Classification of Software Frameworks Utilised in Water Resource Management Modelling

RMM Pradeep[1#] and A Edirisuriya[2]

[1]*Department of Information Technology, Faculty of Computing, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka*
[2]*Department of Computer Engineering, Faculty of Computing, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka*

[#]pradeep@kdu.ac.lk

Abstract - The framework of a particular unit or a system is the structure on which it is built or founded. There appears a conflict in the understanding of frameworks by water resource decision-making professionals and software development professionals. This contradiction affects the quality of the software systems developed for water resource management decision-making. Hence, the objective of the present work is to classify the available understanding of frameworks to contribute to a clear understanding to achieve better and sustainable framework classification to water resource management software system. The present work carried out a systematic review and conceptualised the principle of the framework through an evaluation of interdependencies between presently available understandings. The reviewed environmental modelling frameworks revealed the availability of four different categories such as, Software language foundation, Software on platforms, Techno-business platforms, and Building blocks frameworks. This classification allows the environmental system modellers to understand which framework they will develop and decide in which depth they need to explore technology and business domains.

Keywords: software, system, framework, water resource management, environmental modelling, empirical literature review

I. INTRODUCTION

A. Monolithic to microservice

Monolithic software development refers to the construction of traditional single-tire systems and codes are carrying out the interconnections between user interfaces, business logic and the data that existed in a single platform. Even the modification to the system carries a cost; monolithic had been established as the primary architectural approach to many business systems due to development and implementation simplicity. However, due to the dynamicity in the requirements and advancement of technology, present day trends move towards microservices where user interfaces, business logic, and data are provided and maintained separately as services (Gos and Zabierowski, 2020).

Conceptually the microservices are utilised in automating the complex water resource management process (Wybrands et al., 2021). But such systems need to be evolved from scratch due to absent of fully pledged environmental microservices, dynamics in management process and stakeholder requirements. Hence, the construction is still with the characteristics of monolithic development. This influence the developers to carry out trial and error attempts to accumulate necessary constructional instruments such as user interface guideline, development methodology, security mechanisms while the development (Pradeep and Wijesekera, 2017, 2015b; a). The common acceptance is, such approaches are challenging to achieve the required software qualities as the resources are utilised to discover/invent the basic concepts and artefacts repeatedly (Schmidt, Gokhale and Natarajan, 2004). Nevertheless, due to no proper exemplary works in water resource decision making, it has to follow the monolithic approach (Pradeep and Wijesekera, 2020).

B. Software Framework solves Software Entropy

The monolithic architecture requires adding different components to the developed system. It

results in uncontrollable software architecture - software entropy, the internal levels of disorders in complex or closed systems that get messy over time, same as the second law of thermodynamics (Canfora et al., 2014; Roth, 2017). Roth (2017) highlighted two reasons for software entropy which are common situations for the water resource management systems as: the complexity of the systems and communication requirements across the complex distribution of the system in multidiscipline.

However, as the software frameworks are being developed to depict the subclasses and components for the required scenario, those facilitate better selection of appropriate techniques and methods (Taligent, 1996 as cited in Aksit et al., 1999). Then, software framework could handle the foresaid complexity and communication difficulties. Therefore, it can consider that the most practical solution to manage the software entropy in monolithic development is the utilisation of software framework which is more appropriate to the scenario. Therefore, it required to select a suitable framework for water resource management problem and solution domains.

C. Problem Statement

Even though software frameworks are important, when searching them, contradictory information results from the existing literature. For example, some frameworks describe the technical implementation of the software codes in libraries, while others describe the business artefacts' architectural positioning in software frameworks. Then the situation is getting more complex when searching the suitable water resource modelling software frameworks. As most of such literatures' attention is being paid to the core area of water resource management. Those are simultaneously describing both the technical and business structures under the heading of framework.

Hence selecting the suited is devious and ambiguous as there is no proper classification in the water resource modelling software frameworks. As well as when considering the terminology, it could observe an absence of standardising classification. Therefore, it is vital to demarcate the conceptual boundaries of different software frameworks; then, researchers will select or express the practising frameworks more clearly.

D. Objective

The present work aims to classify the water resource management modelling software frameworks.

II. METHODOLOGY AND EXPERIMENTAL DESIGN

H. Methodology

As the present work based on the knowledge of the previous work, it systematically selected the literature for the study. First, it googles the term software framework and collects the primary keywords of "design pattern, reusable components, modularisation" from different blogs and technical discussion forums. Then through google scholar keyword search, it found the most appropriate research articles. Going through such papers' abstracts, it isolates the essential papers for the study. Then using the connectedpapers.com, an AI website, it develops a connected-paper graph, as shown in Figure 1.0. By studying the literature in the graph links, select the most appropriate literature works for software frameworks.

As the water resource management/environmental modelling frameworks are not included in the previous search, it searches the related papers through Scopus data base using *Publish or Perish 7*® app for the key words "water resource management software, environmental modelling frameworks". From the 200 papers output, it selected only the research articles which satisfied all the following conditions: (1) describe a software tool or software utilisation for a practical implementation of environment/water resource management decision-making process, (2) published in a journal with impact factor more than 2.5, (3) having more than 20 citations and (4) published within the last 20 years. Further, it selected only ten literatures as it can justify the ten are substantial to demonstrate more than 85% accurate view of the population according to the study of Pradeep & Wijesekera, (2012) on the water resource management tool evaluation sample size research.

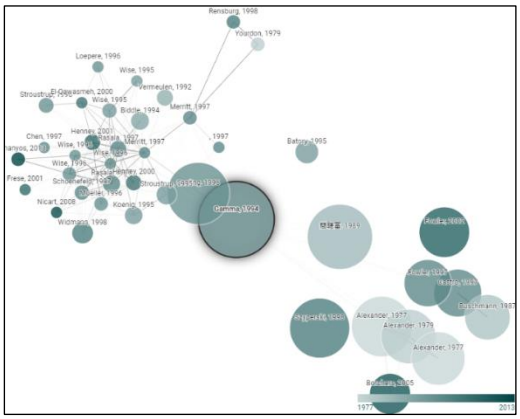

Figure 1.0 Connected Paper graph for Gamma et al., (1994, 1993)
Source: Created through https://www.connectedpapers.com/

I. *Literature Review*

1) Framework Definition: According to the general dictionary meaning, the framework is a supporting structure that can construct something physically, such as a house or bridge. When such construction becomes the conceptual structure of decision, plan, organisation or workflow, the framework consists of interrelated individual or combinations of ideas, information, beliefs, rules, and principles (Cambridge University Press, n.d.). However, the three terms: (1) standard, (2) guideline, and (3) framework, which carrying similar meaning and using concomitantly. Nevertheless, the standards are more mandatory activities or commitments than the recommended activities and commitments described in the guidelines. Then frameworks provide a more conceptual understating of what activities and commitments are required to achieve the required goals (ACRL, 2006; Nolan, 2019). Accordingly, the frameworks are facilitating more precise control and mindful amendments to the activities and commitments.

2) Framework Motivations: Tosolve most software development problems, Parnas (1972) introduced a concept called "Information Hiding". Conceptually, information hiding is a series of activities required to hide the software artefact's design and implementation decisions, but its interface reveals the functionality (Parnas, 1972). Then, within the software, the code blocks were reused, hiding design decisions to reduce development, testing, and modification costs. The concept was extended to the software system levels, and with time, it reuses the predeveloped software with modifications to solve the new problem. In reusing, the reusable component of the predeveloped software may be a set of source codes or a required functionality providing software modules or software units with required inputs and outputs.

Design Patterns: However, with the rapid growth of technology in many dimensions the nature of the problems to be solved was got complex. Then professionals must pay more attention to the problem domain; hence, the solution implementation needs to be carried out with less effort. Therefore, the reusable software artefact becomes popular, but with the advancements in technology more sophisticated solutions were developed. The researchers , Gamma et al. (1993), suggested to express successful design structures using the concepts in object oriented (OO) design paradigm. The design patterns provide clear guidelines for practitioners to arrive at decisions through alternatives and trade-offs when integrating the reusable software artefacts (Gamma et al., 1994, 1993).

Object-Oriented Framework: Even higher-level design patterns are catalogued the development experience for easy teaching and communicating, the software frameworks provide more concrete, practical implementation capability (Gamma et al., 1993). Generally, the software frameworks are reusable semi-finished architectures which can be utilised in the different application domain. The reusability of software is an essential requirement that initially fulfils with call-back procedures, then functions and Abstract Data Types (ADT). However, the object-oriented (OO) concepts-introduced inheritance and dynamic approaches assist in developing powerful frameworks which facilitate the reuse of whole/sub software systems, including the design. Therefore, the frameworks could standardise the internal parts to a specific domain (Schmidt, Gokhale and Natarajan, 2004). Pree (1994) called this scenario an application framework; however, the programmer needs to develop such after critical examination of the architecture and implementation details of the software artefacts. Therefore, he suggested utilising the meta-patterns where it identifies hot spots (the flexible components in the framework), white spots (template method: defines the abstract definition

of the algorithm - Gamma et al., 1993), and grey spots (hook method: provide communication between classes - Wirfs-Brock et al., 1990). This hot-spot-driven approach of Pree has made a foundation for object-oriented software frameworks (Pree, 1994, 1995; Pree and Sikora, 1997).

Component Framework: At the OO involve, it found that one main problem associated with the concept as, OO depends on the language and sometime compiler. Then it required to redevelop objects when needs to implement in another programming languages. Therefore, to independent from the language foundations, the component framework was evolved. The componentwear concept abides by the object-oriented concept as its encapsulation is regarding the data and methods amalgamation (Smith, 1997; Pree, 1997). Then component framework describes the capability of language-independent components utilisation to remove the barrier with the OO software framework. However, from the design, the components are language-independent groups of classes, but carefully developed to provide functionalities that users see as a black box and language independent (Scherp and Boll, 2005).

Aspect-oriented Framework: Further, the development efforts required to cater secondary set of user requirements such as optimisation of resources (memory, network, processor). Then, this opens another dimension of requirements that exceeds the need for reusability. Then Kiczales et al. (1997), suggested the Aspect-oriented programming concept. There the "aspects" are properties of the system while those behave like classes on OO design. This concept was influenced to develop an aspect-oriented framework based on the system's non-functional aspect and business rules (Silva, Braga and Masiero, 2004). These frameworks are dependent on the fundamental hook method, then even today, these frameworks are appearing in practising (Kant and Gupta, 2015).

Metadata-Based Frameworks: This extended the limitations of a small number of functional variabilities in the aspects-oriented frameworks. The use of metadata allows the extension of the behaviours, and the framework facilitates dealing with a large number of functional variabilities. However, the framework describes the code level implementation and metadata process (Guerra et al., 2013).

Service-oriented Frameworks: This encapsulates the deployable component models which provide the service independent to the platform and server. The components are built on the OO architecture, but those benefit distributed computing where inter-process communication is required (Bieber and Carpenter, 2001).

Application Frameworks: This is not a diversified concept, and it refers to the primary OO /component frameworks describe earlier. However, the application frameworks are concerned with more abstract architectures regarding the complex business units or application domains (Fayad and Schmidt, 1997). Then, all frameworks described up to this point explain the code level implementation of the frameworks. There the frameworks provide quality software through four inheritances. First, the modularity of frameworks stable and standardise the volatile implementation requirements, and secondary, reusability reduces the programmer's effort in repeating the reconstruction of developed solutions. The third is expendability of the framework achieved through the popular hook method, while the fourth is runtime architecture which ideally hooks the domain-specific process to the invoked event by implementing the reactive dispatching mechanism.

2) Frameworks Classification schemas: When reviewing the above literature findings, frameworks are described under multiple schemas. Out of those, "scope" is one of the schemas which mattered in the evolvement of different frameworks. Then Fayad & Schmidt, (1997), classify the application frameworks into three considering the scope as (1) System infrastructure frameworks: a developer-oriented local software architecture to standardise language processing tools (2) Middleware integration frameworks: A distributed and commonly available software architectures of distributed applications and components (3) Enterprise application frameworks: A comprehensive software architecture for enterprise-level business application. Accordingly, the middleware and enterprise

architectures are taken more time to develop from scratch, but users can commercially acquire. Further, the enterprise framework provides the entire application's infrastructure and functionalities, which is absent at system infrastructure and middleware integration frameworks (Mili et al., 2002). Then the decision of selecting a suitable enterprise framework is dependent on the stability, adequacy, and economy of the framework. In the same way, Krajnc & Heričko, (2003) summarised seven schemas including "scope", such as approach (the approach taken to develop framework such as OO, component, aspect describe above), Extensibility (framework facilitation of Whitebox, Gray box, Blackbox, glass box), Standardisation (based on the availability of standardisation and/or standardisation authority), Granularity (the simplicity of the implementation and utilisation), License (free or commercial) and Format (framework is either logical, physical, source code or binary code). Then present work found this classification schemas is substantial to the present work.

Well-known Frameworks: However, apart from the technical descriptions and classification, most software engineers simply practice frameworks in their day-to-day developments. Then these frameworks technically available via an application programming interface (API) and supported by software development kits (SDKs). Then using these frameworks, developers automate the business logic of the solutions without bothering the fundamental activities related to communication, system software and hardware. There are popular framework groups suite with the need of the developers, such as (1) Framework for web applications: These frameworks are developed to handle the internet-inherent characteristics such as unstructured-big-dynamic data management, interoperability, cross-platform management, communication and interconnection (Jazayeri, 2007). These frameworks are fundamentally built on different language frameworks such as Angular on JavaScript, Django on Python, and Larval on PHP. (2) Data Science Frameworks: These frameworks assisted the engineers to change the data into action by facilities to data science-related activities of Ask, Acquire, Assimilate, Analyse, Answer, Advise, and Act (Andrade, 2015). Examples are Apache Spark (multi-language support analytical engine framework), PyTorch (open-source machine learning framework) and TensorFlow (end-to-end open-source machine learning framework). (3) Frameworks for Mobile Development: These frameworks are proving the point to point (P2P) data management with platform-specific and hybrid mobile app development capabilities (Spindler, Grossniklaus and C.Norrie, 2009). Ionic (open-source framework for cross-platform native app development), Xamarin (.net platform-based framework), and Kivy (Python-based embedded and enterprise applications framework) are few examples of tons. (4) GIS Application Development Frameworks: These frameworks provide spatial data manipulation and geoprocessing facilities to automate the nosiness processes (Luaces et al., 2005). Few examples are ArcGIS Web Application Developer Framework - ADF (enabled Java and .NET to integrate GIS functionalities) and QGIS Framework (open-source framework for developing GIS functionalities and its applications).

3) Framework Related to Water Resource Management: As the present works main intention is to classify the water resource software frameworks, it critically reviewed seven water resource modelling software framework and two general environmental modelling software frameworks to understand how those are explaining under the term framework.

Water Resource Modelling Frameworks: This section summarised the seven water resource-related articles, with the major components include in described framework.

Andreadis et al., (2017) developed a framework for hydrological modelling and data assimilation software framework, which can nowcast and forecast using the hydro model. The framework named Regional Hydrologic Extremes Assessment System (RHEAS) is constructed with the concepts related to data, GIS model, Hydro Model, Crop model, and the users.

Sood et al., (2018) Smart flood management framework is developed to integrate IoT, big data and High-performance Computing for smart flood management. The framework describes the IoT layer, Fog layer, Data Analysis and Presentation Layer.

Abebe et al.,(2019) developed the Coupled Flood-Agent-Institution Modelling framework (CLAIM) to assess the different scenarios for flood risk effect on human and environment utilising Agents, Institutions, Urban environment, Physical processes, and External factors as the essential components.

Tightly couple Hydro and GIS modelling framework (PIHMgis) is developed by Bhatt et al. (2014), for construct the water management user interface. The Data development, Hydrological model, Data analysis, Domain composition, Data access library, and Shared geodatabase are the framework's building blocks.

Wang et al., (2018) integrated the different information sources to construct a high-resolution urban flood model when developing a water resource modelling framework. The fundamental concepts in the framework architecture are DEM Revision, Flood modelling and, Flood information extraction (multiple data sources)

The Groundwater Visualisation System (GVC) is a software framework that displays data and animate the water information utilising a conceptual hydrogeological model and third party inputs. Cox et al.(2013) developed this framework unitising the layers of Database, Data collection, GVC package, Simulation outputs, 3D Geo-model, Analysis, and Image/video.

Welsh et al., (2013) Source Integrated Modelling System (IMS) is a framework that integrates the models in river systems using layers such as Graphic interface, command line, service, application services, the simulation engine.

Web-based flood forecasting system (WFFS) is an online multiuser-multi-expert interacting framework for flood forecast whilst in an emergency. Li et al., (2006) used Data conversion, Flood forecast model, Calibration of the model, Forecasting and Flood analysis as the main components of the framework.

Environmental software frameworks: Apart from the seven articles, three others on environmental software discipline. Out of them, Parker et al., (2002) developed Integrated assessment and modelling (IAM), a framework that integrated the major components of environmental modelling such as Stakeholders, Scales, Issues, Disciplines, Models. Further, Object Modelling System Version 3 (OMS3) is a software-oriented environmental modelling framework developed by David et al., (2013). The framework components are described as Products, Development Tools, Knowledge base, and Recourses.

III. RESULT AND DISCUSSION

A. *Software framework construction*

Then when considering the software artefacts frameworks, it can observe interdependent concepts when developing different types of frameworks. However, all the software frameworks fundamentally describe how the software artefact codes and behaviour should be handled as a thumb rule. Figure 2.0 shows the amalgamation of all the considered software framework concepts.

Accordingly, it could review that the software development industry's software frameworks are always documenting and describing the software artefacts' internal construction and implementation details. Then the depth of the different fireworks is varying from call-back procedures to enterprise-level architectures. Further, as those can be commercialised, standards and licence types developed by organisations.

B. Classification through schemas

Then it reviewed how the studied ten environmental and water resource modelling frameworks are describing those software frameworks. According to the available descriptions, it attempted to categorise them utilising the seven schemas of Krajnc & Heričko (2003) describe above. Then it found RHEAS, WFFS and OSM3 frameworks are constructed based on the software artefact-based frameworks. As well as all frameworks could be categorised into the same subclasses of five schemas as extensibility: black-box, standardisation: absent, granularity: simple, license: free and format: logical format. However, apart from RHEAS, WFFS and OSM3, all other frameworks show only somewhat relativity to the characteristics of sub-classes. For the “approach” and “scope” schemas, frameworks were classified only considering the

conceptual relativity to the sub-class. See Table 1 for the classification analysis.

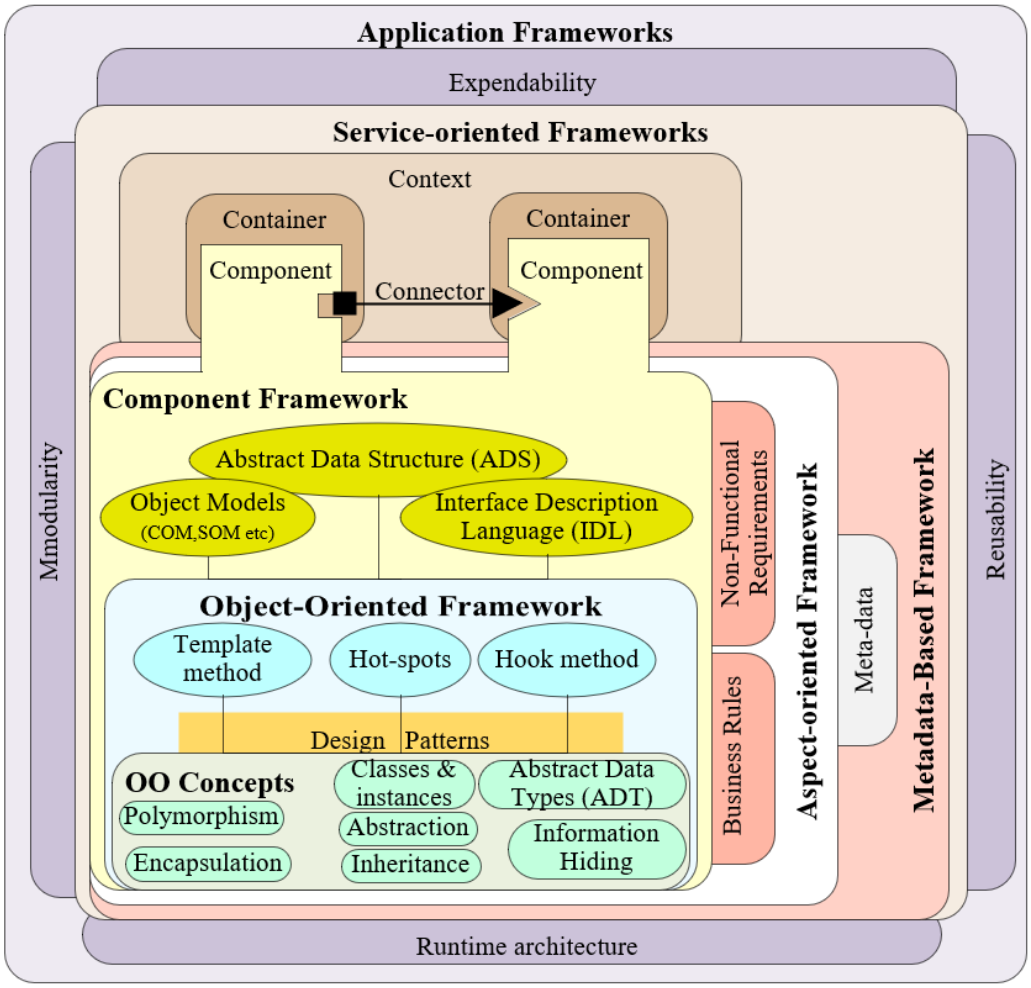

Figure 2.0 Interdependencies of Software Frameworks

Table 1: Environmental/ water resource Software Framework Vs Software framework classification schema

Environmental/ water resource Software Framework	**Software framework classification schema**		**Developed or utilised Software Framework**
	Approach	Scope	
Smart flood management framework (Sood et al., 2018) IF 2.79, Cites 49	Aspect*	Middleware*	No direct software framework described. A conceptual framework
CLAIM (Abebe et al., 2019) IF 4.8, Cites 25	Agent-based*	Enterprise*	
PIHMgis - (Bhatt, Kumar and Duffy, 2014) IF 4.8, Cites 107	Component*	Enterprise*	
Urban flood modelling FW (Wang et al., 2018) IF 4.8, Cites 71	Aspect*	Enterprise*	
GVS - (Cox et al., 2013) IF 4.5, Cites 43	Aspect*	Enterprise*	
IMS (Welsh et al., 2013) IF 4.8, Cites 167	Aspect*	Enterprise*	
IAM (Parker et al., 2002) IF 4.8, Cites 315	Aspect*	Enterprise*	
RHEAS (Andreadis et al., 2017) IF 2.74, Cites 23	Object-Oriented	System infrastructure	Developed through OO Software framework
Web-based flood forecasting system - WFFS (Li et al., 2006) IF 3.88, Cites 46	Web-Based	Enterprise	Enterprise JavaBeans (EJB), CORBA, DCOM, and Java RMI-IIOP
OMS3 (David et al., 2013) IF 4.8, Cites 165	Component	Middleware	Modular Modelling System (MMS), OMS1
Note: *relates conceptually only Environmental/ water resource Software Framework column contain impact factor (IF) of the journal and citation received (Cites) figures of the article			

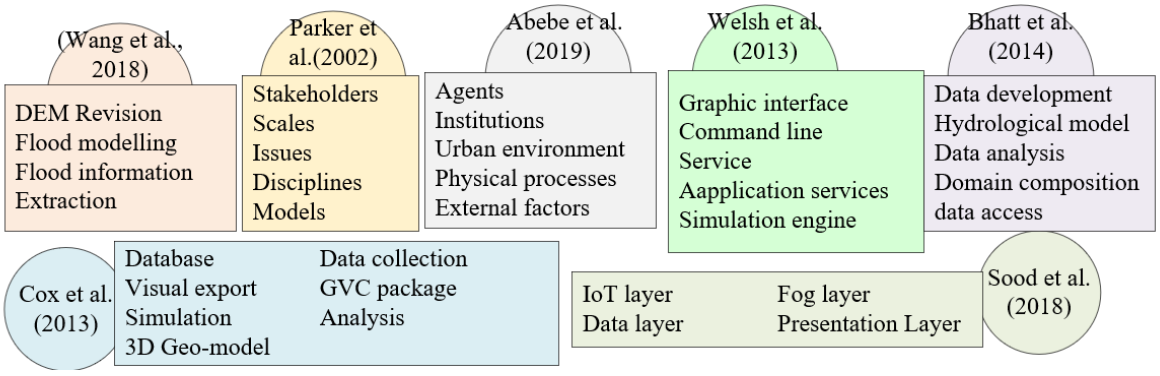

Figure 3.0 Water Resource Modelling Framework Components in studied literatures.

J. Water Resource Software Framework components

The first seven frameworks of Table 1 only showing the general characteristics of the software framework schemas. Hence, the present work mapped the components identified as the main building blocks of the framework by the authors (Figure 3.0). Then it can observe all the components are describing the tools, method and techniques related to water resource data capturing, processing, and visualising. Then described components can be categorised as "business objects and logics" in the water resource and environmental modelling discipline.

K. Water Resource Modelling Framework Classification

Generally, the software frameworks enable integrating all the required reusable components to the problem/solution domain and explain the interoperations and communication between the components (Petty et al., 2014). According to the present analysis, these components vary from functions, procedures, ADTs, hotspots, object models, ADS, IDL, containers, context, services, non-functional requirements, business rules and meta-concepts in different explanations. Then it can realise that both the software and environmental software frameworks are considering both the software architecture and system architecture (Gacek et al., 1995; Medvidovic and Taylor, 2010). Nevertheless, most of such frameworks describe the conceptual framework of the system architecture. Those consist of system components, connections between them, stakeholders, functional and non-functional needs with specific needs, such as IoT, 3D visualisation/simulation (need to fulfil to attain the business requirements).

Then reviewing all these findings, it developed the levels of the software frameworks as shown in Figure 4.0. The dark colour rounded boxes show the conceptual components for each level, and White colour rounded boxes show the examples. Each level's conceptual ingredients become the part of ingredients of the next higher level. However, the utilisation of such a part is optional and depends on the construction of the upper-level framework.

Note:

- DODAF: Department of Defence Architecture Framework of USA
- MODAF - Ministry of Defence Architecture Framework, UK
- NAF: The NATO Architecture Framework

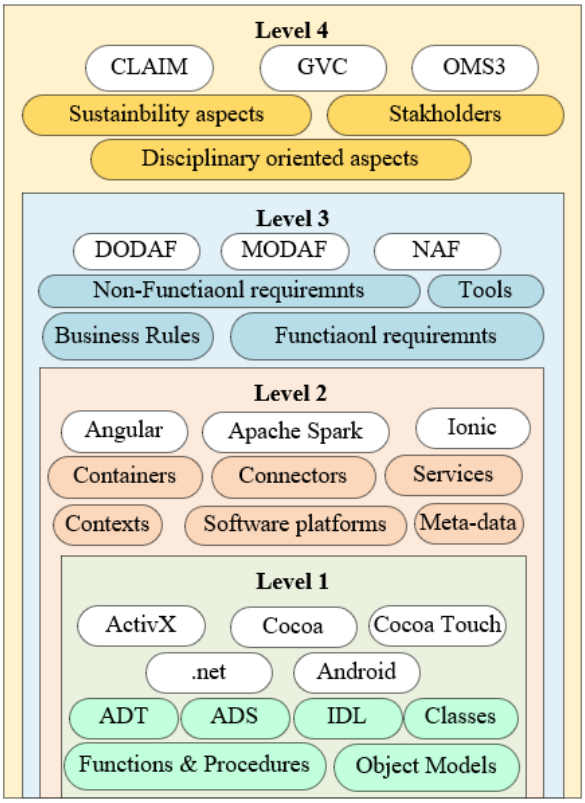

Figure 4.0 Classification of Water resource management Software Frameworks

According to these levels, the conceptual components of practical water resource modelling software frameworks have appeared in Level 3 & 4. Examples of two such software requirement scenarios are shown in Table 2.0.

Table 2.0 Examples for Level 3 and 4

Scenario 1: Optimise the urban watershed culverts	
Level	3
Conceptual component	Non-functional requirement
Specification of the Conceptual component to the scenario	• The maximum diameter of the culvert should be less than 4 meters. • When placing multiple culverts in the same location, maintain at least one culvert diameter gap in-between
Considerations for the above Conceptual component	• Related functional requirements. • What tools needs • Most suited Hydro model
Scenario 2: Urban runoff management	
Level	4
Conceptual component	Stakeholders
Specification of the Conceptual component to the scenario	• The list of stakeholders in according to disciplines • Non-functional requirements of each stakeholder
Considerations for the above Conceptual component	• Manage the conflict requirements among the disciplines. • Arrive the sustainable solutions

Table 3.0 Classification of Water Resource Modelling Software Frameworks

Class (Level)	**Main undertake**
Software Language foundation (Level 1)	Establishes software foundation that includes elements of the software language. It provides the facility to coding the libraries via API. Develop Software Components
Software on platforms (Level 2)	Interrelates the classes/modules to the processes in technological platforms. Develop Software Packages
Techno-Business platforms (Level 3)	Integrates the processes and component which required to interact withing different technological and business objective. Develop Software Systems.
Building blocks (Level 4)	Assembles the major components need to be integrated to construct multidisciplinary systems. Develop Models of software systems.

However, Level 1 & 2 describe the full technical details that align with the software engineering aspects. Then available frameworks can be directly used, sometimes with delta addons. Therefore, level 1 & 2 frameworks are independent of the discipline-business rules and aspects.

Then it can observe Level 1 to 4 are dependent on each other but, Level 1 & 2 dominated by technical aspects while Level 3 & 4 by management/scientific aspects. Further, the Level 1 framework is dominated by individual software programming language foundation, while Level 2 is the platform where the software operates. In the same way, Level 3 frameworks explain both the concerns on the individual discipline (business model) and technical tools, while Level 4 conceptualise the building blocks of interdisciplinary aspects.

Reviewing all these empirical findings can classify the software system frameworks for water resource management as shown in Table 3.0.

IV. CONCLUSION

The term framework is utilising with different meaning at different activities in the software automation process. In the analysis and design stages, it referred to the architecture of the software. When in the coding stage, the programming modularisation and construction of optimised code blocks represent the frameworks. However, when system automation, the attention of the framework exceeds the software automation to the conceptual optimisation of the input-processes-outputs with the business rules.

Then in water resource management software automation, it required to build integrated environment model. For that it needs to properly plan the sustainable decision-making software systems, utilising the optimised code blocks. A close review of the present work in such approach, it could isolate four framework levels in construction of integrated environmental model-based water resource management software.

These levels are starting from the highly technical descriptions- the concepts related to foundation of the software construction. Then in the following by level to level, frameworks collaborate with the system's managerial and/or scientific perspectives reducing the technical details. The final level describes the managerial and scientific concept integrations with less or no technical detail. Then it can be considered as a conceptual foundation of the software system.

Then with such understating, present work contributes the framework-level classes as (Level 1) Software Language foundation, (Level 2) Software on platforms, (Level 3) Techno-Business platforms, and (Level 4) Building blocks frameworks.

Then, the system designers and environmental software modellers will be able to utilise this classification as the fundamental guideline to select or build the suited frameworks for their water resource management problem/solution. Then, it will reduce the conflicting determinations on the frameworks.

However, as this classification is mainly based on the conceptual relation of “software” and “system” differentiations, the defined framework levels are more valuable to software and system developers in the environmental modelling discipline.

ACKNOWLEDGEMENT

As the present finding is based on the collective contribution of the previous works, the authors pay their gratitude to all the authors who are indicated in the reference section.

REFERENCES

Abebe, Y.A., Ghorbani, A., Nikolic, I., Vojinovic, Z. and Sanchez, A., 2019. A coupled flood-agent-institution modelling (CLAIM) framework for urban flood risk management. *Environmental Modelling and Software*, 111, pp.483–492.

ACRL, 2006. Guidelines, standards, and frameworks. In: *ACRL’s Guide to Policies and Procedures*. [online] American Library Association. Available at: <http://www.ala.org/acrl/resources/policies/chapter14#14.1>.

Aksit, M., Tekinerdogan, B., Marcelloni, F. and Bergmans, L., 1999. Deriving Object-Oriented Frameworks from Domain Knowledge. In: Mohamed E. Fayad, D.C. Schmidt and R.E. Johnson, eds. *BBuilding Application Frameworks: Object Oriented Foundations of Framework Design*. [online] New York, USA: John Wiley & Sons Inc.pp.169–198. Available at: <https://research.utwente.nl/en/publications/deriving-object-oriented-frameworks-from-domain-knowledge>.

Andrade, A., 2015. *Data science framework overview*. [online] Data Science Field Guide. Available at: <http://datascienceguide.github.io/data-science-framework> [Accessed 1 May 2021].

Andreadis, K.M., Das, N., Stampoulis, D., Ines, A., Fisher, J.B., Granger, S., Kawata, J., Han, E. and Behrangi, A., 2017. The Regional Hydrologic Extremes Assessment System : A software framework for hydrologic modeling and data assimilation. *PLoS ONE*, [online] 12(5), pp.1–22. Available at: <http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0176506&type=printable>.

Bhatt, G., Kumar, M. and Duffy, C.J., 2014. A tightly coupled GIS and distributed hydrologic modeling

framework. *Environmental Modelling & Software*, 62, pp.1–15.

Bieber, G. and Carpenter, J., 2001. Introduction to service-oriented programming (rev 2.1). *OpenWings Whitepaper*, [online] pp.1–13. Available at: <ftp://81.20.17.8/books/Counterpart 07/Service Oriented Architecture and Programming/ServiceOrientedIntroduction.pdf >.

Cambridge University Press, n.d. Framework. In: *Cambridge English Dictionary*. [online] Available at: <https://dictionary.cambridge.org/dictionary/english/framework> [Accessed 17 May 2021].

Canfora, G., Cerulo, L., Cimitile, M. and Di Penta, M., 2014. How changes affect software entropy: An empirical study. *Empirical Software Engineering*, [online] 19(1), pp.1–38. Available at: <https://link.springer.com/article/10.1007/s10664-012-9214-z>.

Cox, M.E., James, A., Hawke, A. and Raiber, M., 2013. Groundwater Visualisation System (GVS): A software framework for integrated display and interrogation of conceptual hydrogeological models, data and time-series animation. *Journal of Hydrology*, [online] 491, pp.56–72. Available at: <http://www.sciencedirect.com/science/article/pii/S0022169413002321>.

David, O., Ascough, J.C., Lloyd, W., Green, T.R., Rojas, K.W., Leavesley, G.H. and Ahuja, L.R., 2013. A software engineering perspective on environmental modeling framework design: The Object Modeling System. *Environmental Modelling and Software*, [online] 39, pp.201–213. Available at: <http://dx.doi.org/10.1016/j.envsoft.2012.03.006>.

Fayad, M.E. and Schmidt, D.C., 1997. Object-oriented application frameworks. *Communications of the ACM*, 40(10), pp.32–38.

Gacek, C., Abd-Allah, A., Clark, B. and Boehm, B.W., 1995. On the Definition of Software System Architecture. *The First International Workshop on Architectures for Software Systems*, [online] (April), pp.85-95. Available at: <http://csse.usc.edu/csse/TECHRPTS/1995/usccse95-500/usccse95-500.pdf>.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J., 1993. Design patterns: Abstraction and reuse of object-oriented design. In: O.M. Nierstrasz, ed. *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer.pp.406–431.

Gamma, E., Vlissides, J., Helm, R. and Johnson, R., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. *Structural Competency for Architects*. Pearson.

Gos, K. and Zabierowski, W., 2020. The Comparison of Microservice and Monolithic Architecture. *2020 IEEE 16th International Conference on the Perspective Technologies and Methods in MEMS Design, MEMSTECH 2020 - Proceedings*, pp.150–153.

Guerra, E., Buarque, E., Fernandes, C. and Silveira, F., 2013. A flexible model for crosscutting metadata-based frameworks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7972 LNCS(PART 2), pp.391–407.

Jazayeri, M., 2007. Some trends in Web application development. *FoSE 2007: Future of Software Engineering - IEEE*, pp.199–213.

Kant, T. and Gupta, M., 2015. Redesign of Hot Spots using Aspect-Oriented Programming. *International Journal of Computer Applications*, 117(20), pp.11–14.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M. and Irwin, J., 1997. Aspect-oriented programming. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1241, pp.220–242.

Krajnc, A. and Heričko, M., 2003. Classification of object-oriented frameworks. *IEEE Region 8 EUROCON 2003: Computer as a Tool - Proceedings*, B, pp.57–61.

Li, X.-Y., Chau, K.W., Cheng, C.-T. and Li, Y.S., 2006. A Web-based flood forecasting system for Shuangpai region. *Advances in Engineering Software*, [online] 37(3), pp.146–158. Available at: <http://www.sciencedirect.com/science/article/pii/S0965997805000931>.

Luaces, M.R., Brisaboa, N.R., Paramá, J.R. and Viqueira, J.R., 2005. A generic framework for GIS

applications. *Lecture Notes in Computer Science*, 3428, pp.94–109.

Medvidovic, N. and Taylor, R.N., 2010. Software architecture theory and practice. In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. Cape Town, South Africa: IEEE Computer Society.pp.471–472.

Mili, H., Fayad, M., Brugali, D., Hamu, D. and Dori, D., 2002. Enterprise frameworks: Issues and research directions. *Software - Practice and Experience*, 32(8), pp.801–831.

Nolan, A., 2019. *Governance : Understanding guidelines , frameworks & standards*. [online] Compliance. Available at: <https://www.spector.ie/blog/understanding-guidelines-frameworks-and-standards-from-a-governance-standpoint/#:~:text=A framework is a conceptual,and their appetite for risk.> [Accessed 17 May 2021].

Parker, P., Letcher, R., Jakeman, A., Beck, M., Harris, G., Argent, R., Hare, M., Pahl-Wostl, C., Voinov, A., Janssen, M., Sullivan, P., Scoccimarro, M., Friend, A., Sonnenshein, M., Barker, D., Matejicek, L., Odulaja, D., Deadman, P., Lim, K., Larocque, G., Tarikhi, P., Fletcher, C., Put, A., Maxwell, T., Charles, A., Breeze, H., Nakatani, N., Mudgal, S., Naito, W., Osidele, O., Eriksson, I., Kautsky, U., Kautsky aa, E., Naeslund ab, B., Kumblad ab, L., Park ac, R., Maltagliati ad, S., Girardin ae, P., Rizzoli af, A., Mauriello ag, D., Hoch ah, R., Pelletier ai, D., Reilly aj, J., Olafsdottir ak, R. and Bin al, S., 2002. Progress in integrated assessment and modelling. *Environmental Modelling & Software*, [online] 17(May 2001), pp.209–217. Available at: <www.elsevier.com/locate/envsoft>.

Parnas, D.L., 1972. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12), pp.1053–1058.

Petty, M.D., Kim, J., Barbosa, S.E. and Pyun, J.J., 2014. Software frameworks for model composition. *Modelling and Simulation in Engineering*, 2014.

Pradeep, R.M.M. and Wijesekera, N.T.S., 2012. Selecting a Usability Evaluation User Group – A Case Study the Development of a Hydro-GIS Tool Aiming Urban Flood Mitigation. In: *Civil Engineering Research for Industry Symposium (CERIS) – 2012*. [online] Moratuwa: Universiry of Moratuwa. Available at: <http://dl.lib.mrt.ac.lk/handle/123/9922>.

Pradeep, R.M.M. and Wijesekera, N.T.S., 2015a. Development of Security Stamp for Desktop Spatial Data Modification in Unrestricted Access Platform. In: *8th Intrernational Research Conference*. [online] Rathmalana: General Sir John Kotelawala Defence University. Available at: <http://ir.kdu.ac.lk/handle/345/1057>.

Pradeep, R.M.M. and Wijesekera, N.T.S., 2015b. Modification of User Friendliness in to a HydroGIS Tool. In: *8th Intrernational Research Conference*. [online] Ratmalana: General Sir John Kotelawala Defence University. Available at: <http://ir.kdu.ac.lk/handle/345/1138>.

Pradeep, R.M.M. and Wijesekera, N.T.S., 2017. Predictive cum Adaptive Systems Development Methodology for HydroGIS Tool Development. *10th International Research Conference - 2017*.

Pradeep, R.M.M. and Wijesekera, N.T.S., 2020. Incorporating Stakeholder Concerns in Land Information Systems for Urban Flood Management. *Array*.

Pree, W., 1994. Meta Patterns - A Means For Capturing the Essentials of Reusable Object-Oriented Design. In: M. Tokoro and R. Pareschi, eds. *European Conference on Object-Oriented Programming- ECOOP 1994*. Heidelberg: Springer, Berlin.pp.150–162.

Pree, W., 1995. State-of-the-art Design Pattern Approachess— An Overview. *Technology of Object-Oriented Languages and Systems (TOOLS 95)*, pp.1–11.

Pree, W., 1997. Component-based software development - A new paradigm in software engineering? *Software-Concepts and Tools*, 18(4), pp.169–174.

Pree, W. and Sikora, H., 1997. Design patterns for object-oriented software development. *Proceedings - International Conference on Software Engineering*, pp.663–664.

Roth, S., 2017. *Clean C++: Sustainable software development patterns and best practices with C++ 17*. [online] *Clean C++: Sustainable Software Development Patterns and Best Practices with C++ 17*. Schleswig-Holstein: Apress. Available at: <https://link.springer.com/book/10.1007%2F978-1-4842-2793-0>.

Scherp, A. and Boll, S., 2005. A lightweight process model and development methodology for component frameworks. In: *Proceedings of the tenth International Workshop on Component-Oriented Programming*.

Schmidt, D.C., Gokhale, A. and Natarajan, B., 2004. Leveraging Application Frameworks. *ACM Queue*, 2(5), pp.66–75.

Silva, M.T., Braga, R. and Masiero, P.C., 2004. Evolução Orientada a Aspectos de um Framework OO. In: *Workshop de Manutenção de Software Moderna*.

Smith, R., 1997. *Implementing a Typed Object Calculus*.

Sood, S.K., Sandhu, R., Singla, K. and Chang, V., 2018. IoT, big data and HPC based smart flood management framework. *Sustainable Computing: Informatics and Systems*, [online] 20, pp.102–117. Available at: <http://www.sciencedirect.com/science/article/pii/S2210537917302469>.

Spindler, A. de, Grossniklaus, M. and C.Norrie, M., 2009. Development Framework for Mobile Social Applications. In: P. van Eck, J. Gordijn and R. Wieringa, eds. *Advanced Information Systems Engineering. CAiSE 2009. Lecture Notes in Computer Science*. [online] Berlin, Heidelberg: Springer, Berlin, Heidelberg.pp.275–289. Available at: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84903221029&partnerID=tZOtx3y1>.

Wang, Y., Chen, A.S., Fu, G., Djordjević, S., Zhang, C. and Savić, D.A., 2018. An integrated framework for high-resolution urban flood modelling considering multiple information sources and urban features. *Environmental Modelling and Software*, 107(July 2017), pp.85–95.

Welsh, W.D., Vaze, J., Dutta, D., Rassam, D., Rahman, J.M., Jolly, I.D., Wallbrink, P., Podger, G.M., Bethune, M., Hardy, M.J., Teng, J. and Lerat, J., 2013. An integrated modelling framework for regulated river systems. *Environmental Modelling & Software journal, Elsevier*, 39, pp.81–102.

Wirfs-Brock, R., Wilkerson, B. and Wiener, L., 1990. *Designing Object-Oriented Software*. New Jersey: Prentice Hall.

Wybrands, M., Frohmann, F., Andree, M. and Marx Gómez, J., 2021. *WISdoM: An Information System for Water Management*. [online] Springer International Publishing. Available at: <http://dx.doi.org/10.1007/978-3-030-61969-5_10>.

AUTHOR BIOGRAPHIES

Maj. R.M.M. Pradeep, BSc (Hons) in MIS, MSc (Civil) University of Moratuwa.

Maj. Pradeep is a Senior Lecturer in Department of Information Technology, Faculty of Computing, General Sir John Kotelawala Defence University, Sri Lanka. His research interests lie in HydroGIS framework, System analysis and design and software modelling & processing. Pradeep has served on roughly twenty-four years in different assignments in arbovirus vector research, military IT resource management and Computing and Geoinformatics research.

Dr. Ananda Edirisuriya, BSc (Hons) (Maths) (USJ, Sri Lanka), Postgraduate Dip (Stat.) (UOC, Sri Lanka), MSc (Comp. Sci.) (China), PhL(Comp. Sci.) (Sweden), PhD (Comp. Sci.) (Sweden). is a Senior Lecturer in Department of Computer Engineering, Faculty of Computing, General Sir John Kotelawala Defence University, Sri Lanka. He was the former Head of the Department of Computer Science, University of Sri Jayewardenepura, Sri Lanka. His main research interests are Model driven approaches for Enterprise Information Systems Design, Data Engineering, Application Frameworks and Software Quality Improvements. Dr Ananda has served more than thirty years in different research, administration, and industrial positions as a senior academic and consultant.