Law and Adaptivity in Requirements Engineering

Silvia Ingolfo
Department of Information Engineering and Computer Science,
University of Trento, Italy
ingolfo@disi.unitn.it

Vítor E. Silva Souza*
Computer Science Department,
Federal University of Espírito Santo (Ufes), Brazil
vitorsouza@inf.ufes.br

***Abstract*—The great impact that law has on the design of software systems has been widely recognized in past years. However, little attention has been paid to the challenge of coping with variability characterizing the legal domain (e.g., multiple ways to comply with a given law, frequent updates to regulations, different jurisdictions, etc.) on the design of software systems. This position paper advocates the use of adaptation mechanisms in order to support regulatory compliance for software systems. First we show an example of how *Zanshin*, a requirements-based adaptation framework, can be used to design a system that adapts to legal requirements to accommodate legal variability. Then we examine how legal texts can be analyzed as sources for parameters and indicators needed to support adaptation. As motivating running example we consider legal situations concerning the Google driverless car and its recent legalization in the highways of Nevada and soon also in California.**

***Index Terms*—Legal variability, Regulatory compliance, Adaptation framework, Requirements engineering**

I. Introduction

The influence of law on the design of software is growing as new legislations around the world attempt to control its impact on social and private life: tablet application for doctors to share a patient's record in accordance with privacy policies, on-line systems to pay taxes obeying fiscal laws, and even driverless cars respecting traffic law. All software systems need to be designed from the beginning in a law-aware fashion to make sure they comply with applicable laws.

Ensuring compliance of a software is a very expensive proposition. For example, in the Healthcare domain alone, it has been estimated that organizations have spent US$17.6 billion over a number of years to align their systems and procedures with the Health Insurance Portability and Accountability Act (HIPAA). Non-compliance costs (fines, prosecutions, etc.) are generally almost three times higher than the costs of ensuring compliance [1]. As a result of this great impact that regulations have on requirements, new techniques have been developed to allow organizations to align their software requirements with the law [2], [3], [4].

The variability characterizing the space of compliance solutions affects software in different ways. Introducing a new requirement or an amendment to an existing one could make a new set of norms applicable or trigger a particular exception for which a norm is no longer applicable. Amendments to a law often introduce new rules but also new ways to comply. For example a little over a year ago some changes were proposed to the HIPAA Privacy Rules:[1] as these changes come into force, the software systems where the HIPAA Section was applicable need to be realigned to meet these new provisions. Software systems need therefore to be designed so that they adjust and adapt gracefully to such changes.

Moreover, legal variability comes also from differences in legislations. This issue is very prominent in the case of traffic laws and regulations. Recently, the American Insurance Institute for Highway Safety (IIHS) launched an initiative to promote awareness of existing differences in state traffic laws. For example, even though in most states the minimum age for unsupervised driving is 16, across the US this age ranges from 14 to 17.[2] It becomes evident that any system supporting a human driver needs to be aware and adapt depending on the legal context.

Uncertainty from the environment and variability of solutions are amongst the topics of study in the research area of adaptive software system design [5]. Adaptive systems treat this uncertainty in a dynamic way, changing the system's behavior to a different variant when needed. Likewise, the law needs to be handled dynamically in order to accommodate differences in legislation and changes in regulations. For instance, in an autonomous car, drivers could indicate the maximum amount of dollars they are willing to spend for speeding tickets, and then driving across different states the car settings adapt according to applicable limits and fines.

In this paper we report on very early work, on the application of adaptive software system design techniques to the problem of designing compliant software. As in our earlier work, our approach is founded on concepts adopted from Requirements Engineering (RE). On one side existing adaptation techniques can be tailored to accommodate legal compliance requirements. "Legal requirement" is a type of software requirement that reflects the need for the software to comply with applicable laws.[3] We illustrate this with an RE-based approach for the design of adaptive systems called *Zanshin* [7], [8]. On the other side, we illustrate how legal texts can be useful sources for adaptation rules, and sketch a preliminary methodology to deal with this issue.

* Work done while in Trento as PhD student/post-doc.

[1]http://www.hhs.gov/news/press/2011pres/05/20110531c.html.

[2]http://www.iihs.org/laws/mapunsuperviseddrivingage.aspx.

[3]For example, "The software [in charge of driving a driverless vehicle] should respect a stop sign", or "The software [managing patients record in a hospital] should not disclose Patient Health Information to third parties". Once identified, these legal requirements can be added to other elicited requirements [6].

Our focus on RE is motivated by the role requirements play both in law compliance and adaptation. In the former, applicable laws have a great impact in the system's requirements and, as such, should be analyzed in combination [9], [2]. As for the latter, adaptation takes place when the system is not fulfilling requirements, and constitute the most important element of any feedback loop [10].

The rest of the paper is structured as follows. Section II introduces the running example of a driverless car. In section III we apply the *Zanshin* adaptivity framework to the running example, while in section IV we propose a high-level methodology for extracting useful information from legal texts. Finally section V concludes.

II. The Google Driverless Car Example

A driverless car — also self-driven car, robotic car or autonomous car — is "*an autonomous vehicle capable of fulfilling the human transportation capabilities of a traditional car. As an autonomous vehicle, it is capable of sensing its environment and navigating on its own. A human may choose a destination, but is not required to perform any mechanical operation of the vehicle*" [11]. Many car companies are working on this type of projects,[4] however it is thanks to Google that a new prototype was recently put on the streets.

In June 2011 in fact, the US state of Nevada passed a law concerning the operation of driverless cars in its highways and less than a year later the DMV of Nevada issued the first license for a "self-driven car". In the following year, a bill was proposed in California to "establish rules and regulations covering the safe operation of driverless cars on the state's highways" [12].

The Google project is at the moment still in testing phase and the car never operates unmanned [13]. These vehicles are equipped with radars, cameras and lasers, and are programmed to drive autonomously in traffic. According to the New York Times [14], the Google driverless car drove at the speed limit along the highways of Nevada, left the freeway, drove through the city traffic of Mountain View (stopping at stop signs and for red lights), and a voice even announced the directions the car was following (like "turning left ahead").

For purposes of this paper, we consider that the software system that manages the car is composed of three main parts:[5]

1) A monitoring system: a part of the software that is in charge of managing the special equipment needed to sense the environment (cameras, sensors, lasers, etc.);
2) A navigation system: a part of the software that manages the geo-localization of the car and chooses the directions to the destination (usually performed by a GPS system);
3) A vehicle operator: a piece of software that manages the car in all its basic features (drive in the traffic, obey traffic rules, etc.) and takes care of the user request.

[4]E.g., GM: http://www.wired.com/autopia/2008/01/gm-says-driverl/.

[5]Given the limited scope of the example, we will focus on the requirements of the software parts related to the operation of the car. We will skip the requirements of the passenger, who is nonetheless an important actor in the driverless car software system.

In the running example we focus on the requirements of the software that takes care of the operation of the vehicle (the element 'vehicle operator'). First when a software operates a vehicle, it needs to be able to *manage the mechanic components* (engine, break, clutch, etc.) in order to actually have the vehicle respond to basic commands (like stopping, moving, accelerating, etc.). Then, the software should provide some basic features allowing the car to operate in an *environment* with other elements (like avoiding collision with other vehicles). Also, the software operating the vehicle has to manage *user requests*, providing the passenger with the possibility to decide/change the destination, and other operations regarding the possible configuration of the car (e.g. driving mode). Furthermore when operating the vehicle on the streets, this task needs to be performed in accordance with the *traffic law*, so it must be aware of the rules and limitation that are given by the State/Country where it operates. These four basic features are the top requirements of this software component, and an example of a high-level goal model of these requirements is illustrated in Fig. 1.

III. Adaptation of Legal Requirements

Requirements-based approaches for the design of adaptive systems are concerned with, among other things, the elicitation of requirements for adaptation. Going through the literature in this area of research (e.g., [15], [16]), one can find many different proposals for adaptive systems design, many of which focus on requirements.

As previously introduced, laws can have a big impact in many kinds of software systems, some of which might also require adaptation features to deal with its own complexity or the uncertainty of its surrounding environment. In this section, we show that existing techniques for the design of adaptive systems can be used to accommodate the need to adapt to legal requirements. In particular, we will use the *Zanshin* framework [7], [8] to illustrate examples of this need.

Zanshin is based on the idea that adaptivity is implemented by a monitor-adapt feedback loop that reads from the system's requirements model what should be monitored and what to do in case monitoring indicates failures. These are represented in the model by *Awareness Requirements* (*AwReqs*) [7] and *Evolution Requirements* (*EvoReqs*) [8], respectively.

On the monitoring side, *AwReqs* represent constraints on the states that other requirements can assume during their execution at runtime. Back to the vehicle operator component of the driverless car shown in Fig. 1, examples of *AwReqs* could be "goal *Avoid collision* should never fail" or "task *Use GPS to locate (a parking spot)* should have 75% success rate". These *AwReqs* could be elicited, for instance, by asking stakeholders about features that are essential to guarantee the quality of the service provided by the system.

The vehicle operator of the driverless car has to obey the traffic laws and, consequently, its specifications also include legal requirements such as the goal *Obey traffic law* and its refinements. The feedback loop mechanism applies to both types of requirement as we need to be sure that also the

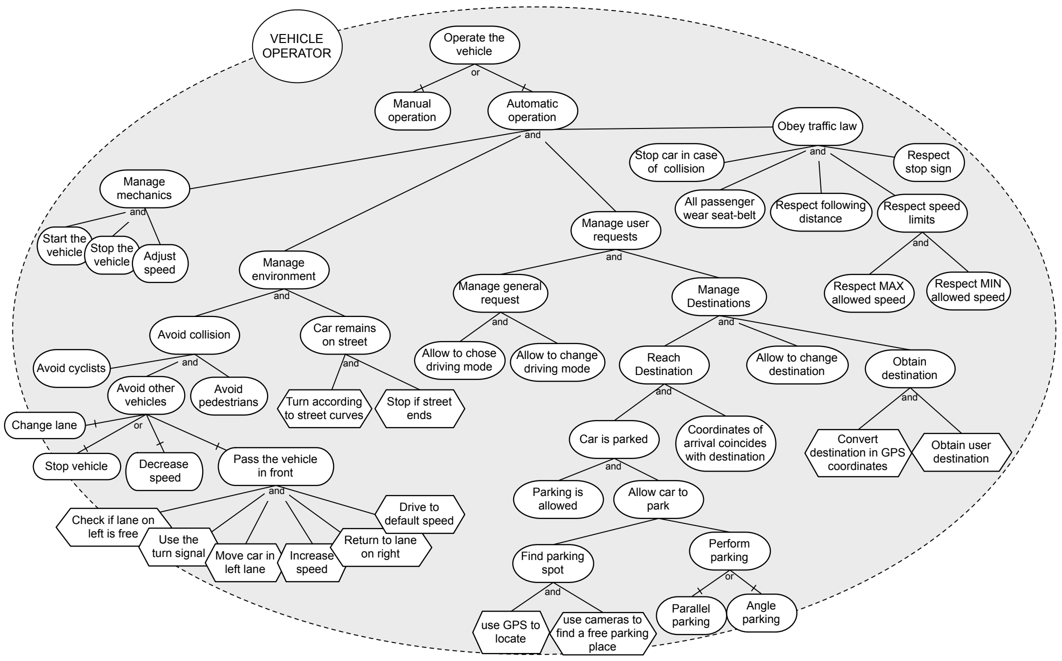

Fig. 1. A goal model representing requirements of the vehicle operator software component.

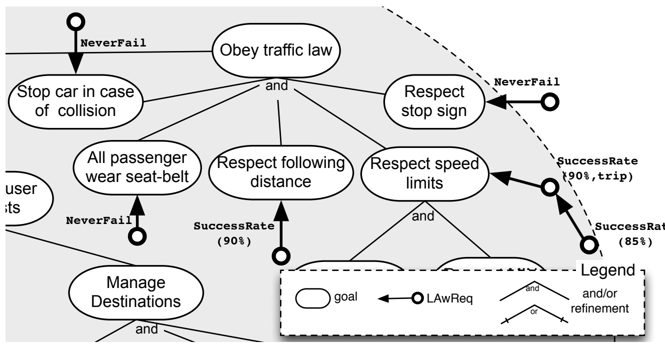

Fig. 2. LAwReq examples elicited based on the Driverless Car model.

legal prescriptions that apply to our system are respected. In this spirit, we categorize these *AwReqs* as *Legal Awareness Requirements* (*LAwReqs*): the class of legal requirements that lead to feedback loop functionalities. In other words, *LAwReqs* will talk about the states that *legal requirements* can assume at runtime. This distinction, however, is for categorization purposes only, and does not affect the *Zanshin* framework.

Fig. 2 shows a few examples of *LAwReqs*, represented using the same syntax as regular *AwReqs* (cf. [7]). Regulations that are identified as being more critical — such as stopping in case of collision, wearing seat-belts and respecting stop signs — are associated with "never fail" constraints, whereas less critical parts of the law have their deniability (failure) tolerated up to certain points: the car should respect both the following distance and the speed limits 90% of the time. For more details on how *Zanshin* operationalizes monitoring, refer to [7].

On the adaptation side, *EvoReqs* prescribe what to do in case an *AwReq* fails, in order to adapt the system. As before, this component of the feedback loop could be applied to *LAwReqs* as well, specifying counter-measures whenever the system has reached a level of noncompliance with the law that should not be tolerated. Adaptation strategies can consist of precise actions, including changes in the model itself. For example, if a passenger is not wearing her seat-belt, the driverless car first issues a warning, then, if the problem persists, it can decrease its speed and ultimately park itself, resuming the trip only when the situation is resolved.

Another strategy for adaptation is to look for a *system parameter* that can be reconfigured the same way a control system (e.g., a thermostat) tunes its variables (the heating/cooling power) to keep its output (the room temperature) as close as possible to a desired value. The speed of the car, the driving style (e.g., conservative, aggressive), preferable routes (e.g., highway, in the city), etc. are examples of possible parameters for the driverless car.

In the above, we have seen legal requirements as targets for monitoring and adaptation. However, the law can also be used as a source in the elicitation of these requirements for adaptation. In the next section, we discuss this issue, sketching an approach for the design of adaptation requirements based on legal documents.

IV. Law as Source of Adaptation

An important aspect of software adaptation comes from the identification of information representing the relationship among the parameters of the system and indicators that it is operating properly (in *Zanshin*, for example, *AwReqs* represent such indicators). The inclusion of these elements in the requirements model provides an essential link between the possible system configurations and its measured output, supporting the design of adaptation features.

In this section we claim that legal texts can be useful sources for identifying new indicators of requirements convergence and parameters that can be tuned at runtime to help maintain such indicators close to desired reference values. In other words, the law can be the source of requirements for adaptation. The support of legal variability (e.g., difference in legislation, changes in regulations) can then be supported by dynamically configuring and managing these indicators and parameters.

Existing solutions for text-based analysis of legal documents are generally aimed at identifying the rules/norms[6] (i.e., rights, obligations, permissions, etc.) the system should comply with. In RE for example, Maxwell et al. [4] show how production rule models can be used to extract software requirements from regulatory texts. Breaux et al. [3] propose a methodology to extract rights and obligations from legal texts using semantic models in order to help the analysts establish compliance of a set of requirements. In the well established field of AI&Law for example, Agnologni et al. [19] adopt an ontology-based approach in order to support both the formalization of normative rules, and the link between these rules and the corresponding part in a business model. Wyner et al. [18] show how a linguistically-oriented approach can identify and extract high-level elements of normative rules from regulations (e.g., agents, deontic modals, exception clauses, etc.). Biagioli et al. [17] explore automatic methodologies for helping the manual identification of the type of normative rule and its elements. Palmirani et al. [20] propose a model for recognizing, understanding and normalizing the normative references of legal texts and standardize such references to increase interoperability of information systems.

All these techniques and methodologies provide an important basis for the textual analysis of legal documents. AI&Law techniques generally focus on the identification of the legal elements and components of the normative rules in the legal text, while RE approaches look for a methodological link between legal texts and a compliant set of software requirements. The idea of this paper is that legal texts can be a useful source for enriching the requirements model of an adaptive system, and a specific solution is missing to combine existing techniques to provide effective support in the enrichment of such models. Despite the fact that compliance with laws is not necessarily the main goal of an adaptive system, it is still very important that it follows and respects the applying regulations. This is especially the case, for instance, of driverless cars.

The enrichment of a model of requirements with elements helpful for adaptation (e.g., parameters and indicators) is a key and non-trivial task [21], [22], [23]. Automatic text-based techniques can be envisaged to identify these elements in legal texts, however, requiring these elements to be 'measurable' may greatly increase the complexity of this automatic task. Analysts — who are not necessarily familiar with the legal domain at hand or its details — may need guidance to structurally enrich these models. Thus, we need a methodology for augmenting a goal model of the requirements of an adaptive system with parameters and indicators coming from the law.

As mentioned earlier, the differences between traffic laws in the US is an evident example of legal variability. The software running on a driverless car (or a hypothetical '*smart*' car equipped with a software to detect and limit the speed according to the type of road) must be able to adapt to the different rules and limitations across the states the vehicle is running. The use of parameters and indicators can be indeed of great help to bridge these differences. The single parameter would remain the same in the model (e.g., `speedOfCar`, representing the speed the car should be currently running, or `numberOfDrinks` representing the max number of drinks the car would allow a driver to have) while the official legal value (e.g., $\texttt{speedOfCar} < 50\,\text{Mph}$, $\texttt{numberOfDrinks} < 2$) would be dynamically configured and evaluated according to the location of the vehicle (e.g., California with maximum speed 50 Mph, while Nevada 55 Mph).

In the following we will sketch a preliminary methodology that takes as input a goal model of the requirements of the system (e.g., Fig. 1) and a piece of law. As output it returns the requirements model expanded with parameters and indicators coming from the legal text. As an example of legal text, we will use an extract of the article of the California Vehicle Code regulating the driving hours:[7]

> §21702.(a) No person shall drive [...] for more than 10 consecutive hours nor for more than 10 hours spread over a total of 15 consecutive hours. Such person shall not drive any such vehicle until eight consecutive hours have elapsed. Regardless of aggregate driving time, no driver shall drive for more than 10 hours in any 24-hour period unless eight consecutive hours off duty have elapsed.

Step 1. The first step of the methodology is to *identify the object regulated* in the analyzed piece of law at hand. We call *object* the element that the law is regulating and for which rules and limitations are in place, e.g., the driver's blood alcohol content (BAC), the number of passengers, the speed of the car, the distance of the car from an object, the way a car can/cannot be operated, etc. In the example above, the object regulated is the amount of hours a vehicle is driven by the same person. As mentioned earlier, this step could benefit from automatic text-based techniques to help identifying or suggesting possible regulated objects in the legal text.

Step 2. As second step we need to evaluate if the *object is measurable and quantifiable* (in general), and if it can be measured by the system. Once again, specific automatic techniques could be of help and support this task with suggestion to the analysts. In the example above, the amount of hours can be measured by a piece of software and no special equipment would be needed in a car in order to quantify the number

[6]In the field of AI&Law the term 'rule' or 'provision' [17], [18] is generally used to identify what in RE is called 'norm' [3], [6].

[7]As stated in Section II, currently these types of car cannot be operated unmanned so all rules are still in force for the person on board. For the complete text of the law, see http://www.dmv.ca.gov/pubs/vctop/d11/vc21702.htm.

of driving hours. On the other hand, for example, the BAC is a measurable and quantifiable object but cars are usually not equipped with devices for its testing. Should the designed system be aimed at making drivers more aware of this very important issue, it could be arranged to equip the vehicle with a breathalyzer and evaluate to include this device in the design of the system. An example of a regulated object that cannot be measured/quantified is the way a car can/cannot be operated (e.g., §21712(d) "A person shall not ride in the trunk of a motor vehicle").

Step 3. As third and last step we need to evaluate, for each object analyzed in the previous step, *if and which adaptation-related elements to include* in the requirements model. As briefly introduced in Section III, in *Zanshin* such elements could be *LAwReqs*, *EvoReqs* or system parameters.

Measurable objects: if the object is measurable by the system and is relevant enough to be used as an indicator, a *LAwReq* can be added to the model. For example, in Fig. 2, *LAwReqs* are associated with goals such as *All passengers wear seat-belts* and *Respect speed limits* presumably because different pieces of the law indicated these behaviors as mandatory.

In the case of article §21702.(a) shown earlier, a new goal *Respect limitations on driving hours* should be added to the model with an associated *LAwReq*, as shown in Fig. 3. The way the goal is operationalized depends on whether the object is *controllable or not by the system*. When *not controllable*, we refine the goal into a *domain assumption* and monitor the information from the environment. As can be seen in the figure, this is the case of the driving hours object.

When an object is *controllable* by the system, the latter can take a more proactive approach into satisfying the goal. Take, for instance, the BAC example previously mentioned. To satisfy goal *BAC within limit* we could install a breathalyzer and not start the car until the driver has blown in it and the BAC has been verified. In these cases, we refine goals into *tasks*.

Non-measurable objects: if an object cannot be directly measured by the system or a solution to do it is not feasible, then a proxy solution could be envisaged by the analyst in order to evaluate the legal object. This is actually the case shown in Fig. 3 for goal *BAC within limit*, which is operationalized by task *Ask user for number of drinks*.

System parameters: controllable objects, regardless if relevant enough to lead to indicators, can become system parameters if they have an effect on the success rate of an *AwReq*.[8] For instance, other articles from the California Vehicle Code mention the car's current speed. Clearly, the value of this particular element affects the satisfiability of goal *Respect speed limits* and, thus, the *LAwReq* associated to it.

As shown in Fig. 3, parameters are represented in the model by diamonds connected to the elements on which they have an effect (e.g., `speedOfCar` in Fig. 3).

Counter-measures: finally, the analysis of legal text can also lead to the precise adaptation strategy to be used in case a

[8]The cross-effect analysis of parameters and indicators is a task addressed by a dedicated process in the *Zanshin* framework.

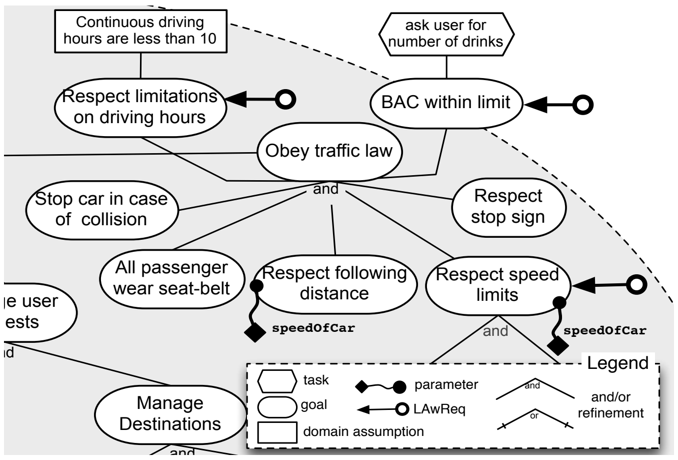
Continuous driving hours are less than 10
ask user for number of drinks
Respect limitations on driving hours
BAC within limit
Obey traffic law
and
Stop car in case of collision
Respect stop sign
All passenger wear seat-belt
Respect following distance
Respect speed limits
speedOfCar
speedOfCar
and
Manage Destinations
Legend
task
goal
domain assumption
parameter
LAwReq
and/or refinement

Fig. 3. An example of how new goals and indicators coming from the law can be included in the Driverless Car model.

LAwReq is not satisfied, leading to the inclusion of *EvoReqs* in the requirements model. In the case of article §21702.(a), for instance, parking the car and not allowing the same driver to drive it for the next eight hours is a possible response to the failure of the *LAwReq* associated with goal *Respect limitations on driving hours*.

LAwReqs can also be set up in order to prevent the system from breaking the law instead of just remedying the situation. For instance, a different domain assumption considering not 10, but 8 or 9 driving hours could be elicited with an associated *LAwReq* that, when not satisfied, would configure the car with a more cautious setting (e.g., using lower speeds) and periodically notify the driver until it reaches the limit of hours.

In this preliminary methodology we have shown how a legal text can be used as source to elicit parameters and indicators for adaptation. The enrichment of a requirements model with these elements can provide an important support in dealing with the problem of legal variability caused by differences in the legislations. Future work will be dedicated to the investigation of a more complete methodology, in order to provide thorough support to the analyst. Moreover, in order to deal with the more complex problem of applicability of norms — related to the variability caused by elements such as exceptions and derogations in legal texts — we will investigate the possibility of linking the requirements model directly with a model of the law that can deal this issue (e.g., Nòmos 2 [9]).

V. Conclusions, Limitations, and Future Work

In this position paper we have presented and characterized the important relationship between law and adaptation in requirements engineering. In the *Zanshin* framework [7], [8], adaptation is founded on the idea that if in a current configuration the system is not satisfying its requirements, a new configuration is found that resolves the failure. However when the law comes into play, you look for an alternative way to comply with it, thereby changing the space of alternatives that is available for adaptation. When a system is not compliant, it can either change its behavior or you look for a different part of the law that you can comply with. The variability characterizing the legal domain creates the need for software systems to adapt and respect legal rules. At the same time, this variability creates an important new space of alternatives for adaptation.

We have shown how *Zanshin* can accommodate legal requirements making the system adapt and reconfigure according to the rules specified in the law. Furthermore, legal texts can be a source for identifying new information useful for the adaptation of a software. We have sketched a preliminary methodology that takes as input a requirements model and a law fragment, and returns the model enriched with parameters and indicators coming from legal texts.

We acknowledge that many limitations and challenges remain open in this exploratory study, and need to be addressed in our future work. First, our motivational and running example of the driverless car system has mainly risen from the recent news related to the regulation and legalization of this type of vehicle in two US states. We recognize that this example belongs to a safety-critical type of systems where the use of a more formal approach might be more appropriate. In our future work we plan to identify and experiment with a system that is more suitable to the level of (in)formality associated with our approach.

Secondly, the use of legal texts in our methodology could suffer from the many challenges and difficulties characterizing this type of text. For example, vague and fuzzy words — "driving *too slow*" or "follow *closely* another vehicle" — can be rationalized and operationalized in different ways by analysts. Ambiguous terms typical of legal texts may need to be interpreted and disambiguated for the legislation to be correctly implemented. In our future work we will consider specific techniques and methodologies that could be integrated to help the analyst deal with these particular traits of legal texts. Also we will investigate the possibility to exploit models of law (e.g., Nòmos 2 [9]) that could help clarifying and modeling complicated regulations (e.g., legislation with many exceptions, derogations, or conflicting rules).

Moreover, in this exploratory paper we have seen how the *Zanshin* framework can be deployed to accommodate legal requirements. It would be interesting to try out other approaches from the literature [10].

Lastly, the scalability of the overall approach and methodology needs to be evaluated for a more accurate assessment of the feasibility of our proposal.

Acknowledgment

This work has been supported by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution" (April 2011 – March 2016, http://www.lucretius.eu) as well as Brazilian foundation FAPES (http://www.fapes.es.gov.br) through the PRONEX grant #52272362.

References

[1] Ponemon, "The true cost of compliance. research report," 2011, accessed on September 21, 2011. [Online]. Available: http://www.tripwire.com/ponemon-cost-of-compliance/pressKit/True_Cost_of_Compliance_Report.pdf

[2] S. Ingolfo, A. Siena, and J. Mylopoulos, "Establishing regulatory compliance for software requirements," in *Conceptual Modeling - ER 2011*, ser. Lecture Notes in Computer Science, 2011, vol. 6998, pp. 47–61.

[3] T. D. Breaux, M. W. Vail, and A. I. Antón, "Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations," in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006.

[4] J. C. Maxwell and A. I. Anton, "Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts," in *RE'09*, 2009, pp. 101–110.

[5] B. H. C. Cheng *et al.*, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems*. Springer, 2009, pp. 1–26.

[6] A. Siena, S. Ingolfo, A. Susi, I. Jureta, A. Perini, and J. Mylopoulos, "Requirements, intentions, goals and applicable norms," in *ER Workshops*, 2012, pp. 195–200.

[7] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness Requirements for Adaptive Systems," in *Proc. of the 6$^{\text{th}}$ International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011, pp. 60–69.

[8] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, "(Requirement) Evolution Requirements for Adaptive Systems," in *Proc. of the 7$^{\text{th}}$ International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2012, pp. 155–164.

[9] A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, and J. Mylopoulos, "Capturing variability of law with Nòmos 2," in *Conceptual Modeling - ER 2012*, ser. Lecture Notes in Computer Science, 2012, vol. 7532, pp. 383–396.

[10] Y. Brun *et al.*, "Engineering Self-Adaptive Systems through Feedback Loops," in *Software Engineering for Self-Adaptive Systems*, 2009, pp. 48–70.

[11] Wikipedia, "Autonomous car," 2010. [Online]. Available: http://en.wikipedia.org/wiki/Driverless_car

[12] M. Williams, "Driverless cars move closer in California," 2012. [Online]. Available: http://www.computerworld.com/s/article/9228756/Driverless_cars_move_closer_in_California?taxonomyId=144

[13] G. O. Blog, "What we're driving at," 2010. [Online]. Available: http://googleblog.blogspot.it/2010/10/what-were-driving-at.html

[14] J. Markoff, "Google cars drive themselves, in traffic," 2010. [Online]. Available: http://www.nytimes.com/2010/10/10/science/10google.html?_r=1

[15] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds., *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 5525.

[16] R. de Lemos et al., Ed., *Software Engineering for Self-Adaptive Systems II*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 7475.

[17] C. Biagioli, E. Francesconi, A. Passerini, S. Montemagni, and C. Soria, "Automatic semantics extraction in law documents," ser. ICAIL '05, 2005, pp. 133–140.

[18] A. Wyner and W. Peters, "On rule extraction from regulations," in *JURIX*, 2011, pp. 113–122.

[19] T. Agnoloni and D. Tiscornia, "Extracting normative content from legal texts," in *MCIS*, 2010, p. 4.

[20] M. Palmirani, R. Brighi, and M. Massini, "Automated extraction of normative references in legal texts," ser. ICAIL '03, 2003, pp. 105–106.

[21] S. Fickas and M. S. Feather, "Requirements Monitoring in Dynamic Environments," in *Proc. of the 2$^{\text{nd}}$ IEEE International Symposium on Requirements Engineering*. IEEE, 1995, pp. 140–147.

[22] B. H. C. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty," in *Model Driven Engineering Languages and Systems*, 2009, pp. 468–483.

[23] W. N. Robinson and S. Purao, "Monitoring Service Systems from a Language-Action Perspective," *IEEE Transactions on Services Computing*, vol. 4, no. 1, pp. 17–30, 2011.