A Constraint-based Approach to Deal with Self-Adaptation: The Case of Smart Irrigation Systems

Asmaa Achtaich[1], Nissrine Souissi[2], Ounsa Roudies[5]
CRI, Université Panthéon Sorbonne Siweb
Univ. Mohammed V Paris, France[1]
ENSMR[2]
Siweb, Univ. Mohammed V Rabat, Maroc[2, 5]

Camille Salinesi[3], Raúl Mazo[4]
CRI, Université Panthéon Sorbonne Paris, France[3, 4]
Lab-STICC, ENSTA Bretagne Brest, France[3]
GIDITIC, Universidad EAFIT Medellín
Colombia

***Abstract*—Smart irrigation is a specific application of the IoT, where devices composed of sensors and actuators, collect environmental data, like soil humidity, air temperature and brightness, in order to lunch or plan irrigation cycles. These systems function according to a configuration that dictates the way in which every component should operate. Static configurations are limited, as they only represent a set of fixed requirements. However, in domains such as the IoT, technology is continuously evolving, and various users, sometimes with various needs, interact with the system. This leads to dynamic requirements, which are fulfilled by dynamic configurations. This purpose uses the case of an irrigation system to illustrate such requirements, and proposes a constrained-based approach to design self-adaptive smart irrigation systems.**

Keywords*—IoT; irrigation systems; smart; constraint; product lines; self-adaptive systems

I. Introduction

Irrigation is the application of water to a farmed land, to accompany the growth of plants. Efficient irrigation is the main key enabler for durable and profitable agricultural production. Along with the composition of the soil and the temperature and brightness of the environment, the amount of water administered to a plant is crucial to its prosperity. The term "irrigation system" essentially refers to a device or a machine that (semi) automates the process of irrigation, like drip lines, sprinklers or center pivot irrigation systems. Aside from the ones that still require man force, these devices are programmed to water a surface with a specific amount of water, at a certain frequency, or when instructed to do so. These "traditional" irrigation methods have proved wasteful, and in some cases, ineffective. Thus, the need for a controlled alternative, which answers to real time needs.

Smart irrigation systems (SIS) refer nowadays to all of the above, and they are concerned with various aspects of the agricultural process; soil fertility, temperature and brightness monitoring and adjusting, and humidity management, of both soil and air. This can be attainable by assembling a variety of smart devices, like humidity sensors, temperature sensors, brightness sensors, cooling systems, adjustable bulbs, humidifiers, various irrigators, etc. All of which have the ability to communicate, interact with the environment, and in some cases, modify it. Hence, smart irrigation systems became one of the leading applications of the Internet of Things (IoT) [1]; a paradigm that connects smart devices to enable services from the most basic ones to the most complex, innovative and sophisticated. Today's irrigation systems can monitor real time indicators, like the humidity, temperature or brightness, and react accordingly. Depending on the planted plant, and its requirements in terms of moisture, heat and light, an irrigator can regulate the soil humidity, an air conditioner can maintain an optimal temperature, and shades or bulbs can correct the brightness, all in response to measures collected from various sensor types. Some of these actuators can also be connected to a weather based system, and use local weather data to adjust irrigation, heating and lightning schedules [2].

Typical SIS is programmed to collect data from specific devices, and react to a set of parameters in a predefined manner. Authors in [3], [4] and [5] implemented smart farming and irrigation systems using wireless sensor networks [6], to monitor moisture level, daylight intensity and other relevant information, and automatically plan irrigation cycles, water the farmland, or notify users with appropriate times to water. So far, similar works have indeed brought major benefits when compared to manual irrigation. However, some limitations have arisen. First, real life circumstances are sometimes unpredictable, then, user's needs are dynamic, and finally, the system itself, is prone to various evolutions (software and hardware). Therefore, static configuration of SIS becomes unsatisfactory.

The goal of this paper is to approach these limitations by designing the SIS as a dynamic constraint satisfaction problem, where the system is described as variables that abstract the various component of the system. And the user requirements are translated as constraints that restrict that value of these variables in specific domains. To achieve this goal, the Action Research methodology is adopted, specifically, the cyclical process [7]. The first step in the process is the *diagnosis.* It consists of studying the paradigm in order to identify its main limitations from a distinct point of view. Then an *action plan* is elaborated, from the various existing forms of action, to approach the problems identified. Following this step, the *action* planned is realized and implemented, and its consequences are studied and discussed. Finally, the main lessons learned are documented in the form of conclusions.

Following these steps, Section 2 investigates the problems related to current trends in designing SIS. Section 3 presents the action plan: an irrigation case is introduced in two scenarios, and the form of action endorsed in this paper is

described. Section 4 presents an implementation of the planned action and discusses the results. Section 5 presents the related work before conclusion in Section 6.

II. Related Work

For several years, smart irrigation systems have been tailored to fulfill a specific set of fixed requirements. Authors like [4] and [5], propose SIS that monitor and act according the established use cases defined at design time. However, the only form of adaptation that is managed in their work is limited to manual parameterization of decisive parameters [8] (When to irrigate? How long? What temperatures can be harmful, etc).

Designing smart systems in general and smart irrigation systems in particular, as self-adaptive systems, have attracted a lot of interest in the recent decade. Authors [9] like propose an SIS that works hand in hand with a simulation system in a closed controlling loop, where data is continuously fed to the simulator from the perception layers of the system, analyzed then sent back to control the actions. While authors in [10] adopted a cognitive approach, by correlating past actions and results to identify good data delivery paths and recommend intelligent adaptations.

The works of [11] and [12] propose self-adaptation mechanisms for the IoT in general, as they respectively built an intelligent gateway that learns users' behavior and interactions, and implemented a self-adaptive OS-based and reconfigurable embedded system according to objectives such as quality of service, performance, or power consumption.

While these works improve the self-adaptation capabilities of SIS systems, to the best of our knowledge, they focus on execution variability management, related to various changes in the context of the SIS, rather than deal with dynamic requirements, related to variability in time and space. The first refers to various possible configurations of the SIS for various situations, and the second, refers to possible evolutions of the SIS, in time.

III. Problem Investigation

A. Overview of Smart Irrigation Systems

Global food demand is exponentially increasing [13], while the need to conserve resources, like electricity or water, is becoming significant too [14]. Therefore, the interest in smart irrigation systems expands. Consequently, several research efforts have been made in this domain for the last decade, along with various solutions that have penetrated the market as well.

Authors in [15] for example use a GPS based remotely controlled robot to monitor real time field data, and perform irrigation tasks respectively, like watering, securing or fertilizing the field. Likewise, in [16], the irrigation system collects soil, moisture, temperature, humidity and light measures through sensors, and transfers them to a web server, which enables the corresponding actuators according to its preset optimal measures. Various other works tackle the problem in similar fashion, with slightly different technologies. For example, some use thermal imaging instead of conventional sensors to schedule irrigation cycles [17], and other authors focus on mapping the gap between communication technologies by combining heterogeneous devices [18].

Leader industrial solutions, like GreenIQ [19] or Rachio [20] share similar core concepts. As a matter of fact, while they may differ in specificities, the combination of their features includes the monitoring of soil conditions (humidity and tension), weather, evaporation rates, and the use of water by the plants to plan appropriate watering schedules. Therefore, as illustrated in Fig. 1, the main capabilities of SIS can be summarized in an operational MAPE-K loop [21]. According to the knowledge base, the monitor calls specific sensors for data measure, which are analyzed by comparing them to the optimums required. Then a planner determines the action to take, which can vary from planning irrigation cycles, to launching air conditioners or light bulbs, to doing nothing, when all measure comply.

From a technological point of view, as described in Fig. 2, context sensors and actuators usually operate with microcontroller(s), and are configured to transfer real time collected data at a specified frequency to a remote data collector and analyzer, through the Internet. A gateway interfaces devices that communicate with different communication protocols to the internet. A web, mobile or cloud-based collector, receives sensor data, and according to its configuration, requests the actuators to modify its context.

The cornerstone of any smart application is the configuration of its composing devices, and the configuration of the controllers (if any) that manage the overall operations. For example, the configuration of sensors dictates the frequency at which they send data to the collector, the optimal measure under or above which they could enable embedded alarms, or the battery level required to uphold a Wi-Fi connection. The configuration of an irrigator can specify the pressure of the water to administer as well as the duration if this process. The rotation can also be specified in the case of sprinklers or the speed of wheels in the case of center pivot irrigation systems. The configuration of an air conditioner can indicate the temperature to maintain in a greenhouse or the fan speed and direction. The configuration of the controller contains the parameters that trigger specific actions; like waking up a slave device, initiating a service, or transferring data between devices.

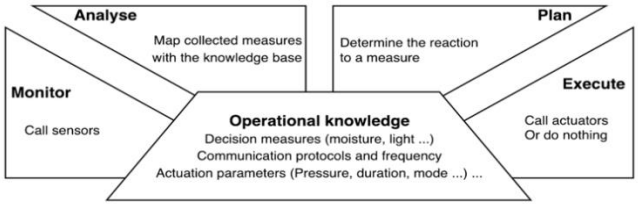

Fig. 1. Implementation of an Operational MAPE-K loop for SIS.

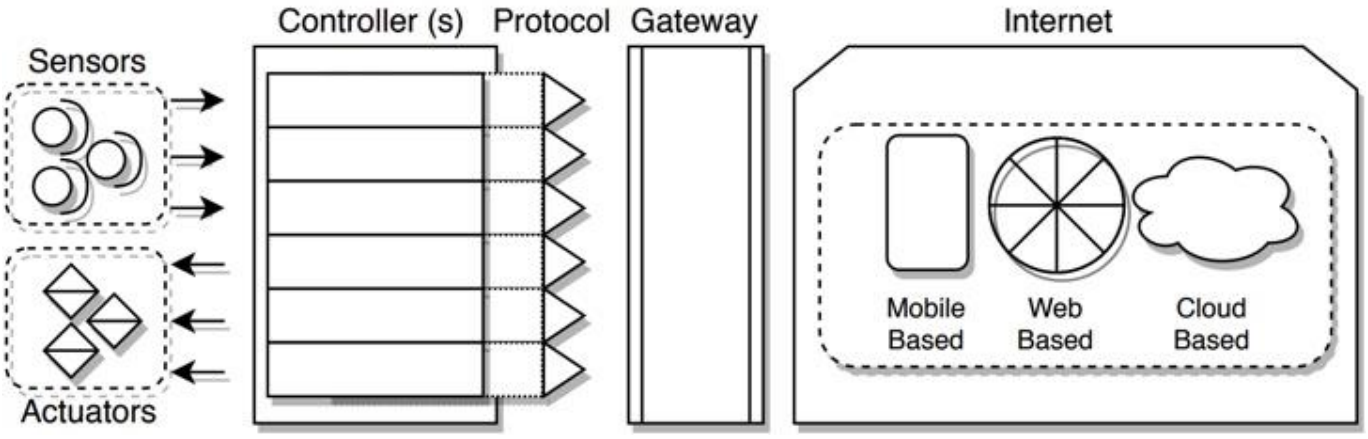

Fig. 2. Overview of a SIS Communication Loop.

While some SIS may offer dynamic parameterization options, to allow users to modify their optimal parameters, reset information about the plantation context, or be informed about the needs of their plants, the smart irrigation system host a static configuration. This configuration defines the pre-set mean of communication between various devices (e.g. the humidity sensors use a ZigBee based communication, the smart meter communicates through Wi-Fi, etc.), determines the devices responsible for achieving specific tasks (e.g. the sprinklers irrigate the field, the mains water provides water to the sprinkler), and may determine the reaction to specific changes in the environment (e.g. When the moisture of the soil is at a specific level, the system shall lunch the sprinkler).

B. Limitations of Static Configurations

A static configuration is defined during the development or implementation process, and remains unchanged over the lifetime of the SIS. However, the more these applications grow in size, complexity and users' expectations, the more maintenance and supervision efforts are required. Surely, the dynamic context of agricultural fields, the dynamic market of irrigation devices, and the various stakeholders that interact with a SIS require a more flexible and autonomous behavioral management, as illustrated in Fig. 3. These characteristics of SIS introduce the following challenges.

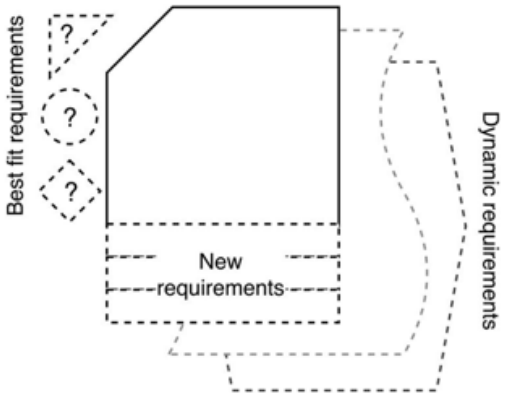

Fig. 3. Multiple Sources of Dynamic behaviour.

1) Best-fit configurations: The context of a SIS refers to everything that surrounds it, and has an impact on it. Some of these indicators influence the fulfillment of core requirements, while other provides information about the overall performance. The measure of humidity, temperature and brightness for example has an impact on the prosperity of the plant. Some other context indicators provide insight over other factors that are just as important to users, like the amount of water applied, the remaining battery level of devices, or the level of security established for communications. Depending on the collected context data, which provides an overview of the environment in which a system is running at real-time, *best fit requirements* specify the configurations that provide the best results, under the current circumstances.

2) Dynamic configurations: As time goes by, the final users and execution context of SIS evolve, in quality, and quantity. The first one refers to the evolutions/change of users' needs. For instance, when Maria sit up a Greenhouse to farm Bio Angelicas, being financially efficient was her first concern. As her business turned more profitable, the speed and quality of the products became more relevant. The configurations that are related to both aspects are therefore modified. The second one refers to the evolution of users themselves, thus their respective configurations. For instance, after a lucrative launch, Maria decided to sell her business. The new owners' approach and needs are slightly different from hers, which should be project on the configuration of the SIS too.

3) New configurations: Connected devices (e.g. Internet of Thing devices or IoT) are growing technical and scientific fields with great potential for innovation and development. Communication protocols, identification mechanisms, operating systems, and even new devices are entering the market with an unprecedented frequency. Installation, enrollment and maintenance cost can thus expand. New requirements describe the positioning of new components within the SIS, how they can interface and interact with previous components, and how they are supposed to operate.

IV. Action Planning

In order to tackle the problems introduced in the previous section, this paper proposes to design smart irrigation systems as self-adaptive systems (SAS) [22], that can automatically modify their behavior in the face of a changing context, to best answer a set of requirements. Thus, with respect to the MAPE-K loop framework in Fig. 4, the SIS monitors the context and the state of devices. The collected measures are mapped with the knowledge base, which englobes information about the application domain (e.g., sunflower cultivation), information about available resources (e.g., water), the distribution and characteristics of the actuators (e.g., water sprinklers), the context optimums (e.g., temperature, wind and humidity), and the user requirements. The compliance of the active configuration is determined, and the resulting action is planned. It could lead to a reconfiguration of the SIS, or to nothing when all measures comply. In order to achieve this goal, a case of an irrigation system is presented in order to illustrate the various problematics discussed above. Then, the core concepts used to design the SIS as a SAS re introduced and defined.

A. The Irrigation Case

To demonstrate the specificities in terms of requirements and illustrate the desired scenarios expected from the SIS, scenarios of an irrigation case are introduced in the following. The first scenario describes the requirements of a new company that wishes to construct a SIS that answers various needs of users, depending on the current context situation of the SIS. The second scenario presents the case of Maria, a new entrepreneur who decides to use GreenLife Solutions, with her own specific requirements.

1) Scenario 1: Best-fit requirements: GreenLife Solutions is a (fictive) company that wishes to build a SIS that monitors change in the environment, and maintains the required levels of moist, temperature and brightness, all while maintaining a good compromise between accuracy, and energy and water efficiency, to insure service durability. In their first prototype, they assume that a farm is rectangular, and needs up to 4 sprinklers to cover the field area. Along with the sprinklers, a drip line and a rooftop are installed to insure hybrid irrigation. The drip line consumes the least water, and the sprinklers are more accurate. The sprinklers can vary in rotation (360°, 240°, 180°, and 90°) and water pressure (20 to 40).

The irrigation devices are connected to two sources of water. The rainwater tank collects rain water, and the mains water is distributed by the local water provider. A water meter measures the level of water remaining in the tank and the water consumption from the mains source. The two sources are enabled alternatively. A pump is only required to be functional when the rainwater mode is enabled.

To collect information about the general state of the field, several sensors are to be installed on premise: Humidity sensors (HS) collect information about the humidity of the soil. Some humidity sensors have embedded alarms or lightbulbs, which can tinkle and twinkle. A HS can be slave (asleep until needed), or master (active). At least one master HS should be active in the fleet. A rain sensor (RS) is installed outside of the greenhouse. A light sensor (LS) detects the brightness inside the greenhouse. The rooftop can also alter the level of soil humidity and the brightness of the field. It can open partially to ventilate, open completely when raining, just enough to water the field, and it can close using the opaque roofing or the transparent one. These different configurations depend on the measures transmitted from the LS, the HS and the RS.

2) Scenario 2: Dynamic and new requirements: Maria owns a small farm in the countryside. She has been trying to plant Bio Angelica for years, but her work schedule does not allow her to move back and forth from the city to the farm. Angelica is a plant that grows better in partial shade, and needs specific attention in terms of irrigation. Therefore, to help Maria supervise the process remotely, the GreenLife solution was recommended to her.

From Maria's perspective, the SIS must activate the sprinklers as long as the water consumption from the smart meter is below the maximum allowed. The first two sprinklers have a 180° rotation, while the second two have a 360° rotation. They all pump water in a 30 psi. When the consumption of water goes higher than the maximum, the SIS switches to the dripline. The rooftop is activated for irrigation when the RS detects rain. The rooftop uses the opaque roofing when the brightness in the field is above the normal measure, and switches to the transparent roofing when the brightness is lower.

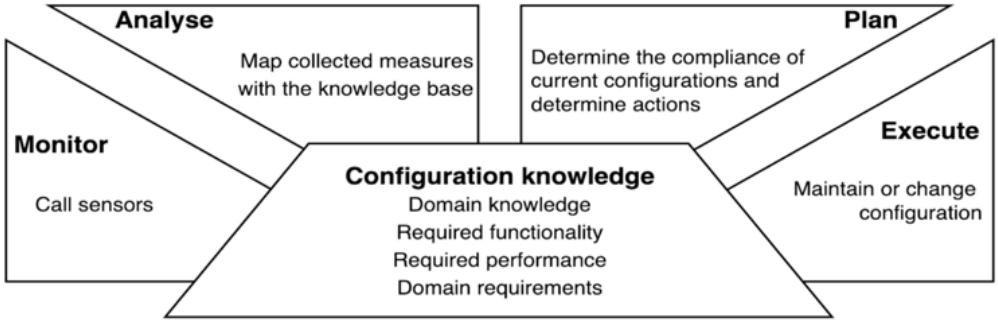

Fig. 4. Implementation of the SIS in the MAPE-K loop Framework.

By default, the water is distributed from the rainwater, as long as the water level in the tank is normal, and the power is on. If one of the two conditions is not fulfilled, the switch lever moves to the Mains water. The rain sensor broadcast information about rain instantly at summertime and with a time span during winter, due to air humidity. To avoid disturbance during the night time, all alarms are deactivated. In the morning, the bulbs are deactivated. All displays and alarms are disabled when there is no human presence detected around the field. When the electricity consumption is normal, all HS are in the master mode. This number is minimized when the electricity consumption becomes higher.

After the period of a year, Maria was not fully satisfied of the outcome of the greenhouse. She decided to worry less about the water and energy efficiency, and concentrate more on accurate means. Her new specifications were the following: "The sprinklers are the only mean of irrigation. Respectively, the rotation and pressure of the first two sprinklers is 240° and 30 psi and of the second two is 360° and 40spsi. Humidity sensors are all on master mode. The fertilizer shall be connected to the sprinklers to fortify the plants. The rooftop is always on opaque mode in the mornings, and is transparent in the evenings and at night.

B. A Constraint-Based Smart Irrigation System

1) Constraint programming: Constraint programming is a declarative paradigm used to solve real world problems, and can be used as a formalism to specify systems and processes, described in the form of variables. Instead of defining an algorithm that describes the instructions needed to solve a specific problem, a constraint program defines the properties that the solution is required to have, and delegates the decision-making tasks to a solver. In this approach, the program is called a Constraint Satisfaction Problem (CSP), and is defined in terms of variables and dependencies that constraint the valuations of these variables in their respective domains.

The SIS can be modeled as a CSP, where variables are abstractions of all the elements of the system, as illustrated in Fig. 5. They can represent a device, a component or a function of the system or its environment. The constraints defined over variables translate the user's requirements. Thereupon, they can define a restriction on the value of a parameter, determine whether or not a component should be enabled, or communicate a preference or a choice of the user. Eventually, the solution derived from the SIS as a CSP specifies a configuration of the system, which fulfils the requirements of the users in an active context.

2) Dynamic constraint satisfaction problems: Static constraint satisfaction problems can be proven efficient for the specification of systems such as SIS. However, as discussed above, these systems undergo continuous change, even after their initial specification. Reasoning about such dynamic environments is beyond CSP, as they stipulate that the set of variables and constraints is known and fixed beforehand. Dynamic Constraint Satisfaction Problems (DCSP) provide the necessary mechanisms to progressively analyze different sets of variables/constraints, for the same problem. In this paper, the DSCP is viewed as a sequence of CSP, where constraints are added/removed from the problem incrementally [23].

In the case of SIS, dynamic requirements are translated as new constraints that dynamically modify the problem. Furthermore, as the elements composing the SIS are prone to evolution, their abstractions as variables may or may not be actively part of a version of the problem.

3) Flexible constraints: For the most part, typical CSPs deal with hard constraint that shall be satisfied to solve the overall problem. However, real world problems are more flexible in nature. therefore, flexible constraints were introduced to allow the specification of problems that are over-constrained. With this approach, even when some of the flexible constraints cannot be satisfied, a useful solution can still be derived [24].

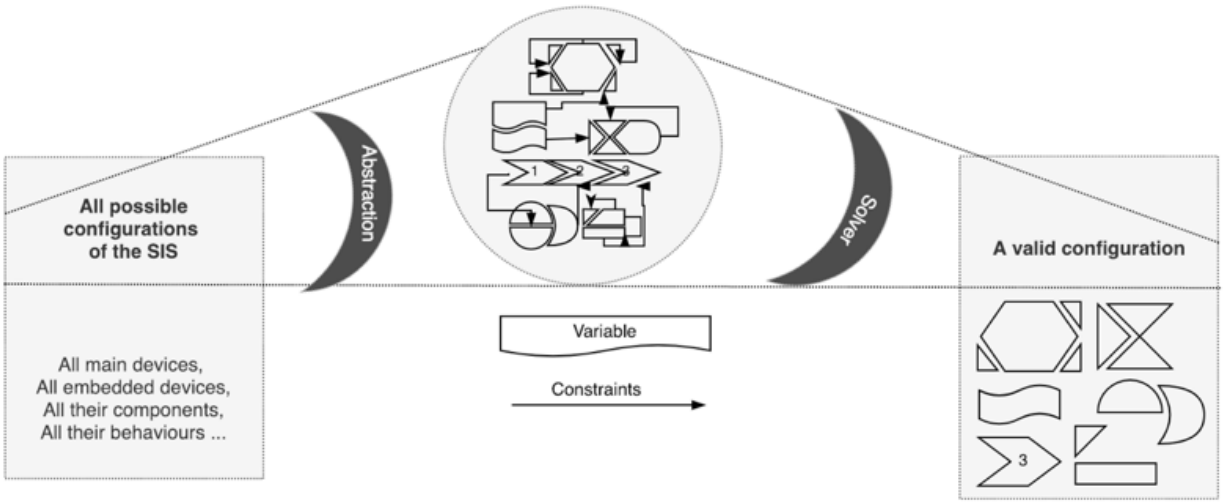

Fig. 5. Representation of a SIS as a CSP.

V. Action and Discussion

The specification of the SIS, as described by GreenLife Solutions, then by Maria and her partner are implemented in Minizinc[1], respectively in the form of CSP and DCSP.

In this section, each scenario is described in the form of a constraints problem; a configuration is generated, and is analyzed to verify its compliance with the needed functionality and performance.

A. Best-Fit Requirements as a CSP

1) Representation: The scenario described in (1)) can be designed as a CSP. The various components of the SIS are abstracted into variables, and the rules that govern the nature of their relationship are defined as constraints over these variables. This approach was first proposed by [25] and [26] and applied to the IoT in cases like [27].

First, the elements of the SIS, along with the functional and nonfunctional requirements they are supposed to fulfill, are declared as variables. Functional requirements (FR) can be satisfied or not. Thus, the domain in which they are solved is Boolean. Nonfunctional requirements (NFR) are of a more relaxed nature. They can be defined over a wider domain. This helps define different priorities to the different NFRs depending on user preferences and expectations.

Finally, components that realize completely the FR, or partially the NFR, are declared as Boolean or numeric variables; the variables that can be selected or omitted in a configuration are defined over a Boolean domain. Similarly, elements that define parameters that are instantiated depending on the runtime state are defined in their respective domains.

```
%Declaration of the functional requirements
var 0..1: Maintain_Plants;
var 0..1: Irrigate;
var 0..1: Monitor;
%Declaration of nonfunctional requirements
var int: EnergyEfficiency ;
var int: WaterEfficiency ;
var 0..4: HSAccuracy ;
var 0..4: HSEnergyEffciency; …
% Declaration of the  SIS components
var bool: Dripline;
var bool: Sprinkler1;
var bool: Sprinkler2; …
%Declaration of integer variables
var 20..60: Pressure1;
var 90..360: Rotation1; …
```

According to the irrigation and monitoring rules defined in Section (1), the system elements are connected, thus constrained are presented in the following code snippet.

```
%FR constraints
Cst1_1 : constraint Maintain_Plants = 1;
Cst1_2 : constraint Maintain_Plants * 2 = Irrigate + Monitor;
%Examples of irrigation constraints
Cst1_5 : constraint Irrigate  >= FertilizerUnit; % A Fertilizer unit is optional
Cst1_11 : constraint Irrigate>=Dripline /\ Irrigate>=Rooftop /\ Irrigate*4>=Sprinkler1+Sprinkler2+Sprinkler3+Sprinkler4 /\ ((Dripline + Rooftop +(Sprinkler1+Sprinkler2+Sprinkler3+Sprinkler4))=1 \/ (Dripline + Rooftop + (Sprinkler1+Sprinkler2+Sprinkler3+ Sprinkler4) =4)) /\ Dripline * Rooftop * (Sprinkler1+ Sprinkler2+ Sprinkler3+ Sprinkler4)=0;  %Mutual exclusion between sprinklers, Dripline and Rooftop
Cst1_15 : constraint Sprinkler1 >= 1 <-> (Rotation1 >= 1); %if the sprinklers are selected, the pressure shall be instantiated as well
%Examples of monitoring constraints
Cst1_27 : constraint Monitor = RS; % The rain sensor is mandatory
Cst1_29 : constraint HS1 >= Master1 /\ HS1 >= Slave1 /\ HS1 = Master1 + Slave1 ; %A humidity sensor can either be Master or slave
```

The impact that every component of the SIS has on its performance is defined by claims, which are maximized according to the preferences set by the users, towards an appropriate solution.

```
%Example of the impact of components on performance
Cst1_38 : constraint C1 > 0 <-> ((Sprinkler1 + Sprinkler2 + Sprinkler3 + Sprinkler4 > 0) -> IrrEnergyEfficiency <= 2) /\ (Dripline > 0 -> IrrEnergyEfficiency = 3) /\ (Rooftop >= 0 -> IrrEnergyEfficiency >= 4); %The sprinklers use most of the water, then the dripline, and finally the rooftop is the least consuming
%Maximization function
Cst1_50 : constraint SIS_Config = Energy_Preference * EnergyEfficiency+ Accuracy_Preference * Accuracy + Water_Preference * WaterEfficiency;
solve maximize (SIS_Config);
```

2) Discussion: All the constraints defined above compose the CSP. Once run by the solver, a configuration of the SIS is supplied. Chart 1 illustrates two configurations derived from the same CSP, each corresponding to the preferences of Maria, then Sophia, her partner. Maria needs the system to run in an energy efficient (**EE**) manner, while Sophia requires accuracy (**A**). Theses preferences can be set respectively by instantiating the parameters *Accuracy_Preference; Energy_Preference and Water_Preference*, with their respective weights. Fig. 6 shows (in red), the configuration generated when accuracy (A) has the biggest weight. Similarly, the charts delineate (in blue) an energy efficient (EE) configuration.

[1] https://www.minizinc.org/

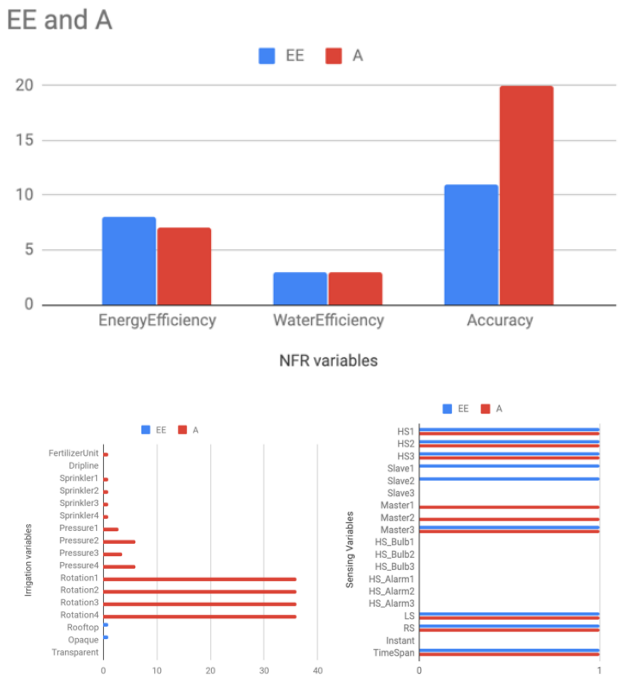

Fig. 6. The Configuration of the SIS (Scenario 1).

As expressed by the constraints presented below, the configurations mainly differ in the use of Sprinklers and rooftop (Table I: Irrigation variables). The first being energy consuming but accurate as it receives data from the sensors and reacts accordingly. Then the second being energy efficient, but inaccurate as irrigation depends on an unpredicted element with is rainfall. Another variable between both configurations is the number and state of humidity sensors (Table I: sensing variables). Indeed, selecting multiple HS increases the accuracy of the collected data. The more master HS there are, the more data is representative of the field's actual state. Similarly, the less active HS there are, and the more they are in the Slave state; the less energy is consumed, at the expense of accuracy. Other invariable are similar as they do not impact either accuracy not energy efficiency according to the CSP.

B. Dynamic and New Requirements as a DCSP

1) Representation: The way Maria intends to use the SIS is tightly related to fluctuations in the context. Thus, to maintain satisfying performance under various conditions, she specifies different requirements to achieve the same goal. Therefore, the constraints introduced to the CSP differ, depending on current context.

The context can also be defined in constraint programming as a set of variables, which are abstractions over a part of the system's environment that has an impact on the activity or quality of the SIS. They can be monitored at runtime by sensing or by reporting, and are thus dynamic. Context elements that are valuable to Maria's case are declared below.

For each context situation (an instantiation of context variables to specific values), a set of requirements shall be fulfilled (a set of constraints shall be validated). Vice versa, as a new context situation occurs, requirements that were once mandatory become obsolete (thus, the previous set of constraints becomes irrelevant, and the new one shall be validated instead).

```
%Declaration of context variables
0..1: Smartphone_Availability;
0..1: Power_failure;
0..1: Rain;
int: Light;
int: MaxLight;
int: TankCapacity;
0..TankCapacity: TankLevel;
int:MinTankWater;
int: MainsWaterConsumption;
int: MaxMainsConsumption;
1..12: Month;
1..31: DayOfMonth;
00..24:Time;
1..10: int
```

Adding requirements to an existing SIS is a two steps process. The first one lies in defining the new constraints (representative of the new requirements), and the second process consists of relaxing unrefined constraint, as to not contradict with new specifications. This is presented in the following, in the form of Reified and Flexible constraint.

a) Reified constraints: The reification of a constraint *c* is the association of a Boolean variable *B* to a constraint *C*. When *B* is true/false, the reified constraint shall be satisfied/unsatisfied, respectively. Similarly, when C is satisfied/unsatisfied, B should be set to true/false, respectively. In some constraints, a simple implication is used instead of a full reification, to avoid inconsistency with previous constraints.

```
%Constraints on global events
Cst2_1 : constraint Year<=2 <-> GlobalRC1;
Cst2_2 : constraint Year>2  <-> GlobalRC2;
```

The requirements expressed during the first year, as described in (2)), are specified in the code bellow. The occurrence of an event (eg: rain), sets the value of the related reified constraint to *True* (eg: RC10=True), therefore enforcing the constraints reified by RC10, (e.g. Rooftop= 1;). Similarly, all the other requirements that shall be fulfilled under specific circumstances are translated into constraints that shall be satisfied when the related reified constraints are true.

```
%----------------First year requirements
%Examples of constraints that express Events -- First Year
Cst2_10 : constraint (GlobalRC1 /\ Month >=5 /\ Month <10) <-> RC2_3; %Summer
Cst2_11 : constraint (GlobalRC1 /\ Smartphone_Availability = 1 )<-> RC3; %Humain presence
Cst2_15 : constraint (GlobalRC1 /\ Power_failure = 1) <-> RC6; %No electricity
Cst2_17 : constraint (GlobalRC1 /\ Light <= MaxLight) <-> RC8; %Normal bright
Cst2_19 : constraint (GlobalRC1 /\ Rain = 1) <-> RC10; %Raining
Cst2_20 : constraint (GlobalRC1 /\ (TankLevel >= MinTankWater /\ Power_failure = 0)) <-> RC11; %Water provided from Rainwater
Cst2_22 : constraint (GlobalRC1 /\ (MainsWaterConsumption >= MaxMainsConsumption) /\ RC10=0)<-> RC13; %Water consumption is Hight
%Examples of reified constraints and their respective actions -- first year
Cst2_39 : constraint RC2_3 -> TimeSpan=1;
Cst2_40 : constraint RC3 -> Treshold_Alarm >= 0 /\ HS_Alarm1 >= 0 /\ HS_Alarm2 >= 0 /\ HS_Alarm3 >= 0 /\ HS_Bulb1 >= 0 /\ HS_Bulb2 >= 0 /\ HS_Bulb3 >= 0;
Cst2_43 : constraint RC6 -> (Master1 + Master2 + Master3) <=2 /\ RC4=1 ;
Cst2_47 : constraint RC8 -> Transparent = 1;
Cst2_49 : constraint RC10 -> Rooftop= 1;
Cst2_50 : constraint RC11 -> Rainwater=1;
```

Similarly, the expectations of Maria for the second year are expressed as a new set of requirements, translated into new constraints. Therefore, a new CSP is to be solved, since the previous constraints are no longer relevant.

```
%----------------Second year requirements
%Examples of constraints that express Events-- Second year
Cst2_25 : constraint GlobalRC2 /\ (Time >=10 /\ Time <16) <-> RC17;
Cst2_27 : constraint GlobalRC2 <-> RC14;
Cst2_31 : constraint GlobalRC2 <-> RC20;
%Examples of reified constraints and their respective actions -- Second year
Cst2_53 : constraint RC14 -> Sprinkler1+ Sprinkler2 + Sprinkler3 + Sprinkler4=4 /\ Pressure1=30 /\ Pressure2=30 /\ Pressure3=40 /\ Pressure4=40 /\ Rotation1=240 /\ Rotation2=240 /\ Rotation3=360 /\ Rotation4=360;
Cst2_56 : constraint RC17 -> Opaque = 1;
Cst2_59 : constraint RC20 -> RS + LS + Instant + TimeSpan = 0;
```

b) Flexible constraints: As the SIS evolves, the constraints that were once mandatory may become irrelevant or reduced in priority. Therefore, constraints that once defined a static structure of the SIS can be subject to change. For example, a component that was mandatory (Rain sensor) may become undermined (RS=0). The irrigation and monitoring components, presented in 4.1. can be transformed into flexible constraints. The goal is to maximize their satisfiability rather than solve them. If no contradictory constraint is introduced in time, the problem is likely to be satisfied completely. However, if it is the case, some of the constraints are allowed to remain unsatisfied, without inconsistency warnings. Constraints (Cst1_10, Cst1_26, Cst1_27) are not satisfied in the light of new requirements. However, since they are now flexible, their unsatisfiability does not generate inconsistencies.

```
%----------------Example of flexed requirements
Cst2_8 : constraint Flex6 <-> Rooftop >= Opaque /\ Rooftop >= Transparent /\ Rooftop >= Opaque + Transparent /\ Opaque + Transparent <= Rooftop;
Cst2_22 : constraint Flex19 <-> Monitor = LS;
Cst2_25 : constraint Flex20 <-> Monitor = RS;
```

C. Discussion

According to these constraints, the configurations of the SIS, during the *first*, then the *second* year, which correspond to the context variables presented in Table I, are respectively represented by *Red* and *Blue*, in the Fig. 7.

The configurations generated indeed in accordance with Maria's requirements. For the first year, if it is raining (T1:rain=1), then the rooftop shall be selected (T2: Rooftop = 1). Other irrigation means are momentary disabled (T2: Dripline=0 & Sprinklers=0), until another configuration specifies otherwise. Human presence is detected around the field (T1:Smartphone_Availability=1), therefore, display devices are enabled including humidity sensor' alarms and bulbs, and the tank's threshold alarm (T2: HS_Bulb(1,2,3)=1, HS_Alarm(1,2,3)=1, & Treshold_Alarm=1). However, since it's the morning (Time=15), bulbs are inefficient (HS_Bulb(1,2,3)=0), as they consume energy without playing a crucial role, therefore, they are disabled (T2:HS_Bulb(1,2,3)=0). As the energy state is normal (T1: Power_Failure=0), all humidity sensors are active (HS(1,2,3)=1), and are on master mode to maximize data precision. The brightness level is higher than required for a healthy plant, therefore, the opaque shades are enabled (T2: Opaque=1). The level of water in the tank is above the minimum required, therefore, it will be responsible for providing irrigation water (T2:Rainwater=1), and will thus require a working pump (Rw_Pump=1) as well. Finally, as it is winter time (Month=11), and since air humidity is high, the RS is likely to mistake faug with rain, thus, the decision about rain is made with a timespan (T2:Timespan=1) to insure the righteousness if the information. During the second year, irrigation shall only be supplied using sprinklers (Sprinkler(1,2, 3, 4)=1), with specific ratios. All humidity sensors are on master mode. And all display devices are disabled.

TABLE. I. INSTANCES OF CONTEXT VARIABLES

Power Failure	Rain	Light	Max Light	Tank Capacity	Tank Level
0	1	51509	46332	20000	7681
MinTank Water	**Mains Water Consumption**	**MaxMains Consumption**	**Smartphone Availability**	**Month / DayOfMonth**	**Time**
998	220977	200000	1	11/ 2	15

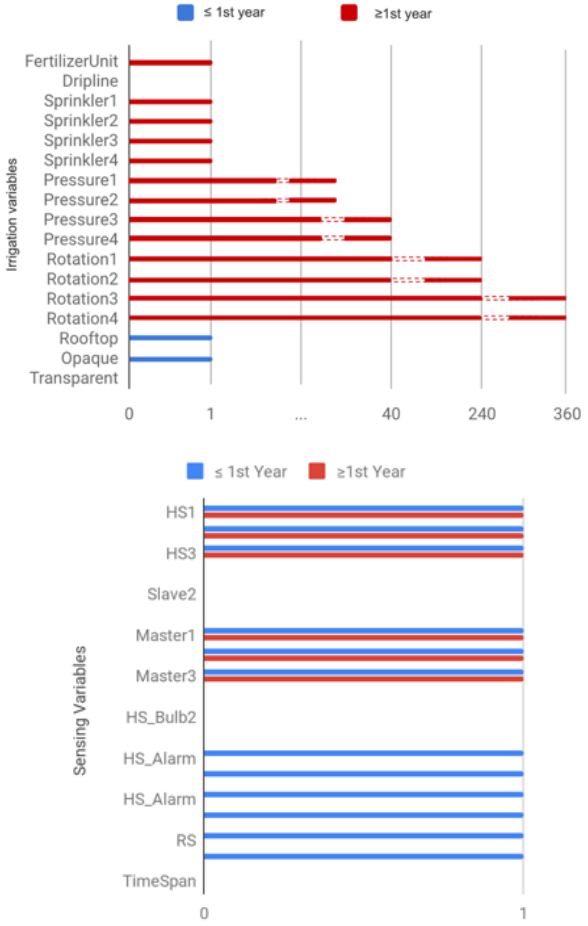

Fig. 7. The Configuration of the SIS (Scenario 2).

VI. CONCLUSION AND PERSPECTIVES

The investigation of the domain of smart irrigation systems shows that contemporary ecological challenges on the one hand, and society's constant need for sophisticated technology on the other hand are big factors that impact the state of the art of the internet of things in general, and smart irrigation systems in particular. Consequently, the generation of SIS, that simply consisted of reporting on environmental data, and reacting or planning irrigation cycles, is on the verge of becoming obsolete. This is mainly a result of user's requirements that are perpetually changing, and technology that is constantly evolving.

The action taken in this paper consists of representing SIS's components as variables, and their relations as constraints, forming a dynamic constraint satisfaction problem. Through a solver, this paradigm finds a solution that (best) fits the (dynamic) requirements of users. Therefore, instead of enrolling a SIS that has a set configuration, designing it as a dynamic and flexible constraint satisfaction problem allows the specification of these dynamic requirements, and therefore, the generation of configurations that fully or partially satisfy them, on the go. Thanks to sensors and various types of data that can be provided to irrigation actuators, SIS can now be designed to accompany dynamic environments, but also, dynamic requirements in time and place.

The snippets of constraint code introduced above are manually written in a Minizinc editor. Being that as it may, it is a fastidious, time consuming and illegible way of documenting constraints. Implementing the main concepts to build such programs proves essential. Therefore, as a perspective, a language that allows the specification of dynamic requirements in fleets of connected devices, and enables the generation of configurations that comply with real time contexts proves substantial.

ACKNOWLEDGMENTS

This work was supported by the Moroccan « Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de la Formation des Cadres », by the « French Embassy in Morocco », and by the « Institut Français du Maroc ».

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Futur. Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, 2013.

[2] D. Delgoda, H. Malano, S. K. Saleem, and M. N. Halgamuge, "Irrigation control based on model predictive control (MPC): Formulation of theory and validation using weather forecast data and AQUACROP model," Environ. Model. Softw., vol. 78, pp. 40–53, Apr. 2016.

[3] T. C. Meyer and G. P. Hancke, "Design of a smart sprinkler system," in IEEE Region 10 Annual International Conference, Proceedings/TENCON, 2016.

[4] N. Sales, O. Remedios, and A. Arsenio, "Wireless sensor and actuator system for smart irrigation on the cloud," in 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 693–698.

[5] X. Zhang, J. Zhang, L. Li, Y. Zhang, and G. Yang, "Monitoring Citrus Soil Moisture and Nutrients Using an IoT Based System," Sensors, vol. 17, no. 3, p. 447, Feb. 2017.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Comput. Networks, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[7] R. L. Baskerville and A. T. Wood-Harper, "A critical perspective on action research as a method for information systems research," J. Inf. Technol., vol. 11, no. 3, pp. 235–246, Sep. 1996.

[8] C. Kamienski et al., "Smart Water Management Platform: IoT-Based Precision Irrigation for Agriculture.," Sensors (Basel)., vol. 19, no. 2, p. 276, Jan. 2019.

[9] H. Luo, P. Yang, Y. Li, and F. Xu, "An Intelligent Controlling System for Greenhouse Environment Based on the Architecture of the Internet of Things," Sens. Lett., vol. 10, no. 1, pp. 514–522, Jan. 2012.

[10] F. Al-Turjman, "Introduction to Cognition in IoT," in Cognitive Sensors and IoT, Routledge, 2017, pp. 1–4.

[11] B. M. Hasan Alhafidh and W. Allen, "Design and Simulation of a Smart Home managed by an Intelligent Self-Adaptive System," 2016.

[12] Y. Eustache and J.-P. Diguet, "Specification and OS-based implementation of self-adaptive, hardware/software embedded systems," in Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis - CODES/ISSS '08, 2008, p. 67.

[13] FAO, "Global agriculture towards 2050," High Lev. Expert Forum-How to Feed world 2050, pp. 1–4, 2009.

[14] Credence Research, "Global Precision Irrigation Systems Market-Growth, Share, Opportunities and Competitive Analysis, 2016 – 2023," California, 2016.

[15] N. Gondchawar and P. R. S. Kawitkar, "IoT based Smart Agriculture," Int. J. Adv. Res. Comput. Commun. Eng., vol. 5, no. 6, pp. 838–842, 2016.

[16] P. Rajalakshmi and S. Devi Mahalakshmi, "IOT based crop-field monitoring and irrigation automation," Proc. 10th Int. Conf. Intell. Syst. Control. ISCO 2016, 2016.

[17] M. Roopaei, P. Rad, and K. K. R. Choo, "Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging," IEEE Cloud Comput., vol. 4, no. 1, pp. 10–15, 2017.

[18] A. Gulati and S. Thakur, "Smart Irrigation Using Internet of Things," in 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018, pp. 819–823.

[19] "GreenIQ Help Center." [Online]. Available: https://support.greeniq.com/hc/en-us.

[20] "Rachio 3 Smart Wi-Fi Sprinkler Controller - New Home Watering Solution." [Online]. Available: https://www.rachio.com/rachio-3/.

[21] IBM, "Autonomic Computing White Paper: An Architectural Blueprint for Autonomic Computing," IBM White Pap., no. June, p. 34, 2005.

[22] J. Andersson et al., "Software Engineering Processes for Self-Adaptive Systems," Softw. Eng. Self-Adaptive Syst. II, pp. 51–75, 2013.

[23] R. Dechter and J. Pearl, "Network-Based Heuristics for Constraint-Satisfaction Problems," in Search in Artificial Intelligence, New York, NY: Springer New York, 1988, pp. 370–425.

[24] E. C. Freuder and R. J. Wallace, "Partial constraint satisfaction," Artif. Intell., vol. 58, no. 1–3, pp. 21–70, Dec. 1992.

[25] D. Hughes et al., "An experiment with reflective middleware to support grid‐based flood monitoring," Concurr. Comput. Pract. Exp., vol. 20, no. 11, pp. 1303–1316, Aug. 2008.

[26] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes, "Constraint Programming as a Means to Manage Configurations in Self-Adaptive Systems," Spec. Issue IEEE Comput. J. "Dynamic Softw. Prod. Lines," vol. 45, no. October 2015, pp. 56–63, 2012.

[27] A. Achtaich, N. Souissi, R. Mazo, O. Roudies, and C. Salinesi, "A DSPL Design Framework for SASs: A Smart Building Example," EAI Endorsed Trans. Smart Cities, vol. 2, no. 8, p. 154829, Jun. 2018.