ERC Advanced Grant 2017
Research proposal [Part B2]

White-Box Self-Programming Mechanisms

WhiteMech

– Principal investigator (PI): **Giuseppe De Giacomo**
– Host institution: **Università degli Studi di Roma "La Sapienza"**
– Proposal duration: **60 months**

Contents

This part of the proposal should be read in conjunction with Part B1, which is to be considered an integral part of the project description.

a State-of-the-art and objectives

a.1 Motivation

We are witnessing an increasing availability of **mechanisms** that offer some form of programmability. Obvious examples are software in our computers and mobile devices, as well as intelligent machines, such as cognitive robots, self-driving cars, flying drones, etc., which are becoming a reality. In particular, programmable mechanisms are increasingly becoming important in three contexts considered of pivotal importance in today's economy, namely **Manufacturing**, **Internet of Things**, and **Business Process Management**. These three areas will act as specific testbeds of the science and technology developed within WhiteMech.

There are compelling reasons to introduce **self-programming abilities** in these systems, such as **self-adaptability** to changing users and environment conditions exploiting information gathered at runtime [142], and **automated exception handling** to suitably **recover from unexpected situations** [117].

The current technology, e.g., **autonomic computing**, which has long advocated self-configuration, self-healing, self-optimization, and self-protection, however, is essentially based on **preprogrammed solutions** by IT professionals [97]. That is, although sophisticated languages and methodologies for streamlining the development of adaptation and exception handling recovery procedures are available, IT professionals, software engineers and programmers, are still writing all the code by hand. As a result, in spite of the progresses in the organization of the software development process, this traditional way of tackling automated reactions in mechanisms is showing serious limitations. In many application areas, it is simply too **costly** and **error-prone** to delegate to software engineers to list and handle all possible adaptation tasks that may arise in the mechanism execution. In fact, especially when applications have to handle widely unexpected circumstances, stemming from the interaction with the real world or with humans taking decisions based on unmodeled circumstances, as indeed in Smart Manufacturing, Internet of Things and Business Processes Management, it is considered simply **infeasible** to determine apriori all possible adaptations that may be needed at runtime [117].

In this state of affairs, the fantastic progress in Machine Learning that we have seen in recent years is attracting a lot of attention. **Machine Learning** is considered a powerful tool to avoid **preprogrammed solutions** in favor of **learned solutions**, and there are indeed great and fully justified expectations of what machine learning can bring about in relieving humans from having to preprogram mundane adaptation tasks.

However, in machine learning, we typically don't have an undestanding of how and why a certain solution has been chosen. This **lack of understandability of machine learning solutions** is increasingly becoming a concern in both the AI and CS scientific communities, [136, 1], and has been recently taken up by DARPA, through the DARPA-BAA-16-53 "Explainable Artificial Intelligence (XAI)" program.[1] For example, the ACM Statement on Algorithmic Transparency and Accountability [1] says: "*There is also growing evidence that some algorithms and analytics can be opaque, making it impossible to determine when their outputs may be biased or erroneous.*"

Beyond the opaqueness of machine-learning-produced solutions, there are also widespread concerns about the **safety** of using such solutions and whether they are thrustworthy.[2] For any system that operates autonomously, there must be guarantees that suitable safety constraints will always be respected. The relevant safety constraints must be specified in a language that human users understand. The system must be able to explain its decisions so that human users can understand the system's operation and confirm that it does follow the relevant rules of conduct. This is crucial for building users' **trust** in the system.

[1] http://www.darpa.mil/program/explainable-artificial-intelligence

[2] These concerns have been widely discussed, see for example https://futureoflife.org/background/benefits-risks-of-artificial-intelligence/.

Hence on the one hand there is a widespread recognition in the AI and CS communities that the objectives of WhiteMech, that is, building self-programming mechanisms that are white-box, are very important. On the other hand, we are currently stuck between two extremes: either we provide preprogrammed solutions, but this is not cost-effective or even feasible for certain applications, or we use machine learning to produce solutions, but this often means that we give up transparency and accountability, and ultimately human comprehensibility.

WhiteMech aims at addressing the realization of white-box self-programming mechanisms on **radically different bases**. It will leverage **Reasoning about Action** and **Generalized Planning** in AI, which provide explicit representation required for being white-box and automated synthesis required for self-programming. However traditional Reasoning about Action and Planning are by far too simple to be used off-the-shelf for realizing white-box self-programming mechanisms. So **WhiteMech intends to revolutionize them by introducing rich objectives, semantically characterized data, and componentization.** To do so WhiteMech will take elements from **Verification and Synthesis** in Formal Methods and **Data-aware Processes** in Databases, and combine the four areas in a novel way to get the right balance between power and effectiveness. Table 1 summarizes which areas contribute to which WhiteMech's objectives (cf. B1).

Table 1: Areas and WhiteMech objectives.

	Objective 1	Objective 2	Objective 3	Objective 4	Objective 5
Generalized Planning	✓	✓			✓
Verification and Synthesis	✓	✓		✓	✓
Reasoning about Action		✓	✓		
Data-Aware Processes		✓	✓		

Below we describe the state-of-the-art and the deep changes proposed in WhiteMech.

a.2 Objective 1: Equip mechanisms with general self-programming abilities

a.2.1 Generalized Planning

Self-programmability is essentially the ability of a mechanism to adapt to new, unexpected conditions, while still being able to achieve its goals. In this view, self-programmability requires that a mechanism be able to produce a plan or a strategy that achieves a desired goal, every time conditions change. In other words, the mechanisms must be able to **plan for a goal**. WhiteMech will investigate an approach based on Planning in AI as a starting point to obtain self-programmability.

Planning is a branch of AI that addresses the problem of generating a course of actions to achieve a desired goal, given a description of the domain of interest and its initial state. The area is central to the development of intelligent agents and autonomous robots. Within WhiteMech, the planning paradigm provides a valuable set of theoretical and practical tools to tackle self-programming: as conditions change, a mechanism can simply build new plans for the desired goals. This can be done very efficiently; indeed, in the last decades, Planning has seen spectacular progresses in terms of scalability, through the use of heuristic search and symbolic approaches [86, 89]. One aspect of Planning in AI that is crucial in WhiteMech, is that **goals are dynamic**, that is, goals are produced continuously, as the agent operates. However, while in Planning a plan for the next goal must be built only after the current goal is achieved, mechanisms have the option to drop the current goal or combine it with the next one, thus adding a complication that cannot be dealt with straightforwardly by standard Planning techniques.

In Planning, the domain where a mechanism operates is described by a set of atomic facts, called *fluents*, and by a set of *actions*, each described by preconditions and effects. In a state, only actions whose preconditions are fulfilled can be executed. The execution of an action yields changes on the truth value of fluents, which, in turn, lead the domain to a new state. This is the reference *Model* of Planning, over which a mechanism can reason to come up with a successful plan. Such model, coming directly from **Reasoning about Action** in KR [131], is embedded in the de-facto standard Planning Domain Definition Language (PDDL) [122, 88].

Planning models have a number of important features for WhiteMech. Firstly, they are **human-comprehensible**, in that fluents and actions describe the domain of interest in a high-level terminology, readily accessible to humans. Secondly, they constitute implicit representations of finite-state transition systems, and are thus readily **verifiable** by standard verification techniques, such as *Model Checking*[45, 12, 113], which typically operate on finite-state systems. By incorporating these features in a mechanism model, **WhiteMech will retain both human-comprehensibility and model verifiability.**

Further, Planning offers the possibility of extracting *Explanations* –see, e.g., [82] for a fresh survey– by querying the model and the plan. **WhiteMech will retain the possibility of querying the model**, which will be a distinctive feature of the project, thus tackling the challenges posed by the DARPA *Explainable AI* Project and the scientific community [136, 1].

Unfortunately, although Planning embeds properties desirable for WhiteMech, it cannot be adopted off-the-shelf to achieve self-programmability, as **traditional Planning models are too coarse to capture important mechanism features**, such as *relational data manipulation* or *componentization.* To address the former, WhiteMech will introduce **rich semantic descriptions** from Knowledge Representation, to which the *PI has contributed significantly over the years [65, 54, 60, 139, 140, 59, 55, 57, 58, 13]*; in this way, mechanisms will be specifiable which incorporate relational data manipulation capabilities, thus **lifting models to a first-order state representation**. To address the latter **componentization will be incorporated in WhiteMech**: as typical of Service-Oriented Computing [24, 146, 19, 64, 51], mechanisms will consist of components, suitably coordinated and orchestrated, each with its own features. *The PI has pioneered work on composition of stateful-services (services that react depending on the state there are in) [17, 16] and later extended these results to handle behavior compositions, e.g. of cognitive robotics systems in AI [137, 138, 64, 51].*

Also the **goal specification formalism of traditional Planning is too limited for mechanisms**. In Planning, goals are specified by boolean combinations of fluents and are considered fulfilled by a plan when a state is reached that satisfies the boolean combination. WhiteMech, instead, aims at handling full-fledged temporal specifications analogous to those used in model checking and reactive synthesis [45, 12, 126], but over a finite time-horizon. For this reason, WhiteMech will adopt non-traditional specification formalisms, such as *LTL and LDL on finite traces, recently proposed by the PI together with Moshe Vardi (Rice U, Huston) [69, 66, 67] and adopted in generalized forms of Planning in AI [149, 39] and in declarative business processes in BPM [152, 47, 61]*, or safe/co-safe LTL/LDL formulas, which have been shown to be more expressive than expected, while remaning **well-behaved** [80, 79, 108, 77]. Besides offering enough expressive power to capture all mechanism features, these formalisms allow to sidestep the notorious difficulties encountered when dealing with their infinite-horizon counterparts. Solvers for these are indeed substantially simpler that those for general reactive synthesis, as based on reachability and safety games, which are amenable to efficient implementations. Importantly, the new formalisms adopted in WhiteMech **will retain the human-comprehensibility featured by traditional goals**.

Planning comes in a variety of forms, each designed to capture certain domain features. In **Classical Planning**, where actions are deterministic and plans are action sequences, finding a plan that, from the initial state, leads the domain to a state satisfying the goal is a PSPACE-complete (reachability) problem. In **Nondeterministic Planning**, instead, actions are non-deterministic and a solution is a function, known as a *policy*, which returns the action to execute, given the current state (a policy is in fact a strategy, which can be memoryless in these problems). Nondeterministic Planning is further split, based on whether the agent can or cannot observe all of the features characterizing the state, into **Fully observable (FOND)**, which is EXPTIME-complete in the domain description, and **Partially observable (POND)**, which is 2EXPTIME-complete, cf. [133]. All of these variants have been, and currently are, deeply investigated, compared and solved by several AI research groups all over the world, which have devised numerous solution strategies, then implemented, tested and optimized in actual planners. The results achieved include, to mention some: advances in FOND planning that have greatly improved the efficiency in this setting [107, 119, 85, 102, 124]; early POND planners, such as CNLP [125], Cassandra [129], and Graphplan (e.g., SGP [6]); model-checking-based planners, such as MBP [18] and BBSP [134]; planners based on heuristic search, such as Contingent-FF [93], "POND" [26], and the more recent CLG [2, 3, 23].

These efforts, spanned over several decades, have led to the development of a sort of **science of search algorithms for Planning**, which has allowed to confine the inherent complexity of Planning within a set of hard instances, while keeping most cases of practical interest efficiently solvable [128, 147, 112, 61]. Since we expect mechanisms to require, in most cases, solving problems from well-behaved classes, WhiteMech will exploit the body of knowledge built in decades of research in Planning and extend algorithms and heuristics to handle generalized forms of Planning. In particular, recent work by the **PI has explored connections between generalized forms of planning and synthesis** when goals are temporally extended: goals are expressed as requirements on the entire execution instead of the final state of the execution [99, 9, 41, 68, 10, 38]. *Promising exploratory results by the PI and other [50, 141, 69, 66, 39] are available*, but such results are far from clarifying the picture both from a theoretical perspective and from an algorithmic perspective. Nonetheless, it appears possible to **compile** the more general forms of synthesis problems which reduce to reachability and co-reachability/safety games **into synthetic planning domains so as to exploit the algorithmic insights from Planning**, Classical Planning, FOND and POND, to efficiently solve these games. In particular, it appears possible to exploit the correspondence between FOND and 2-player games, and the significant recent advances in the computational efficiency of FOND planning that have produced FOND planners that scale well in many domains (e.g., NDP [4], FIP [85, 84], MyND [119], Gamer [102] and PRP [124]).

The crucial point of the approach is that Planning-based solvers embed very effective domain-independent heuristics, continuously improved by a dedicated scientific community. Importantly, no specific modeling formalism is needed to take advantage of the advances, and to obtain such efficiency [83]. Hence, in spite of the notable changes in the model and the goals we will consider in WhiteMech, **we expect that the planning algorithms employed will scale up** as they do in standard Planning.

a.2.2 Verification and Synthesis

Apart from Planning, WhiteMech will draw on the rich modeling and algorithmic techniques from Formal Methods, particularly program verification and synthesis.

The main approach to program verification is *model-checking*. This is the problem of automatically determining whether a system model satisfies a formal specification expressed in logic [45, 12]. This field, for which the founders and proponents won two different Turing awards, has been used successfully for complex systems, and many hardware and software companies use this approach in practice for, e.g., verification of VLSI circuits, communication protocols, software device drivers, real-time embedded systems, or security algorithms.

The same community has been developing *synthesis*, i.e., an approach to automatically construct a system that satisfies a given specification. Of particular relevance is synthesis applied to *reactive systems*: those that maintain an ongoing interaction with an external environment.[3] Such *Reactive Synthesis* is best viewed as a game between the uncontrollable environment and the program to be synthesized. A correct program can then be viewed as a winning strategy in this game, and thus synthesis reduces to finding such a strategy [44, 126]. Over the years, the formal-methods community has exploited this game-theoretic viewpoint and developed a **comprehensive and mathematically elegant theory of reactive synthesis** connecting logics, automata theory and games [154, 105, 106, 104, 155, 90, 43, 21, 22, 80, 116, 81, 87, 78, 20, 5, 76, 25, 96].

In spite of the rich theory developed for program synthesis, little of this theory has been reduced to practice. Some researchers argue that this is because the realizability problem for linear-temporal logic (LTL) specifications is 2EXPTIME-complete [126, 135]. However, this argument is not compelling. First, experience with verification shows that even nonelementary algorithms can be practical, since the worst-case complexity does not arise often (cf., the model-checking tool MONA [75]). Furthermore, in some sense, synthesis is not harder than verification. This may seem to contradict the known fact that while verification is linear in the size of the model and at most exponential in the size of the specification [45], synthesis from LTL specifications is 2EXPTIME-complete. There is, however, something misleading in this fact: while the complexity of synthesis is given with respect to the specification only, the complexity of verification is given with respect to the specification and the

[3]This is in contrast to classical non-reactive programs that transform static inputs to static outputs.

program, which can be much larger than the specification. In particular, it is shown in [135] that there are temporal specifications for which every realizing program must be at least doubly exponentially larger than the specifications. Clearly, the verification of such programs is doubly exponential in the specification, just as the cost of synthesis.

We believe there are a number of reasons for the **lack of practical impact of the theory of reactive synthesis**. We list 3 of these, and include principles and directions for overcoming them within WhiteMech.

(i) **The focus on expressing both the specification and the model in LTL (as in reactive synthesis) hides the relevant complexities**. Exactly as in model checking, we need to distinguish the **complexity wrt the model of the mechanisms** and the **complexity wrt the specification of its tasks (goals)**. In this light, the complexity of solving games with LTL objectives is PTIME-complete in the size of the model, i.e., EXPTIME-complete in the compact representation of the model, and 2EXPTIME-complete in the size of the specification formula of the goal. Thus, even in the case that the model is given compactly (as in Planning), the complexity wrt the model is much lower than the complexity wrt the specification. This is a particularly important observation, since in general the model is going to be much larger than the goals. **WhiteMech will embrace this important distinction.**

(ii) **The specification languages lead to constructions that are too complex**. The approach to LTL synthesis or solving LTL games involves two main steps. First, constructing parity tree-automata that realize all winning strategies, and second, testing such automata for emptiness. The first step can be done using determinization of Büchi automata. Unfortunately, determinization of these automata is not as simple as determinization of ordinary automata (on finite strings), and has been notoriously resistant to efficient implementations [151]. Alternative constructions that avoid determinization [106, 104] did not prove to be radically more efficient [79]. The second step involves solving parity games. This problem, solvable in quasi-polynomial time, is not known to be in PTIME [27] and has been attacked with many different algorithms.[4] However, there are relevant classes of specifications, coming from generalized planning in AI and declarative business processes, which **sidestep these difficulties altogether**. These advocate the use of linear-temporal logics over *finite traces*, i.e., LTL_f and its MSO-complete extension LDL_f. The principal technical advantage of using these is that they involve *ordinary* automata on finite words which are indeed amenable to quite good implementations [152, 69, 66, 67, 149, 39]. **WhiteMech will build upon these well-behaved classes of specifications.**

(iii) **Techniques are optimized for solving difficult synthesis problems, including puzzle-like ones**. Reactive synthesis is typically applied to low-level problems in which there is no reason to think that solutions are easy to find –the planning literature calls these puzzle-like problems. On the other hand, it is unrealistic to require WhiteMech to handle such artificial problems. Instead, just as in Planning, WhiteMech will aim at solving large problems that are not puzzle-like. Following this assumption, Planning has obtained a spectacular improvement in the last two decades, relying on heuristics which ultimately can be seen as abstractions that relax details of the problem at hand. **WhiteMech will apply such heuristics to synthesis.** WhiteMech will also explore new techniques from the formal-methods community, such as those for bounded-synthesis where small controllers are explored before large ones [80, 79, 77], which appear to be quite promising [98].

Finally, we mention that in Verification and Synthesis, **parallel solvers for synthesis games** based on a domain decomposition approach have been recently developed [7, 8]. The idea is easy and effective: partition the problem in sub-problems to be solved in parallel and then combine efficiently their solutions to get the one for the original problem. Benchmarks on random games have proved the usefulness of this idea over the Zielonka Algorithm [157], both in terms of computational cost and of scalability. Specifically, it has been proved an improvement factor linearly dependent on the number of cores used and an irrelevant communication overhead among the executed processes. **WhiteMech intends to explore the possibility that parallel algorithms offer.** This will be done in *collaboration with Camil Demetrescu at Sapienza and Nello Murano at U. Naples.*

[4]See `https://github.com/tcsprojects/pgsolver`.

a.3 Objective 2: Make self-programming mechanisms comprehensible and verifiable by humans

The third ingredient of the WhiteMech approach is **Knowledge Representation**, in particular **Reasoning about Action**. The key property of white-box self-programming mechanisms is the ability of describing the domain in which they operate, their specification, the programs they generate and the relationship between the two in human terms. To do so, the domain where the mechanism operates, the mechanism itself, its capabilities and limitations, as well as its specifications and goals, need to be formally described in terms of concepts that are comprehensible by humans. WhiteMech will consider this issue upfront, by founding white-box self-programming mechanisms on the area of AI called **Knowledge Representation**.

The PI is a prominent member of the scientific community working on Knowledge Representation. He is indeed the single researcher with most papers appeared in the Flagship Conference of the community: the International Conference on Principles of Knowledge Representation and Reasoning (KR) ranked A according to CORE.*

Knowledge Representation stems from a deep tradition in logic, which aims at building systems that know about the world they operate in and are able to act in an informed way in it, as humans do. A crucial feature of these systems is that knowledge is represented "*symbolically*" and that "*reasoning procedures*" are able to extract consequences of such knowledge as new symbolic representations. Such an ability is used to deliberate in an informed fashion the course of actions to take, understanding the consequences of the actions performed. Knowledge Representation has developed enormously since its origins in diverse directions and subareas[120, 109, 110]. Here we focus mainly on the area of Reasoning about Action.

Reasoning about Action takes a first-person view of an agent.[5] The agent has: a representation of the domain in which it operates; a representation of the possible actions in terms of preconditions and effects formally expressed over the representation of domain; and a representation of complex agent behaviors and capabilities, typically described through high-level programs, whose atomic instructions and tests correspond, respectively, to actions and queries over the representation of the domain. By reasoning on these representations, the agent understands what an action or a certain behavior will bring about, thus gaining the ability to make informed decisions on which action to choose or to exclude in relation to its current tasks/goals/specifications. Reasoning about Action has been studied in depth through comprehensive frameworks, such as that of Situation Calculus [121, 131]. As mentioned, it has deeply influenced the models used in Planning, including PDDL.

The PI has deeply contributed to the development of Reasoning about Action, especially within the framework of Situation Calculus, by introducing: the high-level programming languages ConGolog and IndiGolog; the distinction between off-line and online execution; the notion of execution monitoring and recovery; the idea of interleaving execution and planning; and the notion of ability [65, 54, 139, 140].

Importantly, the work on the Situation Calculus, as well as in many other frameworks, is based on a **First-Order Representation of the State**: the state of the agent is represented in terms of predicates/relations. This gives rise to infinite-state transition systems which are typically impossible to verify directly. *Recently, the PI has devised a set of novel results showing the effective computability of expressive variants of the original full-fledged (predicate based) Situation Calculus [57, 58, 56, 13, 36].* Moreover, the techniques for applying belief revision to transition systems proposed in [92, 40], open up the possibility of grounding computationally the notion of "model revision" for mechanisms.

WhiteMech intends to represents the state of mechanisms as well as the state of the enviroment in a semantically rich fashion. To do so we rely on Description Logic and Ontology-Based Data Access.

Description Logics are the formalism of election in for representing the information on the domain of interest in terms of objects grouped in classes or concepts and relationships among such classes. Moreover **description logics** are nowadays considered the standard formalism for ontologies and deeply shaped semantic technologies including the current W3C standard OWL 2.

[5]This contrasts with the work in Multi Agents Systems, where typically a third person view (a "designer" view) is adopted.

The PI has deeply contributed in the development of description logics. First he worked on the correspondence between expressive description logics and modal logics of programs such as Dynamic Logic [53, 31]. Then more recently, together with his group in Rome he developed one of the best know light weight description logics, DL-lite [29], which is essentially able to formalize UML class diagrams and Entity-Relationship diagrams, while keeping inference and conjunctive query answering tractable (the latter in AC0, the same cost as standard SQL). DL-lite in turn made it possible to develop ***Ontology Based Data Access****, which can be considered the most successful use of semantic technologie for data integration [127, 144, 100].*

The role of these technologies in WhiteMech is to represent in terms of an ontology chosen so has to capture the doman on interest in which the mechanism is operating as well as the (high-level) data structures of the mechanism itself in terms that are comprehensible by humans as e.g., in [148]. Technically from a point of view of Reasoning about Action, (the intensional part of) such ontologies act as so called state-constraints, i.e., assertions that must hold in every state. Introducing state constraints in reasoning about actions is notoriously problematic [111, 131]. However *the PI has recently shown the feasibility of combining action theories with ontological representations in description logics [32, 30, 91, 37].* In particular, WhiteMech intends to exploiting the techniques for efficient query-aswering provided by DL-lite and Ontology Based Data Access, especially in the write-also variants, explored recently [63, 52].

Notice that on the other hand WhiteMech does not aim at defining new concrete representation languages. Instead it intends to use well-known formalisms such as BPMN, UML, OWL, etc. as concrete languages, though with a precise formal semantics to allow for automated reasoning, verification and synthesis, see e.g., [15, 62].

a.4 Objective 3: Make self-programming mechanisms data-aware

Crucially, differently from current work in Planning and in Verification and Synthesis, which operate with a propositional representation of the state, WhiteMech does not want to discard relational data, and hence it will need to consider a first-order or relational representation of the state. Apart form Reasoning about action, the other area that is looking into this issue is Data-Aware Processes in Databases.

Data-Aware Processes emerged, during the last decade, as an integrated framework to simultaneously capture the (relational) data of interest in an organizational environment, together with the (business) processes operating over such data towards achieving the strategic objectives of the organization [34].

Traditionally, data and process management have been investigated in mutual isolation, so as to attack the complexity of the organization with the usual *divide et impera* approach. This has led to very successful languages, methods, and approaches in each area, such as UML class diagrams and ER diagrams for data modeling, and BPMN and EPCs for process modeling. However, this isolation falls short when one wants to understand the organization as a whole, take informed decisions, and model end-to-end processes by uniformly combining their business logic and the underlying information systems [132, 72, 130]. This is even more critical if one considers that, in the last decade, we have witnessed a progressive shift from traditional repetitive and predictable production processes, to complex and unpredictable knowledge-intensive processes [153, 94, 103].

As a reaction to this increasing trend, a new generation of data-aware process models and execution environments has arose, with business artifacts [94] and object-centric approaches [103] as its main representatives. In parallel, a flourishing literature on the foundations of verification for such rich models has been produced, devising a plethora of sophisticated decidability results obtained by fine-tuning the interaction between the process and the relational data component (see [156, 34] for two comprehensive surveys on this challenging topic).

The key issue is that relational data in the states gives rise to infinite transition systems which are generally problematic to analyze. *However the PI has already shown within the EU FP7-ICT-257593 ACSI: Artifact-Centric Service Interoperation, that such difficulties can be overcome in notable cases [16, 28, 11, 35].* A key decovery is that under natural assumptions these infinite-state transitions systems admit faithful **abstractions** to finite-state ones, hence enabling the possibility of using the large body of techniques developed within Verification and Synthesis in Formal Methods. Moreover important advancements in undestanding how to deal with such complexity have been established as well as relationships with Reasoning about Action in KR [91, 14, 37, 36, 13]. We will leverage on these ideas to lift our results so as to handle data.

In this light, WHITEMECH will exploit data-aware processes to obtain a coherent, rich representation of mechanisms that enjoys two fundamental properties: being amenable to verification, it paves the way towards automated reasoning; being data-aware, it makes it possible to formulate explanations that simultaneously take into account the specification of the process and the context in which it is being applied.

At the same time, WHITEMECH will expand the boundaries of this research areas along two crucial directions. First, while the majority of data-aware process modeling languages depart from standard modeling languages, WHITEMECH will focus on the enrichment of reference process modeling languages (such as BPMN and variants of Petri nets) with data-related aspects, building on recent, seminal results [70, 123, 62]. Second, while the vast majority of the literature has mainly focused on modeling and verifying data-aware processes against temporal/dynamic properties of interest, WHITEMECH will develop, for the first time, sophisticated forms of synthesis.

a.5 Objective 4: Support component-based approaches

WHITEMECH intends to **support component-based approaches.**

Most synthesis techniques are not compositional. This is a major methodological issue. Most current theory of program synthesis assumes that one gets a comprehensive set of temporal assertions as a starting point. In the context of WHITEMECH and the applications we have in mind, this is not realistic. A more realistic approach would be to assume an evolving formal specification: temporal assertions can be added, deleted, or modified. Since it is rare to have a complete set of assertions at the very start of the design process, there is a need to develop compositional synthesis algorithms. Such algorithms can, for example, refine designs when provided with additional temporal properties [104, 79, 5]. Are these the right citations? The idea of composing solutions has been pioneered in service-oriented computing, and indeed the work by the PI pioneered this approach [17, 50, 64, 33].

An interesting approach is that of bounded synthesis [80, 79, 77] in which the LTL formula is first transformed into a coBüchi automaton on trees which is then further reduced a safety game given the maximum size of the controller to be synthesized. Then iteratively the size in incremented until strategy is found or an exponential limit is reached which guarantees that no strategy will be found at all. This incrementally nature is quite interesting (see below), but construction still too complex for being applied in real cases [98]. !!!!!!!!!!!!!!!!!!

Interfaces capture essential properties of components while hiding irrelevant details. Interface theories [46] provide composition operators, compatibility notions (is the composition of a set of components legal?), and conditions for substitutability (when can one component be replaced by another without compromising the properties of the overall system?) that enable component-based design and alleviate state explosion. Compositionality and incrementality raise a number of questions in the context of synthesis: How to separately synthesize controllers for individual components, or for the same component but with respect to different properties, and then combine them into a single controller? How to derive the component interfaces? Because components take different forms depending on the application domain (they can be pieces of hardware, of software, or of models written in a high-level language such as Simulink or UML), there is no unique, one-size-fits-all interface model. For instance, interfaces carry different information in synchronous vs. dataflow components, or when reasoning about I/O dependencies vs. correctness vs. timing and performance properties [115, 114, 150]. Domain knowledge is key in identifying the right level of abstraction and information content of the interfaces. In conjunction with this, we will develop methods to derive composite interfaces from basic interfaces automatically, thus maximizing the synergy of human and synthesizer.

a.6 Objective 5: Integrate stochastic decision and reinforcement learning

The dynamics and control of many systems is inherently uncertain, and this uncertainty must be taken into account when one deploys algorithms that seek to automatically choose adequate behaviors. So far, we discussed non-deterministic models, but stochastic models are much more popular in many areas because they enable us to quantify our uncertainty, allowing for more informed risk analysis. Popular models include Markov Decision Processes (MDPs), partially observable MDPs (POMDPs) and stochastic games (SG). Moreover, some of the algorithms used to address such domains make stochastic choices. For example, Monte-Carlo search algorithms based their choice on stochastic samples of possible futures, reinforcement learning (RL) algorithms must explore the space, often stochastically, while optimal decisions in adversarial settings often require randomization. It is important that also in these settings, we will be able to provide safety guarantees and express clearly the objectives of the system and require properties (such as conformance with regulations, etc.).

In the extreme case of an agent acting in an unknown environment, such as an agent faced with a choice between two buttons to press, with no knowledge of the implications of each choice, there is little we can do. But the application domains we are concerned with are quite different, as there is usually some understanding of the domain, and hence, an approximate model. What we seek is to improve on manual choices in this domain, and support adaptivity. Thus, even without a model, we can provide the algorithm with advice on its action choice. Simple advice can take the form of "don't use action a in state s". This is easy to integrate into all existing algorithms. More interesting advice is about long term behavior "make sure to cool the engine every now and then" ,"don't touch a pot that has been heated for some time". Even more interesting advice can take the form of a skeleton for the program: "while it is hot, make sure the window is open, and if its does not cool, turn on the air-conditioning". Note that such advice can be provided without having an explicit model in mind.

But even when a model exists, such in MDPs and POMDPs, probably the most popular stochastic models used for planning, the ability to express and use such information is crucial. However, these models specify their objectives using a reward function a function that associates a real value with every state of the world, representing how desirable it is. These models do not support the specification of more complex long term requirements, such as not operating the system is some mode for long periods of time, or ensuring that requests are ultimately serviced, or seeking to have to solution conform to some program skeleton.

We believe that tools developed and enhanced by the PI and co-authors allow us to address these issues. Specifically, one can describe the above constraints and objectives using the expensive language of LDLf. Using this language, one can describe program skeletons and long-term behaviors. This allows us to restrict and guide the behavior of RL algorithms as well as the programs generated by solving MDPs and POMDPs.

With these formulas, we can describe constraints that restrict the exploration of the algorithm to within this safe zone. Second, as means of specifying additional desiderata that is, while the RL algorithm learns about the rewards associated with diverse states of the world and the world dynamics, we can also inject additional rewards that steer it towards behaviors that satisfy these formulas. This, of course, applied to solving known models as well using this language, we can inject additional rewards or constraints that the solution algorithm will either optimize or satisfy. What is most important, is that this can be done efficiently! These formulas can be encoded as finite-state machines that can be used to augment the state space in a way that is transparent to existing algorithms, and typically, quite efficiently. In on going work, we show how these methods can be integrated into classical dynamic programming, as well as search-based methods, and during the project, we intend to enhance these tools, as well as integrate them with algorithms for learning state machines in order to provide more advanced methods for RL and inverse RL that can take into account such specifications.

a.7 Driving Application Contexts

WhiteMech will ground its scientific results in diverse real **application scenarios** to demonstrate the current utilization of the scientific achievements within the project, notably the application of *software systems for automating business processes in dynamic contexts*, i.e., smart environments based on IoT.

Our current world is increasingly connected through a large amount of connected devices, typically embedded in electronics and software and equipped with sensors and actuators, that enable the objects to sense, (re-)act, collect and exchange data via the Internet: this is the Internet-of-Things (IoT). IoT could benefit from Business Process Management (BPM) in terms of structuring analysis about procedures. Business processes represent a specific ordering of tasks and activities across time and place to serve a business goal. Process synthesis based on IoT data can enable an even more complete analysis of processes and realize unused potential for process optimization. Notable application scenarios are the ones of dynamic BPM systems adopted in dynamic contexts (e.g., emergency management [118]), of smart spaces, including public ones as TeverEterno (cf. http://www.tevereterno.it/) and of smart manufacturing, also known as Industry 4.0[6]; a typical manufacturing plant uses information technology, sensors, motors/actuators, computerized controls to manage each specific stage or operation of a manufacturing process. The challenge is to integrate individual stages of manufacturing production enabling data sharing throughout the plant, with the purpose to allow complete production lines and entire plants to run with real-time flexibility in order to conserve energy and optimize outputs [42].

Thus the interplay of IoT, AI-based techniques, cloud computing, Software-as-a-Service (SaaS), and BPM creates the so-called *cyber-physical environments* and give rise to the concept of *Cyber-Physical Systems* (CPSs). Their role is to monitor the *physical processes* enacted in cyber-physical environments, create a virtual copy of the physical world and make decentralized decisions, by introducing methods of self-optimization, self-configuration, self-diagnosis, and intelligent support of workers in their increasingly complex work [143]. Synthesis of components is the main ingredient allowing the complete run-time flexibility. The fundamental role is played by the processes orchestrating the different actors (software, humans, robots, etc.) involved in the CPS. Such *cyber-physical processes* (CPPs), whose enactment is influenced by user decision making and coupled with contextual data and knowledge production coming from the cyber-physical environment, require *Cognitive Process Management Systems* (CPMSs) [95].

A conventional Process Management System (PMS) is a software system that manages and executes processes on the basis of *process models* [74]. The basic constituents of a process model are *tasks*, describing the various activities to be performed by process participants. The procedural rules to control such tasks, described by so-called "routing" constructs such as sequences, loops, parallel, and alternative branches, define the *control flow* of the process. A PMS, then, takes a process model (containing the process' tasks and control flow) and manages the process routing by deciding which tasks are enabled for execution. Once a task is ready for execution, the PMS assigns it to those participants capable of carrying it on. The representation of a single execution of a process model is called a *process instance* [73]. A PMS is said to be *cognitive* when it involves additional processing constructs that are at a semantic level higher than those of conventional PMSs. These constructs are called *cognitive BPM constructs* and tend to include data-driven activities and constraints, goals, and plans [95]. Their usage can open opportunities for new levels of automation and support for CPPs, such as - for example - *the automated synthesis of flexible strategies at run-time exploiting solely the process knowledge and its expected evolution.*

As an example, in the production process of an Italian ceramic sanitary ware factory, a CPS is used to manage and control the working of the robot lines employed in the various steps of the production process of a sanitary item. Each element of the factory is wired and smart, i.e., attached to sensors and actuators, and has the necessary software to operate as part of the CPS, which provides continuous monitoring of the big data generated by the robot lines involved in the production process. Furthermore, a typical production process also includes humans whose intelligence and actions are necessary for the given process. In the ceramic industry, a new product (e.g. a washbasin) is designed (usually with the objective of attracting new customers) with the help of Computer-Aided Design (CAD) tools, which enable designers to produce sophisticated geometric parts that can be examined and, later, modified by manufacturing engineers to ensure trouble-free manufacturing. Starting from

[6]cf. H. Kagermann, W. Wahlster and J. Helbig: Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group, 2013, Frankfurt, http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

a CAD model, an initial mould model of the product is generated. A mould model has has an higher volume (of about 11%-12%) if compared to the final product's volume, since in the subsequent steps of the production process (*drying*, *glazing* and *baking*) it will lost part of its volume and will be affected by deformation influenced by different factors (gravity, humidity, quality of the ceramic mixture, glazing temperature, etc.). The *deformation* of ceramic materials during the drying and baking steps of the ceramics is a well-known problem, as it changes the shape of the ceramic element and makes the final product different to the one thought at design-time [145]. Since it is well known that when a failure occurs in ceramic materials, it is often catastrophic, instantaneous, and total [101], mould design of ceramic products is today performed by a trial-and-error approach and by means of skilled technicians. This makes the production of a new sanitary item as a cyclic process - from the concept to the final product - in which the mould model is continuously modified (by generating several prototypes of the model) to contrast the deformation, with the purpose to reach the exact design dimensions thought at design-time for the final product's shape. Also in the most advanced factories, exceptions and product defects (e.g., deformations) are identified and mitigated only *after* the fact has happened, i.e., after the final step of the production process. Consequently, the time and the related costs required for a new component to enter in production are high (this cycle can run up to 18 months) and strongly limit the productive capacity and flexibility of ceramic factories. To reduce the production time of new products, several ceramic factories might in the future install 3D scanners in the different steps of the production line. The scanners will use structured-light (PLS) technology for capturing digital 3D scans of the sanitary surfaces by generating high-resolution digital 3D models, that can be used to track the behaviour of product deformations during the various stages of the production line. Data detection made by 3D scanning will then be used to develop mathematical models that can help to predict deformations of ceramic items and supporting an optimized design of the CAD models. However, the data collected by 3D scanning are thought to be used to optimize the design of the CAD model *off-line*, while a CPMS could act *on-line*, by first detecting if the deformations of the mould model - during one of the steps of the manufacturing process - are different from the ones expected when developing the initial CAD model, and then by completely changing the manufacturing process to produce one or more procedures that possibly mitigate the entity of the deformation. For example, a possible process can instruct the maintenance crew which is present on site to remove a prototype of the mould model before the manufacturing process is completed (e.g., if a deformation after the drying stage is evaluated as critic and not anymore "adjustable", it is useless to glaze and bake the prototype). A further procedure may command the oven to modify its temperature to fix some deformation caused by the drying step by making more effective the baking step. Exploiting the big data generated by the 3D scanners and having the system to detect and resolve disturbances at run-time before they escalate and result in product defects, with the target to optimize the whole production process by reducing the time to production, is the concrete outcome of applying synthesis techniques studied in WhiteMech to the smart manufacturing scenario.

The PI and his group at Sapienza have contributed to all these fields, see e.g., [48, 49, 71]. Moreover the PI has applied advanced science to real-cases in the area Semantic Data Integration where he and his group have invented the Ontology-Based Data Access paradigm, possibly the most successful approach for Semantic Data Integration [127, 144, 100]. Such an approach has matured to the point that the PI founded a Sapienza start-up ***OBDA Systems*** *(`http://www.obdasystems.com`) to commercially exploit it in real data integration scenarios.*

a.8 High Risk, High Gain

Recent foundational results by the PI chart a novel path that within WhiteMech will revolutionize **Reasoning about Action** in KR and **Generalized Planning** in AI by introducing rich objectives, data, and componentization in order to produce a **breakthrough in engineering self-programming mechanisms that are human-comprehensible and safe by design.** Moreover WhiteMech aims at being the spark that will bring together and cross-fertilize four distinct research areas, namely **Reasoning about Action** in *Knowledge Representation*, **Data-aware Processes** in *Databases*, **Verification and Synthesis** in *Formal Methods*, and **Generalized Planning** in *AI*. The PI has profoundly contributed to all these areas, and he is one of the most prominent AI scientist leading this cross-fertilization.

WhiteMech is a **high risk, high gain project**. It is **high gain** since, if successful, it will result in a radically more useful automated mechanisms than what we have today, namely **white-box self-programming mechanisms**, unleashing full potential of **self-programmability** and removing the main barriers to the uptake of automated mechanisms in real business context, namely *predefined forms of automation*, and difficulties in *formally analyzing their automated behavior in human terms.* Given the crucial role that automated mechanisms plays in our modern economy and society and that **white-box self-programming mechanisms** have the potential to resolve a number of central hurdles, this implies a potentially very high impact on computer science as well as in crucial business contexts, facilitating the already ongoing uptake of automated mechanisms in industry eventually also on economy and society. As a result, WhiteMech is **very timely** and of **greatest significance for European science**.

The WhiteMech involves **high risk** because the ultimately we need to merge explicit representation, with advanced form of process synthesis/coordination and refinement and identify novel and useful islands of effective feasibility that WhiteMech aims at is technically extremely challenging, much more so than the related Verification and Planning both of which have made tremendous progresses in the last decade, touching upon several notorious open problems in computer science. On top of that, the path from theoretical analysis to practically efficient algorithm, which we plan to fully explore within WhiteMech, is a huge challenge given that we aim realizing real automated mechanisms to be deployed in real business scenarios.

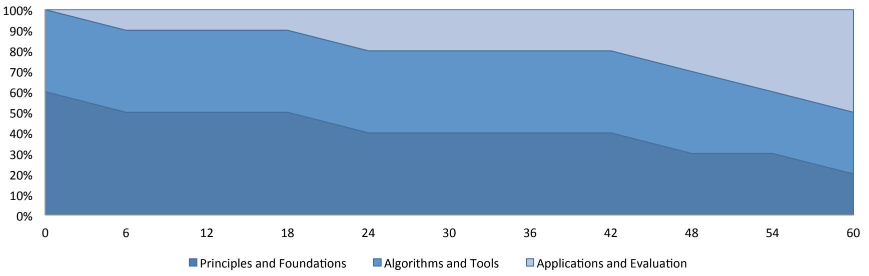
Despite the challenges that WhiteMech will face, as discussed above, the general **feasibility** of WhiteMech is demonstrated by the PI original, exploratory publications [69, 66, 67, 47, 64, 51], and follow ups [149, 39].

b Methodology

The scientific work in WhiteMech will be methodologically structured into 3 broad research streams:

– **Principles and Foundations** that will deal with the scientific foundations of white-box self-programming mechanisms.
– **Algorithms and Tools** that will deal with the development of practical algorithms, optimizations and tools for realizing such mechanisms.
– **Applications and Evaluation** that will evaluate white-box self programming mechanisms in the three business critical driving applications mentioned above.

The percentage of work in the 3 streams in the various years of the projects is reported in figure below.



The **Principles and Foundations** and **Algorithms and Tools** streams will cut across 6 work-packages (WPs) roughly corresponding to the 5 project objectives (cf. B1). The **Applications and Evaluation** stream will be further refined into separate WPs with 3 driving applications. Specifically the WPs are detailed in the following.

WP1: Querying and Synthesis in Finite State Models (m0–m36)

Aim: Develop the core technology for white-box self-programming mechanisms in a finite state setting *(cf. Objective 1 and 2)*. The WP produces a key milestone below at month 18. After the milestone the WP continues with refinements until month 36.
Milestone: (M1) Presentation of first iteration of documents and associated software for theory, algorithms and tools for Querying and Synthesis in Finite State Models (m18).
External collaborators: Hector Geffner (UPF, Barcelona), Blai Bonet (U. Simon Bolivar, Caracas),

and Malte Helmert (U. Basil, Switzerland) on Planning; Moshe Vardi (Rice U., USA) on automata-based verification and synthesis; Sasha Rubin (U. Napoli, Italy), and Benjamin Aminof (TU Wien, Austria) on game-based verification and synthesis; Nello Murano (U. Napoli, Italy) on devising parallel algorithms to run on multicore GPUs [7, 8].

WP2: Querying and Synthesis in Component Based Models (m6–m36)
Aim: Develop the core technology for componentization of white-box self-programming mechanisms in a finite state setting *(cf. Objective 4)*. The WP produces a key milestone below at month 24. After the milestone the WP continues with refinements until month 36.
Milestone: (M2) Presentation of first iteration of documents and associated software for theory, algorithms and tools for Querying and Synthesis in Relational State Models (m24).
External collaborators: Sebastian Sardina (RMIT, Melbourne, Australia), Alfonso Gerevini (U. Brescia, Italy) for composition in generalize planning, Moshe Vardi (Rice U. USA) composition in verification and synthesis, Brian Logan and Natasha Alechina (U. Nottingham, UK) on composition in smart manufacturing.

WP3: Querying and Synthesis in Relational State Models (m18–m54)
Aim: Lift the WhiteMech technology from the finite state case to a first-order setting which is able to deal with relational data *(cf. Objective 3)*. The WP produces a key milestone below at month 36. After the milestone the WP continues with refinements until month 54.
Milestone: (M3) Presentation of first iteration of documents and associated software for theory, algorithms and tools for Querying and Synthesis in Relational State Models (m36).
External collaborators: Yves Lesperance (York U., Toronto, Canada), Hector Levesque (U. Toronto, Canada), Yongmei Liu (Sun Yat-sen U., Guangzhou, China) on reasoning about actions aspects; Rick Hull (IBM Research, USA), Jianwen Su (UCSB, USA), and with Alessio Lomuscio (Imperial College, London, UK) on data-aware processes.

WP4: Querying and Synthesis in OBDA State Models (m24–m54)
Aim: Move the WhiteMech technology from flat relational states to a setting with semantically rich state description *(cf. Objective 2,3)*. The WP produces a key milestone below at month 42. After the milestone the WP continues with refinements until month 54.
Milestone: (M4) Presentation of first iteration of documents and associated software for theory, algorithms and tools for Querying and Synthesis in OBDA State Models (m42). **External collaborators:** Diego Calvanese and Marco Montali (U. Bolzano, Italy), and Ernest Teniente (U. Politecnica de Catalunya, Spain).

WP5: Advanced Querying for What-If Analysis (m30–m54)
Aim: Develop the technology for advanced forms of querying of white-box self-programming mechanisms that enables What-If Analysis *(cf. Objective 2)*. The WP produces a key milestone below at month 48. After the milestone the WP continues with refinements until month 54.
Milestone: (M5) Presentation of first iteration of documents and associated software for theory, algorithms and tools for Advanced Querying for What-If Analysis (m48).
External collaborators: Daniele Magazzeni (King's CoIlege London, UK) on Explainable Planing, and Sasha Rubin (U. Napoli, Italy), and Benjamin Aminof (TU Wien, Austria) on counterfactuals in synthesis.

WP6: Integrating MDPs and RL Components (m12–m54)
Aim: IntegrateWhiteMech technology to handle also MDPs and RL Components *(cf. Objective 5)*. The WP produces a key milestone below at month 36. After the milestone the WP continues with refinements until month 54.
Milestone: (M6) Presentation of first iteration of documents and associated software for theory, algorithms and tools for Integrating MDPs and RL Component (m36).
External collaborators: Ronen Brafman (Ben-Gurion U. Israel), Hector Geffner (UPF, Barcelona, Spain), Blai Bonet (U. Simon Bolivar, Caracas).

WP7: Applications and Validation

Aim: Apply and validate WhiteMech technology in 3 concrete application domains, namely: *Smart Spaces* (IoT), *Process-aware Information Systems* (BPMN), *Smart Manufacturing* (Industry 4.0).

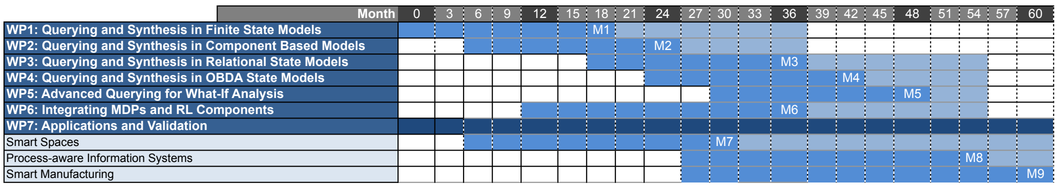
Milestones:

– (M7) release of demonstrator in the context of Smart Spaces of finite-state mechanisms: evaluates results of WP1 and WP2 (m30).

– (M8) release of demonstrator in the context of Process-aware Information Systems of relational-state and OBDA-state mechanisms: evaluates results of WP3 and WP4 (m54).

– (M9) release of demonstrator in the context of Smart Manufacturing of full-fledged mechanisms: evaluates results of WP5 and WP6 and the whole project (m60).

The WP timings is reported in the figure below.



c Resources

Researchers of the group presenting this proposal that will participate in WhiteMech include: Full professors: Giuseppe De Giacomo (PI, h-index=68), Tiziana Catarci (HCI, h-index=35), Maurizio Lenzerini (KR&DBh-index=74), Riccardo Rosati (KR&DB, h-index=50); Associate professors: Ioannis Chatzigiannakis (IoT&I4, h-index=31), Camil Demetrescu (Parellel algorithms, h-index=26), Luca Iocchi (Planning, h-index=37), Domenico Lembo (KR, h-index=32), Massimo Mecella (BPM, h-index=30); Assistant professors (ricercatori): Fabio Patrizi (Planning, h-index=18), Antonella Poggi (KR&DB, h-index=23), Andrea Vitaletti (IoT&I4, h-index=21). PostDocs: Andrea Marrella, Marco Ruzzi, Domenico Fabio Savo and Valerio Saltarelli. In addition, the project will hire 4 Senior PostDocs (SPD) for 3 years each (*Ricercatore a Tempo Determinato di tipo A* cost €45.887 per year of which 75% on the project), 6 Junior PostDocs (JPD), 3 with 3 years contracts and 2 with two year contracts, (*Assegno di Ricerca* €30.600 per year of which 90% on the project), and a total of 8 PhD students (PhD), each on a 3 year program (€16.433 per year of which 70% on the project). % begintable[h!] These new personnel will be allocated as follows: SPD1 will work on Algorithmic aspects of Planning and SPD2 will work on Verification and Synthesis, they both will start at the beginning of the project and wil work mainly for WP1 and WP2. SPD3 will work on aspect lifting finite-state techniques to first-order representations, s/he will start at month 24 and s/he will mostly be involved with WP3 and WP4. SPD4 will work on integrate stochastic decision and reinforcement learning in WP6 and will start working at month 12. JPD1, JPD2, JPD3 (each consisting of 2+3 year contracts, not necessarily for the same person) will work for the entire duration of the project on applications WP7 and software development. While JPD 4 will work on WP1 and WP2, and JPD5 on WP3 and WP4. Finally JPD6, on a 2 year contract, will work on WP5 from month 30.

The **total cost** of the project is €2.499.197. The **direct costs** cover personnel cost for €1.719.357, including the cost of researchers already staffed, and the new openings described above. Giuseppe De Giacomo's cost corresponds to about 70% of full time involvement. The other researchers will work in smaller proportions, because they are involved in other projects as well. Overheads amount to 25% of direct costs. **Equipment costs** €11.000 cover the acquisition of a GPU rack to experiment with parallelization of the planning/synthesis process. **Travel costs** amount to €153.000, and include both the cost for participation to international conferences and workshops, and dissemination and communication events of the personnel, as well as the cost for visiting external collaborators listed in the project. The project does not have any subcontracting cost. However, as stated before, the project will benefit from the numerous collaborations that our group has with external groups and €90000 are allocated for the visit of external collaborators to our Department. €12000 are allocated for publication expenses. Notice that amount is limited because fortunately some of the major journals in AI have open access by default, e.g. JAIR, and AIJ (open access through the IJCAI site). Finally the cost of €14.000 for the audit certificate is also included.

c.1 Budget

Cost Category			**Total in Euro**
Direct Costs	**Personnel**	PI	326.200
		Senior Staff	291.600
		Postdocs	826.083
		Students	276.074
		Other	0
	i. Total Direct Costs for Personnel (in Euro)		1.719.357
	Travel		153.000
	Equipment		11.000
	Other goods and services	Consumables	0
		Publications (including Open Access fees), etc.	12000
		Other (please specify)	90000
	ii. Total Other Direct Costs (in Euro)		0
A - Total Direct Costs (i + ii) (in Euro)			1.763.426
B - Indirect Costs (overheads) 25% of Direct Costs (in Euro)			440.856
C1 - Subcontracting Costs (no overheads) (in Euro)			0
C2 - Other Direct Costs with no overheads (in Euro)			0
Total Estimated Eligible Costs (A + B + C) (in Euro)			0
Total Requested Grant (in Euro)			0

Please indicate the duration of project in months:	60
Please indicate the % of working time the PI dedicates to the project over the period of the grant:	70%

References

[1] ACM U.S. Public Policy Council and ACM Europe Policy Committee. Statement on algorithmic transparency and accountability. ACM, 2017.

[2] A. Albore, H. Palacios, and H. Geffner. A translation-based approach to contingent planning. In *IJCAI*, pages 1623–1628, 2009.

[3] A. Albore, M. Ramírez, and H. Geffner. Effective heuristics and belief tracking for planning with incomplete information. In *ICAPS*, 2011.

[4] R. Alford, U. Kuter, D. S. Nau, and R. P. Goldman. Plan aggregation for strong cyclic planning in nondeterministic domains. *Artif. Intell.*, 216:206–232, 2014.

[5] R. Alur, S. Moarref, and U. Topcu. Compositional synthesis of reactive controllers for multi-agent systems. In *CAV*, pages 251–269, 2016.

[6] C. Anderson, D. Smith, and D. Weld. Conditional effects in graphplan. In *AIPS*, pages 44–53. AAAI Press, 1998.

[7] R. Arcucci, U. Marotta, A. Murano, and L. Sorrentino. Parallel parity games: a multicore attractor for the zielonka recursive algorithm. In *ICCS*, pages 525–534, 2017.

[8] R. Arcucci, A. Murano, G. Perelli, and L. Sorrentino. A parallel model for reachability games. In *Submitted*, 2017.

[9] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. Math. Artif. Intell.*, 22(1-2):5–27, 1998.

[10] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2):123–191, 2000.

[11] B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali. Verification of relational data-centric dynamic systems with external services. In *PODS*, pages 163–174, 2013.

[12] C. Baier, J.-P. Katoen, and K. Guldstrand Larsen. *Principles of Model Checking.* MIT Press, 2008.

[13] B. Banihashemi, G. De Giacomo, and Y. Lespérance. Abstraction in situation calculus action theories. In *AAAI*, pages 1048–1055, 2017.

[14] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of agent-based artifact systems. *J. Artif. Intell. Res. (JAIR)*, 51:333–376, 2014.

[15] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artif. Intell.*, 168(1-2):70–118, 2005.

[16] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based semantic web services with messaging. In *VLDB*, pages 613–624, 2005.

[17] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *ICSOC*, pages 43–58, 2003. 10 year most influential paper award at ICSOC'13.

[18] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *IJCAI*, 2001.

[19] P. Bertoli, M. Pistore, and P. Traverso. Automated composition of web services via planning in asynchronous domains. *Artif. Intell.*, 174(3-4):316–361, 2010.

[20] R. Bloem, K. Chatterjee, S. Jacobs, and R. Könighofer. Assume-guarantee synthesis for concurrent reactive programs with partial information. In *TACAS*, pages 517–532, 2015.

[21] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3), 2012.

[22] A. Bohy, V. Bruyère, E. Filiot, N. Jin, and J. Raskin. Acacia+, a tool for LTL synthesis. In *CAV*, pages 652–657, 2012.

[23] B. Bonet and H. Geffner. Flexible and scalable partially observable planning with linear translations. In *AAAI*, pages 2235–2241, 2014.

[24] A. Bouguettaya, Q. Z. Sheng, and F. Daniel, editors. *Web Services Foundations*. Springer, 2014.

[25] R. Brenguier, J. Raskin, and O. Sankur. Assume-admissible synthesis. *Acta Inf.*, 54(1):41–83, 2017. Work partially supported by the ERC inVEST (279499) project.

[26] D. Bryce, S. Kambhampati, and D. E. Smith. Planning graph heuristics for belief space search. *J. Artif. Intell. Res.*, 26:35–99, 2006.

[27] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *STOC*, pages 252–263, 2017.

[28] D. Calvanese, G. De Giacomo, R. Hull, and J. Su. Artifact-centric workflow dominance. In *ICSOC*, pages 130–143, 2009.

[29] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[30] D. Calvanese, G. De Giacomo, D. Lembo, M. Montali, and A. Santoso. Ontology-based governance of data-aware processes. In *RR*, pages 25–41, 2012.

[31] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *PODS*, pages 149–158, 1998.

[32] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Actions and programs over description logic ontologies. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, 2007.

[33] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Regular open APIs. In *KR*, pages 329–338, 2016.

[34] D. Calvanese, G. De Giacomo, and M. Montali. Foundations of data aware process analysis: A database theory perspective. In *Proc. of PODS*, 2013.

[35] D. Calvanese, G. De Giacomo, M. Montali, and F. Patrizi. Verification and synthesis in description logic based dynamic systems. In *RR*, pages 50–64, 2013. Best paper award.

[36] D. Calvanese, G. De Giacomo, M. Montali, and F. Patrizi. First-order μ-calculus over generic transition systems and applications to the situation calculus. *Information & Computation*, 2017. To appear.

[37] D. Calvanese, G. De Giacomo, and M. Soutchanski. On the undecidability of the situation calculus extended with description logic ontologies. In *IJCAI*, pages 2840–2846, 2015.

[38] D. Calvanese, G. De Giacomo, and M. Y. Vardi. Reasoning about actions and planning in LTL action theories. In *KR*, pages 593–602, 2002.

[39] A. Camacho, E. Triantafillou, C. J. Muise, J. A. Baier, and S. A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, pages 3716–3724, 2017.

[40] M. Carrillo and D. A. Rosenblueth. CTL update of kripke models through protections. *Artif. Intell.*, 211:51–74, 2014.

[41] S. Cerrito and M. C. Mayer. Bounded model search in linear temporal logic and its application to planning. In *TABLEAUX*, pages 124–140, 1998.

[42] S. Chand and J. Davis. What is smart manufacturing. *Time Magazine Wrapper*, pages 28–33, 2010.

[43] K. Chatterjee and T. A. Henzinger. Assume-guarantee synthesis. In *TACAS*, pages 261–275, 2007.

[44] A. Church. Logic, arithmetics, and automata. In *Proc. International Congress of Mathematicians, 1962*. institut Mittag-Leffler, 1963.

[45] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[46] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT*, pages 148–165, 2001.

[47] G. De Giacomo, R. De Masellis, M. Grasso, F. M. Maggi, and M. Montali. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *BPM*, pages 1–17, 2014.

[48] G. De Giacomo, C. Di Ciccio, P. Felli, Y. Hu, and M. Mecella. Goal-based composition of stateful services for smart homes. In *OTM*, pages 194–211, 2012.

[49] G. De Giacomo, M. Dumas, F. M. Maggi, and M. Montali. Declarative process modeling in BPMN. In *Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings*, pages 84–100, 2015.

[50] G. De Giacomo, P. Felli, F. Patrizi, and S. Sardiña. Two-player game structures for generalized planning and agent composition. In *AAAI*, 2010.

[51] G. De Giacomo, A. E. Gerevini, F. Patrizi, A. Saetti, and S. Sardiña. Agent planning programs. *Artif. Intell.*, 231:64–106, 2016.

[52] G. De Giacomo, D. Lembo, X. Oriol, D. F. Savo, and E. Teniente. Practical update management in ontology-based data access. In *ISWC*, 2017.

[53] G. De Giacomo and M. Lenzerini. Tbox and abox reasoning in expressive description logics. In *KR*, pages 316–327, 1996.

[54] G. De Giacomo, Y. Lespérance, and H. J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

[55] G. De Giacomo, Y. Lespérance, and C. J. Muise. On supervising agents in situation-determined congolog. In *AAMAS*, pages 1031–1038, 2012.

[56] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Sardiña. Verifying ConGolog programs on bounded situation calculus theories. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 950–956, 2016.

[57] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Vassos. LTL verification of online executions with sensing in bounded situation calculus. In *ECAI*, pages 369–374, 2014.

[58] G. De Giacomo, Y. Lespérance, F. Patrizi, and S. Vassos. Progression and verification of situation calculus agents with bounded beliefs. *Studia Logica*, 104(4):705–739, 2016.

[59] G. De Giacomo, Y. Lespérance, and A. R. Pearce. Situation calculus based programs for representing and reasoning about game structures. In *KR*, 2010.

[60] G. De Giacomo, H. J. Levesque, and S. Sardiña. Incremental execution of guarded theories. *ACM Trans. Comput. Log.*, 2(4):495–525, 2001.

[61] G. De Giacomo, F. M. Maggi, A. Marrella, and F. Patrizi. On the disruptive effectiveness of automated planning for LTL_f-based trace alignment. In *AAAI*, pages 3555–3561, 2017.

[62] G. De Giacomo, X. Oriol, M. Estañol, and E. Teniente. Linking data and BPMN processes to achieve executable models. In *CAISE*, pages 612–628, 2017.

[63] G. De Giacomo, X. Oriol, R. Rosati, and D. F. Savo. Updating dl-lite ontologies through first-order queries. In *ISWC*, pages 167–183, 2016.

[64] G. De Giacomo, F. Patrizi, and S. Sardiña. Automatic behavior composition synthesis. *Artif. Intell.*, 196:106–142, 2013.

[65] G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *KR*, pages 453–465, 1998.

[66] G. De Giacomo and M. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564, 2015.

[67] G. De Giacomo and M. Vardi. $\mathrm{LTL_f}$ and $\mathrm{LDL_f}$ synthesis under partial observability. In *IJCAI*, pages 1044–1050, 2016.

[68] G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *ECP*, pages 226–238, 1999.

[69] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.

[70] R. De Masellis, C. Di Francescomarino, C. Ghidini, M. Montali, and S. Tessaris. Add data into business process verification: Bridging the gap between theory and practice. In S. P. Singh and S. Markovitch, editors, *Proc. of AAAI*, pages 1091–1099, 2017.

[71] L. de Silva, P. Felli, J. C. Chaplin, B. Logan, D. Sanderson, and S. M. Ratchev. Synthesising industry-standard manufacturing process controllers. In *AAMAS*, pages 1811–1813, 2017.

[72] M. Dumas. On the convergence of data and process engineering. In *Proc. of ABDIS*, volume 6909 of *LNCS*, pages 19–26. Springer, 2011.

[73] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of Business Process Management.* Springer-Verlag Berlin Heidelberg, 1st edition, 2013.

[74] M. Dumas, W. M. van der Aalst, and A. H. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology.* John Wiley & Sons, 1st edition, 2005.

[75] J. Elgaard, N. Klarlund, and A. Møller. MONA 1.x: New techniques for WS1S and WS2S. In *CAV*, pages 516–520, 1998.

[76] J. Esparza, J. Kretínský, J. Raskin, and S. Sickert. From LTL and limit-deterministic büchi automata to deterministic parity automata. In *TACAS*, pages 426–442, 2017. Work partially supported by the ERC inVEST (279499) project.

[77] P. Faymonville, B. Finkbeiner, M. N. Rabe, and L. Tentrup. Encodings of bounded synthesis. In *TACAS*, pages 354–370, 2017. Supported by the European Research Council (ERC) Grant OSARES (No. 683300).

[78] J. Fearnley, D. A. Peled, and S. Schewe. Synthesis of succinct systems. *J. Comput. Syst. Sci.*, 81(7):1171–1193, 2015.

[79] E. Filiot, N. Jin, and J. Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.

[80] B. Finkbeiner and S. Schewe. Bounded synthesis. *STTT*, 15(5-6):519–539, 2013.

[81] S. Fogarty, O. Kupferman, M. Y. Vardi, and T. Wilke. Profile trees for büchi word automata, with application to determinization. *Inf. Comput.*, 245:136–151, 2015.

[82] M. Fox, D. Long, and D. Magazzeni. Explainable planning. In *XAIP*, 2017.

[83] G. Frances, M. Ramirez, N. Lipovetzky, and H. Geffner. Purely declarative action representations are overrated: Classical planning with simulators. In *IJCAI*, 2017.

[84] J. Fu, A. C. Jaramillo, V. Ng, F. B. Bastani, and I. Yen. Fast strong planning for fully observable nondeterministic planning problems. *Ann. Math. Artif. Intell.*, 78(2):131–155, 2016.

[85] J. Fu, V. Ng, F. B. Bastani, and I. Yen. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *IJCAI*, pages 1949–1954, 2011.

[86] H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning.* Morgan & Claypool Publishers, 2013.

[87] B. Genest, D. A. Peled, and S. Schewe. Knowledge = observation + memory + computation. In *FoSSaCS*, pages 215–229, 2015.

[88] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619 – 668, 2009.

[89] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning and Acting.* Cambridge University Press, 2016.

[90] A. Harding, M. Ryan, and P.-Y. Schobbens. A new algorithm for strategy synthesis in ltl games. In *TACAS*, pages 477–492, 2005.

[91] B. B. Hariri, D. Calvanese, M. Montali, G. De Giacomo, R. De Masellis, and P. Felli. Description logic knowledge and action bases. *J. Artif. Intell. Res. (JAIR)*, 46:651–686, 2013.

[92] A. Herzig, M. V. de Menezes, L. N. de Barros, and R. Wassermann. On the revision of planning tasks. In *ECAI*, pages 435–440, 2014.

[93] J. Hoffmann and R. I. Brafman. Conformant planning via heuristic forward search: A new approach. *Artif. Intell.*, 170(6-7):507–541, 2006.

[94] R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *Proc. of ODBASE*, pages 1152–1163, 2008.

[95] R. Hull and H. R. Motahari Nezhad. Rethinking BPM in a cognitive world: Transforming how we learn and perform business processes. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, volume 9850 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2016.

[96] P. Hunter, G. A. Pérez, and J. Raskin. Reactive synthesis without regret. *Acta Inf.*, 54(1):3–39, 2017. Work partially supported by the ERC inVEST (279499) project.

[97] IBM. An architectural blueprint for autonomic computing. IBM White Paper, 2005.

[98] S. Jacobs, R. Bloem, R. Brenguier, A. Khalimov, F. Klein, R. Könighofer, J. Kreber, A. Legg, N. Narodytska, G. A. Pérez, J. Raskin, L. Ryzhyk, O. Sankur, M. Seidl, L. Tentrup, and A. Walker. The 3rd reactive synthesis competition (SYNTCOMP 2016): Benchmarks, participants & results. In *SYNT@CAV*, pages 149–177, 2016.

[99] F. Kabanza, M. Barbeau, and R. St.-Denis. Planning control rules for reactive agents. *Artif. Intell.*, 95(1):67–11, 1997.

[100] E. Kharlamov, D. Bilidas, D. Hovland, E. Jimenez-Ruiz, D. Lanti, H. Lie, M. Rezk, M. Skjaeveland, A. Soylu, G. Xiao, D. Zheleznyakov, M. Giese, Y. Ioannidis, Y. Kotidis, M. Koubarakis, and A. Waaler. Ontology based data access in statoil. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2017. In print.

[101] W. D. Kingery. Introduction to ceramics. 1960.

[102] P. Kissmann and S. Edelkamp. Gamer, a general game playing agent. *KI*, 25(1):49–52, 2011.

[103] V. Künzle, B. Weber, and M. Reichert. Object-aware business processes: Fundamental requirements and their support in existing approaches. *Int. J. of Information System Modeling and Design*, 2(2):19–46, 2011.

[104] O. Kupferman, N. Piterman, and M. Y. Vardi. Safraless compositional synthesis. In *CAV*, pages 31–44, 2006.

[105] O. Kupferman and M. Vardi. Synthesis with Incomplete Informatio. *ICTL*, 1997.

[106] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542, 2005.

[107] U. Kuter, D. S. Nau, E. Reisner, and R. P. Goldman. Using classical planners to solve nondeterministic planning problems. In *ICAPS*, pages 190–197, 2008.

[108] B. Lacerda, D. Parker, and N. Hawes. Optimal policy generation for partially satisfiable co-safe LTL specifications. In *IJCAI*, pages 1587–1593, 2015.

[109] H. J. Levesque. On our best behaviour. *Artif. Intell.*, 212:27–35, 2014.

[110] H. J. Levesque. *Common Sense, the Turing Test, and the Quest for Real AI.* MIT Press, 2017.

[111] F. Lin and Raymond. State constraints revisited. *J. Log. Comput.*, 4(5):655–678, 1994.

[112] N. Lipovetzky and H. Geffner. Best-first width search: Exploration and exploitation in classical planning. In *AAAI*, pages 3590–3596, 2017.

[113] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.

[114] R. Lublinerman, C. Szegedy, and S. Tripakis. Modular code generation from synchronous block diagrams: modularity vs. code size. In *POPL*, pages 78–89, 2009.

[115] R. Lublinerman and S. Tripakis. Modular code generation from triggered and timed block diagrams. In *RTAS*, pages 147–158, 2008.

[116] Y. Lustig and M. Y. Vardi. Synthesis from component libraries. *STTT*, 15(5-6):603–618, 2013.

[117] A. Marrella, M. Mecella, and S. Sardiña. Intelligent process adaptation in the smartpm system. *ACM TIST*, 8(2):25:1–25:43, 2017.

[118] A. Marrella, M. Mecella, and S. Sardiña. Intelligent process adaptation in the smartpm system. *ACM TIST*, 8(2):25:1–25:43, 2017.

[119] R. Mattmüller, M. Ortlieb, M. Helmert, and P. Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, pages 105–112, 2010.

[120] J. McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 756–791, 1957.

[121] J. McCarthy and P. J. Hayes. Some Philosophical Problems From the StandPoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

[122] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, CT, 1998.

[123] M. Montali and A. Rivkin. DB-nets: on the marriage of colored petri nets and relational databases. *Transactions on Petri Nets and Other Models of Concurrency*, 2017.

[124] C. J. Muise, S. A. McIlraith, and J. C. Beck. Improved non-deterministic planning by exploiting state relevance. In *ICAPS*, 2012.

[125] M. Peot and D. E. Smith. Conditional nonlinear planning. In J. Hendler, editor, *Proc. 1st Int. Conf. on AI Planning Systems*, pages 189–197, 1992.

[126] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

[127] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[128] F. Pommerening, M. Helmert, and B. Bonet. Higher-dimensional potential heuristics for optimal classical planning. In *AAAI*, pages 3636–3643, 2017.

[129] L. Pryor and G. Collins. Planning for contingencies: A decision-based approach. *Journal of AI Research*, 4:287–339, 1996.

[130] M. Reichert. Process and data: Two sides of the same coin? In *Proc. of OTM*, volume 7565 of *LNCS*, pages 2–19. Springer, 2012.

[131] R. Reiter. *Knowledge in Action.* MIT Press, 2001.

[132] C. Richardson. Warning: Don't assume your business processes use master data. In *Proc. of BPM*, volume 6336 of *LNCS*, pages 11–12. Springer, 2010.

[133] J. Rintanen. Complexity of planning with partial observability. In *ICAPS*, pages 345–354, 2004.

[134] J. Rintanen. Distance estimates for planning in the discrete belief space. In *AAAI*, pages 525–530, 2004.

[135] R. Rosner. *Modular Synthesis of Reactive Systems.* PhD thesis, Weizmann Institute of Science, 1992.

[136] S. Russell, D. Dewey, and M. Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4), 2015.

[137] S. Sardiña and G. De Giacomo. Realizing multiple autonomous agents through scheduling of shared devices. In *ICAPS*, pages 304–312, 2008.

[138] S. Sardiña and G. De Giacomo. Composition of congolog programs. In *IJCAI*, pages 904–910, 2009.

[139] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the semantics of deliberation in Indigolog - from theory to implementation. *Ann. Math. Artif. Intell.*, 41(2-4):259–299, 2004.

[140] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the limits of planning over belief states under strict uncertainty. In *KR*, pages 463–471, 2006.

[141] S. Sardiña and N. D'Ippolito. Towards fully observable non-deterministic planning as assumption-based automatic synthesis. In *IJCAI*, pages 3200–3206, 2015.

[142] R. Seiger, S. Huber, and T. Schlegel. Toward an execution system for self-healing workflows in cyber-physical systems. *Software & Systems Modeling*, pages 1–22, 2016.

[143] R. Seiger, C. Keller, F. Niebling, and T. Schlegel. Modelling complex and flexible processes for smart cyber-physical environments. *Journal of Computational Science*, pages 137–148, 2014.

[144] J. F. Sequeda and D. P. Miranker. A pay-as-you-go methodology for ontology-based data access. *IEEE Internet Computing*, 21(2):92–96, 2017.

[145] D. Sighinolfi. Experimental study of deformations and state of tension in traditional ceramic materials. *Materiały Ceramiczne*, 63:226–232, 2011.

[146] S. Sohrabi, N. Prokoshyna, and S. McIlraith. Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, pages 319–334, 2009.

[147] M. Steinmetz and J. Hoffmann. State space search nogood learning: Online refinement of critical-path dead-end detectors in planning. *Artif. Intell.*, 245:1–37, 2017.

[148] M. Tenorth and M. Beetz. Representations for robot knowledge in the knowrob framework. *Artif. Intell.*, 247:151–169, 2017.

[149] J. Torres and J. Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *IJCAI*, pages 1696–1703, 2015.

[150] S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee. A theory of synchronous relational interfaces. *ACM Trans. Program. Lang. Syst.*, 33(4):14:1–14:41, 2011.

[151] M. Tsai, S. Fogarty, M. Vardi, and Y. Tsay. State of büchi complementation. *Logical Methods in Computer Science*, 10(4):1–27, 2014.

[152] W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.

[153] W. M. P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: A new paradigm for business process support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.

[154] M. Y. Vardi. An automata-theoretic approach to fair realizability and synthesis. In *CAV*, 1995.

[155] M. Y. Vardi. From church and prior to PSL. In *25 Years of Model Checking - History, Achievements, Perspectives*, pages 150–171, 2008.

[156] V. Vianu. Automatic verification of database-driven systems: a new frontier. In *Proc. of ICDT*, pages 1–13, 2009.

[157] Wieslaw. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.

A Ethics