Combining Data-Driven and Knowledge-Based AI Paradigms for Engineering AI-Based Safety-Critical Systems

Juliette MATTIOLI[1], **Gabriel PEDROZA**[2], **Souhaiel KHALFAOUI** [3,5], **Bertrand LEROY** [4,5]

[1] Thales, France - [2] CEA List, U. Paris-Saclay, France - [3] Valeo, France - [4] Renault, France - [5] IRT SystemX, France,
juliette.mattioli@thalesgroup.com - gabriel.pedroza@cea.fr - souhaiel.khalfaoui@valeo.com - bertrand.leroy@renault.com

This work received French government aid under the Investments for the Future program (PIA) within the framework of SystemX Technological Research Institute

Abstract

The development of AI-based systems entails a manifold of doubled-hard challenges. They are mainly due, on one side, to the technical debt of involved engineering disciplines (systems, safety, security), their inherent complexity, their yet-to-solve concerns, and, on the other side, to the emergent risks of AI autonomy, the trade-offs between AI heuristics vs. required determinism, and, overall, the difficulty to define, characterize, assess and prove that AI-based systems are sufficiently safe and trustworthy. Despite the vast amount of research contributions and the undeniable progress in many fields over the last decades, a gap still exists between experimental and certifiable AIs. The present paper aims at bridging this gap "by design". Considering engineering paradigms as a basis to specify, relate and infer knowledge, a new paradigm is proposed to achieve AI certification. The proposed paradigm recognizes existing AI approaches, namely connectionist, symbolic, and hybrid, and proffers to leverage their essential traits captured as knowledge. A conceptual meta-body is thus obtained respectively containing categories for Data-, Knowledge- and Hybrid- driven. Since it is observed that research strays from Knowledge-driven and it rather strives for Data-driven approaches, our paradigm calls for empowering Knowledge Engineering relying upon Hybrid-driven approaches to improve their coupling and benefit from their complementarity.

Introduction

Safety can be defined as "freedom from risk which is not tolerable" (ISO). This definition implies that a safe system is one in which scenarios with non-tolerable consequences have a sufficiently low probability, or frequency, of occurring. Thus, Safety critical systems must be dependable during all their life-cycle, supporting evolution without incurring prohibitive costs. It becomes mandatory that an AI-based Critical System (AICS) does what it has been specified to do (correctness). In the near term, the goal of deploying AI on critical systems motivates research to handle accountability, reliability, suitability, timeliness, etc. Moreover, AICS need to be resilient, safe and (cyber)-secure. Complex mechanisms have to be integrated to ensure both responsibility and accountability of AICS and their outcomes. As any critical system, an AICS needs to be verified, validated, qualified and even certified, following, a suitable development methodology. Explainability is also an issue because AI systems and their decisions need to be explained to integration and maintenance teams as well as end-users in an understandable manner. Auditability for assessing algorithms, data, knowledge, design and integration processes is also a key property that has to be handled. The Fig. 1 highlights various non-functional requirements that have to be verified and demonstrated for a sound AI-based component deployment within an AICS. All such requirements need a sound and trustworthy AI engineering methodology with efficient supporting tools, while addressing various levels of granularity. On one hand they must encompass specific algorithmic domains engineering, including associated data, models and knowledge representations. On the other hand, they must guarantee architecture design correctness up to the complete system engineering cycle.

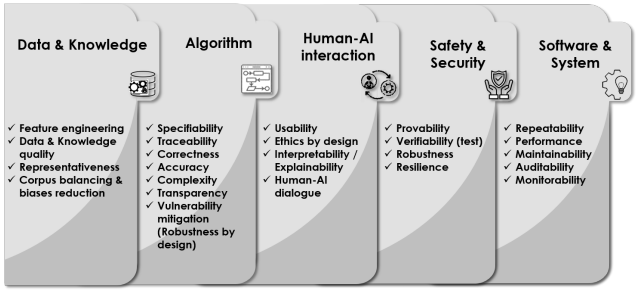

Figure 1: AISCS induced various non functional requirements

By the taxonomy of disciplines involved (Systems, Knowledge, Algorithms, Safety, and Security Engineering), their inherent complexity, the yet-to-solve concerns within each of them (technical debt), and the fact that AICS lay down in the intersection of those disciplines, then AICS development becomes a doubled-hard challenge. Indeed, from an engineering perspective, questions like, *Which are the fundamental notions and features to characterize AICS? Which development languages and methods can sufficiently comprehend and interrelate those notions? How can knowledge be structured to suitably elicit, fulfill and verify require-*

ments? and last, yet not the least, *Whether the bundle of criteria (ranging from data-sensitivity to explainability) can be harmonized and how?*

This position paper introduces the need of a conceptual paradigm which aims to provide a basis upon which answers to previous questions can be elicited. To handle the complexity of the subject, several choices are made. First, since autonomy is a distinctive feature of AICS, and it is intrinsic to safety, the latter is placed as top-priority criterion. Then it is used to guide design, conduct analyses and align the rest of criteria: all in all, we target Safety-Critical AICS (AISCS). Secondly, the proposed paradigm assumes that fundamental notions, as such, should appear in there, irrespective of the development methodology or framework. Last yet not the least, the paradigm also assumes that challenges in AISCS development can be addressed via a body of knowledge amenable to (1) emulate whatever human-beings perceive as intelligence and (2) integrate related concerns and in particular safety. Overall, this paper is dedicated to provide a first specification of a conceptual paradigm as a basis for safe AI-based systems development, in order to design a sound and tooled AI engineering methodology that encompasses with objective of trustworthy AI algorithm engineering, data engineering, knowledge engineering and AI system engineering.

Data-Knowledge-Based Paradigm for Safe AI

In this Section, we shortly describe the background taken as reference for the description of our paradigm and how it is leveraged to constitute the expected foreground.

Paradigm Background

After conducting a brief survey of approaches for AI development (Foggia, Genna, and Vento 2001) (Sun 2015) (Besold et al. 2017), it is observed that, research production is mostly distributed over two big fields, namely **symbolic** (Belle 2020) and **connectionist** (Kasabov 2012). Symbolic approaches are based upon a syntax that is endowed with formal semantics (meaning) useful for properties expression and verification. They have been successfully applied to increase system's trustworthiness in different application domains like health care, automotive, aeronautics, railway (Hofer-Schmitz and Stojanović 2020). Contrary to symbolic, connectionist approaches are not built upon an explicit representation of human expertise: the behaviour is learned from data instances. Connectionist algorithms are based upon a statistical or probabilistic model which is tightly coupled to data sets which are first used for model training, and once tuned, for performance evaluation. The model is often structured as a set of nodes defined by multi-value functions or random variables. The nodes are interconnected, and the links can be randomly weighted by values influencing nodes inputs/outputs (Kasabov 2012). More recently (Sun 2015), **hybrid** approaches integrating symbolic-based and connectionist paradigms have been proposed. Hybrid approaches aim to profit of salient features of both symbolic and connectionist leaving out any potential concurrence (Foggia, Genna, and Vento 2001), (Garnelo and Shanahan 2019). In certain cases, the complementarity between techniques even leads to overcome certain limitations of each other. Indeed, on one side, data-driven AI approaches successfully characterize and capture the salient traits of the data sets. However, being the connectionist models heuristic and agnostic of typical notion-encapsulation archetypes, they lack argumentation necessary for explainability. On the other side, symbolic approaches introduce a semantic layer aligned with the notion-encapsulation archetype, which is amenable for expressing domain knowledge and concerns useful for validation and argumentation.

Paradigm Foreground

The proposed paradigm leverages the background described in previous Subsection in the following manner. First, symbolic, connectionist and hybrid are methods and techniques with distinctive features which are applied to achieve the intended AI functionality. In that respect, once a technique is selected, the engineering choices are mostly oriented to explore and decide HOW and WHEN to apply. Given the referred challenges for AISCS development, and in order to extend the space of engineering choices, it is proposed to consider such techniques as instances of a more abstract meta-structure: an AI-body of knowledge. Such body is meant to include structured information amenable to dissert about WHAT and WHY, in a first place, and in addition HOW and WHEN. Thus, since the connectionist approaches treat and depend upon data sets, then the AI-body of knowledge includes the category of approaches driven by data, *i.e.* a Data-driven AI. Similarly, the symbolic approaches rely upon rules allowing reasoning on terms and notions to infer further knowledge, then the AI-body contains a category of approaches driven by knowledge, *i.e.* a Knowledge-driven AI. A mix of Data- and Knowledge-Driven yields the third category named Hybrid-driven AI.

From the literature survey, the research seems to stray from symbolic AI methods and instead leverage learning-based artificial neural networks. However, some underlining issues of current data-driven approaches such as robustness, fairness, explainability, maintainability, etc. are leading some to call for a return of knowledge-based or some reconciliation of the two main paradigms through Hybrid AI (Garnelo and Shanahan 2019). Following this call, the paradigm proposed hereinafter strives for strengthening Knowledge-driven AIs, targeting a tighter coupling to Data-driven AIs and relying upon the Hybrid-driven approaches.

AI Safety stakes

The conceptual paradigm aims to address several stakes of AISCS. Some of the most salient are listed below.

- *Quantitative safety metrics and methods.* Metrics based upon failure rates have been proven effective to achieve suitable levels of safety. However, new metrics and methods to measure errors and misbehaviours of AI modules are yet to be defined and incorporated.
- *Qualitative safety metrics and methods.* Human interpretability calls for qualitative metrics. Indeed, explainability, risks, and even safety of AI are notions that re-

quire qualitative interpretation (meaning) in order to be assessed.

- *Data modeling and quality.* Data modeling is proposed as a mean for assessment of data features. The influence of data traits (*e.g.*, data diversity (Ashmore and Mdahar 2019), existing vs. possible input values) over the AI modules and their intended functionality need to be assessed and integrated during design.
- *Traceability of safety-related events.* Irrespective of the design method, safety events need to be traceable. In a top-down perspective, high-level safety scenarios should cascade down over the detailed architecture so as to infer dependencies and identify critical subsystems. In a bottom-up perspective, errors/failures at component level need to be propagated upwards to determine safety effects. A structuring layer to support such analyses seems necessary but is still missing.
- *AI safety levels and certification.* AI errors and misbehaviours shall be characterized in such manner that their effects only lead to bearable risks. Assurance and trust on AI can be built upon provable levels of safety incorporated and evaluated all along the development cycle.

The conceptual paradigm aims to be a basis of a framework for representation and integration of previous aspects as well as for inference and assessment. The building process consists in empowering Knowledge Engineering through a better coupling of Data- and Knowledge-driven approaches. The rest of the paper is dedicated to describe the salient aspects and constituents of the proposal.

Empowering Knowledge-based AI systems by Knowledge Engineering

Introduced in 1956, AI is a computer sciences discipline concerned with the theory and development of artificial systems able to perform cognitive activities such as reasoning, knowledge representation, planning, learning, natural language processing, perception and decision. AI includes a wide range of technologies which can be divided into two broad categories: (1) Data-driven AI which includes neural networks, statistical learning, evolutionary computing...; and (2) Knowledge-based AI which focuses on the development of ontology and semantics graphs, knowledge-based systems and reasoning... However, each AI paradigm only focuses on portions of the information and decision chain, leading to solutions that are thus not driven by the global "good decision" goal, making them globally inefficient.

The main AI paradigms

The premises of data-driven AI and knowledge-based AI are fundamentally different (see figure 2). The paradigm of data-driven AI is based on brain-style learning such as neural networks, whereas Knowledge-based AI approaches employ model and knowledge reasoning.

- Often used in the context of pattern recognition, classification, clustering or perception, **data-driven AI** such as machine learning aims at capturing tacit knowledge - knowledge which is difficult or impractical to explicitly or analytically define - through statistical approaches by inferring the inherent structure of a set of examples (input data) that can be used for mapping new data samples.
- (Newell and Simon 2007) claimed that "*Symbols lie at the root of intelligent action*" and should therefore be a central component in the design of artificial intelligence. In its initial form, **knowledge-based AI** focused on the transfer process; transferring the expertise of a problem-solving human into a program that could take the same data and make the same conclusions.

Knowledge-based AI

In Software Engineering, the distinction between a functional specification and the design/implementation of a system is often discussed as a separation of *what* and *how*. During the specification phase, what the system should do is established in interaction with the users. How the system functionality is realized is defined during design and implementation (e.g., which algorithmic solution can be applied). This separation does not work in the same way for a **knowledge-based system** (KBS) which is a computer system that represents and uses knowledge to carry out a task and inference procedures to solve problems that are difficult enough to require significant human expertise for their resolution. Thus, such system has two distinctive features: a knowledge base and an inference engine, where knowledge is then assumed to be given "declaratively" by a set of Horn clauses, production rules, constraints or frames and where inference engines like unification, forward or backward resolution, and inheritance capture the dynamic part of deriving new information.

For instance, constraint programming (CP) is a knowledge-based AI approach (Rossi, Van Beek, and Walsh 2008) for solving combinatorial problems where constraints model the problem and a general purpose constraint solver is used to solve it. The main idea is to propose (1) a modeling language for combinatorial optimization problems (through variable and constraints) and (2) a generic search algorithm able to solve a combinatorial problem described using the modeling language. Mainly, a constraint solver aims at reducing dedicated algorithms implementation costs and constitutes a framework for reuse in combinatorial optimization. In other words, the essence of CP is based on a clean separation between the statement of the problem (the variables and the constraints), and the resolution of the problem (the algorithms) (Heipcke 1999).

Knowledge engineering

Knowledge engineering (KE) is the process of understanding and then representing human knowledge in data structures, semantic models (conceptual diagram of the data as it relates to the real world) and heuristics. Expert systems, constraint programming, ontologies, ... are examples that form the basis of the representation and application of this knowledge. The basic assumption is that both knowledge and experience can be captured and archived in textual or rule-based form, using formalization methods.In its initial form, KE focused on transferring the expertise of a problem-solving human into a program that could take the same data

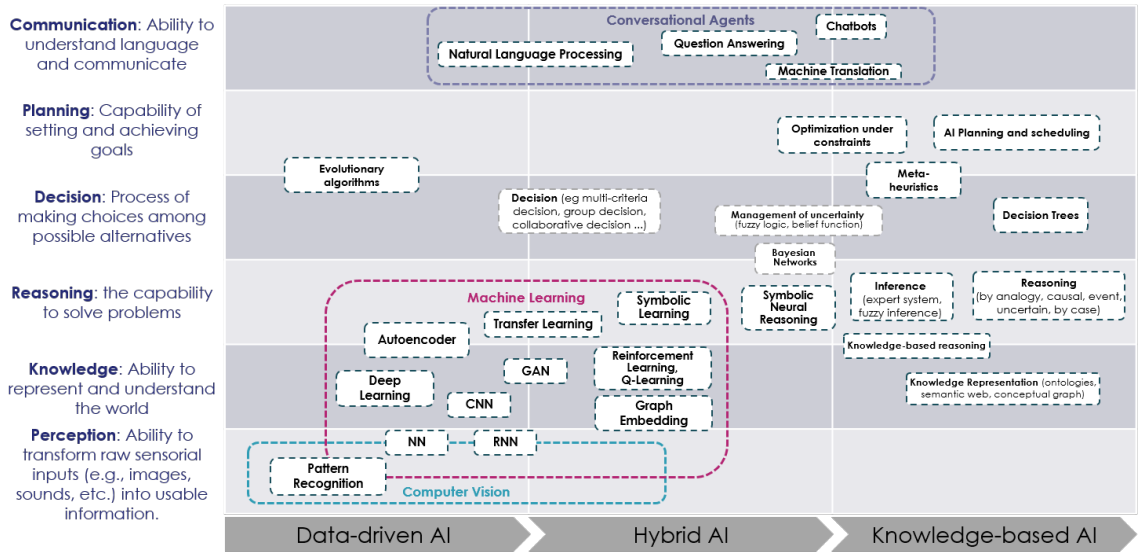

Figure 2: Data-driven AI, Knowledge-based AI and Hybrid AI paradigms illustrated with some techniques

and make the same conclusions. In the 1990s, the KE community shifted gradually to domain knowledge, in particular reusable representations in the form of ontologies. This evolution aimed at alleviating KE limitation to accurately reflect how humans make decisions and more specifically its failure to take into account intuition and "gut feeling", known as "reasoning by analogy". Nevertheless, designing knowledge-based AI component induces some general fundamental problems.

1. **Knowledge discovery**: how do we translate knowledge as it currently exists in textbooks, articles, databases, and human skills into abstract representations in a computer?
2. **Knowledge representation**: how do we represent human knowledge in terms of data structures that can be processed by a computer? How to determine the best representation for any given problem?
3. **Knowledge reasoning**: how do we use these abstract data structures to generate useful information in the context of a specific case? How to manipulate the knowledge to provide explanations to the user?
4. **KBS development lifecycle**: How to verify and update the knowledge base? How to evaluate and validate knowledge-based systems?

Knowledge discovery

While some knowledge is easy to obtain and understand, other knowledge may be difficult to obtain or interpret. In many situations, experts do not have any formal basis for problem solving or for explaining their reasoning process. So they tend to use "rules of thumb" (heuristics) developed on the basis of their experience to help them make decisions. Thus, Knowledge discovery is the process of collecting, extracting, transferring, accumulating, structuring, transforming and organizing (domain) knowledge (e.g., problem-solving expertise) from data and information or from the synthesis of prior knowledge. One of the most important key points of knowledge discovery is to ensure that correct and relevant knowledge is extracted and represented to the stakeholders and decision makers. No matter what kind of knowledge is collected, this process can be realized in a manual way and in an automatic way.

Even if knowledge discovery is today dominated by machine learning (ML) approaches, the iterative execution of the CRISP-DM[1] methodology (Chapman et al. 1999), which is today considered the de-facto standard for knowledge discovery projects, assumes an interaction between domain experts and the data scientists. In practice, the ML model creation process tends to involve a highly iterative exploratory process. In this sense, an effective ML modeling process requires solid knowledge and understanding of the different types of ML algorithms and their parameter tuning (Maher and Sakr 2019), which can be guided by domain knowledge or heuristics (Gibert et al. 2018).

Knowledge Representation and Reasoning

Knowledge Representation and Reasoning (KRR) represents information from the real world for a computer to understand and then utilize this knowledge to solve complex real-life problems. KRR is not just about storing data in a database, it is the study of how what we know can at the same time be represented as comprehensibly as possible and reasoned with as effectively as possibly. One of the main issue is to find the best trade-off between these two concerns.

For (Sowa 2000), "*Knowledge Representation is the application of logic and ontology to the task of constructing computable models for some domain*". Therefore, the way a knowledge representation is conceived reflects a particular insight or understanding of how people reason. The se-

[1]CRISP-DM stands for CRoss Industry Standard Process for Data Mining is a model proposed by a consortium initially composed with DaimlerChryrler, SPSS and NCR

lection of any of the currently available representation technologies (such as logic, knowledge bases, ontology, semantic networks...) commits one to fundamental views on the nature of intelligent reasoning and consequently very different goals and definitions of success. As we manipulate concepts with words, all ontologies use human language to "represent" the world. Thus, ontology is expressed as a formal representation of knowledge by a set of concepts within a domain and the relationships between these concepts. Nevertheless, the "fidelity" of the representation depends on what the knowledge-based system captures from the real thing and what it omits. If such system has an imperfect model of its universe, knowledge exchange or sharing may increase or compound errors during the reasoning process. As such, a fundamental step is to establish effective knowledge representation (symbolic representation) that can be used by future hybrid systems. Symbolic methods may be more adapted to dealing with sparse data, support enhanced explainability and incorporate past human knowledge, while machine learning methods excel at pattern recognition and data clustering/classification problems.

The symbol grounding problem

Developers building knowledge-based systems (KBS), usually create knowledge bases from scratch through a tedious and time-consuming process. First, they have to deal with the diversity and heterogeneity of knowledge representation formalisms and with modeling, taxonomical, and terminological mismatch of different knowledge items, even if they belong to the same application domain. Thus, while the data engineers focus on building the data pipes and data scientists focus on inference methods, knowledge engineers focus on modeling structural use cases and detailing concepts of expert knowledge. Knowledge engineering methods adapt to use cases of knowledge and can model for specific requirements and in many cases produce reusable formats. One of the main limitations of knowledge-based systems lie in the abstract nature of the considered knowledge, in acquiring and manipulating large volumes of information or data, and the limitations of cognitive and other scientific techniques.

Despite of the progress in KE and ontology engineering in the last decade, obstacles remain. Modeling is still a difficult task, as with the choice of the suitable knowledge-based AI technology. Like every model, such a model is only an approximation of the reality. The modeling process is often cyclic. Expert Knowledge, notably via Modeling bias, whereby a human manually designing a model (or part of a model) does not take into account some aspects of the environment in building the model, consciously or unconsciously. New observations may lead to a refinement, modification, or completion of the already built-up model. On the other side, the model may guide the further acquisition of knowledge. Therefore an evaluation of the model with respect to reality is indispensable for the creation of an adequate model. These limitations relate to the so-called **symbol grounding problem** (Harnad 1990), and concern the extent to which representational elements are hand-crafted rather than learned from data. By contrast, one of the strengths of machine learning methods are their ability to discover features in high-dimensional data with little or no human intervention. Several features must be taken into account when developing a KBS:

- Redundancy: are there identical or equivalent knowledge model (such as rules within expert systems, concepts within ontologies, constraints withing constrained solving) that is a special case of another (subsumed)?
- Consistency: Are there ambiguous or conflicting knowledge, is there indeterminacy in its application? Is it intended? Are several outcomes possible, for example, depending on the strategy (the order in which the knowledge models are ordered)?
- Minimality: can the knowledge set be reduced and simplified? Is the reduced form logically equivalent to the first one?
- Completeness: Are all possible entries covered by the knowledge of the set?

Thus, a good KBS must have properties such as:

- Representational Accuracy: It should represent all kinds of required knowledge.
- Inferential Adequacy: It should be able to manipulate the representational structures to produce new knowledge corresponding to the existing structure.
- Inferential Efficiency: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
- Acquisitional Efficiency: The ability to acquire new knowledge easily using automatic methods.

Another key concern in knowledge based modeling is stability. How much variability is there between instances of the problem? How stable is the solution method to small changes? Is the problem very dynamic? What happens if (a small amount of) the data changes? Do solutions need to be robust to small changes? Many such questions need to be answered before we can be sure that a particular knowledge based AI technique is a suitable technology.

Knowledge-driven AISCS Engineering

ML based AISCS engineering is often portrayed as the creation of a ML/DL model, and its deployment. In practice, however, the ML/DL model is only a small part of the overall system and significant additional functionality is required to ensure that the ML/DL model can operate in a reliable and predictable fashion with proper engineering of data pipelines, monitoring and logging, etc. To capture these aspects of AI engineering we defined the ML algorithm engineering pipeline (see fig. 3), where we distinguish between requirements driven development, outcome-driven development and AI-driven development. As the starting point, data must be available for training. Based on data engineering, there are various ways to collect and qualify data set and divide it to training, testing, and cross-validation sets. Engineering activities have to be encapsulated as a series of steps within the pipeline such as:

- **1) Problem specification**, including the Operational Design Domain (ODD), that is the description of the specific

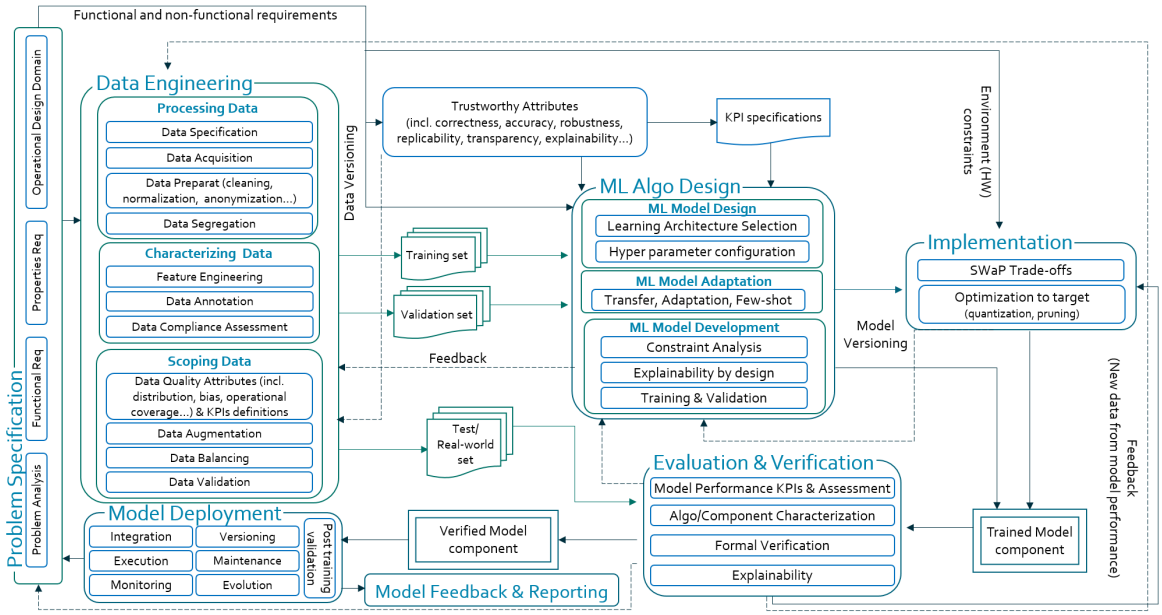

Figure 3: Proposed ML algorithm engineering pipeline

operating condition(s) in which a safety-critical function or system is designed to properly operate, including but not limited to environmental conditions and other domain constraints. These requirements describe the specific function that the ML items should implement as well as the safety, performance, and other requirements that the ML items should achieve.

- **2) Data engineering**, including data collection, preparation and data segregation propose some guidelines. A machine learning model requires large amounts of data, which help the model learning about system objectives and purpose. Before it can be used, data needs to be collected and usually also prepared. Data collection is the process of aggregating data from multiple sources. The collected data needs to be sizable, accessible, understandable, reliable, and usable. Data preparation, or data pre-processing, is the process of transforming raw data into usable information.
- **3) ML Algorithm Design**, after feeding training set to the ML algorithm, it can learn appropriate parameters and features. Once training is complete, the model will be refined by using the validation data-set. This may involve modifying or discarding variables and including a process of tweaking model-specific settings (hyperparameters) until an acceptable accuracy level is reached.
- **4) Implementation**, to develop ML components, we have to decide on the targeted hardware platform, the IDE (Integrated Development Environment) and the language for development. There are several choices available. Most of these would meet our requirements easily as all of them provide the implementation of AI algorithms discussed so far, but sometimes we have to take into account embedded constraints.
- **5) Evaluation and verification**, after an acceptable set of hyperparameters is found and the model accuracy is optimized, we can finally test our model. Testing uses our test dataset and is meant to verify/demonstrate that our models are correct and guarantee some required properties such as robustness and/or explainability. Based on the feedback, we may return to training the model to improve correctness, accuracy and robustness, then adjust output settings, or deploy the model as needed.
- **6) Model Deployment** in the overall system with respect to safety and cyber-security system requirements. Learning assurance case methods can be used.

Then, an ML algorithm has to be designed or selected on existing ML library (such as Scikit Learn (Pedregosa et al. 2011)), to provide a ML model together with its hyperparameters. Next, the model is trained with the training data. During the training phase, the system is iteratively tuned so that the output has a good match with the "right answers" in the training material. This trained model can also be validated with different data. If this validation is successful – with any criteria we decide to use – the model is ready for deployment, similarly to any other component.

Conclusion

"*Data-driven AI is the AI of the senses, and knowledge-based AI is the AI of meaning*"[2]. This is why, in order to cover all cognitive capacities, the future lies in the hybridization of these two paradigms, which are often placed in opposition to AI. Indeed, the shortcomings of deep learning align with the strengths of knowledge-based AI, which

[2]David Sadek, VP Research Technologies and Innovation at Thales

raises the possible benefits of hybridization. First, thanks to their declarative nature, symbolic representations can easily be reused in multiple tasks, which promotes data efficiency. Second, symbolic representations tend to be high-level and abstract, which facilitates generalization. Lastly, because of their propositional nature, symbolic representations are amenable to human understanding. AI algorithms need relevant observations to be able to predict the outcome of future scenarios accurately, and thus, data-driven models alone may not be sufficient to ensure safety as usually we do not have exhaustive and fully relevant data. Nevertheless, as any critical system, an AISCS needs to have well defined development methods from its design to its deployment and qualification. This requires a complete tool chain ensuring trust at all stages, as:

1. Specification, knowledge and data management,
2. Algorithm and system architecture design,
3. AI functions characterization, verification and validation,
4. Deployment, particularly on embedded architecture,
5. Qualification, certification from a system point of view.

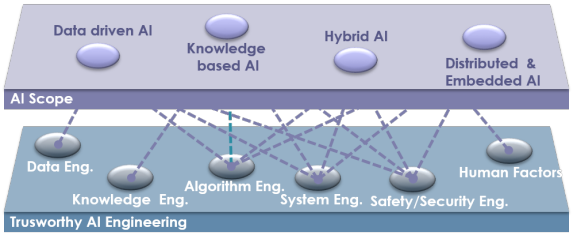

Figure 4: Revisiting all engineering disciplines for a sound deployment of AISCS

All that demands a sound and tooled AI engineering methodology that encompasses, with objective of trustworthy AI algorithm engineering, data engineering, knowledge engineering and AI system engineering by addressing the issues described above. Academic research already proposes solutions towards AI certification[3], industry should take over now (see Fig. 4). At French national level major industrial players in the fields of Automotive, Aeronautics, Defense, Manufacturing and Energy (Air Liquide, Airbus, Atos, EDF, Naval-Group, Renault, Safran, SopraSteria, Thales, Total and Valeo) with the support of academic partners (CEA, INRIA, IRT Saint Exupéry and IRT SystemX) are collaborating together to address such issues through the French National Program "**Confiance.ai**" (https://www.confiance.ai/). Based on the specifications described above, this program aims to bridge the gap between AI Proof of Concepts and AI deployment within critical systems toward certification by providing an interoperable engineering workbench to support AI processes and practices through methods and tools during the overall lifecycle of the AI-based system.

References

Ashmore, R.; and Mdahar, B. 2019. Rethinking Diversity in the Context of Autonomous Systems. *Safety-Critical Systems Symposium 2019*, 175–192.

Belle, V. 2020. Symbolic Logic meets Machine Learning: A Brief Survey in Infinite Domains. *CoRR*, abs/2006.08480.

Besold, T. R.; d'Avila Garcez, A.; Bader, S.; Bowman, H.; Domingos, P.; Hitzler, P.; Kuehnberger, K.-U.; Lamb, L. C.; Lowd, D.; Lima, P. M. V.; de Penning, L.; Pinkas, G.; Poon, H.; and Zaverucha, G. 2017. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. arXiv:1711.03902.

Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; and Wirth, R. 1999. The CRISP-DM user guide. In *4th CRISP-DM SIG Workshop*.

Foggia, P.; Genna, R.; and Vento, M. 2001. Symbolic vs. connectionist learning: an experimental comparison in a structured domain. *IEEE Transactions on Knowledge and Data Engineering*, 13(2): 176–195.

Garnelo, M.; and Shanahan, M. 2019. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29: 17–23.

Gibert, K.; Izquierdo, J.; Sànchez-Marrè, M.; Hamilton, S. H.; Rodríguez-Roda, I.; and Holmes, G. 2018. Which method to use? An assessment of data mining methods in Environmental Data Science. *Environmental modelling & software*, 110: 3–27.

Harnad, S. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3): 335–346.

Heipcke, S. 1999. Comparing constraint programming and mathematical programming approaches to discrete optimisation—the change problem. *Journal of the Operational Research Society*, 50(6): 581–595.

Hofer-Schmitz, K.; and Stojanović, B. 2020. Towards formal verification of IoT protocols: A Review. *Computer Networks*, 174: 107233.

Kasabov, N. 2012. Evolving spiking neural networks for spatio-and spectro-temporal pattern recognition. In *2012 6th IEEE International Conference Intelligent Systems*, 27–32.

Maher, M.; and Sakr, S. 2019. Smartml: A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. In *The 22nd EDBT*.

Newell, A.; and Simon, H. A. 2007. Computer science as empirical inquiry: Symbols and search. In *ACM Turing award lectures*, 1975.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12: 2825–2830.

Rossi, F.; Van Beek, P.; and Walsh, T. 2008. Constraint programming. *Foundations of Artificial Intelligence*, 3: 181–211.

Sowa, J. F. 2000. Guided tour of ontology. *Retrieved from*.

Sun, R. 2015. Artificial Intelligence: Connectionist and Symbolic Approaches. In Wright, J. D., ed., *International Encyclopedia of the Social & Behavioral Sciences (2nd Edition)*, 35–40. Oxford: Elsevier, second edition edition.

[3]https://www.deel.ai/