Proceedings of the Federated Conference on Computer Science and Information Systems pp. 293–297
DOI: 10.15439/2017F226
ISSN 2300-5963 ACSIS, Vol. 11

Failure Analysis for Adaptive Autonomous Agents using Petri Nets

Mirgita Frasheri, Lan Anh Trinh, Baran Cürüklü, Mikael Ekström
Mälardalen University
Västerås, Sweden
Email: {mirgita.frasheri, anh.lan, baran.curuklu, mikael.ekstrom}@mdh.se

***Abstract*—Adaptive autonomous (AA) agents are able to make their own decisions on when and with whom to share their autonomy based on their states. Whereas dependability gives evidence on whether a system, (e.g. an agent team), and its provided services are to be trusted. In this paper, an initial analysis on AA agents with respect to dependability is conducted. Firstly, AA is modeled through a pairwise relationship called willingness of agents to interact, i.e. to ask for and give assistance. Secondly, dependability is evaluated by considering solely the reliability attribute, which presents the continuity of correct services. The failure analysis is realized by modeling the agents through Petri Nets. Simulation results indicate that agents drop slightly more tasks when they are more willing to interact than otherwise, especially when the fail-rate of individual agents increases. Conclusively, the willingness should be tweaked such that there is compromise between performance and helpfulness.**

I. Introduction

DEPENDABILITY IS, and has been for a decade, a promising research direction for researchers and is considered central in designing systems that are intended to work closely with and for humans (e.g. automotive, airborne, and service robots). Originally, it was devised from software development areas and can be stated by Avizienis *et al.* [1] as "*the ability to deliver service that can justifiably be trusted*". The dependability of a system is evaluated by one, several or all of the attributes including availability, reliability, safety, integrity, and maintainability. The implementation of dependability starts with the understanding of the threats to the system dependability: The threats consists of failures, errors, and faults. The link between the three is known as fault-error-failure chain. A failure happens when the services provided by a system do not comply with its specification. An error affects the services and leads to the failure of the system. The hypothesized cause of an error is a fault. Therefore, there are four means that have been developed to protect the system dependability which are fault prevention, fault removal, fault forecasting, and fault tolerance, whereof fault tolerance is discussed in this work. It is deducible that the fault is the root of every failure appearing inside and outside of the system. The most pressing challenge is how to predict the frequency of faults and the moment a fault occurs, thus fault analysis is utilized to minimize the probability of faults and fault prediction is applied to give an estimate of when faults happen in the system. Thereafter, other means are developed to protect the dependability with respect to the analysis of the fault. In order to conduct fault analysis, various approaches such as Petri Net (PN), fault tree analysis (FTA), failures modes effects and criticality analysis (FMECA), and hazard operability (HAZOP) have been introduced in the work of Bernardi *et al.* [2]. However, in this work PN is chosen because this framework provides a probability approach for fault analysis and fault prevention in both development and operational stages of designing a system. For instance as an extension of PN, a stochastic Petri net could be combined with Markovian models to evaluate the probability of the current state and the probability of future fault events for a fault prognosis process. In this work, a Colored Time PN (CTPN) is utilized for fault analysis in a system of adaptive autonomous agents.

On the other hand, agent autonomy represents a widely discussed topic in the literature. It can be described in two dimensions: self-directedness, i.e. autonomy in determining one's own goals, and self-sufficiency, i.e. autonomy in carrying out a task or goal without outside help [3]. Autonomy is also a relational concept [4]. There is autonomy from the environment, which is defined by how much an agent is independent from the stimuli coming from outside. Moreover, there is autonomy from other agents – social autonomy – that is defined by how much an agent is independent from the influences of those other agents. Castelfranchi [4] grounds the concept of autonomy on dependence theory. Consequently, an agent A doing a task might realize that it needs a resource, know-how, a plan, or an action that it does not have in order to achieve its goal. If there is an agent B that can provide A with what it needs, then A will depend on B for that specific need. As a result A is not autonomous from B. When the dependencies change, so does the autonomy of the parties involved. Adaptive autonomy specifically enables the agent to decide on its own autonomy [5]. In this work, it means that agents can choose when to depend on others, or when to be depended upon based on their current circumstances.

The aim of this paper is to analyze an initial concept of an adaptive autonomous agent developed previously [6] with respect to fault tolerance by using Petri Nets. The following sections are organized as follows. First related work both on Petri Nets and adaptive autonomy is discussed. Thereafter, the initial agent model is described and it is shown how the willingness to interact shapes the agent's autonomy. Next, the failure analysis is conducted. Simulations are run in order to

show how the willingness to interact, failure frequency and the number of tasks dropped by the agents relate to each other. Finally results are described and discussed.

II. RELATED WORK

A. Dependability

The assessment of system dependability, as aforementioned, is based on the basic attributes. Depending on the specific applications, different attributes are used to measure the dependability of a system. In the development of robotics application, with regards to the reliability, a group of researchers from India [7] proposed an approach to assess various parameters that make a multi-robot system more reliable. The proposed method is a combination of PN and fuzzy lambda-tau. Another model and analysis for multi-robot based on stochastic PN is introduced by Sheng *et al.* [8]. However, the former research focused on reliability analysis for a navigation system, the latter one studied of a reliable model for multi-robot exploration. These researches are limited to these particular systems.

In the study of fault tolerance analysis for a group of agents, there are some researches on modeling and analyzing a distributed system of agents using PN and extended PN. For instance, Ammour *et al.* [9] introduced a stochastic Petri net combined with Markovian models to evaluate the probability of the current state and the probability of future fault events for a fault prognosis process in discrete event systems. In the work of Sun Chen *et al.*, [10] an adaptive consensus of fault tolerance for a multi-agent system was proposed. Another automated analysis of fault tolerance for a reliable communication system is introduced by Stoller and Schneider [11]. A model, analysis of fault tolerance for a distributed multi-agent system using Time Colored Petri Nets are provided by Boukredera *et al.* [12]. In the work of Kristan *et al.* [13], based on PN tool, the activities of agent in a complex multi-agents system are analyzed. Recently, a combination of a fuzzy method and PN introduced in [14] is proposed to analyze a sequential failure in complex industrial systems. Although some effective techniques for faults analysis have been developed, there are lacks of works focusing on the analysis of faults for adaptive autonomous multi-agent systems. Therefore, a method based on CTPN is proposed for failure analysis in the context of autonomous adaptive agents.

B. Agent Autonomy

Alongside adaptive autonomy, there are several other similar concepts such as adjustable autonomy, collaborative control, mixed-initiative interaction, and sliding autonomy. Adjustable autonomy refers to a system in which the human operator is the one to decide on the levels of autonomy of the agent [5]. Mixed-initiative interaction makes it possible for either the human or the agent to decide on autonomy levels [5]. Collaborative control allows the human and agent to resolve their inconsistencies through dialogue [15]. Sliding autonomy refers to a system which can switch between full tele-operation and autonomy on a task level, i.e. the agent can be tele-operated with respect to a task T_1 whilst executing autonomously a task T_2 [16]. While this list is not exhaustive, it still represents the most encountered terminology in the literature and serves to build a general picture of the landscape in the field. Worth noting is that there has been a paradigm shift in how dynamic autonomy is handled, from the 10 levels of autonomy [17] to team-work approaches in which autonomy is not fixed but rather changes depending on the interdependencies between agents [18]. Moreover, the authors have proposed a design methodology which identifies possible interdependencies in a system, and thus helps the designer provide support for them in the implementation phase.

Several works aim at providing comparison between systems with and without adaptive autonomy [19], and between systems with different approaches to autonomy [5] [20]. Systems such as RIAACT [21] are oriented toward meeting real-time requirements of adjustable autonomous agents. Whereas, STEAM [22] is an agent architecture which adds support for teamwork. Specifically, team operators, i.e. reactive team plans, are introduced. The agents also have their individual plans which do not require teamwork. Kaa [23] is a system developed on top of the KAoS policy system [24]. KAoS implements policies to orchestrate multi-agent behavior, and Kaa allows for their modification on run-time. This solution is centralized.

In this paper, the willingness to interact is used to shape agent autonomy. Additionally, the decision of whether to interact, i.e. whether to ask or to give assistance, is considered as internal to the agent. There is no hierarchy between them, nor fixed levels of autonomy.

III. BACKGROUND ON PETRI NETS

A Petri net (PN), from a mathematical perspective, is a bipartite graph built from a set of tuples (U, V, W), where U and V are a set of places and set of transitions respectively. Meanwhile, W is a set of arcs used to link from a place to transition and vice versa. Noting that there is no any connections between places as well as between transitions. The arcs running out from a place to a transition are known as input places of transitions, whereas the contrary arcs, i.e, running out from a transition to place, are called output places of transitions. PN, therefore, with regards to two types of flows from a transition, is described as a set of five tuples (U, V, W, W^-, W^+), in which W^- defines the output weights and W^+ for the input weights from a transition. The number of tokens inside a place is named marks. The stage of PN is completely determined based on the marks of all places, therefore a marking M is used to present the current state of PN. The marking M is expressed by a vector $[M(p1), M(p2), ..., M(pi), ..., M(pn)]$, in which n is the number of places and $M(pi)$ is the mark of place pi. Let W^- is expressed by a two dimensional matrix of weights where each element $W^-(pi, tj)$ is determined as the number of tokens allowed to move from the place pi to the transition tj. Similarly, W^+ is formulated by the weight $W^+(tj, pi)$ from the transition tj to the place pi. It is noted

that $1 \leq j \leq m$, in which m is the number of transitions. Thus, the marking is updated by the following equation.

$$M'(p) = M(p) + W^{+}(t,p) - W^{-}(p,t), \forall p. \qquad (1)$$

Let M_0 being an initial marking, the tuple (U, V, W, W^{+}, W^{-}) is extended to $(U, V, W, W^{+}, W^{-}, M_0)$. The marking M is reachable if M' is the result of applying a sequence of update equation (1), starting from M. The state-space analysis of PN shows a full graph of all possible markings that are reachable from M_0 and all possible paths of transition from a marking to another. However, the complexity of the state-space analysis dramatically increases with respect to the number of places and transitions of a PN network. Thus, the state-space analysis is completely suitable for the small PN network.

There are various types of extension of PN. The colored PN (CPN) is one of them, in which CPN utilizes different color to mark tokens [25]. Additionally, each type of token separately fires with the transition. The transition behavior for different colors is decided by the arc expressions which are built from operators and functions. Another extension type is the stochastic PN in which a time delay is added to each transition, and a random variable is used to estimate the firing rate. A probabilistic inference, therefore, is utilized to analyze the state-space of the PN network. In this work, a hybrid PN called colored time PN (CTPN) is applied to deal with both non-deterministic and deterministic variables of the time delay. CPNTools [25] are utilized to model the proposed system and perform the failure analysis.

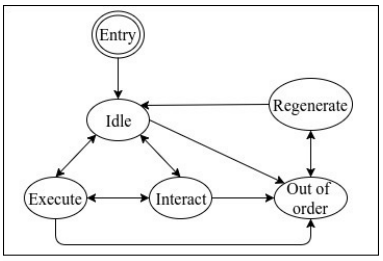

Figure 1. Agent Model

IV. AGENT MODEL

The agent is realized as a state machine with 5 operational states which are $idle$, $execute$, $interact$, $regenerate$ and out_of_order (Figure 1). An agent starts its operation in the $idle$ state, in which it will generate a task with a probability P. As long as it is in $idle$ the agent has not committed to any task. A change of state occurs either if a request for help is received, or if the agent has generated a task. In the former, the agent switches to the $interact$ state and decides whether it will give assistance or not (not interruptible process). This decision will depend on the agent's willingness to give assistance (defined probabilistically). In the positive case the agent will put the task in a queue and switch to $execute$, otherwise it will discard the request and resume what it was doing (either back in $idle$ or $execute$). In the latter, the agent will put its own generated task into a queue and will change its state to $execute$. At the beginning of the execution of a task the agent decides on whether it needs outside assistance or not. This is based on the agent's willingness to ask for assistance. If it does, it will select the agent which was perceived as most helpful in the past and make a help request. This is a blocking operation. Nonetheless, the agent waits for a reply for a finite amount of time, after which it will return to idle regardless of the outcome. In principle, the agent could receive a request while being in the execute state as well. If the agent reaches critical levels of battery, i.e. lower than some predefined threshold, it will switch to out_of_order. As of now, the agent will immediately switch to $regenerate$, in which the recharge takes place, and into *idle* right after. The willingness to ask and give assistance make up the agent's willingness to interact, i.e. they are like the two sides of a coin. Modeling these elements mathematically is part of the research for adaptive autonomous agents. Nonetheless, for the purposes of this work, they are expressed through probabilistic distributions, as is explained in the next section.

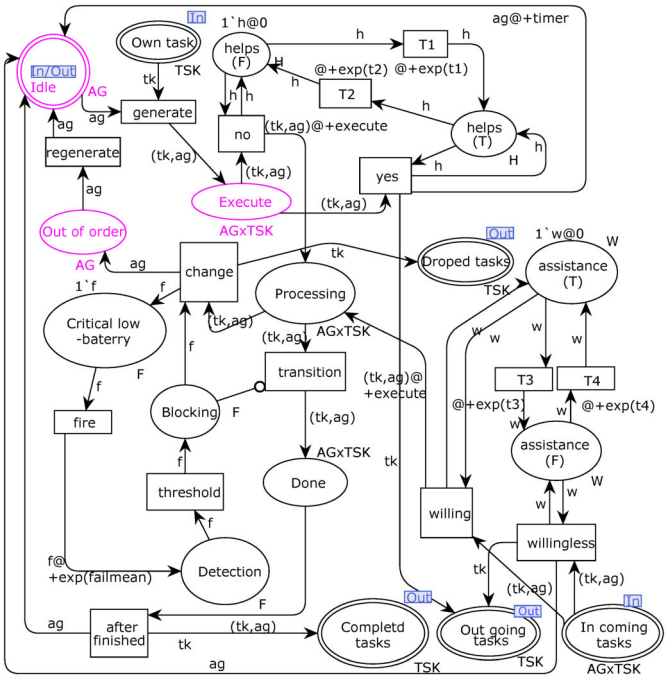

Figure 2. Design of the *agent module*

V. FAILURE ANALYSIS

Based on the description given in the previous section the agent module is designed. Moreover, all agents share the same structure which is represented by PN (Figure 2). All tasks are generated by the agents. The agent once completing the task will put the completed tasks in a submodule known as *completed tasks*. Otherwise, the task will be put on a submodule called *dropped tasks*. The agent, right after, will send a helping request. If there is an available agent which is willing to give assistance, the agent will take the task from a

place named *tasks need helps* and try to finish the unfinished tasks. The agent, while doing the task, is being checked for its energy. When the agent's battery is in critical low level, the agent will drop the task and change into *out-of-order* state. Thereafter, it is assumed that the agent goes into recharging itself. After this process has completed, the agent will be regenerated and shift into *idle* state and be ready to do a new task.

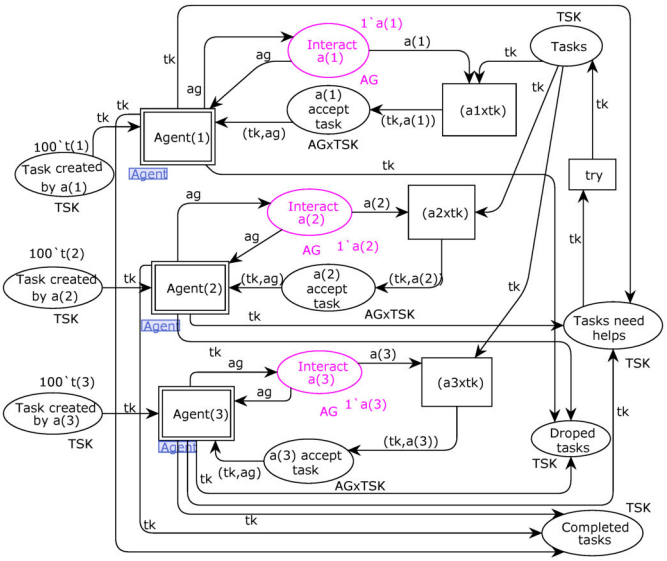

Figure 3. Design of the *task management module*

The *task management module* (Figure 3) is utilized as a main module to analyze the failures happening in the system. It is supposed that there are three agents $(a1, a2, a3)$. The agents are placed in the *task management module* and their internals correspond to the one in the *agent module* (Figure 2). Moreover, Figure 3 also represents when the agents are in the *interact* state. Each agent generates one hundred tasks to evaluate the performance of fault analysis of dropping tasks due to the lacks of energy. The tuple of parameters (t_1, t_2, t_3, t_4) which are modeled by exponential distributions, present the willingness to interact of the agent. In detail, t_1 and t_2 describe the non-willingness and willingness of asking for help, meanwhile, t_3 and t_4 present willingness and non-willingness to give assistance respectively. The executing time of each agent for a task is assumed $\tau_e = 20sec$. Meanwhile, the fail event (out of power) of an agent is modeled by a random variable of the exponential distribution with the mean time λ_e (value *failmean* in agent). Once the agent is out of power, the agent will fall into failed state. The failure of the system is analyzed based on a probability of failure which is given by $prob = dt/gt = dt/(dt + ct)$, in which dt is the number of dropped tasks, gt is the number of generated tasks, and ct is the number of completed tasks.

VI. Results

The parameters in the simulation are configured so as to assess how the system behaves as the agent gets more dependent on other agents. Specifically, for the case in which $t_1 = t_2$, the agents are configured with the same probability of asking for help and of not asking for help once it generates a new task. Meanwhile, $t_1 < t_2$ reveals that the agent is more dependent on other agents to complete the tasks. A similar meaning is expressed with the two configurable parameters t_3 and t_4, where $t_3 > t_4$ shows that the agent is more willing to help other agents to complete a task. Overall, the cases $t_1 = t_2$, $t_1 < t_2$ combined with $t_3 = t_4$, $t_3 > t_4$ are used to evaluate the failures happening in the system. Moreover, the dropped tasks as well as the completed tasks are computed to give the probability of failure in the system with respect to the changes of those parameters. The fail mean varies from 10 to 100 time units $(10 \leq \lambda_e \leq 100)$. It can be seen that the results reflect what is expected, i.e. there is an increase in dropped tasks when the agents need more help from the others as well as they are willing to give assistance. Meanwhile the dropped tasks decrease when the intervals between two fail events (fail mean) are prolonged. Table I shows that the probability of failure increases when t_2 increases. The results illustrated in rows $1-3$ and $7-9$ present that the probability of failures decrease when the agents are less willing to give assistance and the fail means are longer. Moreover, it is possible to observe that being more willing to help results in slightly more tasks dropped than being more willing to ask for help – especially for longer fail means. This is because, it is possible for an agent which gives help for a task to ask for help for the same task. Finally, the results in the last three rows $(10-12)$ express the worst case in which the agents are more dependent to the others and also are willing to give assistance. Therefore, the system will fall into the failed state with a higher probability.

Table I
Presentation of the numerical results. D - dropped tasks, C - completed tasks, P - Probability (expressed in %)

Fail Mean	$t_1 = t_2, t_3 = t_4$			$t_1 < t_2, t_3 = t_4$			$t_1 = t_2, t_3 > t_4$			$t_1 < t_2, t_3 > t_4$		
	D	C	P	D	C	P	D	C	P	D	C	P
10	284	16	94.67	288	12	96	293	7	97.67	295	5	98.33
20	265	35	88.33	271	29	90.33	265	35	88.33	271	29	90.33
30	237	63	79	245	55	81.67	245	55	81.67	245	55	81.67
40	211	89	70.33	219	81	73	221	79	73.67	224	76	74.67
50	188	112	62.67	202	98	67.33	215	85	71.67	208	92	69.33
60	174	126	58	187	113	62.33	191	109	63.67	187	113	62.33
70	156	144	52	158	142	52.67	176	124	58.67	174	126	58
80	138	162	46	156	144	52	169	131	56.33	168	132	56
90	121	179	40.33	134	166	44.67	147	153	49	151	149	50.33
100	118	182	39.33	127	173	42.33	139	161	46.33	137	163	45.67

VII. Conclusion

In this work, the formulation of CTPN has been introduced for the failure analysis of a system composed of a group of adaptive autonomous agents. The failure analysis is evaluated by the probability of dropped tasks. The CTPN models have been designed to probabilistically evaluate the failure rate of the system. From the experimental results, the analysis has shown the correlation of the probability of failure of the system (rise in the number of dropped tasks) with the failure mean of an agent. Moreover, it could be observed that a high

willingness to interact when the fail mean is low induces a slightly higher number of dropped tasks. It is possible to conclude that, while being willing to interact and collaborate is a desired characteristic for an agent, it will not always result in higher performance. Consequently, the agents need proper reasoning mechanisms that will allow them to make the best of any situation they are in. In some cases this might mean being helpful, and in others it might mean focusing on one's own tasks/goals.

In the future, the proposed approach will be extended to deal with a more complex agent architecture in which relevant factors that should shape the willingness to interact will be identified and analyzed. Consequently, appropriate models will be developed so as to reflect the impact of these factors and will replace the distributions used in this work. Ultimately, agents should be able to find a balance between helping each other, completing their tasks and face failures of different frequencies. Another concern relates to the potential scalability of such approach by considering a bigger population of agents in ranges of 10, 20, 50 etc. Finally, the actual communication between the agents will be addressed as well in future PN models, as it is central to the adaptive autonomous agent model described in this paper.

VIII. Acknowledgments

The research leading to the presented results has been funded by the EUROWEB+ project (1^{st} author) and the research profile DPAC - Dependable Platforms for Autonomous Systems and Control project, funded by the Swedish Knowledge Foundation (2^{nd}, 3^{rd}, and 4^{th} author).

References

[1] A. Avižienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004. [Online]. Available: https://doi.org/10.1109/TDSC.2004.2

[2] S. Bernardi, J. Merseguer, and D. C. Petriu, *Model-Driven Dependability Assessment of Software Systems*. SPRINGER, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-39512-3

[3] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, B. Van Riemsdijk, and M. Sierhuis, "The fundamental principle of coactive design: Interdependence must shape autonomy," in *Coordination, organizations, institutions, and norms in agent systems VI*. Springer, 2011, pp. 172–191. [Online]. Available: https://doi.org/10.1007/978-3-642-21268-0_10

[4] C. Castelfranchi, "Founding agent's 'autonomy' on dependence theory," in *Proceedings of the 14th European Conference on Artificial Intelligence*. IOS Press, 2000, pp. 353–357.

[5] B. Hardin and M. A. Goodrich, "On using mixed-initiative control: A perspective for managing large-scale robotic teams," in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, 2009, pp. 165–172. [Online]. Available: https://doi.org/10.1145/1514095.1514126

[6] M. Frasheri, B. Çürüklü, and M. Ekström, "Towards collaborative adaptive autonomous agents," in *9th International Conference on Agents and Artificial Intelligence 2017 ICAART, 24 Feb 2017, Porto, Portugal*, 2016. [Online]. Available: https://doi.org/10.5220/0006195500780087

[7] S. Sharman, N. Sukavanam, N. Kumar, and A. Kumar, "Reliability analysis of complex robotic system using petri nets and fuzzy lambda-tau methodology," *International Journal for Computer-Aided Engineering and Software*, vol. 27, no. 3, pp. 354–364, 2009. [Online]. Available: https://doi.org/10.1108/02644401011029925

[8] W. Sheng, Q. Yang, and N. Xi, "Modeling, analysis and design for multi-robot exploration based on petri nets," 2004.

[9] R. Ammour, E. Leclercq, E. Sanlaville, and D. Lefebvre, "Faults prognosis using partially observed stochastic petri nets," in *International Workshop on Discrete Event Systems*. IEEE, 2016, pp. 472–477. [Online]. Available: https://doi.org/10.1109/WODES.2016.7497890

[10] S. Chen, D. W. Ho, L. Li, and M. Liu, "Fault-tolerant consensus of multi-agent system with distributed adaptive protocol," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2142–2155, 2015. [Online]. Available: https://doi.org/10.1109/TCYB.2014.2366204

[11] S. D. Stoller and F. B. Schneider, "Automated analysis of fault-tolerance in distributed systems," *Formal Methods in System Design, Springer Science*, vol. 26, pp. 183–196, 2005. [Online]. Available: https://doi.org/10.1007/s10703-005-1492-2

[12] D. Boukredera, R. Maamri, and S. Aknine, "Modeling and analysis of reliable contract net protocol using timed colored petri nets," in *International Conference on Web Intelligence and Intelligent Agent Technology*. IEEE, 2013, pp. 17–24. [Online]. Available: https://doi.org/10.1109/WI-IAT.2013.85

[13] M. Perše, M. Kristan, J. Perš, G. Mušič, and S. K. G. Vučkovič, "Analysis of multi-agent activity using petri nets," *Pattern Recognition*, vol. 43, pp. 1491–1501, 2010. [Online]. Available: https://doi.org/10.1016/j.patcog.2009.11.011

[14] A. D. Torshizi and S. R. Hejazi, "A fuzzy approach to sequential failure analysis using petri nets," *International Journal of Industrial Engineering and Production Research*, vol. 21, no. 2, pp. 53–60, 2010.

[15] T. Fong, C. Thorpe, and C. Baur, *Collaborative control: A robot-centric model for vehicle teleoperation*. Carnegie Mellon University, The Robotics Institute, 2001, vol. 1.

[16] J. Brookshire, S. Singh, and R. Simmons, "Preliminary results in sliding autonomy for assembly by coordinated teams," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, 2004, pp. 706–711. [Online]. Available: https://doi.org/10.1109/IROS.2004.1389435

[17] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, vol. 30, no. 3, pp. 286–297, 2000. [Online]. Available: https://doi.org/10.1109/3468.844354

[18] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis, "Coactive design: Designing support for interdependence in joint activity," *Journal of Human-Robot Interaction*, vol. 3, no. 1, pp. 43–69, 2014. [Online]. Available: https://doi.org/10.5898/JHRI.3.1.Johnson

[19] S. K. Barber, A. Goel, and C. E. Martin, "Dynamic adaptive autonomy in multi-agent systems," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 2, pp. 129–147, 2000. [Online]. Available: https://doi.org/10.1142/S0218001401001015

[20] M. J. Barnes, J. Y. Chen, and F. Jentsch, "Designing for mixed-initiative interactions between human and autonomous systems in complex environments," in *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, 2015, pp. 1386–1390. [Online]. Available: https://doi.org/10.1109/SMC.2015.246

[21] N. Schurr, J. Marecki, and M. Tambe, "Riaact: A robust approach to adjustable autonomy for human-multiagent teams," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, 2008, pp. 1429–1432.

[22] M. Tambe, "Agent architectures for flexible, practical teamwork," in *Proc. of the 14th National Conf. on AI, USA: AAAI press*, 1997, pp. 22–28.

[23] J. M. Bradshaw, H. Jung, S. Kulkarni, M. Johnson, P. Feltovich, J. Allen, L. Bunch, N. Chambers, L. Galescu, R. Jeffers *et al.*, "Kaa: policy-based explorations of a richer model for adjustable autonomy," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, pp. 214–221. [Online]. Available: https://doi.org/10.1145/1082473.1082506

[24] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, 2003, pp. 93–96. [Online]. Available: https://doi.org/10.1109/POLICY.2003.1206963

[25] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer Verlag, 2003.