ECA-driven Architectural Connectors Meet Rewrite Logic and Django for Smoothly Developing Adaptive Sound and Efficient AI-Powered Knowledge-Intensive Software Applications

Nasreddine Aoumeur* Kamel Barkaoui Gunter Saake
University of Science and Technology (USTO), Algeria

Laboratoire CEDRIC, CNAM, 292 Saint Martin, 75003 Paris - FRANCE

ITI, FIN, Otto-von-Guericke-Universität Magdeburg, Germany

Abstract. Artificial Intelligence (AI) with its Machine-Learning (ML) vertiginous advances are dramatically reshaping our way of developing software either as (prompting) GenerativeAI- or purely ML-based ones. Nevertheless, this newly emerging AI-Era of software is unfortunately not significantly benefiting from decades of investigations and findings around software-engineering (SE) concepts, principles and methods. The resulting is plethora of GenerativeAI- and ML-based software: *Black-boxed* rigid *ill-conceptually* and completely *isolated* from our "ordinary" yet mostly disciplined software landscape.
The aim of this paper is to contribute in leveraging such unsatisfactory Prompting- and ML-based software form to be well-conceptually, dynamically adaptable by *intrinsically* fitting it within our "ordinary" domain-oriented software landscape: We refer generically to as AI-Powered (knowledge-intensive) applications software; thereby reconciliating Domain- and AI-Experts instead of contemporarily miserable 'confrontation'. We achieve such promising endeavour by exactly capitalizing on best advanced SE concepts and principles
More precisely, we are putting forward an innovative stepwise integrated model-driven approach that smoothly exhibits the following conceptual, founded and technological milestones. Firstly, any structural features are semi-formally modelled as UML components intrinsically thereafter mapped into (ordinary and *ML-based*) Web-*Services*. Behavioural crucial features are then captured as intuitive *business rules* mostly at the *inter-service interactions*. Secondly, for the *precise conceptualization* of such inter-service behavioural rules, we are proposing tailored graphically appealing stereotyped primitives as ECA-driven architectural (inter-service) *connector glues*, we refer to as ECA-driven interaction *laws*. Thirdly, for the rigorous certification, while staying ECA-Compliant we are tailoring Meseguer's true-concurrent rewriting logic and its strategies-enabled MAUDE language for that purpose. Last but not least, for the efficient implementation we are proposing a four-level implementation still ECA-Compliant architecture, by relying on modern software technologies including python-empowered API with Django and its REST framework and Visual-Studio enterprise as advanced IDE. All approach milestones and steps are extensively illustrated using a quite realistic AI-powered software application dealing with Brain Tumor diagnostics while stressing all its benefits, with at-top reliability, dynamic-adaptability, self-learning and understandability.

Keywords: ECA-driven architectural interaction laws, UML and Service-orientation, Machine-Learning (ML), KNN, Brain-Tumor, Reliability and Adaptability, Rewriting-Logic, Domain- and AI-Experts, Django REST and Python API.

1 Introduction

Artificial Intelligence (AI) with its Machine-Learning (ML) vertiginous advances are dramatically reshaping our manner of developing software either as (prompt-

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 95-112, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152007

ing) GenerativeAI- or purely ML-based ones [1–4], we refer to as *AI-Powered* software, and this in almost any application domain: healthcare, finance, logistics, tourism, just to name a few. Indeed, Machine-Learning (ML) as well as Generative AI capabilities, lead by AI-Experts or Data-Scientists, transcend by far the (human) Domain-Experts reasoning and deduction capabilities when it comes to *efficiency* and time response, adding to that the tremendous ability in *automatically* learning and reasoning about hundreds if not thousands of *years* of (symbolic-)knowledge (i.e datatsets), whereas human Domain-Experts (like Doctors, Engineers, etc.) could at most and error-prone and only "daily" handle couple of years of (meaningful) knowledge-intensive experience. Moreover, with the recent AI-Agentic ubiquity [5], extremely complex applications can be handled in a self-automatic manner.

Nevertheless and despite all these advantages and tremendous benefits, we argue and demonstrate in this contribution that these "pure" *AI-centered* software, when compared to our huge ordinary (non-intelligent and non-automated) software landscape, they exhibit several severe if not *life-critical* deficiencies, including: (1) being algorithmic and henceforth *black-boxed*, *rigid*, *ill-conceptually* and thereby hardly tractable by essence; (2) consequently in most *mission-critical* applications, such as healthcare, finance, space, etc. they remain untrustful to be largely embraced (hallucinations phenomenon); (3) although pretended to be (AI-centered) "software", they did not at-all integrate with our *long-term experienced* "ordinary" disciplined software landscape which is mostly conceptually founded, adaptable, understandable and therefore highly reliable; (4) This unfortunate "artificial" dissection between, AI-centered and ordinary software, has further triggered dangerous confrontation between Domain- and AI-Experts leading to high anxiety and fears for domain-experts to be soon "kicked out" by this emerging AI-Era; we endeavor in this research direction to bring things straight!, within it domain-experts and/or software-engineers remain by far intelligently unmatched when it comes to supervision and meaningful (ECA-driven) knowledge-centricity exhibition vrs. (meaningless symbolic) datasets! and therefore they ultimately deserve keeping the overhand decision-making.

The aim of this paper is therefore to contribute in leveraging such unsatisfactory prompting- and ML-based intelligent software towards more disciplined *conceptualization*, transient *adaptability* mainly by *intrinsically* and judiciously fitting them within that "ordinary" (knowledge-intensive) software landscape, we refer generically to as AI-Powered knowledge-intensive software applications. Thereby we are endeavouring *reconciliating* Domain- and AI-Experts instead of the contemporarily miserable 'confrontation'. We achieve such promising endeavour by exactly capitalizing on best advanced *software-engineering* (SE) concepts, principles and methods, widely investigated, worked-out and more importantly practiced over more than four to five decades of founded and applied SE research [6]. More specifically, in this paper we propose to judiciously and smoothly bring together the following SE milestones, we found necessary in leveraging AI-centered and ordinary software development towards harmoniously unified model-driven AI-powered knowledge-intensive runtime adaptable and reliable software application development approach:

– *UML-Classes and -Components as Services [7]:* For coping with structural features of any AI-powered application to be developed at both ordinary and ML-based perspectives as will be detailed later, while behavioural features are

firstly intuitively elicited using (even-driven) *business rules* with high focus on the externalized inter-service interactions;

- *Service-orientation (and cloud-computing) [8]:* For straightforwardly perceiving such (UML-based) structural features as (Web-)services providing basic properties and operations and exhibiting minimal necessary *intra*-service behaviours while leaving ;
- *Precise ECA-driven architectural glue-based conceptualization of any inter-service behaviors:* As dynamically transient ECA-driven *architectural connectors* [9–11] with very appealing graphically-enhanced primitives, with at first-stance high-prioritization on Domain-Experts ordinary-based ones as will be later justified and illustrated, we refer to as (functional- and ML-based) ECA-driven *interaction laws*;
- *Rewrite logic for formal rapid-prototyping sound validation and verification:* A tailoring instantiation of the yet rule-based Meseguer's rewriting logic [12] coupled with its true-concurrent strategy-driven MAUDE language is forwarded at both ordinary- and ML-sides;
- *Efficient yet ECA-driven compliant Django-based service-oriented four-level implementation architecture:* By calling on best modern technologies, namely *Django REST framework* for persistent Web-Services, *Python-API* Requests [13,14] and Object-orientation for a decoupled invocation of such Web-Services at the participant interfaces and their interaction ECA-driven levels.

The rest of this paper is organized in conformance with the (five-level) different phases of the approach, we are subsequently referring to as $\mathcal{AR}$ECA4AI$\mathcal{SA}$ (*Adaptive Reliable ECA-driven AI-Powered knowledge-intensive service-oriented Software Applications*), where each phase will be explained illustrated and validated using the running real-size case study tackling Brain Tumor diagnostics [15]. But first to shed more light on the approach itself, its background, milestones are objectives, we holistically and illustratively summarize it with focus on its stepwise development life-cycle.

In the third section, we then focus on the semi-formal graphical UML-based structural features applied to the Brain-Tumor application domain; important to stress here is that AI-Expertise is very requested even at this early requirements elicitation phase: A fact that makes our approach so valuable conceptually. The fourth section, to be lead by the Domain-Expert, is devoted to the precise ECA-driven architectural conceptualization of the behavioural features of the application at-hand from functional perspective, namely the brain tumor case study, by the ECA-driven inter-services interaction laws. The fifth crucial section of this contribution concerns the ML-centered ECA-driven inter-services interaction laws, lead by a dedicated AI-Expert / Data-Scientist, where for sake of simplicity and readability we opted here just for the KNN (K-Nearest-Neighbor) model with appropriate DataSet both perAI-Powered ceived as Web-Services. The resulting rule-based execution of such KNN-Algorithm, in terms of accuracy and prediction, is first superposed (as PUT in REST) on the dedicated ML-based Web-Service. Finally, these ML-based results are then to be exploited again by the Domain-Expert (doctor and his/her staff in this case) to boost his/her decision towards more AI-Powered trustfulness and preciseness. Once such intrinsically adaptable ECA-driven interaction laws, being it "ordinary" functional or ML-centered ones, have been precisely conceptualized and revisited, the next fifth section aims at coping with the formal mathematically-based underpinning. That is, this fifth section concerns the formal validation by rapid-prototyping and verification by

model-checking by smoothly shifting the ECA-driven interaction laws into a tailoring instantiation of Meseguer's rewriting logic and its true-concurrent MAUDE language, we apply on the Brain Tumor case-study. Finally, in the sixth section, for efficient technological implementation we propose yet an ECA-compliant four-level architectural centered around Django REST framework and Python API-Requests all in all developed on the basis of the Enterprise Visual Studio Integrated Development Environment. This paper is finally wrapped up by some remarks and further promising perspectives.

2 $\mathcal{AR}$ECA4AI$\mathcal{SA}$: Illustrated Steps and Applied Milestones

In some detail, what we are forwarding consists of an integrated stepwise model-driven approach, we refer to as $\mathcal{AR}$ECA4AI$\mathcal{SA}$, for developing *Adaptive Reliable ECA-driven AI-Powered knowledge-intensive service-oriented Software Applications*. As graphically illustrated in Figure 1, the approach is thus covering all software life-cycle, from requirements elicitation through precise conceptualization and formalisation for validation to an efficient compliant technologically modern implementation. This is, the approach could be understood and grasped as five-based explicit and transparent phases:

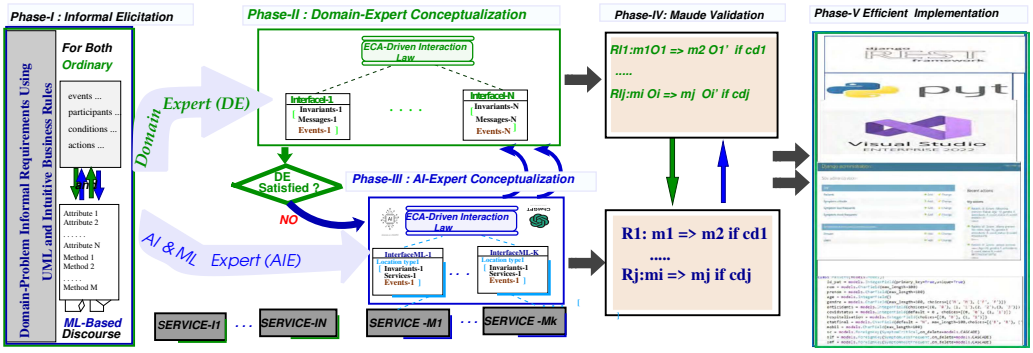

Fig. 1. $\mathcal{AR}$ECA4AI$\mathcal{SA}$ as a Disciplined and Adaptive ECA-Centred Architectural Approach for Service-Oriented AI-Powered Software Applications.

– $\mathcal{P}$**hase I— Semi-formal structural UML-Diagrams [7] and Rule-based [16] Requirements elicitation and modelling:** As UML method with its different diagrams remains the defacto standard for semi-formally -graphically modelling any domain application at-hand lead by domain-experts, we are adopting it with more focus on the structural features, namely UML-class and-components diagrams, we later smoothly perceive as Web-Services. For the behavioural features, we propose instead of dispersed dynamics UML-diagrams rather (event-driven) business rules, which are by essence intrinsically adaptable and intuitively business-driven. Moreover, we intuitively formulate such business rules more at the external inter-class resp. inter-service communication level, so that they can dynamically be adapted, upgrated, replaced and/or combined depending on the involved service participants *context* and configuration [17, 18]. In that sense, the intra-service will be drastically reduced to invariants and basic necessary properties and operations. Besides that "ordinary" usual domain-side UML- and Business Rules lead by the Domain-Experts, we are creatively proposing at this early stages of development, to

also allow and/or enforce the AI and ML-Experts to perceive à-la-UML at-least their predominant "(hyper)parameters" associated to any selected to-be subsequently implemented ML-based model(s) resp. algorithm(s) as well as all relevant properties and operations to be applied on involved DataSets. In this paper, as aforementioned we restrict ourselves to the simple KNN [13] (K-Nearest-Neighbor) algorithm, where besides the generic *percent* between training and test we have more specifically the *number of neighbors*: We could then both reason on them via tailored ECA-driven rules as will be later more highlighted using the Brain Tumor case-study. Of course for the case of deep-learning models, those hyper-parameters are numerous and quite complex to be managed and reason about by the respective AI-Expert and Data-Scientist, making that phase more interesting and valuable.

- $\mathcal{P}$**hase II— Domain-Expert ECA-driven precise Conceptualisation:** As we already stressed, our main objective is to *firstly* architecturally re-structure, previously UML- and rule-based elicited "ordinary" domain applications, around precise yet graphically appealing inter-services ECA-driven architectural *interaction laws*, so that our targeted software application become intrinsically dynamically adaptable and highly transparent and reconfigurable [17–21]. Concretely, as will be detailed subsequently, for instance, doctors and their staffs bring their knowledge-intensiveness experience for diagnosing Brain-Tumor mostly in the pattern of Event-Conditions followed by Actions and decisions to be taken, where conditions are extracted from patient records, associated symptoms and so forth; all in all as participant interfaces from different Web-services (e.g. Patient, Symptoms, antecedents disease, radio-image, etc.).
- $\mathcal{P}$**hase III— AI- and ML-based ECA-driven Conceptualization**: Process-speaking this phase is in fact two-steps based: (1) AI- and ML-based Step and (3) Domain-AI-Powered Step. In the following we highlight the essence and essential of each of these two important steps.
  1. **ML-based ECA-driven Conceptualization**: Depending on the complexity and the pre-defined objectives of the application at-hand, in several situations domain experts may found themselves vainly enforced to resort to AI- and ML-capabilities by calling on associated AI-Experts (Data-Scientists or even AI-Robot) for handling overwhelmingly complicated situations that transcend by far their capabilities. Another quite deliberately emerging option consists in resorting to AI-Empowering for "strategic/operational/economic" reasons such as efficiency, trustfulness, staff deficiency, cost effectiveness, etc.

     Independently of whatsoever reason is triggering this relevant step, all what we require to stay *service-oriented* and domain-expert-driven decision-handling, is that the resulting outputs of this step, that is at-least the computed accuracy and respective prediction/classification, have to be put in the respective Web-Service.
  2. **AI-Powered Domain-Expert ECA-driven Conceptualization**: The ultimate step consists thus for the Domain-Expert in capitalizing on the resulting output of the previous step, namely the recorded accuracy and prediction resp. classification lead by the AI-Expert. The resulting should be new *AI-powered* ECA-driven interaction laws bringing with them more trust, satisfaction, automation and transparency besides the intrinsic adaptability and understandability.

important to point out here is that depending on the degree of AI-Automation and trust of the governing enterprise or institution, even the second phase could be jumped to focus directly on this AI-based third phase; the other way around where AI-Automation is less trusted and/or established consists then in focussing only on the second phase leading to a pure "ordinary" domain-based application development: These are the tremendous benefits and flexibility of externalizing at the inter-service architectural level any (ordinary and/or AI-based) behavioural-features as ECA-driven business rules. Moreover, as we demonstrated in [17], we can go even further by separating between functional and (IOT-centered) context-aware ECA-driven interaction laws.

- $\mathcal{P}$**hase IV— (Rule-based) Rewriting-Logic and its** MAUDE **Language for Formal Validation and Verification:** Once the phases II and III have been satisfactory conceptualized and all associated (ordinary as well as ML-based) ECA-driven arhcitectural interaction laws deeply revisited and definitively agreed on, we pass to the formal underpinning of such arhcitectural conceptualization. For that crucial phase, we are proposing as semantical framework adequately tailored instantiation of yet rule-based Meseguer's Rewriting Logic [22] and its MAUDE language. This allows us not only true-concurrent validation by rapid-prototyping with the aim to detect conflicting, redundant and/or inconsistent ECA-driven interaction laws, but also to achieve formal (temporal-based) model-checking verification.
- $\mathcal{P}$**hase V— Efficient ECA-driven SO Implementation:** Once the conceptualization on both the ML-driven side as well as the ordinary one is satisfactory addressed and formally validated / verified, it remains the efficient implementation. For that purpose we are putting forwards a Django-based [23] advanced (Web-Service) REST framework-based implementation architecture. Furthermore, in order to still staying ECA-driven compliant, that architecture consists in a four-level one with: (1) At the button are the participating persistent Web-Services; (2) At the client-level an explicit Python API requests is interacting with any involved Web-Service; (3-4) To facilitate the formulation of the ECA-driven rules as fourth-level, an object-oriented intermediate third-level is proposed to blur out the complexity of JSON resp. XML Web-Service requests and responses.

3 Phase I: Semi-formal UML- and Rule based Requirements: The Brain Tumor Case

Brain tumors [15, 24, 25] represent in their simple description an uncontrolled proliferation of abnormal cell division in the brain. They are classified according to world health organization based on: 1) their originating cell and histology such as Glioma and meningioma 2) their grade of malignancy from grade I to grade IV, begin tumors with low grade I and II, and malignant tumors with high grade III and IV, and 3) their histogenetic biology (molecular biology).

Approved by the *"Cancer.Net"* Editorial Board (httr:r//www.cancer.net/about-us/cancernet-editorial-board): Symptoms of a brain tumor can be *general* or *specific*. General symptoms are mainly caused by the pressure of the tumor on the brain or spinal cord, whereas specific symptoms are more caused when a specific part of the brain is not correctly working. As depicted in Figure 2, we have straightforwardly described most of such (general as well as specific) symptoms as UML-classes, where besides that we have described the patient class with its

most usual properties. Without delving deeper into such symptoms and patient properties, which are more related to the medical field and its domain-experts, we summarize their essentials as UML-Components (later on as Web-Services) in the following, while spitting between these domain-based (green-colored) and ML-based ones (blue-colored).

3.1 Tumor Brain Domain UML-based entities and their rules

As depicted in Figure 2 besides the Patient, the Doctor component (i.e. Classes) are be described in order to diagnostic that Brain-Tumor symptoms and afterwards treating them, where according to *"Cancer.Net"* such symptoms could categorized into general or specific. More precisely, such different UML-based classes as (Web)Services could be summarized as follows:

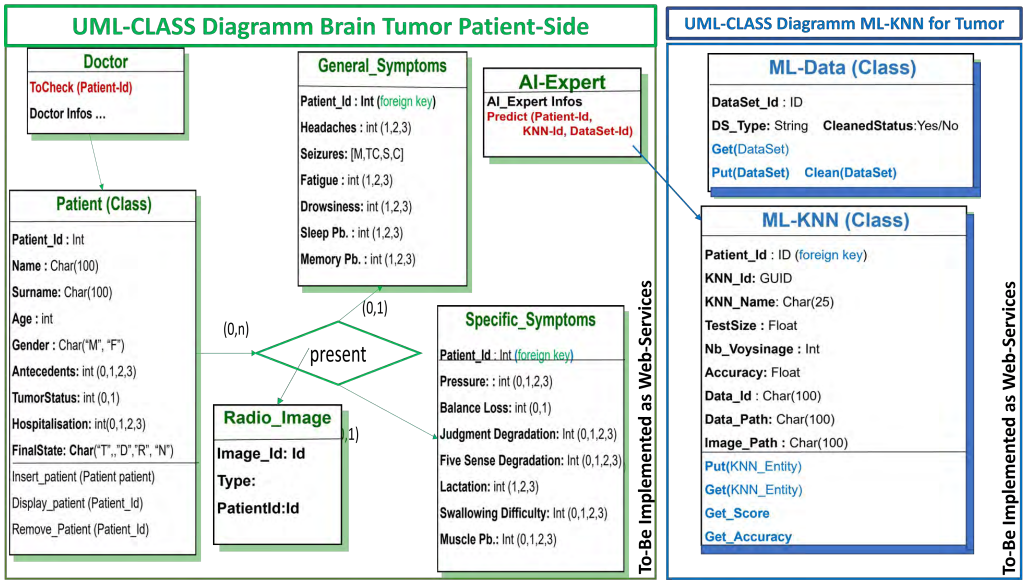

Fig. 2. *ARECA4AISA*—**Phase I**: The Brain Tumor Application Domain as UML-based Semi-Formal Description (with KNN-Class as ML-Based Modelling)

Patient UML-Class (Service) : Besides the usual personal information (eg. patient-Id, Name, Surname, Age, Gender), to facilitate the diagnostics while being still Event-driven important properties to consider include: The *TumorStatus* with the value 1 as being positively suffering from Brain Tumor (we could also opt for a more detailed classification as detailed for instance in [15]); *Antecedents* are previously suffering from severe diseases such Diabetic, Pressure or Cancer, etc; *Hospitalisation* referring to the degree of hospitalisation we have for simplicity numerically valued from 0 to 3; *FinalState* as the post-treatment status including those symbolic values (T:InTreatment, D:Died, H:InHospital, N:Normal). Well established CRUD operations, such as Create or Update any Patient are also to be considered here.

General Symptoms as UML-Class (Service) : Those include headaches, Seizures, Fatigue, Drowsiness, Spell Pb. as well as memory problems. For sake simplicity, most of such properties have been numerically valued from 1 to 3 depending on the severity of the case.

Specific Symptoms as UML-Class (Service) : Similarly the specific advanced symptoms include: Pressure, Balance-Loss, degradation of Judgement, Degradation of the five sense specifically the vision, Lactation, the swallowing difficulties as well as muscle problems.

Radio-graphical Image as UML-Class (Service) : The radio image could besides basic properties such as its Id as well as the Patient Id include more specific technical information such as the *Type* as trhe severy of the case and further information we are skipping here for sake of simplicity.

Doctors as UML-Class Service : This may include all relevant medical staff information such doctors, nurses and so on. Essential here is the event "ToCheck" triggering the starting of the diagnostics and thereby initiating the associated Even-driven interaction law.

Appointments, Medicaments and more:SKIPPED : As we pointed out, there is no hindering whatsoever to be cope with real-size application about Brain-Tumor by involving further required UML-based entities such as Appointments, Medicaments, and so on, as they are perceived as almost automatically Django REST-based Web-Services.

3.2 Tumor Brain AI- and ML-Centered UML-based entities and their rules

What deserves more attention in our approach is undoubtedly the "unusual" new "blue" right-side, where AI-Experts are requested to describe à-la-UML at-least the main (hyper)parameters of the ML-based models and algorithms as well as the properties of the respective DataSets and the main operations acting on such ML-based models. As aforementioned, as the focus lies more on the approach itself, we have decided to deal with one of the simplistic model, namely the KNN [26]. For that purpose, the AI-Expert need at least the following properties and operations: (1) *KNN_Id* as a primary Id for each created KNN-object instance; *Patient_Id* as foreign key referencing the concerned patient; optional *KNN_Identifier*; (3) *TestSize* as percent between training and test sets to play with in any ECA-driven rule with the aim to improve the accuracy; *Nb_Voysinage* as the number of Neighbourhood also to reason on it in any ECA-driven rule; The *Accuracy* and *Prediction*. Furthermore, as operations we need a (REST) *Put* operation to record the result (such as the prediction and related accuracy) as well as a (REST) *Get* operation requesting such results from the KNN-(Web-)Service for further treatment as we will illustrate subsequently. Similarly the to-be applied DataSet service should have the straightforward properties, such as *DataSet_Id*, *Size*, etc. and CRUD operations besides *Cleaning*, *Cleansing*, etc..

3.3 Informal Domain-based Rule Governing Brain-Tumor Application: Simple Illustration

At early requirement stages, *business rules* represent the best available modelling concepts to cope with competitiveness and evolution, while defining the behavioural aspects and any business logic of the application at-hand at the *business-level* and at different levels: Strategic, operational and technical [27, 28]. Business rules are mostly expressed in terms of Event-Conditions-Actions (ECA) forms [16], which suits well with the service-oriented paradigm based on (triggering) invocations/subscribing and discovering.

For our Brain Tumor running case, it is obvious that besides the symptoms, being it general or specific, also patient properties such *old-age* and historical records such as antecedent chronical diseases play decisive role for the ultimate decision-making of medical staff. Herewith just one possible rule could be for instance formulated as follows:

Rule (for Brain Tumor Diagnostics) *"If the patient is over 62 and suffers from at-least one antecedent chronical disease and presents most of* general *symptoms and his/her brain-radiography is over the value of 1 (severe) THEN the medical domain-expert staff decides that he/she definitely presents a Brain Tumor and has to be hospitalized for at-least X days and need to take the following medicaments (skipped here) ".*

Following the medical staff experience, a second rule could combine well defined *specific* symptoms when exhibited with a severe brain-radiography valuation. A third rule could concern both general and specific symptoms etc. We want just to stress how beneficial and crucial to follow a *transparent and explicit* rule-based formulation of requirements instead of opting for hard-coded or ad-hocly ill-conceived business logic (see [17, 18, 21] for more detail).

4 Phase II: Disciplined ECA-driven Architectural Service-oriented Conceptualisation

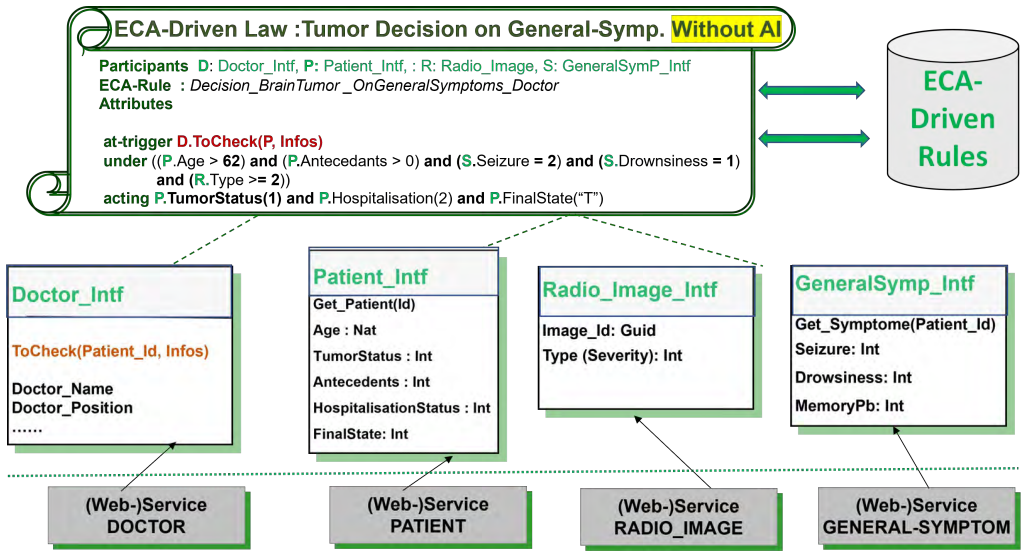

Fig. 3. $\mathcal{AR}$ECA4AI$\mathcal{SA}$—**Phase II**: The ECA-based Functional (Ordinary) Rules from the DOMAIN_EXPERT for the Brain Tumor Case

Due to space-limitation again, we refrain from delving into the architectural foundation and technical background behind the forwarded ECA-driven architectural disciplined conceptualization, where interested reader may specifically consult [18, 21]. That is, we restrict ourselves in directly projecting any informal rules with their (UML-based) services from the previous section into associated disciplined precise inter-service architectural interaction laws. That is, as depicted in Figure 3, in accordance with the previously described first informal rule, on initiating the triggering event, here *ToCheck(Patient_Id)*, the medical

staff call on the different involved Web-Services to get all properties and operations required, through tailored *interfaces*, to accordingly formulate the conditions and actions to undertaken in accordance. As depicted in Figure 3, such transient ECA-driven rigorous conceptualisation, with the pattern *at-trigger-under-acting*, is precise enough yet quite understandable to be directly implemented in any object-oriented resp. service-oriented programming language; in our case we opted for Microsoft .NET C# under Visual-Studio Enterprise.

Besides that preciseness, the advantages of such ECA-driven architectural conceptualization are obviously manyfold, including in particular: (1) The transparency and clarity in formulating the business logic in the form of self-explained ECA-based primitives (2) The intrinsic runtime adaptability as we can either adapt it or bring another ones (second or third afore-described rules). That means for our Brain-Tumor case, for instance, the staff medical depending the patient and the environment specificities may opt for the rule that fits at best that patient; (3) in this respect as we demonstrated in [17,18] even the current *contextual* conditions could be involved boosting the staff decision; and (4) the straightforward compliant efficient implementation as will be detailed later.

5 Phase III: Disciplined ECA-driven architectural ML-Based Service-oriented Conceptualisation

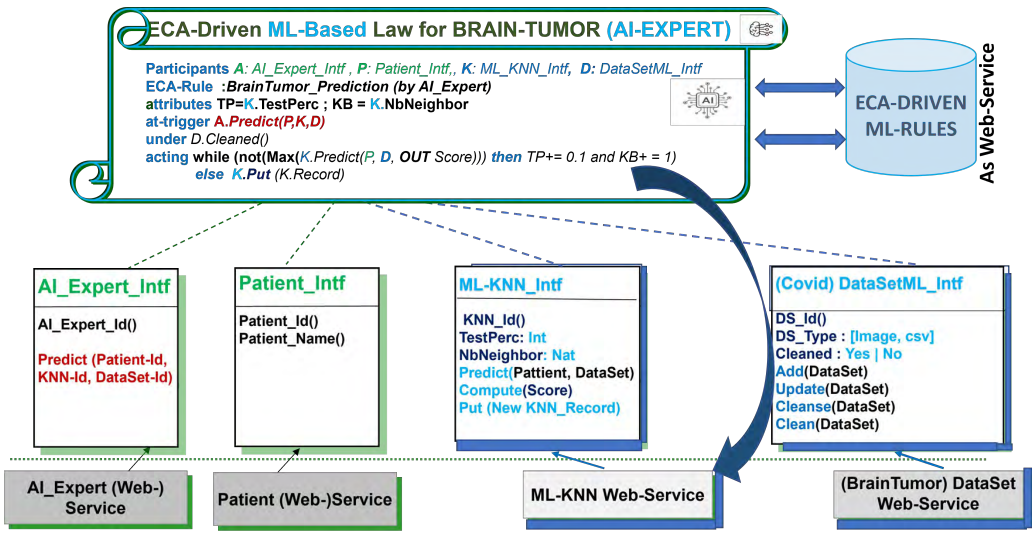

Fig. 4. $\mathcal{AR}$ECA4AI$\mathcal{SA}$—**Phase III(a)**: The ECA-based ML-centered Rules from the AI_Expert for the Brain Tumor Case

As we already stressed, with the rapidly emerging of the AI-Era with all its tremendous advantages on software development, healthcare in particular have been embracing it at vertiginous pace, although unfortunately still mostly in ad-hoc hard-coded rigid manner. In particular, for the diagnostics of complicated diseases such as Brain-Tumor, Covid-19, Alzheimer etc. [29], where the ML-based solutions become unavoidable. As we are distancing ourselves from the algorithmic-centricity of such AI-based solutions, as we afore-summarized, for this third phase we are proposing two intrinsically interrelated steps: Step (III-a) allowing in a transparent and transient manner conceptualizing any suitably ML-based

architectural inter-service ECA-driven interaction law selected by the corresponding AI-Expert/Data-Scientics/AI-Agent to be afterwards performed; whereas the second step (III-b) consists in coming back again to the Domain-Expert with the same pattern but by benefiting from such AI-Expert results.

5.1 Phase III-a AI-Expert ECA-driven Architectural ML-Based Conceptualisation

At this stage, we are going to capitalize on the ML-based UML-centered modelling from the first phase, I depicted in the right-hand of Figure 2, and on its straightforward mapping into Web-Services, including *KNN-Web-Service* and *KNN-DataSet Web-Service* and optionally *AI-Expert Web-Service*. More precisely, as graphically detailed in Figure 4, by assuming that the AI-Expert has opted for an ECA-driven architectural rule that allows him/her to incrementally looping on the percent-Training-Test set and the K-Neighbour until reaching best accuracy-scoring, expressed by the self-explained conditional expression `while(not(Max(Predict(P,D,OUT Score))) then TP+=0.1 and KB+=1`—where as described `TP` represents the `TestPercent` while `KB` represents the `NumberOfNeighbours`.

This elegant ECA-Driven demonstrates how flexible and very expressive are the ML-based ECA-Driven laws for capturing even very complex computations at the business-level. Once such optimal scoring is reached it remains just to persistently pushing the resulting prediction and that reached accuracy-scoring into the respective KNN-Web-Service through a *REST PUT* operation.

Again the advantages at this ML-based architectural ECA-driven ML-centred conceptualisation are manyfold, including: (1) the freedom for the AI-Expert to opt for any ML-model/algorithm he/she found mostly suitable resp. by comparing the resulting prediction and accuracy results as well as reasoning about them for optimisation purpose–the same concerns the to-be tackled associated DataSets; (2) As for "ordinary" rules, the new AI-based ECA-driven are also on-the-fly adaptable with high flexibility for their conceptualkisation (using when necessary additional local attributes); (3) Last but least, since we are implementing such rules themselves as *persistent* Web-Services, the AI-Expert as well as the Domain-Expert can benefits over the years from their *gained experiences* and improve thereby their decision-making, by capitalizing on such gained knowledge-intensiveness as "event-driven rule-based" DataSets while formulating any newly generated rule.

5.2 Phase III-b Domain-Expert AI-Powered ECA-Rules : Applied on the Brain Tumor Case

At this intermediate step, we assume that the AI-Expert "instantly" has already performed his/her ECA-driven prediction tasks, as we just explained, while putting the resulting prediction and accuracy-scoring into the respective persistent associated Web-Service, namely the KNN-Web-Service. The next expected crucial and ultimate step consists thus in boosting the Domain-Expert by AI-powering revisiting his/her previous ECA-driven architectural rules while fully-exploiting such ML-based predictions and accuracy-scoring results. For our Brain-Tumor case study, as depicted in Figure 5, the medical-staff now requesting (using for instance REST-based GET) such Brain-Tumor prediction and accuracy results from the concerned KNN-Web-Service and accordingly incorporate them while conceptualizing such new AI-powered ECA-driven rule and thereafter taking the most

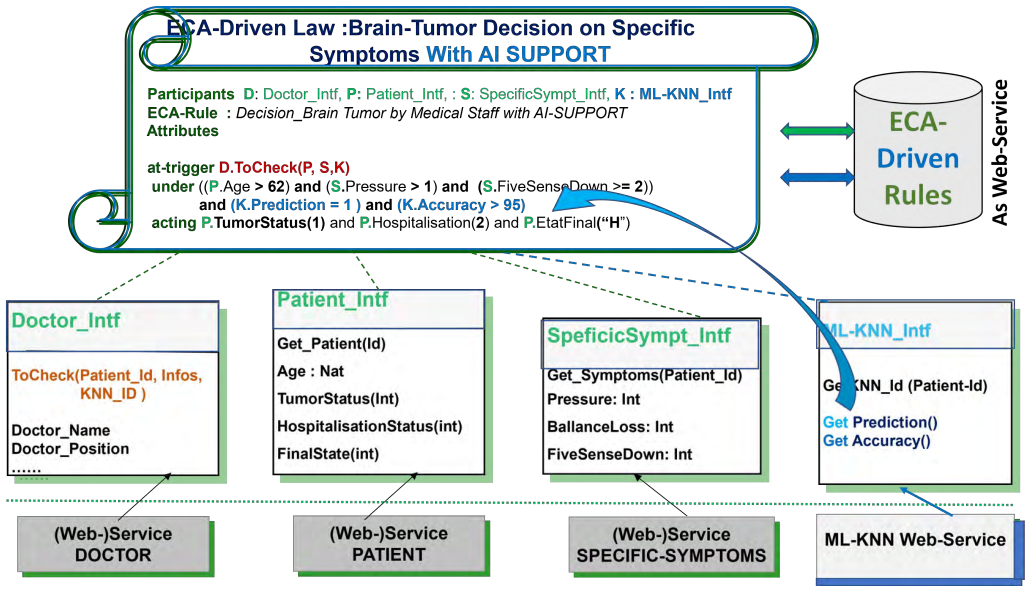

Fig. 5. $\mathcal{AR}$ECA4AI$\mathcal{SA}$—**Phase III(b)**: Domain Expert AI-Powered ECA-Rules Benefiting from ML-based Results

trustful decision, for instance by enforcing the accuracy to be more than 95%. We should note that for sake for adaptability, we have conceptualized the second rule which is based on the Brain-Tumor specific symptoms such as `Seizure`, `BalanceLoss` and `FiveSenseDegradation`—instead of the first rule modeled in Figure 3 based on the general symptoms and radio-image (not needed here in the presence of AI-prediction and accuracy!). That is, the medical staff has to decide themselves by estimating on the basis of their accumulated experience which accuracy percent is to be trusted for their specific patient on the basis of the reached KNN-results.

6 Phase IV: Leveraging MAUDE for Validating the $\mathcal{AR}$ECA4AI$\mathcal{SA}$ Architectural Approach

In order to result in a compliant MAUDE-based formal foundation with rapid-prototyping abilities of the approach conceptual-level, we have first to leverage MAUDE to service-orientation. Towards achieving this, this section first overviews the main concepts of the MAUDE language, using the Domain-Expert ECA-Driven rule given in Figure 3. We then present how to conceive their services and interfaces in MAUDE.

Recalling that, we have decided for a MAUDE-based declarative implementation, due to the following potential benefits. Firstly, rewriting logic (RL) is a unified framework for true-concurrent systems, promoting thus distribution in modern software-intensive systems. Secondly, MAUDE is highly efficient allowing millions of rewritings per-second with its Linux-based WorkStation implementation. Thirdly, with its intrinsic reflection capabilities [30], MAUDE promotes separation and explicit controlling of rules execution using strategies. Fourthly, by adopting rewriting-logic and its MAUDE we are also benefiting from years of own research-experience in that respect [31]. Last but not least, for certification purpose, MAUDE is endowed with an LTL-based built-in model-checker [30].

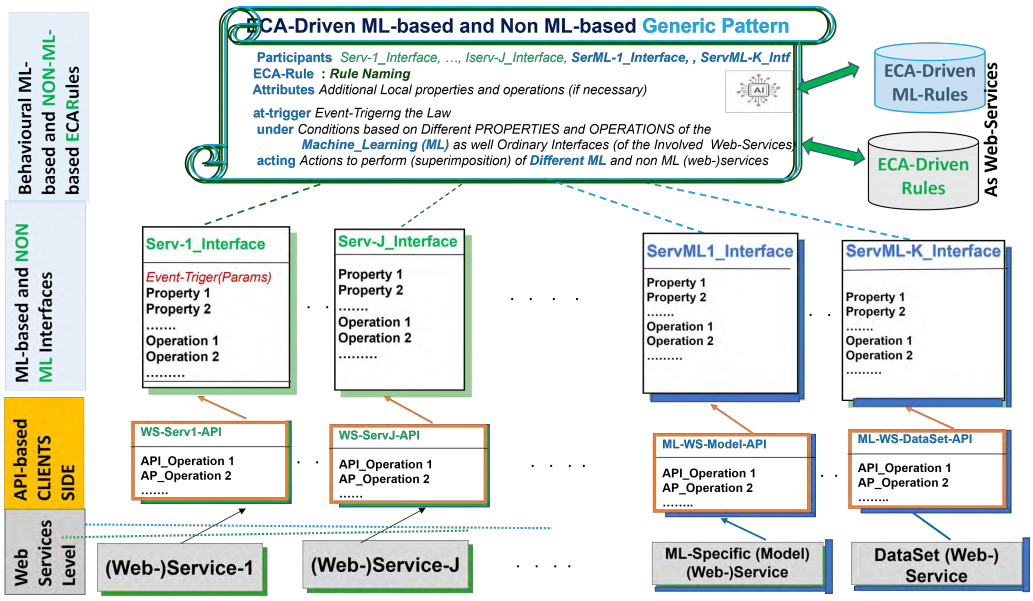

Fig. 6. $\mathcal{AR}$ECA4AI$\mathcal{SA}$—**Phase V**: ECA-Driven ML-BASED FOUR-LEVEL Django-Centred Architecture

6.1 Rewriting logic and MAUDE overview

Rewriting Logic [12] represents a unified model of concurrency. It interprets rewrite rules as a change through a categorical foundation. It further unifies the object paradigm with concurrency. Rewriting logic goes along with a very rich and efficient object-oriented programming / specification language called MAUDE [30].

In MAUDE, object states are conceived as tuples of the form $\langle Id : C | at_1 : v_1, .., at_k : v_k \rangle$; where Id stands for the object identity, C for its class while $atr_1, ..., atr_k$ denote attribute names with respective current values $val_1, ..., val_k$. Messages can be concurrently sent / receive to such object states. Both object and message instances flow together in the so-called *configuration*, as multiset governed by the union operator denoted by '_ _' . The precise definition of this configuration in MAUDE itself takes the form.

```
fmod Configuration is
  protecting ID        **** provides OId, CId and AId .
  sorts Configuration Object Msg .
  subsorts OId < Value , Attribute < Attributes .
  subsorts Object Msg < Configuration .
  op _:_ : AId Value → Attribute .
  op _,_ : Attribute Attributes → Attributes
  op ⟨_:_|_⟩ : OId CId Attributes → Object.
  op _ _ : Configuration Configuration → Configuration .
endfm.
```

The changes effect of messages on targeted object states is modelled using rewrite rules. The effect of such rules result generally in (attributes) state changes of some participating objects, creation / deletion of some objects, absorption of involved messages and emerging of some new messages.

ECA-Driven Brain-Tumor Rule: A holistic object-oriented based specification of the ECA-Drive rule given in Figure 3 takes the form:

```
omod DECISION_BRAIN_TUMOR is
```

```
protecting REAL, INT, STRING
class P_INTF | TumorStatus : INT , Hospital: INT, FinalState:Char .
class D_INTF | NAME : STRING , POSITION: INT .
class S_INTF | ANTEC : INT , Seizure: INT, BalanceLoss:INT .
msgs  ToCheck  Get_Patient: OId  Info → Msg .
msg  Get_Sympt(_,_) : OId  OId → Msg .
vars P  D  S : OId , S, B: Int, Info:STRING .
  ********* The ECA-Driven Non-ML Tumor behaviour.*******
rl[ToCheck] ToCheck(P,Info)
  ⟨P : |TumorStatus : 0, Age : A, HOSP : H, ST : S⟩
  ⟨S : |ANT : T, Seizure : S, BalanceLoss : B⟩
    ⇒ ⟨P : |TumorStatus: 1, Age : A, Hospital : 0, FinalStatus : nil⟩
    ⟨S : |ANT : T, Seizure : S, BalanceLoss : B, Hospital : 2, FinalStatus : "T"⟩
      if (A≥62 and T≥ 2 and S≥ 1 and B≥ 1) .
endom
```

With such rules, we can apply them on specific configuration of service interface instances. Due to space limitation and also not to overwhelm the reader with such details, we orient the interested reader to similar work on such configurations [11].

7 Phase V: A Django REST-centered $\mathcal{AR}$ECA4AI$\mathcal{SA}$-compliant Modern and Efficient Implementation Architecture

Fig. 7. $\mathcal{AR}$ECA4AI$\mathcal{SA}$—**Phase V.**: The Patient and KNN Django Web-Services with their respective DJANGO-REST-Methods

For the efficient yet *fully ECA-driven compliant* and modern implementation of the $\mathcal{AR}$ECA4AI$\mathcal{SA}$ approach, we have opted at-a-first-stance for the widely accepted Django-REST Web-Service Framework based environment [32], which is also Python-conform (with its plethoria of ML-based models and algorithms). For the IDE we have opted for Visual Studio Enterprise [33] as it harmoniously encompasses Django (in contrast to other complex IDEs such as PyCharm).

That is, with the endeavor for a modern implementation while staying ECA-driven compliant with the approach conceptual-level and after tremendous programming effort, we have ended up with a four-level architecture as depicted in Figure 6. The first deep level concerns the involved ML-based and "ordinary" Web-Services as we already sufficiently motivated and explained , that is, for our Brain-Tumor case, we have: The Patient, the Symptoms (General and Specific), the Radio-Image Web-Service, the KNN Web-Service and associated DataSets and optionally the Doctors and the AI-Experts ones. The second level, we referred as to API Client, turned out to be necessary to transform usual complicated JSON format received from these Web-Services into a workable understandable *Object-Oriented* model. The third-level concerns the required interfaces (i.e. properties and operations) required for the conceptualization of the associated ML-based and ordinary ECA-driven architectural inter-service interaction laws. Finally, the fourth-level consists in mapping the conceptualized and formally validated ECA-driven (ML-based and non-ML) interaction lawss into Python-based programmed rules.

8 Conclusions

We put forwards an innovative model-driven unified stepwise and highly disciplined approach for rigorously developing AI-Powered adaptive knowledge-intensive software applications. The approach is architecturally service-oriented by essence and ECA-driven covering the holistic software development life-cycle, namely: (1.a) the semi-formal -graphical UML-based elicitation and modelling phase for coping all structural-features straightforwardly mapped into intrinsic Web-Services and this being it ordinary or ML-centric; (1.b) behavioural features are firstly intuitively on their-side captured as (event-driven) business rules mostly at the inter-service communication level; (2) The precise graphically appealing architectural ECA-driven inter-service conceptualization of the informal domain-based intuitive business rules; (3-a) The equivalent precise ECA-driven architectural conceptualization, by the AI-Experts, of the to-be involved ML-centric models and algorithms ECA-driven rules and (3-b) the exploiting of the resulting from (3-a) in terms of predication and accuracy in AI-powering Domain-Experts decisions; (4) The Rewriting-logic and MAUDE-based formalization and validation of both ordinary and ML-based ECA-driven architectural interaction rules; finally (5) the ECA-compliant efficient implementation of the approach on the basis of the DJANGO REST-framework and its sibling Python API Requests framework; all in all development using the model visual studio enterprise IDE and extensively illustrated and validated on the basic of quite real-size case study coping the Brain-Tumor case-study.

As the authors are not aware of any similar research approaches addressing the conceptualisation of AI-Powered knowledge-intensive software applications on the basis of ECA-driven principles at the interaction *architectural-level*, it is relevant to mention some work integrating business rules with Machine-Learning models. First wort- mentioning reseach appeared in [34], where the authors aim at extracting *rules patterns* from any DataSets to improve and more specifically optimize the ML-model resulting outputs. Other work such as [35] or [36] address specific applications by combining business rules and machine-learning. Finally, a successful Rule-Engine for general purposes, called OpenRules [37], is currently

successfully extended to cope with Machine-Learning Models and their optimisations in an elegant manner.

As already planned future investigations, since we are quite confident that this promising research is opening up variety of related research directions by pushing resp. enriching the presented approach milestones further in different directions, that may include: (1) Further case studies for validating the approach at the implementation and efficiency level are required, in this sense we are particularly planning to focus on the "prompting" AI-generative side in the same manner as we did here for the ML-based models; (2) following our previous work on architectural techniques [38] , the forwarded tailoring of Meseguer's Rewriting-Logic could be extended to cope, on the one hand, with strategies [39], so that an optimal guided processing and coordination between the Domain-Experts and the AI-Experts ECA-driven Rules could be achieved. Work on strategies could be found in our previous work in [11]. On the other hand, since MAUDE does not allow direct interconnections to Web-Services, we are planning to benefits from the four-level achieved implementation and generate from the API clients level direct object-oriented (sql-server based) configurations for large-scale prototyping. Last but not least, verification based on model-checking [40] allowed by previous work on MAUDE could also very beneficial before any efficient service-oriented implementation. Finally, to validate the approach on different ML-Models as well on different application domains, further investigations of other implementations tools such ML.NET from Microsoft as well as more experiments especially with deep-learning and other generative AI are needed.

Acknowledgment

The authors would like to thank the paper reviewers for pointing out on some shortcomings that contributed improving the current final version by partially tackling them.

References

1. O. Caelen and M. Blete, *Developing Apps with GPT-4 and ChatGPT.* O'Reilly Publishing, 2024.
2. F. C. Romero J., Inmaculada M., Ed., *Optimising the Software Development Process with Artificial Intelligence.* https://link.springer.com/book/10.1007/978-981-19-9948-2: Springer Nature Link, 2023.
3. S. Navdeep, Ed., *Optimising the Software Development Process with Artificial Intelligence.* BPB Publications, 2023.
4. P. Harrington, *Machine Learning in Action.* Manning Publications, 2012.
5. K. Huang, " Agentic AI: Theories and Practices," vol. 9(5), pp. 1–9, 2025.
6. I.Sommerville, Ed., *Software-Engineering (10th Edition).* Pearson, March 2025.
7. OMG, "UML 2.0: Superstructure Specification. Version 2.0, formal/05-07-04," omg.org, Tech. Rep., 2005.
8. D. Barry, Ed., *Web Services, Service-Oriented Architectures, and Cloud Computing.* Morgan Kaufmann, 2013.
9. N. Richards, M. Ford, Ed., *Fundamentals of Software Architecture: An Engineering Approach.* OREILLY, March 2025.
10. K. Magee and J. Kramer, "Dynamic Structure in Software Architectures," in *Proc. of 4th Symp. on Foundations of Software Engineering.* ACM Press, 1996, pp. 3–14.
11. N. Aoumeur, K. Barkaoui, and G. Saake, "Validating, Composing and Dynamically Adapting Features in Concurrent Product-Lines Applications," in *Proc. of 16th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS'09).* IEEE Computer Society Press, 2009, to appear.

12. J. Meseguer, "Conditional rewriting logic as a unified model for concurrency," *Theoretical Computer Science*, vol. 96, pp. 73–155, 1992.
13. Y. Liu, *Python Machine Learning by Example.* Packt Publishing, 2012.
14. C. Mueller and S. Guido, *Introduction to Machine Learning with Python.* O'Reilly Media, 2017.
15. P. Bhong, "Brain Tumor Detection : CNN Based Brain Tumor Detection for MRI Scan," *International Journal of Advanced Technology and Engineering Exploration*, vol. 9(5), pp. 1–9, 2025.
16. J. Jung, J. Park, S. Han, and L. Lee, "An ECA-based framework for decentralized coordination of ubiquitous web services," *Information Software Technology*, vol. 49, no. 11-12, pp. 1141–1161, 2007.
17. N. Aoumeur and K. Barkaoui, "Pervasive systems development: A stepwise rule-centric rigorous service-oriented architectural approach," *Computer Science & Information Technology*, vol. 12, no. 1, 2023.
18. N. Aoumeur, J. Fiadeiro, and C. Oliveira, "Distribution Concerns in Service-Oriented Modelling," *International Journal of Internet Protocol Technology (IJPT)*, vol. 1, no. 3, pp. 144–158, 2006.
19. L. Andrade and J. Fiadeiro, "Architecture Based Evolution of Software Systems," ser. Lecture Notes in Computer Science, M. Bernardo and P. Inverardi, Eds., vol. 2804. Springer, 2003, pp. 148–181.
20. L. Andrade, J. Fiadeiro, J. Gouveia1, and G. Koutsoukos, "Separating computation, coordination and configuration," 2002.
21. N. Aoumeur, K. Barkaoui, and G. Saake, "A Multi-Dimensional Architectural Approach to Behavior-Intensive Adaptive Pervasive Applications," in *Proc. of 4th International Symposium on Wireless Pervasive Computing (ISWPC'09).* IEEE Computer Society Press, 2009, to appear.
22. N. Marti-Oliet and J. Meseguer, "Rewriting logic as a logical and semantic framework," in *Proc. of First International Workshop on Rewriting Logic*, ser. Electronic Notes in Theoretical Computer Science, J. Meseguer, Ed., vol. 4, 1996, pp. 189–224.
23. A. Mele, *Django 3 by Example: Build powerful and reliable Python Web Applications from Scratch.* Packt Publishing, 2020.
24. Malathi, M. Sinthia, P., "Brain Tumour Segmentation Using Convolutional Neural Network with Tensor Flow," *Asian Pacific Journal of Cancer Prevention*, vol. 20(7), pp. 2095–2101, 2019.
25. Philipsy, E. Radhakrishnan, B., "Brain Tumor Detection Using Standard Deviation and Area," *IJCSN - International Journal of Computer Science and Network*, vol. 8(3), pp. 2277–5420, 2019.
26. P. Majumdar, *Mastering Classification Algorithms for Machine Learning.* MDPI Publications, 2023.
27. B. R. Group, "Defining Business Rules - What Are They Really?" in *www.businessrulesgroup.org*, 2005.
28. P. Kardasis and P. Loucopoulos, "Expressing and Organising Business Rules," *Information and Software Technology*, 2004.
29. J. M. M. K. A. Chen, T. Carter, Ed., *Artificial Intelligence in Healthcare: Recent Applications and Developments.* Springer, 2023.
30. M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and C. Talcott, "All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic," *Lecture Notes in Computer Science (springer)*, vol. 4350, 2007.
31. N. Aoumeur, *On the Stepwise and Disciplined Engineering of Adaptive Service-oriented Applications .* Shaker-Verlag, 2018.
32. C. Stein, *Learning Django 5: Program web apps with complete Django capabilities, ORM, Template, MongoDB and Docker containerization.* Asian Publishing House, 2024.
33. O. J. d. Perrez, Ed., *Visual Studio 2022 In-Depth: Explore the Fantastic Features of Visual Studio 2022 - 2nd Edition.* BPB Publications, 2022.
34. J. Feldman and I. Graham, "Integrating business rules and machine learning technologies," 2021.
35. K. Neelima and S. Vasundra, "Machine learning-based business rule engine data transformation over high-speed networks," *Computer Assisted Methods in Engineering and Science*, vol. 30, no. 1, 2022.
36. B. Chris, F. Stephane, G. David, and al., "Machine learning with business rules on ibm z:acting on your insights," 2018.

37. O. R. Developers, “Decision modeling with business rules, machine learning, and optimization (openrulesdecisionmanager.com),” 2024.
38. N. Aoumeur, *On the Stepwise and Disciplined Engineering of Adaptive Service-oriented Applications*. Shaker-Verlag Publisher (ISBN: 978-3-8440-6054-6), 2018.
39. E. Steven, M.-O. Narciso, M. Jose, and al., “The maude strategy language,” *Journal of Logical and Algebraic Methods in Programming*, vol. 134, 2023. [Online]. Available: https://arxiv.org/abs/2402.00275
40. R. Ruben, M.-O. Narciso, and al., “Strategies, model checking and branching-time properties in maude,” *Journal of Logic and Algebraic Methods Programming*, vol. 123, 2021. [Online]. Available: https://arxiv.org/abs/2401.07680