Component Based Decision Architecture for Reliable Autonomous Systems

Arunkumar Ramaswamy*† and Bruno Monsuez* and Adriana Tapus*
*Department of Computer and System Engineering,
ENSTA-ParisTech, 828 Blvd Marechaux, Palaiseau, France
†VeDeCom Institute, Foundation Mov'eoTec,
2 place Touraine, 78000 Versailles, France
Email:{arun-kumar.ramaswamy; bruno.monsuez; adriana.tapus}@ensta-paristech.fr

***Abstract*—Several decision making algorithms that are developed in robotics domain are found to be useful in automotive industrial applications. In order to use these algorithms in safety critical embedded systems one has to ensure a minimum confidence level, quality assurance, and reliability. In addition, decision algorithms are designed and developed by domain experts and system integrators do not have control over the performance of low level components. Hence, there is a compelling need for a scalable framework, which accelerates the system integration and at the same time is conform to safety levels. This position paper presents a Quality of Service (QoS) based component architecture so as to address this problem. Our approach introduces a component model that segregates the functional and non-functional aspects of decision making.**

I. Introduction

Efficient decisional algorithms are back bone for a system that operate with limited intervention from humans. Decision making, in cognitive sense, arises from arbitrating several mental processes. Similarly, in autonomous systems, the decision outcome is the result of several data processing algorithms in different dimensions. Different algorithms solving various robotics problems have been developed and exposed in the robotics literature. However, there is very limited reuse of existing implementations for building a new application. Component based approaches are widely discussed and recommended for overcoming such problems. Component Based Software Engineering (CBSE) emphasis on the reusable, off-the-shelf components for building an application by abstracting component interfaces with the implementation [1]. The main issue that arises when such heterogeneous components are combined, is in ensuring reliability and safety at the system level. In cyber-physical systems execution correctness encompass both functional and non-functional properties. However, only functional semantics is treated in traditional programming languages. Software modules built on such programming languages becomes unreliable unless it is handled at the component level in the hierarchy. Adaptable components should provide a way to incorporate Non-Functional Attributes (NFAs) along with the data that they consume and produce. System engineers take into account the non-functional properties of components to characterize the system architecture. It necessitates a mechanism in which the NFAs propagate in component interactions so that it will assist system engineers to provide a better architecture for target applications.

The confidence level of the components does not depend only on the efficiency of the implemented algorithm but also on the context in which it is executed such as resource availability, external disturbance, etc. For example, if a component is implementing localization using adaptable particle filters, the quality and response time depends on the number of particles used, which in turn depends on available memory during execution time, the state uncertainty level [2], etc. The component developer cannot test NFAs unless it considers a Worst Case Execution Time (WCET) anticipating the available resources or its component should be already incorporated in the target system. Nonetheless, that will fail the very purpose of reusable component for a reconfiguring system.

Component abstraction using NFAs along with its functionality are crucial for safety critical embedded system. In this work, we also propose extending the component abstraction for parameterizing human behaviors which will help verifying the system where humans are involved in decision making. It is difficult to assess the quality of collaborative decision making unless such modularization of human behavior are incorporated in the architecture.

The paper is organized as follows: Section II reviews related works. Section III identifies several essential features for a decision making architecture. Section IV describes our proposed approach with the help of an example of multisensor fusion in mobile robots. Section V introduces a component model that justifies the approach. Componentizing human behavior is discussed in section VI. And finally, Section VII concludes our paper.

II. Related Works

Hierarchical composition of components are well studied in Ptolemy project [3]. It proposes a model structure and semantic framework to address the issues with heterogeneous component interactions. Computation and communication are clearly separated in order to reuse the component functionality across different domains. But there is little framework support for decision fusion and dynamic configuration of components. In [4] an emergent architecture, which provides services for integration of components is proposed. Reconfigurable interfaces that contain mediators and adapters provide basic infrastructure and the system integrator is free to choose any architecture. Such flexibility may not be reliable for safety critical systems. The authors in [5] discuss a software component using adaptive techniques from control theory. Supervisory logic and functional logic are identified within a component and the components interact via run-time infrastructure, but it does not explain how such interactions are selected using component attributes. A hierarchy of self adaptive properties are identified

978-1-4673-6404-1/13/$31.00 ©2013 IEEE

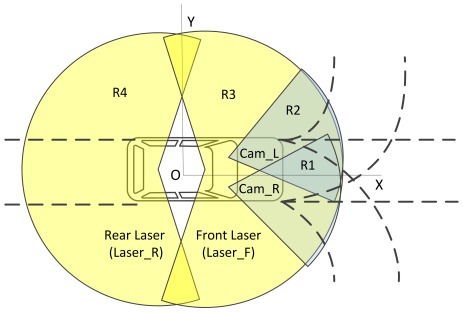

Fig. 1: Mobile robot sensor arrangement

in [6]. In addition to non-functional properties, functionality can also be included for self adaptation. A component is allowed to lower its functionality to an acceptable level in order to improve its performance. Common robot architecture comprises of three layers - Functional, Executive and Planning levels [7]. In [8] a two layers model - Decision and Functional layer - with varying level of granularity to achieve autonomy is adopted.

A priority based command arbitration is used in DAMN architecture for mobile robots [9]. However, priority based voting does not suffice in a complex dynamic environment such as in outdoor vehicles. Decision architecture for autonomous motion in vehicles are examined in [10]. General templates that contains knowledge on specific motion skills are employed for addressing motion planning problem. The interfaces to such components are purely functional and no qualitative measures are used to assess the reliability. Reconfigurable interfaces proposed by [4] are targeted to facilitate system integrators to plug components to slots in the architecture. All these architectures propose different mechanisms to address part of the problem, one architecture emphasize on reconfigurability, another one on voting based arbitration, another one on component level supervision. However, none of these architectures handles non-functional properties to provide a way to adapt the functional behavior of components dynamically so as to take the most reliable decision with respect to the operational context, which is our goal.

III. Architecture Requirements

Features of decision making architecture that help component developer and system integrator to have control over the system reliability are described below.

1) Scalability: A system is said to be scalable if the system composition and its constituent elements have the same fundamental properties. It refers to how easily a system adapts when one or more components are modified or added. The potential of modularity and decentralization are advocated to achieve scalability.

2) Component abstraction: Separating component specification and implementation using component abstraction makes the system extensible, portable, and reusable. Component abstractions standardize functionalities, interfaces, communication mechanisms, timing behavior, etc. The abstractions can be varied during project lifecycle. Initially the abstractions will contain basic functionalities and later as the project progresses extra specifications on non-functionalities, such as performance, timing behavior can be added.

3) Quality Propagation: The main goal of information fusion is increasing its quality. Quality is determined by several factors, such as precision, accuracy, confidence etc. Hence when information flows through computational algorithms, parameters comprising the quality should also flow along with it. The component composition should provide a mechanism in which the quality propagates. Such quality flow should act as shield against faulty behavior and can assist in smooth degradation of the system in case of failure.

4) Framework Supported Dynamic Configuration: The infrastructure needed for configuring component dynamically should be provided by the framework. Therefore, developing system level and low level components from architectural elements are recommended instead of having a separate supervisory control. It also helps in debugging and verification if the same model is used for the entire system. Most of the available architectures are overly concerned about communication mechanisms and adaptable interfaces as way of approaching reconfigurable system.

5) Conflict Resolution: Two methods of conflict resolution mechanisms are prominently used in robotics domain - arbitration and command fusion [11]. In the arbitration method, one or more decisions are selected in such a way that there will be no conflict in the final outcome, while in the command fusion, a new behavior emerges from the contribution of all the individual behaviors.

6) Context Awareness: Context awareness is important for deliberative as well as reactive layer in mobile robots. It helps in intelligent reasoning in higher layers and in sensor fusion in lower layers. In the latter case it helps to extract meaningful information from the multitude of heterogeneous sensors.

7) Information Filters: The decisions can be fused from multitude of sources and it does not happen in a centralized place. There is temporal as well as spatial difference between multiple fusions taking place at different levels. During fusion process, data is getting aggregated, but contrary to "data aggregation logic", data is actually reduced at each fusion process. Each fusion level can be considered as information filters, which narrows down the data to information. Hence architectural support to information filter at various level of hierarchy is essential for a decision making system.

A component based approach is proposed to modelize, to simulate, to distribute and to implement robotic applications. Several approaches for robot architectures is proposed by [7], [4], [12] and [9]. The Proteus project have already a RobotML implementation using a UML profile to model robotic applications [13]. However, all those approaches concentrate on the functional modeling or a specific type of composition or in dynamic reconfiguration. A precursor to the model that we intent to develop may be seen in the Structural Analysis for Real Time systems (SART) method [14], [15]. Because of the specificity and complexity of SART, this method is only used for high safety critical systems. We want to extend this approach by providing each component a behavioral model, a non-functional model as well a mechanism to ensure compositionality of such models.

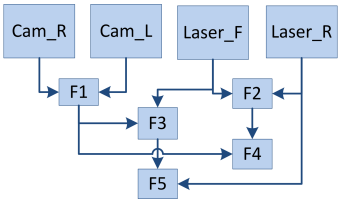

Fig. 2: Fusion Hierarchy

IV. Proposed Approach

A software component that is perfectly reliable in its functionality becomes unpredictable in the dimension of timing behavior. This is because software timing properties are unpredictable even if the same operating environment and initial conditions are assumed. A more dynamic property called Quality of Service (QoS) is associated with components to address the problem. The QoS of a system can be described as the degree to which the system conform to its functional specification meeting its non-functional requirements. The main drawback with a separate supervisory control is that the system level adaptation mechanism is not aware of its component semantics. Components should also be aware of the system level QoS variations. For example, it can increase its performance when more resources are available, while it can lower its functionality to acceptable level in case of resource scarcity. Application aware components are able to trade off its quality with that of system intentions. This approach is best explained using a sensor fusion application because of its inherent redundancy.

A. Illustrative Example

Perception, in mobile robots, can for example be used in several tasks, such as localization, navigation, map building, and obstacle avoidance. Figure 1 shows a simple non holonomic mobile robot that uses video cameras and laser scanners for obstacle detection (e.g., a pair of cameras and a laser scanner are used for front view and a laser scanner for rear view). The goal of the robot is to navigate a given trajectory with the maximum possible speed until it detects an obstacle in its path. Approximate field of view (FOV) and different regions - R1, R2, R3, and R4 based on overlapping FOV is identified in Figure 1. The robot can perform 4 types of motion - forward, backward, right turn, and left turn. The aim of the perception system is to detect obstacles on its path while executing the above motions. A common solution could be the fusion of multisensor data and the generation of a local occupancy map with respect to the robot frame of reference. Assuming the execution rate of fusion process is ν_{fuse} and maximum distance of the obstacle that can be detected is D_{max}, the maximum possible speed of the robot is given by $D_{max} \times \nu_{fuse}$. The system integrator has no control over its performance since the response time of such centralized fusion process is tightly coupled with the hardware and available resources during execution time.

The QoS of the robot is inversely proportional to the time taken to reach its goal position. The system QoS depends on the quality of perception subsystem. In this example, the rate at which the information is provided by perception subsystem is one factor that determines quality of system performance. The proposed approach is derived from the very fact that the quality of data does not depend only on the data itself. It also depends on the context in which it will be used. In this example, the required information about obstacles depends on the motion to be executed in the next timestep. The robot executing a backward motion needs little information on the obstacle on its front. Fusion process is not required in this case since the confidence of the data do not increase because only one sensor is deployed for rear region, R4. Similarly, while executing a sharp left turn, the data which is of prime importance is from front laser and left camera. Hence, restricting the speed of robot on the rate of centralized fusion process is not an appropriate method. The fusion architecture should be designed in such a way so as to maximize the QoS of the overall system rather than delivering high QoS by a subsystem alone. Distributed fusion process adds flexibility of selecting the fusion granularity using system QoS attributes and thereby increasing the overall performance.

The fusion process can be configured in a number of hierarchical ways. Some sensors are easier to merge at lowest level, while some others at higher levels. For example, sensors using the same principle of operation such as lasers and radars are easier to fuse rather than with highly heterogeneous sensors, such as cameras and lasers. These restrictions can be characterized using I/O based approach proposed by [16]. Homogeneous sensors commonly use Data In-Data Out or Data In-Feature Out paradigms. A most common example of Data In-Feature Out characterization is found in stereo vision, where two image data are fused to estimate the depth feature. For heterogeneous sensors the more appropriate characterization is Feature In-Feature Out or Feature In-Decision Out models. The hierarchical fusion process depicted in Figure 2 is used for further discussion. The blocks F1 and F2 fuse data from cameras and lasers, respectively and then the results are fused in F4 at the next level. It is to be noted that data can represent raw data, feature, or decision and this trilogy can be seen as orthonormal to the presented hierarchy. F3 fuses data from F1 with front laser and then with rear laser in next level at F5. The final result after fusing all sensors can be obtained at F4 and F5, wherein sequence of fusion steps is varied according to spatial constraints and sensor types. Although F4 and F5 have two prior fusion steps, the two steps before F4 can be executed in parallel, while it will be sequential in the case of F5. The advantage of latter case is the granularity in which the intermediate steps can bypass the result depending on context.

Although QoS is an abstract concept, the constituent elements are more application specific. For example, if the goal of maintaining a database of surrounding obstacles during navigation is added to the previous goal, then the QoS will have to be modified. In such a case, in addition to time, obstacle properties such as precision and confidence contribute towards QoS of the robot. Now the quality of fusion of process will not any more depend on specific regions but more on its accuracy of the obstacles. Hence, a change in the system objective affects the QoS of the system, which in turn affects low level components. It is to be noted that it is not necessary that functional specification of the subsystems should change when system objective changes. In this example, the functionality of the fusion blocks did not change, only the sequence of fusion process in the hierarchy is changed. Therefore, by separating the performance determining factors and the computation, different objectives can be met with the

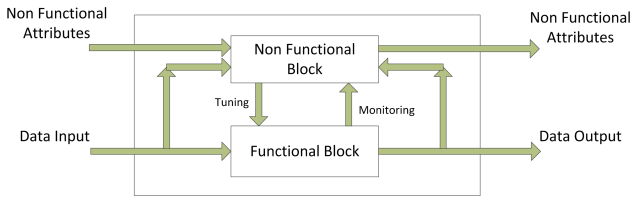

Fig. 3: Component Model

same components. This architectural approach will be feasible only if such model of operation is adopted by all the system components from higher to lower level. A component model facilitating the proposed approach is described in section V.

V. Component Model

NFAs such as safety and reliability are viewed as system level properties. Nevertheless, these parameters may have originated from a micro level, such as sensors, network communication etc. Hence, the flow of NFAs should be handled at the micro as well as at the macro level. Software components should provide a mechanism in which such NFAs are analyzed, compared, manipulated, and transferred. Figure 3 shows such a component model in which functional and non-functional blocks are separated. The functional block computes the output data from the input data and models the functional behavior of the component. Functional blocks implements functional specification and provides interfaces for tuning its own parameters and monitoring its internal state. The monitoring parameters can include the algorithmic state and various attributes of its results, such as confidence, accuracy, resolution, etc. The non-functional block computes non-functional attributes from the input data and from the inferred non-functional attributes that annotates the input data. Non-functional block has mainly two functions, first it is responsible for modifying incoming NFAs with respect to the components performance and it can act as a feedback loop by tuning the algorithm parameters. The component provides all the interfaces that allows to connect to other components. It also encapsulates the additional system services such as scheduling, orchestration of sub-components, etc. Components that encapsulate human behavior can be incorporated in the system that model human actions and provide abstract functions to measure, compare performance with that of system counterparts is discussed in section VI.

The primary objective of fusion process is improving confidence level of information. The confidence level can be evaluated using different criteria such as probability of detection, redundancy, accuracy, etc. In this example, confidence is related to the redundancy of the sensors in the overlapping regions. Confidence on the data will be increased if two sensors confirm each other and it may decrease if their decisions contradict. In other words, the confidence level provided by each fusion level varies at each time step, depending on the situation it handles. For example, Figure 4 shows the confidence level in each component in one instance of timestep where no component provides contradictory data. For simplicity, in the figure, we have assumed that the confidence level of the fusion result will be a simple addition of confidence level of the constituent data. If C_{cam} and C_{laser} represents confidence level of data from camera and laser, respectively, their fusion result is assumed to be $C_{cam} + C_{laser}$. For example, if we are interested only in region $R1$ with a minimum confidence

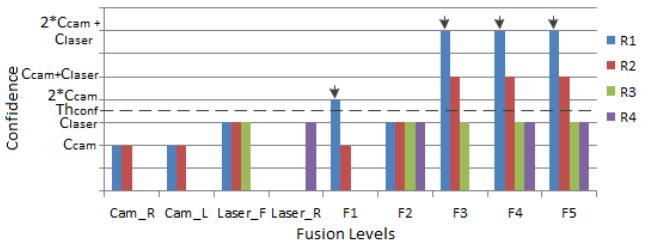

Fig. 4: Available confidence levels at different fusion levels in regions R1, R2, R3 and R4

level of Th_{conf} as shown by dotted line in Figure 4, there are 4 fusion blocks satisfying this constraints indicated by arrow marks. Assume that two components that sources the data contradict each other. The merging process will compute a new confidence level with respect to the confidence level provided by each source. If one source has a very high confidence level and the second has a very low confidence level, the confidence level of the merged data still remains high. But, if both sources have a high confidence level, the merged data will have a much lower confidence level because of the contradiction between both data sources. In this case, it is not recommended to use the data directly from the sources although it is having higher confidence level, hence a more complex QoS is introduced to incorporate more diverse attributes.

A list of example attributes for each block in the fusion hierarchy is shown in Table I. Attributes for camera can be Field of View (FOV), Image Format, Resolution; and for Laser it can be Scan angle, Range, etc. Worst Case Response Time (WCRT) is identified as an important attribute for components involving timing constraints. WCRT can be computed based on the sensor rate, algorithm execution time, communication delay, and its dependency with other components. T_C and T_L represent the frame rate of camera and laser respectively and t_{Fx} represents the time taken by the fusion operation at Fx. For example, T_{F1}, the response time of F1 is the sum of T_C and T_L since it uses the data from Camera and Laser. Although these parameters are highly dependent on target environments such as available resources, platform, clock rate etc, these anticipated environmental conditions can also be included in the list of attributes, as shown in Table I. The QoS of each block can be a predicate function, which encompasses all these attributes to check whether it meets the required quality level. An example of QoS predicate for F3 component is shown in Equation 1.

$$\text{Predicate QoS}(F3) : (\text{WCRT} < 0.6ms) \wedge (\text{Confidence} > 0.8) \wedge ((\text{Resolution} > 0.01) \vee (\text{Rate} > 5Hz)) \quad (1)$$

The boolean predicate given in Equation 1 states that the decision from the F3 component is optimum if it satifies the given conditions.

Two advantages can be identified with respect to this approach: (1) From the set of system compositions, the best possible platform required to execute the system can be guessed. (2) Given the target platform, the system will be able to predict the best possible configurations. In any case, the attributes of components from higher level to lower level will

TABLE I: Attributes of Fusion Blocks

Block	WCRT	Example Attributes
Camera	T_C	FOV, Weather, Format, S/N Ratio
Laser	T_L	Scan angle, Resolution, Range
F1	$T_{F1} : T_C + t_{F1}$	Resolution, Confidence measure, Prior probability, Resources required, Platform dependency, Dependent third party libraries, etc.
F2	$T_{F2} : T_L + t_{F2}$	
F3	$T_{F3} : (T_{F1} \|\| T_L) + t_{F3}$	
F4	$T_{F4} : (T_{F1} \|\| T_{F2}) + t_{F4}$	
F5	$T_{F5} : (T_{F3} \|\| T_L) + t_{F5}$	

be matched with that of the platform for example checking whether WCRT is compatible with the WCET of the platform.

The proposed component model satisfies the architectural requirements identified in section III. *Scalability* of the system architecture can be classified into functional and non-functional scalability. Functional scalability refers to adding more functional competencies to the existing system without disrupting the existing skills. Subsumption architecture proposed by Brooks [12] is a good example of functional scalability. For example, when a mobile robot is in a highly dynamic environment, it has to increase the update rate for certain components, perception, for instance. *Context aware* components are able to increase the non-functional contraints and lowering its functionality to an acceptable level depending on the situation. This is achieved by the proposed component model since it allows monitoring its own functionality by the non-functional block thereby varying the parameters for achieving the above goal. The external and internal communication mechanisms involved in the component is not specified in this paper. The component model should allow composability and compositionality properties. Composability preserves the component property after integration and compositionality allows to predict the system properties from its contituent components. This enables *dynamic configuration* of components during run-time and to prove its correctness [17]. The challenges involved in component abstraction is two fold; in top-down and bottom-up perspective. In top-down approach the component specifications should include complete functional and non-functional attributes included its variation points, self monitoring and adaptation properties so that it can be implemented correctly. Bottom-up approach is used in estimating and guessing an implemented algorithm for its attributes so that it helps in reusing existing components.

Conflict resolution mechanism previously mentioned in section II such as in DAMN architecture [9] implements the arbitration at a single level. By nesting the components in a hierarchical fashion, conflict resolution can be achieved in multiple levels. If a conflict cannot be resolved in lower level with given context information, it can propagate the conflict to higher levels where it has more information to solve the conflict. In addition, these components can be viewed as *information filters* which extract information from the input and maps it to output using some algorithm. By using the same architectural template in all layers from lower level data fusion to higher level information extraction, it is possible to dynamically adjust the granularity and the frequency of operation.

VI. Componentizing Human behavior

The concept of creating context aware components and using QoS for dynamic reconfiguration can be extended to systems involving human intervention. In systems such as assisted teleoperation, the control system shares the authority with that of human counterparts. Even in fully autonomous systems such as in automatic driving, while transitioning from manual to automatic mode, there will be brief period of time in which human and autonomous components coexists. Empirical results show that the performance is degraded if human's behavior is not modeled in the system. Extensive studies have been conducted in Human-Robot Interaction domain to quantify human behavior. The technique for human error rate prediction (THERP) provides a systematic methodology for the quantification of human reliability [18]. However, from the software architecture point of view, there is limited study on how a framework can provide support for human model to coexist with autonomous components.

Since we use attributes of components for dynamic composition and decision making, it is possible to model human behavior using functional and non-functional attributes and it can be interfaced with existing autonomous components. The works of [19] characterizes human-robot interaction using neglect time tolerance, task complexity, and interaction efficiency. These parameters can be mapped to the non-functional attributes of the component model proposed in this paper. In this context, our argument is that, similar to QoS for a navigation or a perception component, the QoS of human behavior component will enable it to blend with autonomous components without exposing that humans are actually involved in the decision process.

Another advantage is in simulation by involving humans in the decision making process. It is possible to verify how the system reacts in complex scenarios when the attributes of humans, such as driving skills, perceptual capability, alertness, stress levels etc. are varied. It helps to simulate various situations, such as assisted parking, automatic lane merging, augmented cruise control, etc. Such simulations enable us to address certain anticipated scenarios, for example, a driver attempting to take control when the vehicle is performing an automatic lane merging maneuver.

VII. Conclusion and Future Work

This paper presented a new component based model where components encapsulate the functional as well as the non-functional behaviors. Functional behavior computes the output of the component with respect to the input data. Non-functional behavior infers from the input data and the non-functional attributes associated with input data, the attributes that will characterize the behavior of the component as well as the quality of the output in various situations. This approach will define a new design methodology for system architecture by using the component specifications and to verify the system during design time if the quality of the output and its services is adequate. This approach can also be used to validate and verfiy an existing implementation by providing each function a sound functional and non-functional model so that extensive simulation can be done on the model. Finally, this component based approach can be extended to modelize human actions at

the global system level. This enables to provide a simulation platform where we can design and validate a system where some actions may be human controlled and some others be completely automated.

Future works will consist of providing a formal specification for the component model. This model will be based on fractal hypergraphs, a specific family of hierarchical graphs [20]. It will include a formal language to specify attributes as well as a mechanism to ensure sound composition.

ACKNOWLEDGMENT

This research is funded by VeDeCoM Institute, a French automotive cluster on mobility research.

REFERENCES

[1] D. Brugali and P. Scandurra, "Component-based Robotic Engineering Part I: Reusable building blocks," *Robotics & Automation Magazine, IEEE*, vol. 16, no. 4, pp. 84–96, 2009.

[2] D. Fox, "KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization," *Advances in Neural Information Processing Systems (NIPS)*, pp. 26–32, 2001.

[3] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming heterogeneity-the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[4] J. Gowdy, "Emergent architectures: A case study for outdoor mobile robots," Ph.D. dissertation, Carnegie Mellon University, 2000.

[5] G. Karsai, A. Ledeczi, J. Sztipanovits, G. Peceli, G. Simon, and T. Kovacshazy, "An approach to self-adaptive software based on supervisory control," *Self-adaptive software: applications*, pp. 77–92, 2003.

[6] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 4, no. 2, p. 14, 2009.

[7] E. Gat *et al.*, "On three-layer architectures," *Artificial intelligence and mobile robots: case studies of successful robot systems*, vol. 195, 1998.

[8] I. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin, "Claraty and challenges of developing interoperable robotic software," in *Intelligent Robots and Systems. Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2003, pp. 2428–2435.

[9] J. Rosenblatt, "Damn: A distributed architecture for mobile navigation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 339–360, 1997.

[10] C. Laugier and T. Fraichard, "Decisional architectures for motion autonomy," *Intelligent Vehicle Technologies, Chapter 11*, vol. 11, pp. 333–391, 2000.

[11] J. Hurdus and D. Hong, "Behavioral programming with hierarchy and parallelism in the darpa urban challenge and robocup," *Multisensor Fusion and Integration for Intelligent Systems*, pp. 255–269, 2009.

[12] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, 1986.

[13] S. Dhouib, S. Kchir, S. Stinckwich, T. Ziadi, and M. Ziane, "Robotml, a domain-specific language to design, simulate and deploy robotic applications," *Simulation, Modeling, and Programming for Autonomous Robots*, pp. 149–160, 2012.

[14] P. T. Ward, *Structured Development for Real-Time Systems: Vol. I: Introduction and Tools*. Prentice Hall, 1986.

[15] D. J. Hatley and I. A. Pirbhai, *Strategies for real-time system specification*. Dorset House Pub., 1988.

[16] B. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 24–38, 1997.

[17] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded system design for automotive applications," *Computer*, vol. 40, no. 10, pp. 42–51, 2007.

[18] B. Kirwan, "The validation of three human reliability quantification techniquestherp, heart and jhedi: Part 1technique descriptions and validation issues," *Applied Ergonomics*, vol. 27, no. 6, pp. 359–373, 1996.

[19] J. W. Crandall and M. A. Goodrich, "Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2. IEEE, 2002, pp. 1290–1295.

[20] N. Vallée, B. Monsuez, and V.-A. Paun, "Extracting logical formulae that capture the functionality of systemc designs," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 2, 2011.