example is inspired by the peer-to-peer model. But the problem with this approach is that it provides no end-to-end workflow description, making it difficult to track the document's global state. Instead, it is better suited for light workflow processes between peers, despite supporting standardized inter-operability technologies.

Web Services and Business Process Execution Language

Web Services Business Process Execution Language (WS-BPEL), [36], explained that Web services, coupled with Business Process Execution Language (WS-BPEL), provide a promising implementation model for inter-organizational workflows, given its suitability for heterogeneous and Internet-based environments. [37], defined Web services as "*an interface that describes a collection of operations that are network-accessible through standardized XML messaging*". They are often viewed as an alternative to traditional middleware solutions such as CORBA (Common Object Request Broker Architecture), COM+ and EJB (Enterprise Java BUS).

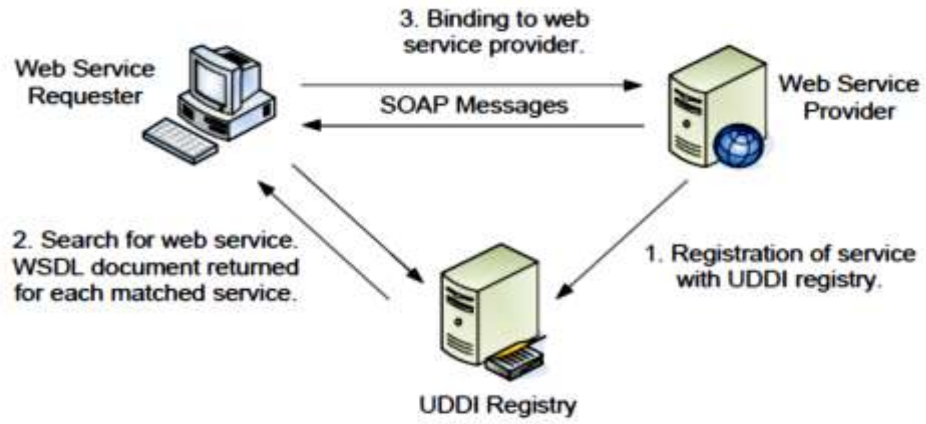
3. Binding to web service provider.
Web Service Requester
SOAP Messages
Web Service Provider
2. Search for web service. WSDL document returned for each matched service.
1. Registration of service with UDDI registry.
UDDI Registry

Figure 3: Web Service Base Model: http://www.w3.org/TR/ws-arch/

They differ from these technologies by providing lightweight business process integration. This is possible as web services operate on top of existing Internet protocols such as HTTP. Web Services Architecture [38] defines a layered model for illustrating the main components of web services. At the top of this stack is the Process layer where WS-BPEL lies. Web services security exists as a vertical component in this stack as it affects all layers.

Web Services Base Model - When a request for a web service is made, the three step "*publish, find, bind*" process is followed. This model is illustrated in Figure.3.

A web service provider advertises their service in a registry so that others can locate it. Registrations and queries to the registry are performed using the Universal Description, Discovery and Integration Protocol [39]. The information published in the UDDI registry includes the publisher's name, a description of the service, binding specifications, etc. A web service requester queries the UDDI registry to find a service. If a matching web service is found, the registry responds with a Web Services Description Language [40], document which details how to bind to this web service. The requester connects to the web service provider by following the specifications in the WSDL document. Finally, the provider responds with the output of the service. Web services messages are communicated via the SOAP protocol as illustrated in [41].

Business Process Execution Language – [42], provides a language for specifying business processes. It extends the web services interaction model by enabling support for business transactions. WS-BPEL defines an interoperable integration model aimed at facilitating automated process integration for both intra-organizational and inter-organizational environments.

Primitive activities supported in WS-BPEL are:

<Invoke> – to invoke an operation of a web service (start an activity).
<Receive> – to wait for an operation to be invoked by another workflow participant.
<Reply> – to generate the response of an operation
<Assign > – to transfer data from one place to another.

Complex activities supported in WS-BPEL include sequences, branching, loops and parallel execution.

SECURITY POLICIES AND ACCESS CONTROL

Focus now shifts to information security, and in particular, security policies and access control. This section explores various policy and access control models, from abstract policy models to the traditional access control models and finally some emerging access control technologies. But to begin, first look at the concepts that are fundamental to information security.

Security Concepts

CONFIDENTIALITY: Confidentiality deals with the protection of sensitive information. This is typically influenced by rules stating the disclosure restrictions of the protected information. Making data incomprehensible is the most common form of confidentiality. This is achieved by using encryption. The two types of encryption techniques which exist include conventional encryption (based on symmetric encryption algorithms) and public-key encryption (based on asymmetric encryption algorithms). Both types are discussed in the following subsections.

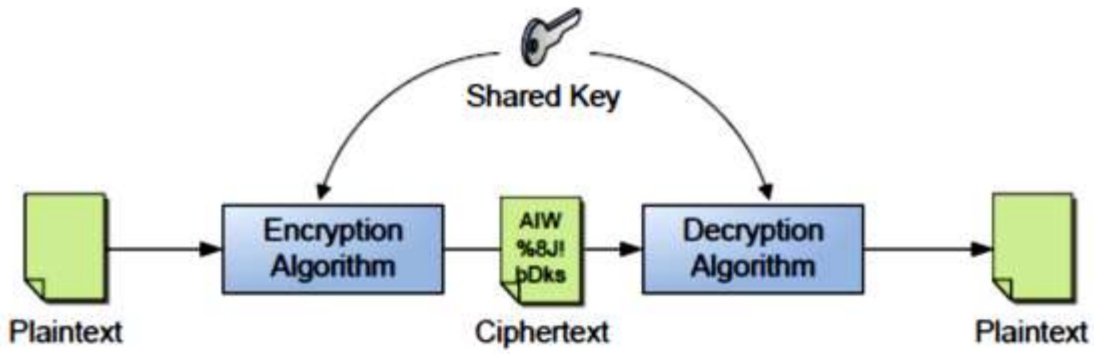

Figure 4: The basic model for conventional encryption: Stallings, W., [43].

Conventional Encryption - Conventional encryption techniques were, for a long time, the only known form of encryption. In fact, these are still the most widely used type today. Symmetric encryption relies on the existence of a shared secret between the message source and the destination. Figure 4 illustrates how conventional encryption works. The plaintext (i.e., the message to be protected) and the shared key (i.e., the shared secret) are passed into the encryption algorithm to produce the cipher text (i.e., the message encrypted). The sender sends the cipher text to the recipient. Once received, the recipient passes the cipher text and the shared key into the decryption algorithm to reveal the plaintext.

When using symmetric encryption, the ability to derive the plaintext from the cipher text alone must be impractical. Additionally, the security of symmetric encryption techniques must rely on the secrecy of the key – not the encryption algorithm. Knowledge of the encryption algorithm should not provide a security risk. Widely accepted symmetric encryption algorithm standards today are Data Encryption Standard (DES), Triple-DES and Advanced Encryption Standard (AES). An overview of these standards can be found as highlighted by [43]. DES operates by splitting data into 64-bit blocks and encrypting them using a 56-bit cipher key. However, concerns exist over this standard. Many believe that the short key size makes it vulnerable to brute-force attacks. Furthermore, the internal structure of DES is not completely open raising concerns that backdoors may exist which enable the deciphering of messages without the key. Triple-DES was proposed to alleviate the fears of brute-force attacks against DES. With Triple-DES, two 56-bit keys are used in a three-step process. Firstly, the plaintext is encrypted with key 1, then the result is decrypted with key 2, before finally, the result of this is encrypted with key 1. This process changes the cost of discovering the encryption keys using a brute-force attack from the order of 2^{56} to 2^{112}.

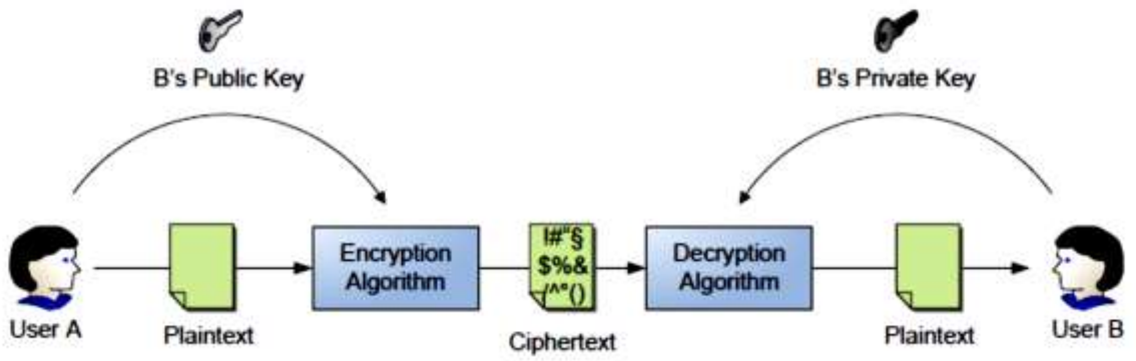

Figure 5: The basic model for public-key encryption: Stallings.

Although Triple-DES addresses the problem of the small key size, it performs relatively slowly when implemented in software and it is not suitable for use in limited-resource platforms. To resolve these problems, Rijndael was proposed in 1997 and this algorithm was later officially standardized as AES in 2001. AES operates fast in both hardware and software, requires little memory to operate, uses 128-bit block sizes, and supports key lengths of 128, 192 and 256 bits. Additionally, the specifications of AES are open and can be found in FIPS PUB 197 [18].

Public-key Encryption - A difficult issue facing conventional encryption methods is key distribution. These methods require either:

i. The communicating parties to already possess the secret key OR

ii. A key distribution centre (KDC) to be in place. But with (2), maintaining total secrecy over the communications is under threat. Key owners are forced to share keys with a KDC which could potentially become compromised.Public-key encryption schemes tackle this issue by using two separate keys rather than a single secret key. Public-key (or asymmetric) encryption algorithms operate by using one key for encryption and the other key for decryption. An important characteristic of asymmetric encryption algorithms is that it must be computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key as illustrated by [44].

Figure 4.2 shows how public-key encryption works. In this example, user B owns a key pair – a public key (to be distributed to everyone) and a private key (to be kept secret). Suppose user A decides to send an encrypted message to B, the message is encrypted by A using B's public key and the resulting cipher text is sent to B. When received, B decrypts the cipher text using his private key to reveal the plaintext. Total secrecy over the communications is achieved as private keys can be generated locally and never need to be distributed to anyone.

[45], explained that the most widely accepted asymmetric encryption algorithm in existence today is the Rivest-Shamir-Adleman (RSA) scheme. RSA is a block cipher and uses mathematical functions (rather than substitution and permutations) to perform encryption/decryption functionalities. A drawback of asymmetric encryption techniques is that they are computationally slower than symmetric encryption techniques. For this reason, it is common practice for a secure communication session to initially use asymmetric encryption for key negotiation before using symmetric encryption once a secret key has been established.

INTEGRITY: Integrity is concerned with protecting messages from unauthorized modifications. To achieve this, digital signatures can be used. Creating a digital signature involves taking a fingerprint of the message (otherwise known as a hash value) and then encrypting this fingerprint with the signer's private key. Digital signatures make use of public-key encryption.

To create the fingerprint, hash functions are used. Hash functions take the form $h=H(M)$ where h represents a fixed-length hash value, H represents the hash function and M represents a variable-length message. [46], highlighted that some important requirements of hash functions include to:

i. be able to be applied to variable-length data.
ii. be able to produce a fixed length output.
iii. be relatively easy to compute for any given message.
iv. be computationally infeasible to derive the message from the hash value (one-way property).
v. be computationally infeasible to find another message which produces the same hash value.
vi. be computationally infeasible to find for any pair *(x, y)*, such that $H(x) = H(y)$.

Hash functions satisfying the first five requirements are known as weak hash functions. For those which also satisfy the sixth requirement (which protects against the birthday attack as explained by Stinson (1995), these are known as strong hash functions.

A simple example of a hash function is one which performs a bit-by-bit exclusive-or (XOR) operation over each block of the message. Alternatively, this simple example could be enhanced by incorporating a one-bit circular shift (rotation) on the hash value after each data block is processed. More elaborate hash function algorithms include the MD5 message-digest algorithm pointed out by [47] and the Secure Hash Algorithm (SHA) illustrated by [48].

To check if message integrity has been maintained, digital signatures need to be verified at the receiver's end. This verification involves a two-step process. Firstly, the receiver decrypts the signature using the signer's public key to reveal the fingerprint of the message. Secondly, the receiver performs its own hash over the message to create a fingerprint and if the two fingerprints match, integrity has been maintained.

AUTHENTICATION: Authentication addresses the question of: "*Are you who you say you are*?" This principle is concerned with verifying an entity's identity claim based on evidence that no-one else can present. According to Summers (1997), authentication can be based on three types of evidence:

i. Something the user knows.
ii. Something the user has.
iii. Something the user is or does.

Passwords are a classic example of authentication based on something that the user knows. Token devices (e.g., memory cards and smart cards) are a common form of authentication based on something that the user has, whereas authentication based on biometrics (e.g., fingerprints and retina scans) fall under the category of something the user is or does. The most basic type of authentication is one-way authentication – otherwise known as simple authentication. This involves a claimant who wishes to authenticate itself with a verifier by sending a proof of identity. A more advanced form of authentication is mutual authentication. This form of authentication enables two-way authentication so that the claimant can be assured that it is authenticating to the correct verifier.

AUTHORIZATION: Authorization refers to the process of granting or denying access to resources. It is closely associated to access control which defines the models and mechanisms for applying restricted access to protected resources. The most common access control models typically consist of three main components:

i. Subjects – the entities (e.g., users) of the system.
ii. Objects – the resources (e.g., files) under protection.

Permissions – the access rights supported by the model. Common access rights include read, write, execute and own. An important aspect of authorization is that it assumes that subjects have been properly and successfully authenticated beforehand. This enables authorization systems to enforce their policies and to make decisions based on the identities of subjects.

NON-REPUDIATION: Non-repudiation is concerned with providing guarantees so that, for example, a sender of a message cannot deny that a message transmission never occurred when, in fact, it did. Similarly, it can be used to prevent a recipient from claiming that it never received a transmitted message when it was successfully received. Dispute resolution is a major reason for non-repudiation.

According to [49], digital signatures are the only means of achieving non-repudiation without involving a notary. Following are the different forms in which non-repudiation can take as slated by [50]:

i. Non-repudiation of creator - prevents the creator from denying ever creating the data.
ii. Non-repudiation of sender - prevents the sender from denying ever sending the data.
iii. Non-repudiation of submission - proof of data transmission for the sender. Prevents the communication system or any recipients from claiming that the data was never transmitted.
iv. Non-repudiation of delivery - prevents the recipient from denying ever receiving the data. It proves to the sender that the data was delivered to the intended recipient.
v. Non-repudiation of receipt - prevents the recipient from claiming that he has never seen the data.
vi. Non-repudiation of use of service - protects a service provider from a user denying that he never used a particular service offered by the provider.

000AUDITING: Auditing refers to the logging of system events and keeping an audit trail. It is concerned with keeping users accountable for their actions. An important characteristic of auditing is that it enhances access control. In principle, if unauthorized actions pass the access control system, then the auditing system detects it. According to [51], auditing enhances access control in the following ways: It acts as a deterrent. Users become more reluctant to perform unauthorized activities if they know that their actions are being tracked. Log files provide evidence for investigating attempted or actual security violations. It helps with discovering security holes in the system; ensures that authorized users do not abuse or misuse their privileges.

TRUST: When examining the issue of trust, it is important to identify who trusts whom on what. The principle of trust suggests a relationship between (at least) two entities based on a particular action. Trust can be categorized into two types – direct trust and third-party trust. The former type describes the situation where two entities have established a trust relationship themselves. The latter refers to an implicit trust relationship between two entities since both entities trust a specific trusted third-party (the third-party determines the trustworthiness of the two entities). Third-party trust is common in public-key cryptography where a Certification Authority (CA) acts as the trusted third party.

SUMMARY AND CONCLUSION

Current research into access control for inter-organizational workflows and cryptographic access control have encompassed many forms, a few of which have been presented in this paper. However, these works have predominately focused on the theoretical side and less on the practical aspects. Hence on a theoretical level, combining inter-organizational workflow, security policies and access control is well understood.

REFERENCES

1. Abouelhoda, M.; Issa, S.; Ghanem, M. (2012). "Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support". *BMC Bioinformatics* 13, pp. 77
2. Alfresco Enterprise 4.0.2 Documentation (2012). http://docs.alfresco.com/4.0/index.jsp?topic=%2Fcom.alfresco.enterprise.doc%2Fconcepts%2Fwf-whatis-workflow.html
3. Cardoso, J., Bostrom, R., and Sheth, A. (2004). *Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications.* Inf. Technol. and Management, Vol. 5(3-4), pp. 319–338.
4. Crampton, J., Martin, K., and Wild, P. (2006). *On Key Assignment for Hierarchical Access Control.* In CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations, pp. 98–111, Washington, DC, USA. IEEE Computer Society.
5. Damiani, E., Di Vimercati, S.D.C., Paraboschi, S., and Samarati, P. (2000). *Design and Implementation of an Access Control Processor for XML Documents.* In Proceedings of the 9th International World Wide Web Conference on Computer Networks: The International Journal of Computer and Telecommunications Networking, pp. 59–75, Amsterdam, The Netherlands, North-Holland Publishing Co.
6. Farrell, S., and Kaijser, P. (1995). *A Non-repudiation Service Architecture and Certification Infrastructure.* In EDITT Workshop, pp. 113–124.
7. Fetscherin, M., and Schmid, M. (2003). *Comparing the Usage of Digital Rights Management Systems in the Music, File and Print Industry.* In Proceedings of the 2003 International Conference of Electronic Commerce, pp. 316–325. ACM Press.
8. Georgakopoulos, D. (2004). *Teamware: An Evaluation of Key Technologies and Open Problems.* Distributed Parallel Databases, 15(1), pp. 9–44.
9. Gottschalk, K., Graham, S., Kreger, H., and Snell, J. (2002). *Introduction to Web Services Architecture.* IBM Systems Journal, 41(2), pp. 170–177.
10. Jablonski, S. and Bussler, C. (1996). *Workflow Management: Modeling Concepts, Architecture and Implementation.* International Thomson Computer Press.
11. Java Cryptography Architecture Reference Guide. (2006). http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html. Sun Microsystems.
12. KnowledgeTree Document Management System (Open Source Edition) (version 3.4.2). (2007). http://www.knowledgetree.com/. JamWarehouse.
13. Koshutanski, H. and Massacci, F. (2003). *An Access Control Framework for Business Processes for Web Services.* In XMLSEC '03: Proceedings of the 2003 ACM Workshop on XML Security, pp. 15–24, New York, NY, USA. ACM Press.

14. Kwok, S. H. (2002). *Digital Rights Management for the Online Music Business.* SIGecom Exch., 3(3), pp. 17–24.
15. Messerges, T. S., and Dabbish, E. A. (2003). *Digital Rights Management in a 3G Mobile Phone and Beyond.* In DRM '03: Proceedings of the 3rd ACM Workshop on Digital Rights Management, pp. 27–38, New York, NY, USA. ACM Press.
16. Miklau, G., and Suciu, D. (2003). *Controlling Access to Published Data Using Cryptography.* In VLDB'03: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 898–909. VLDB Endowment.
17. Misra, J., and Cook, W. (2007). *Computation Orchestration: A Basis for Wide-area Computing.* Software and Systems Modeling (SoSyM), 6(1), pp. 83–110.
18. MySQL Database Server. (2007). http://www.mysql.com/. MySQL AB.
19. National E-Health Transition Authority (NEHTA). (2007). http://www.nehta.gov.au/.
20. NEHTA Industry Seminar (2007): Jurisdiction Summit and Vendor Focus Sessions (Discharge Summary: The Information – Sample Discharge Summary). http://www.nehta.gov.au/index.php?option=com_docman&task=doc_details&gid=309&Itemid=139&catid=150.http://publications.lib.chalmers.se/records/fulltext/238220/238220.pdf.
21. Organization for the Advancement of Structured Information Standards (OASIS). (1993–2008). http://www.oasis-open.org/.
22. Park, J., and Sandhu R. (2002). *Originator Control in Usage Control.* In POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks, pp. 60–66, Washington, DC, USA. IEEE Computer Society.
23. Park, J., and Sandhu, R. (2002).*Towards Usage Control Models: Beyond Traditional Access Control.* In SACMAT '02: Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, pp. 57–64, New York, NY, USA. ACM Press.
24. Pouwels, Ivan & Koster, Ferry. (2017). *Inter-organizational cooperation and organizational innovativeness.* A comparative study. International Journal of Innovation Science. 9. 10.1108/IJIS-01-2017-0003.
25. Riehle, D., and Z¨ullighoven, H. (1996). *Understanding and Using Patterns in Software Development.* Theory and Practice of Object Systems, 2(1), pp. 3–13.
26. Rivest, R. (1992). *RFC 1321 – The MD5 Message-Digest Algorithm.*
    a. http://www.faqs.org/rfcs/rfc1321.html.
27. Rivest, R., Shamir, A., and Adleman, L. (1978). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Commun. ACM, 21(2), pp. 120–126.
28. Rossi, D. (2004). *Orchestrating Document-based Workflows with X-Folders.* In SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 503–507, New York, NY, USA. ACM Press.
29. Rossignoli, C. and Ricciardi, F., (2014) Inter-Organizational Relationships: Towards a Dynamic Model for Understanding Business Network Performance. Springer.
30. Sandhu, R. (1996). *Rationale for the RBAC96 Family of Access Control Models.* In RBAC '95: Proceeding of the 1st ACM Workshop on Role-based Access Control, pp. 1–8, New York, NY, USA. ACM Press.
31. Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). *Role-Based Access Control Models.* IEEE Computer, 29(2), pp. 38–47.
32. Sandhu, R., and Samarati, P. (1994). *Access Control: Principles and Practice.* IEEE Communications Magazine, 32(9), pp. 40–48.
33. Shen, J., and Qing, S. (2007). *A Dynamic Information Flow Model of Secure Systems.* In ASIACCS'07: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, pp. 341–343, New York, NY, USA. ACM Press.
34. Shibboleth Project (version 1.3). (2007). http://shibboleth.internet2.edu/. Internet2.
35. Simon, R., and Zurko, M. E. (1997). *Separation of Duty in Role-based Environments.* In CSFW '97: Proceedings of the 10th IEEE Workshop on Computer Security Foundations, pp. 183–194, Washington, DC, USA. IEEE Computer Society.
36. Singaravelu, L., Wei, J., and Pu, C. (2008). *A Secure Information Flow Architecture for Web Services.* In 2008 IEEE International Conference on Services Computing, pp. 182–189, Los Alamitos, CA, USA. IEEE Computer Society.
37. SOAP (version 1.2). (2007). http://www.w3.org/TR/soap12/. W3C Recommendation.

38. Sohlenkamp, M., Mambrey, P., Prinz, W., Fuchs, L., Syri, A., Pankoke-Babatz, U., Kl¨ockner, K., and Kolvenbach, S. (2000). *Supporting the Distributed German Government with POLITeam.* Multimedia Tools Appl., 12(1), pp. 39–58.
39. Stallings, W. (1995). *Network and Internetwork Security: Principles and Practice.* Prentice Hall.
40. Stallings, W. (2006). *Cryptography and Network Security: Principles and Practice.* Prentice Hall.
41. Stinson, D. (1995). *Cryptography: Theory and Practice.* CRC.
42. Stohr, E., and Zhao, J. L. (2001). *Workflow Automation: Overview and Research Issues.* Information Systems Frontiers, 3(3), pp. 281–296.
43. Summers, R. (1997). *Secure Computing: Threats and Safeguards.* McGraw-Hill.
44. Taverna (2010). http://www.taverna.org.uk/introduction/what-is-a-workflow-management-system/
45. Van der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. (2003). *Workflow Patterns.* Distrib. Parallel Databases, 14(1), pp. 5–51.
46. Web Services Business Process Execution Language (WS-BPEL) (version 2.0). (2007). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. OASIS Standard.
47. Web Services Description Language (WSDL) (version 2.0). (2007). http://www.w3.org/TR/wsdl20/. W3C Recommendation.
48. Web Services Federation Language (WS-Federation 1.1). (2006). http://www.ibm.com/developerworks/library/specification/ws-fed/. BEA Systems, BMC Software, CA,IBM, Layer 7 Technologies, Microsoft, Novell, VeriSign.
49. Web Services Policy (version 1.5). (2007). http://www.w3.org/TR/ws-policy/. W3C Recommendation.
50. Web Services Security (2006). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. OASIS Standard Specification.
51. Workflow Management Coalition – *Terminology and Glossary.* (1999). http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf. Workflow Management Coalition.

EZEAMASOBI CHIBUZOR, H.C INYIAMA, GODSPOWER I. AKAWUKU (2023). WORKFLOW MODELS IN AUTOMATED BUSINESS ENVIRONMENTS. *NEWPORT INTERNATIONAL JOURNAL OF SCIENTIFIC AND EXPERIMENTAL SCIENCES (NIJSES)* 3 (2):60-71.