Georgian Scientists
ქართველი მეცნიერები
Vol. 7 Issue 1, 2025
https://doi.org/10.52340/gs.2025.07.01.48

The development of intelligent case management systems

Zurab Bosikashvili[1], Lolita Bejanishvili[2]

Faculty of Informatics and Control System, Georgian Technical University,77, Kostava Str., 0175, Tbilisi Georgia (Tbilisi), GEORGIA

[1]z.bosikashvili@gtu.ge, orcid.org 0009-0001-8222-2875

[2]lolita.bejanihvili@gtu.ge, orcid.org 0000-0002-4692-0932

Abstract

At the modern stage of development, the processes of document circulation and case management are well studied, and many software systems are successfully used to manage these processes. On the other hand, we are in the technological era of artificial intelligence, which opens up new opportunities for the developing existing information systems. The paper analyzes the weaknesses of modern case management systems and proposes ways to solve them using artificial intelligence methods. While case management processes are shaped by legislation and regulations, ambiguities in their interpretations, their incompleteness, and the challenges of analyzing related documents make it difficult to establish rigid case management schemes regularly. To achieve business goals, it is essential to implement dynamic schemes instead. Consequently, this paper presents the case management process using mixed models. Some aspects of the process are described as fixed workflows, while others are represented as dynamic, configurable processes that utilize established subprocesses, which require planned resolution.

Current case management systems generate valuable knowledge and experience when handling real data, enabling the automation of select processes and decision-making. However, such functionality is often lacking in existing systems.

To address this issue, the paper discusses the expansion of the case management system with a unified intellectual module engine, which includes a knowledge base, mechanisms for working with rules, facts, drawing conclusions, machine learning, planning, and decision-making.

The paper presents a formal meta-model of a case management system, which is designed as a transformation of case components—objects and initial states—into a target state through the selection of sequences of case management operations.

In this context, the state is defined as a set of object attributes and their corresponding values, represented as facts within a knowledge base. The operations are detailed in the knowledge base in the form of rules. When a new situation arises in the case management process, it becomes necessary to re-plan the process. The model formulates the task of selecting the optimal sequence of actions, and the author proposes an algorithm to address this challenge.

The architecture of the intelligent case management system has been developed based on the presented formalism, and its core components have been implemented. The system includes several key architectural components:

1. Case Lifecycle Management Module: This module handles the creation, updating, and closure of cases.

2. Hybrid Intelligent Module: Responsible for managing the knowledge base, this module incorporates machine learning, dynamic planning, and decision-making processes.

3. Validation Module: This component ensures data consistency and accuracy through a combination of static, dynamic, and AI-enhanced validation rules.

4. Configuration Module: This module manages the system's operational functions.

The system developed based on this architecture has been implemented in both government and private sector organizations across various fields.

Key Words: - case management, workflow, case handling, artificial intelligence, inference, rule, planning

1 Introduction

There are various fundamental problems when designing case management systems, which restrict flexibility of the system and prevents its further development. These problems get much more complex, when using Workflow principles in the design of case management systems. Here are some of them:

- In any information system, including Case Management Systems, object structure that is used inside the system can be very complex. These complexity implies multi-attribute and multi-dimension objects. Moreover, set of attributes can be defined dynamically and in particular cases, some of them should be validated & some of them not, while in other cases, all of them or none of them should be validated and so on.

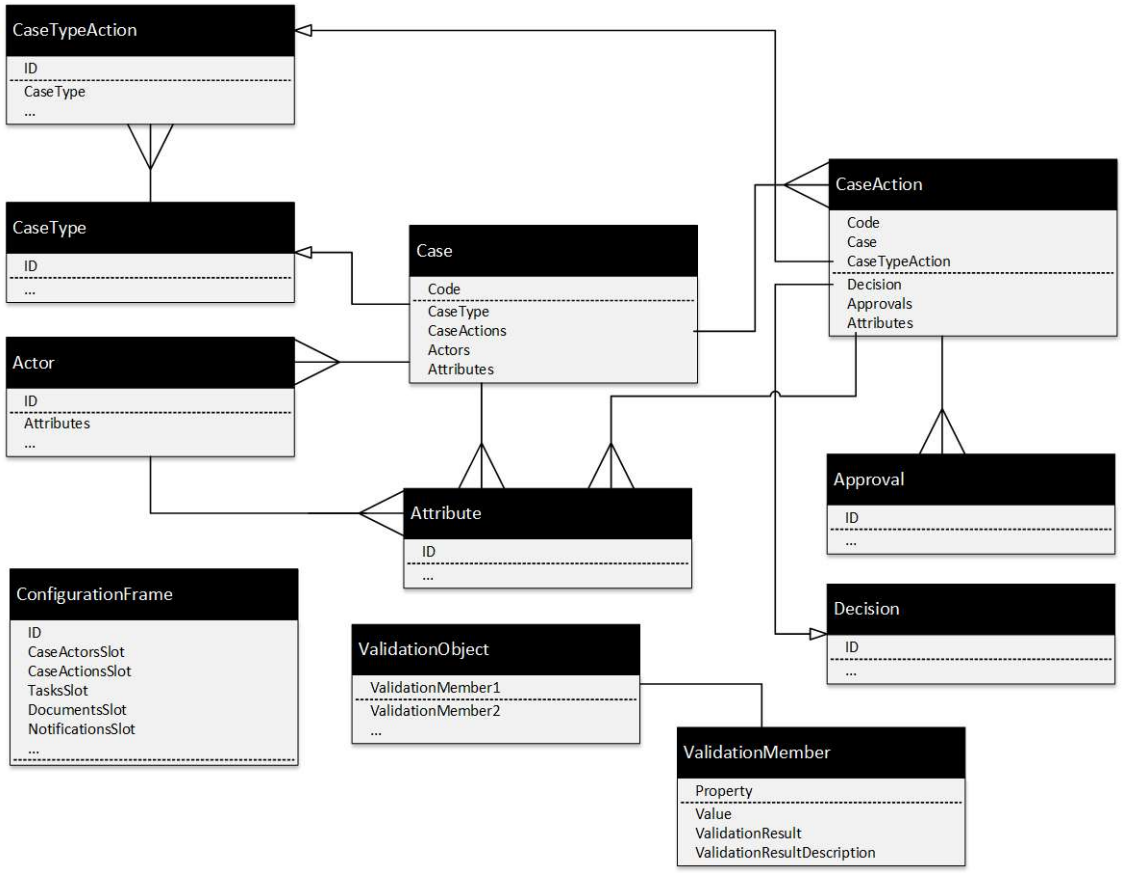
- Due to data structure complexity, there is a validation problem. In addition to “hard”, predefined validation, when validation terms are predefined, more and more often we need handle situations,

when attribute validation logic is depended not only on it's value & validation rule, but also on other attribute values and validation results. As a result, it's

- Impossible to define and "hardcode" all validation rules and we need ability, to have so called "dynamic validation" when validation process combines several rules and several attributes and their validation results.

- Biggest problem of workflow-based case management systems is the limits that workflow puts on the system. Workflow predefines not only all available states in which particular case can be, but also limits ways, by which case can move from one state to other. Because of this limitation, very often its almost impossible to handle non-standard situations.

Figure 1. Case management system class diagram



- Next problem that we have in workflow-based systems, is dependency on rules. Different values of same attributes of same object as an input, can produce different results on output of the rule. To say differently, not only the rules, but also objects attribute values itself, can give information what is next target state of object and more over, what are the ways of reaching that state. But standard principles of workflow systems, does not give us ability, to use data in combination with predefined rules to get next states and ways of reaching that states.

- When changing states of objects or creating new ones, sometimes there is need to generate some new, linked objects (e.g.: changing case state from A to B implies generation of notifications and

some document liabilities). The difficulty is that to achieve this goal, system has to analyze current object with it's attributes and state and make a decision what new linked objects should be generated.

- In working systems, as real data is processed, some knowledge and experience is generated. This newly generated knowledge gives us ability, to automate some processes and decision making. In current systems we can hardly find such functionality.

Very often, we need to auto-combine configuration or validation rules to get a new one, which will be discussed as regular rule an which will be used by system automatically. Also, there is need to use several rules and using transit dependency between them get new state or decision

2 The meta model of Intellectual case management systems

To understand case management system conceptual model, let's describe so called "meta-model" [6]. In this model, we can see main objects of the system.

All of the objects have necessary properties and further expansion is done via attribute universal model which gives us an ability, to expand any object with any combination and type of attributes. For illustration purposes, diagram shoes very simplified and minimal version of real base object model of the system (Fig.1).

System configuration defines *CaseType* object. Using it, one can describe any desired type of the case configuration and also all available operations (*CaseTypeActions*) that are supported by case type. During actual case management process, *Case* object is created for real life case which is depended on *CaseType*.

Case object contains all information about actual case, all current and past operations that where executed on the case (*CaseActions*), actors who are involved in case (*Actors*), and attributes that are used in current case (*Attributes*).

CaseAction represents particular operation information and data that is/was executed on the case. It consists base data, approval data (*Approvals*), decision data (*Decision*) and object attributes (*Attributes*).

Actor object describes concrete actor, who is taking part in case and consists of the info of the actor. Like other objects, it's also extended using object attribute technology (*Attributes*).

But most of all, we are interested in validation and configuration objects: *ValidationObject* & *ConfigurationFrame.* Validation object represents a collection of flat structured objects, where all case related information is projected. Each member of this collection contains information about attribute, name, type, value also it contains validation result and description properties which are filled during validation process and contains succeeds or failure detailed description.

Configuration is done using *ConfigurationFrame* object. This is universal object, which is filled using case current information and provides all necessary info during every operation. It contains available operation information, for case current state, also all the objects that are requested to be generated before or after operation is executed. This functionality is provided using "Slot" mechanism which means that *ConfigurationFrame* represents typed collection of slots and each slot is defined beforehand and return necessary objects when needed.

ConfigurationFrame and *ValidationObject* are those "middleware" objects that connect intellectual module and case management system with each other.

1.1.1. 2.1 The intellectualization of case management systems

Effective solution of all problems, mentioned above, is integration of intellectual engine into the case management systems. Using intellectual engine does not mean radically changing of existing case management processes or excluding workflow principles. In opposite, using intellectual engine gives us ability to use more effectively existing case management & workflow processes and solve problems that exist in most of modern solutions.

Using intellectual engine, makes case management systems much more flexible & configurable. On the one hand, system still suggests users with recommended actions and decisions, on the other hand, we are totally removing "hardcoded" configuration problems and now, system can effectively analyse not only the preconfigured rules, but also existing data and make decisions depending on all this information. Also it handles non-standard situations more flexible. Object structures, states and available ways of going from one state to another are not "tied" to roles any more. With accumulation of actual data, each end every case can be treated individually, without reconfiguring system or adding "hardcoded" rules.

3 A Formal Model of Case Handling

In this section, we will specify the dynamics using a formal model. First, we introduce a formal model describing a case definition. In this model, we abstract from certain entities (e.g., forms) and focus on activities and data objects. Based on this formal model, we describe the execution model for case handling in terms of state-transition diagrams and ECA-rules. Finally, we discuss the relation between the formal model and the entities excluded from the formal model, e.g., forms and actors.

Definition 1. Case definition is a tuple

$$CD = <A, P, O, Q, U, S, F> (1)$$

where

A is a set of *activities definitions*,

$P \subseteq A \times A$ is a *precedence relation* and is *acyclic*,

O is a set of *data object definitions*,
Q is a set of characteristics (*attributes),*
U is a set of *attributes values(domain),*

S is a set of *data objects states* $S = \prod_{i=1}^{n} Si$, *where S*i is set of $o_i \in O$ *data object states.*
F is set of functions $\{f_i \mid f_i: A \rightarrow \Re(S)\}$, which map each *activity* onto $\Re(S)$,($\Re(x)$- denotes the set of all subsets of x). The mappings between this sets is depicted in fig.2.

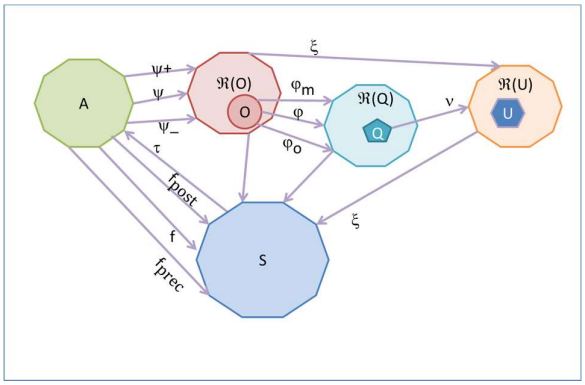


Figure 2. Case data sets transformation diagram.

ψ: $A \rightarrow \Re(O)$, maps each activity to set of joined data objects before start of the activity. ψ_+, ψ_-, specifies a set added and deleted data objects after activity.

φ: $O \rightarrow \Re(Q)$ maps each data object to set of its features(attributes). φ_m, φ_o – is a functions, that defines mandatory and optionally attributes and holds next condition: ∀a ∈ A and ∀o∈ $O_{1=} \psi(a)$, exists φ and q such $O_1 \subseteq \varphi^{-1}(q)$.

ν: $Q \rightarrow \Re(U)$ is a function mapping each data object onto its *domain.*

Definition 2. ∀a ∈ A and ∀o∈ $O_{a=} \psi(a)$, $S_a = \prod_{o \in Oa} \bigcup_{q \in \varphi(o)} \nu(q) \in A$ is a *state* built by values of joined ***a*** *data objects.*

It's clearly, that case state $S = \prod_{a \in A} Sa$.

While activities are an important artefact in case handling, the case is mainly controlled on the basis of states of data objects, associated with the particular case. It is important to stress that not only the life-cycle of activities can be described by states and state transitions, but also data objects. To see this, consider the state transitions that data objects may take as shown in Figure 3. On the creation of a data object, it adopts the undefined state. Data objects can be defined, either by users filling in forms which represent these data, or they can be defined automatically, for example, by running queries against a database and transferring the result values to the data objects. Activities for which data objects are mandatory can be redone (cf. the redo role), which results in a state transition of data objects to the unconfirmed state. By confirming the values, data objects re-enter the defined state.

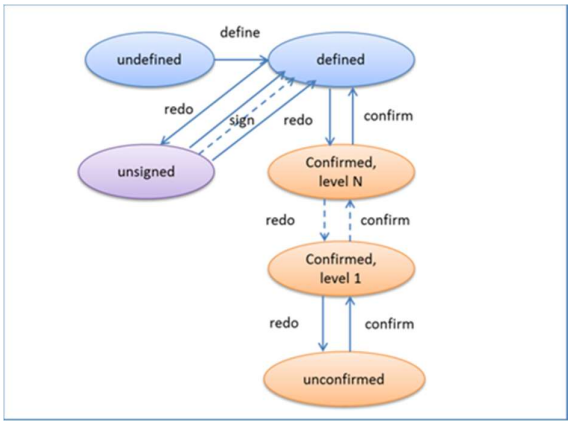


Figure 3: States of data objects.

Definition 3. Denote by o_s special data object and call it by state data object with attribute domain *Uo_s={undefined, defined, unsigned, unconfirmed, confirmed level 1, …, confirmed level n}.* Then

$\psi: A \rightarrow \Re(O \cup o_s)$ and $\nu: Q \rightarrow \Re(U \cup Uo_s)$.

It is useful to define terms describing the relative order of activities within the context of a given case definition. Given a case definition CD, for each activity $a \in A$

- $A_{preceding}(a) = \{a' \in A \mid (a\_, a) \in P+\}$, and
- $A_{subsequent}(a) = \{a' \in A \mid (a, a') \in P+\}$.

where $P^+ = U_{i>0} P^i$ is the non-reflexive transitive closure of P. Case handling systems make use of case definitions to guide users in handling cases. In order to do that, the system has to make sure that a given activity is flagged ready for execution if and only if the preconditions of that activity are met. To be able to specify ifan activity should be executed or bypassed, we use the following auxiliary function.

A function $f_{prec}: A \rightarrow \Re(S)$ defines a the set of precondition states, $f_{post}: A \rightarrow \Re(S)$ defines a the set of postcondition states.

Functions f_{prec} and f_{post} only focus on the data state $s \in S$. Clearly, the data state is not sufficient to determine the dynamics, also the activity state $a \in A$, the causal relations specified by P, and the state-transition diagrams shown in Figures 4. To specify the semantics of case handling we augment the state transitions with rules specified using an Event Condition and proposed in the paper[1-2]. Each state transition is described by a rule of the following form: **IF** *condition*, **THEN** *action.* Write these rules by the presented formalism proposed in the [2].

$$A_1, A_2, \dots, A_n \xrightarrow{R} D(C[\alpha_1, \alpha_2, .., \alpha_m]) \quad (5)$$

where $A_1, A_2, .., A_n$ - elementary or compound terms or even logical expressions. R - is a relation (is A, member Of, part Of, etc.), according to which are performed proofs and it may be satisfy the reflexive, transitive or commutative conditions, that is fixed in the dictionary. D-is a domain, *C-is a*

category of the conclusion which belongs to D. $\alpha_1, \alpha_2,.., \alpha_m$ – are terms which express the values of the attributes and relation of the category *C.*

For the Municipality Management system, social services is defined by rules, some of them have the form:

- *if the statemnt ends or the time has expired and the operation of this operation has been completed with positive decision, then begin the operation of a case study.*

  R1: Decision(@previousActivity(),"Positive") & (State(:y,"Completed") ⊕ State(:y,"Ready")
  →Case{Case(:x), StatementActivity(:y), StudyActivity (:z),State(:z,"Ready")

- *it's possible, that user can carry out second level confirmation, if the first level confirmation is completed and the user has an according role and there is a visa on the document.*

  R2: Role(:x,"Conformer")&Conformed(:y,"Level1") &Visa(:y,"Yes")
  →Case{ Person(:x),Document(:y), Conformed(:y,"Level2")}

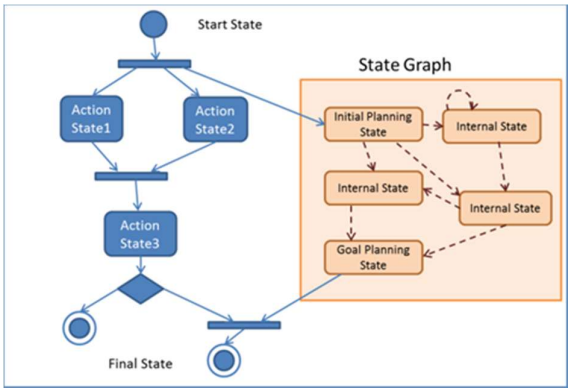
- *An user is participant in the council meeting, if hi is council chairman, hi is or he is a council member, or he is a invited guests.*
  R3: Role(:x," Council Chairman") ⊕ Role(:x," Council member") ⊕ Role(:x," Invited Guests")

  →CouncilOrgan { :y, User (:x), Participant(:x,:y,"Yes")}

3.1 Planning problem

In many cases, the case management process can be presented via mixed models. Part of the process can be described by the fixed workflow and other part can be presented by the case handling model (Fig.4).

In second model, often we have a situation, when we must find the path of activities from some initial state to the goal state (Fig.5). This is a classical plannin problem of the artificial intelligence [5]. let consider the formal model.



Fugure 4. Case management mixed model.

Definition 1. *Planning subproblem* of case handling is a tuple

$$CP = <s_0, S_p, F_p, G > (2)$$

where

$s_0 \in S$ is a initial state,

$S_p \subset S$ is a subset of a set of case data object states,

$F_p \subset A$ is a subset of a set activities,

$G \subset S$ is a goal subset of a set of case data object states.

A planning problem is determined as finding the optimal sequence $f_1,.., f_{n-1}$, that $f_n \circ f_{n-1} \circ .. \circ f_1(s_0) \in G$.

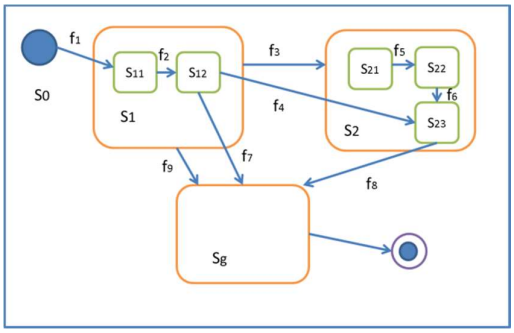


Figure 5. Data state graph.

For a planning problem solution exists many methods[6,8]. The choice of specific methods is out of scope of this paper. We have developed the heuristic search algorithms in the factor state space [4].

4 The realization of the Intellectual case management system

The realization of an intellectual case management system (ICMS) requires a structured approach that ensures flexibility, dynamic decision-making, and adaptability to various business scenarios. This section outlines the key steps and architectural considerations essential for developing such a system, incorporating modern AI advancements.

4.1 System Architecture and Core Components

The ICMS is designed as an extension of traditional case management systems by integrating an intellectual module that utilizes rule-based reasoning, heuristic planning, and modern AI techniques such as machine learning (ML) and natural language processing (NLP) [1]. The core architectural components include:

- **Case Management Core (CMC):** Manages the fundamental case lifecycle, including creation, updates, and closure.
- **Intellectual Module (IM):** Contains the rules engine, planning algorithm components, and AI models responsible for dynamic decision-making and adaptive process flow.
- **Validation Layer:** Ensures data consistency and accuracy through both static, dynamic, and AI-enhanced validation rules.
- **Configuration Layer:** Uses ConfigurationFrame objects to determine the necessary operations and data transformations based on current case states.

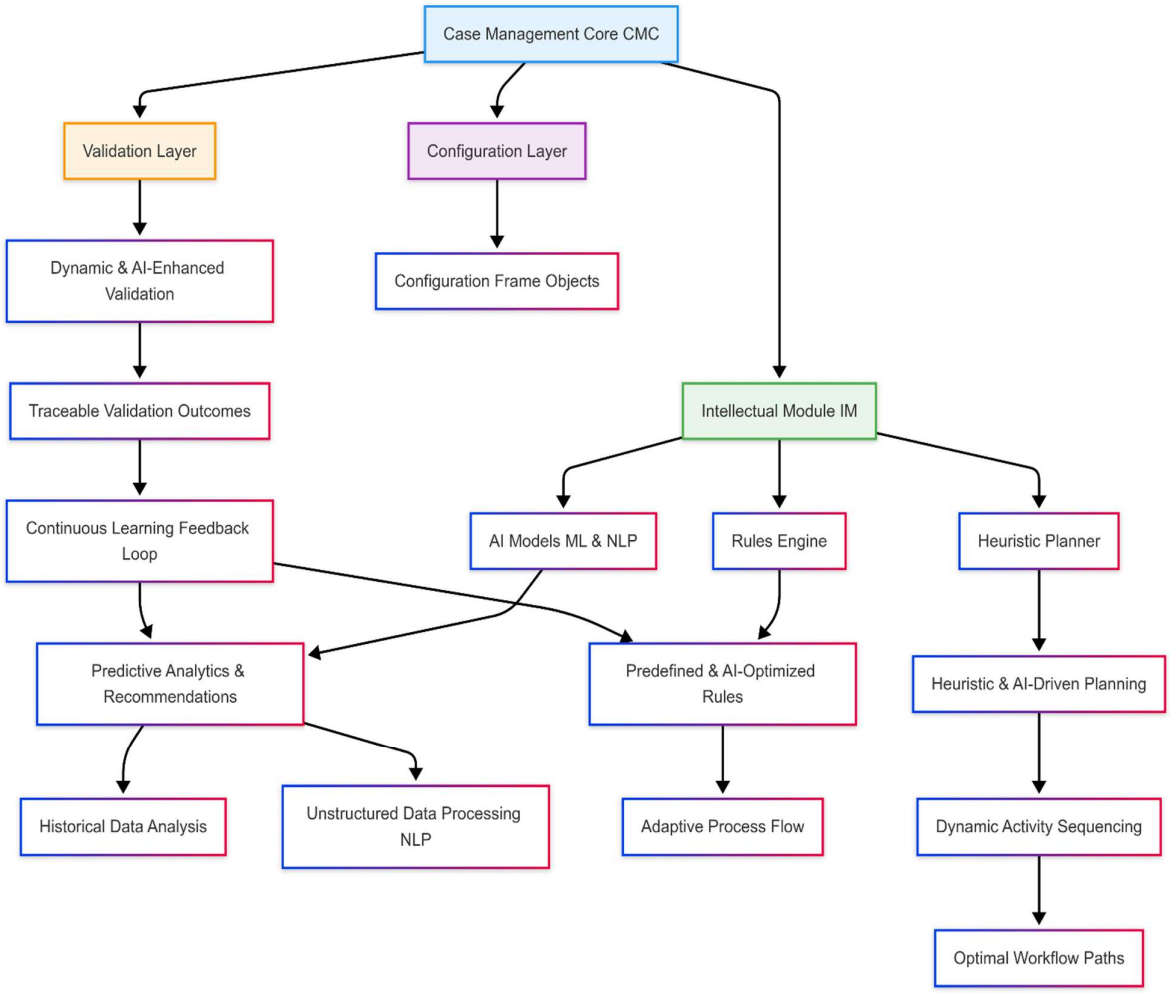


Figure 6. Intellectual Case Management System

4.2 Integration of the Intellectual Module

The IM operates in conjunction with the CMC and leverages accumulated knowledge for intelligent decision-making. Key aspects of integration include:

- **Rules Engine:** Utilizes predefined, dynamically generated, and AI-optimized rules to validate inputs, determine process flows, and make decisions based on case data attributes.
- **Heuristic Planner:** Engages in solving planning subproblems, particularly in cases involving dynamic paths from initial to goal states. The planner leverages heuristic algorithms and AI-driven optimization techniques.
- **AI Models:** Incorporates ML algorithms to analyze historical case data, predict outcomes, and suggest optimal workflows. NLP capabilities allow the system to understand and process unstructured text data for enhanced decision-making.
- **Middleware Interfaces:** ConfigurationFrame and ValidationObject act as mediators between the IM and CMC, facilitating seamless communication and data exchange.

4.3 Realization of Dynamic Validation

Dynamic validation is achieved by:

- Allowing attributes to be validated based on interdependent values, results, and AI-driven insights.
- Using the IM to dynamically construct and adjust validation logic as cases evolve, leveraging ML to learn from previous validation outcomes.
- Recording validation outcomes within the ValidationObject to maintain traceability and enable continuous learning.

4.4 Planning and Decision-Making Implementation

The planning component addresses the need for dynamic activity sequencing. The implementation involves:

- Defining initial and goal states, along with a subset of activities and transitions.
- Applying heuristic and AI-enhanced algorithms to navigate the state space and determine optimal activity sequences.
- Using historical data and reinforcement learning techniques to refine planning algorithms, ensuring efficiency and relevance over time.
- Incorporating AI-based recommendation systems to suggest next steps based on case context and historical patterns.

4.5 Handling Non-Standard Situations

The ICMS excels in addressing non-standard scenarios by:

- Utilizing the AI-enhanced rules engine to identify exceptions, learn from novel situations, and recommend adaptive solutions.
- Allowing the planner to explore alternate paths using AI-driven predictive models when traditional workflows are insufficient.
- Facilitating continuous learning where the system adapts based on newly acquired data and outcomes, using unsupervised learning to detect emerging patterns.

4.6 Implementation Considerations

Key considerations for successful realization include:

- Ensuring modularity for ease of maintenance, scalability, and integration of evolving AI models.
- Prioritizing user-centric design to ensure intuitive interaction with adaptive processes, incorporating AI-powered assistance for better user experience.
- Incorporating robust security measures to safeguard sensitive case data, with AI models to detect and respond to potential security threats.
- Establishing feedback loops and continuous training pipelines for AI models to improve the accuracy and effectiveness of rules and planning strategies over time.

By leveraging modern AI achievements, the realization of an intellectual case management system becomes feasible, offering enhanced flexibility, efficiency, and adaptability to complex, dynamic business environments. This approach ensures the system remains future-proof and capable of handling evolving challenges.

5 Conclusion

In the beginning we defined main problems, wich can be faced when developing and using case management systems with standard schemes. Also we raised problems which are found when implementing case management over workflow. Most natural and flexible way to solve this problems is adding new component, intellectual module in the existing system architecture. After that, all non-trivial business logic, validation, object generation and process flow control will be orchestra via this module. This approach not abandons existing scheme, rather extends it, adds more flexibility and configurability, removes existing problems and handles non-standard situations more naturally. As a result, system users are concentrated on their main tasks rather than avoidance of limitations that are set by existing solutions. Systems that are built on described architecture are successfully implemented

in several government and private sector companies and successfully what reinforces our belief that future development of case management systems is laying on intellectual modules.

References:

[1] Bosikashvili Z., Kvartskhava G., [2024],Using Hybrid Models of AI for Identification of Trees by UAV Images of Forests: I. Machine-learning Component of the Models, WSEAS Transactions on Signal Processing, 20, 39-53, https://doi.org/10.37394/232014.2024.20.5

[2] Bosikashvili Z., [2014], Extansion of the Architecture of Software Systems with Artificial Intelligence Elements, Modern Computer Applications in Science and Education, Editor Constantin Buzatu,75-84, ISBN:978-960-474-363-6

[3] Wang, P. [2013] "NON-AXIOMATIC LOGIC A Model of Intelligent Reasoning," Copyright © 2013 by World Scientific Publishing Co. Pte. Ltd,2013

[4] Bosikashvili Z., [2010], **The blocking meta-heuristics for combinatorial problems solving,** The ACM Digital Library, World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA.

[5] Russell, S. and Norvig, P. [2010] Artificial Intelligence: A Modern Approach, 3rdedn. (Prentice Hall, Upper Saddle River, New Jersey).

[6] Steven M. LaValle. [2006]. Planning Algorithms. Published by Cambridge University Press

[7] Wil M.P, Pallas Athena, [2002] Case Handling: A New Paradigm for Business Process Support, http://wwwis.win.tue.nl/~wvdaalst/publications/p252.pdf

[8] Nilsson, N. J. [1991] "Logic and artificial intelligence," Artificial Intelligence **47**, 31–56.

საქმეთა მართვის ინტელექტუალური სისტემების აგების შესახებ

ზურაბ ბოსიკაშვილი, ლოლიტა ბეჟანიშვილი

რეზიუმე

განვითარების თანამედროვე ეტაპზე, დოკუმენტბრუნვის და საქმეთა წარმოების პროცესები კარგად არის შესწავლილი და არსებობს მრავალი პროგრამული სისტემა, რომლებიც წარმატებით გამოიყენება ამ პროცესების სამართავად. მეორეს მხრივ ჩვენ ვიმყოფებით ხელოვნური ინტელექტის ტექნოლოგიურ ეპოქაში, რომელიც ახალ შესაძლებლობებს აჩენს, არსებული ინფორმაციულის სისტემების განსავითარებლად. ნაშრომში გააანალიზებულია თანამედროვე საქმეთა წარმოების სისტემების სუსტი მხარეები და შემოთავაზებულია მათი გადაწყვეტის გზები ხელოვნური ინტელექტის მეთოდების გამოყენებით. თუმცა საქმეთა წარმოების პროცესებს კანონმდებლობა და რეგულაციები განსაზრვრავენ, მაგრამ მათი არაცალსახა ინტერპრეტაციების, არასრულობის და პროცესში შემოტანილი დოკუმენტების კონტენტის ანალიზის სირთული გამო შეუძლებელია ყოველთვია საქმეთა მართვის ხისტი სქემების შექმნა. აუცილებელია დინამიური სქემების შემოტანა ბიზნეს მიზნის მისაღწევად.

აქედან გამოდინარე, ნაშრომში საქმეთა მართვის პროცესი წარმოდგინდება შერეული მოდელების სახით. პროცესის ნაწილი აღიწერება ფიქსირებული სამუშაო პროცესის სახით, ხოლო ნაწილი დინმიური აწყობადი პროცესების სახით ფიქსირებული ქვეპროცესების გამოყენებით, რომელთა გადაწყვეტაც უნდა დაიგეგმოს. არსებულ საქმეთა წარმოების სისტემებში რეალური მონაცემების დამუშავებისას წარმოიქმნება გარკვეული ცოდნა და გამოცდილება, რაც გვაძლევს შესაძლებლობას, განხორციელდეს ზოგიერთი პროცესის და გადაწყვეტილების მიღების ავტომატიზაცია. ამჟამინდელ სისტემებში ჩვენ ძნელად ვიპოვით ასეთ ფუნქციონირებას.

ამ პრობლემის გადასაწყვეტად ნაშრომში განიხილება საქმეთა წარმოების სისტემის გაფართოება უნიფიცირებული ინტელექტუალური მოდულით-ძრავით, რომელიც მოიცავს ცოდნის ბაზას, წესებთან, ფაქტებთან მუშაობის, დასკვნების კეთების, მანქანური სწავლების, დაგეგმვის და გადაწყვეტილებების მიღების მექნიზმებს.

ნაშრომში შემუშავებულია განვითარებული საქმეთა წარმოების სისტემის ფორმალური მეტა მოდელი, რომელიც წარმოდგინდება როგორც საქმის კომპონენტების-ობიექტების, საწყისი მდგომარეობათა გარდაქმნა მიზნობრივ მდგომარეობაში, საქმის წარმოების ოპერაციების მიმდევრობათა შერჩევით.

მდგომარეობა განიხილება, როგორც ობიექტების ატრიბუტების და მათი მნიშვნელობების ერთობლიობა, რომელიც აღიწერება ცოდნის ბაზის ფაქტების სახით, ხოლო ოპერაციები აღიწერება ცოდნის ბაზაში წესების სახით. საქმის წარმოების პროცესში ახალი სიტუაციის გაჩენის შემთხვევაში საჭიროა პროცესის გადაგეგმვა, რისთვისაც მოდელის ფარგლებში ფორმულირდება ოპტიმალური ქმედებათა მიმდევრობის შერჩევის ამოცანა და შემოთავაზებულია მისი გადაწყვეტა ავტორის მიერ შემუშებული ალგორითმით. კონფიგურაციის ან ვალიდაციის წესების ავტომატურად გამოსაყენებლად ნაშრომში შემოთავაზებულია მანქანური სწავლების კლასიფიკაციის მოდელების გამოყენება ღრმა ნეირონული ქსელების საფუძველზე.

წარმოდგენილი ფორმალიზის ბაზზე შემუშავდა საქმეთა მართვის ინტელექტუალური სისტემის არქიტექტურა და განხხორციელდა მისი ბირთვის რეალიზაცია. ძრავი მოიცავს შემდეგ ძირითად არქიტექტორულ კომპონენტებს, როგორიცაა საქმის სასიცოცხლო ციკლის მართვის მოდული (საქმის შექმნა, განახლება და დახურვა), ჰიბრიდული ინტელექტუალური მოდული, რომელიც პასუხისმგებელია ცოდნის ბაზის მართვზე, მანქანურ სწავლებაზე, დინამიური დაგეგმვის და გადაწყვეტილებების მიღების პროცესებზე, ვალიდაციის მოდული უზრუნველყოფს მონაცემთა თანმიმდევრულობას და სიზუსტეს როგორც სტატიკური, ასევე დინამიური და AI-ით გაძლიერებული ვალიდაციის წესების მეშვეობით და კონფიგურაციის მოდული უზრუნვეყოფს სისტემის ოპერაციათა მართვას.

აღწერილ არქიტექტურაზე აგებული მოდული არის დანერგილი განსხვავებულ სფეროებში მომუშავე სამთავრობო და კერძო სექტორის კომპანიებში.