Adaptación vía Controladores Discretos Multi-Capa

1 Objetivos

La necesidad creciente de sistemas de software que pueden lidiar en tiempo de ejecución con cambios inesperados ha reforzado la importancia de tener métodos ingenieriles sólidos para la construcción de sistemas adaptativos [39]. Adaptabilidad es un requerimiento amplio que apunta a poder continuar operando a pesar de cambios en tiempo de ejecución de la disponibilidad de recursos, necesidades de usuarios, y condiciones inesperadas en el entorno como por ejemplo fallas. Adaptabilidad requiere que el sistema, dinámicamente, se reconfigure y optimice ocultando al usuario, en la medida de lo posible, este comportamiento complejo

Arquitecturas para sistemas adaptativos [22, 16, 25] son típicamente multi-capa, combinando ciclos MAPE (monitor, analysis, plan and execute) de control de distintos niveles de abstracción. Los niveles más bajos suelen ser de baja latencia y enfocan más en objetivos tácticos que no requieren de memoria (es decir "state-less"), mientras que los niveles superiores suelen centrarse en objetivos estratégicos de alta latencia y con memoria.

Para los niveles superiores, que suelen llamarse de planificación, los sistemas adaptativos utilizan representaciones explicitas del entorno de ejecución [25, 16]. Estos modelos de comportamiento (ej. Sistemas de transiciones etiquetadas) son procesados mediante técnicas algorítmicas (ej. planning [31, 41] y síntesis de controladores [35, 11]) para producir en tiempo de ejecución estrategias operacionales para que el sistema pueda lograr sus objetivos.

Estos modelos de comportamiento son idealizaciones del entorno real. Lo son porque la complejidad algorítmica de tratar con un modelo "completo" sería intratable en tiempo y espacio. Además, presunciones sobre el ambiente pueden ser aprovechadas para lograr objetivos de sistema más sofisticados (muy poco puede ser garantizado en un mundo en donde "todo" puede salir mal).

Todo modelo, pues, introduce incertidumbre [14] y por lo tanto riesgo. No podemos estar seguros que nuestras presunciones son validas, y por lo tanto tomamos riesgo al planificar en función de ellas. Uno de los objetivos detrás de la investigación en sistemas adaptativos es mitigar el riesgo (que surge de incertidumbre al tiempo de diseño) mediante decisiones en base a la información disponible en tiempo de ejecución [39].

A pesar del balance crucial que debe lograrse entre riesgo, robustez y la capacidad de garantizar propiedades complejas, los diseños existentes para sistemas adaptativos sólo contemplan un modelo del entorno sobre el cual hacer planificación [41, 11, 42, 13]. Con lo cual fijan el nivel de riesgo y los objetivos que son alcanzables, y se exponen a una falta de robustez: Cuando las presunciones sobre el ambiente no son válidas el sistema falla completamente o continua sin garantía alguna.

Nuestro *objetivo principal* es desarrollar diseños para sistemas adaptativos, y en particular para el nivel de planificación, que soporten múltiples niveles de riesgo y permitan transicionar, de acuerdo a la validez de las presunciones en cada nivel de riesgo, entre distintos niveles de garantía funcional. Esperamos que un diseño de este tipo permita una degradación y mejora gradual según la validez de las presunciones.

La beca solicitada se enmarca en un proyecto cuya *hipótesis general* es que la síntesis de controladores de eventos discretos puede ser el motor de razonamiento que permita proveer en tiempo de ejecución estrategias correctas por construcción para el nivel de planificación en sistemas adaptativos.

Síntesis de controladores construye automáticamente estrategias operacionales a partir de un modelo del entorno y un objetivo de sistema. Dichas estrategias pueden ser construidas en la capa de planificación en tiempo de ejecución en base a información que se va recolectando dinámicamente y requerimientos nuevos. Luego dichas estrategias pueden ser ejecutadas por un interprete.

La *hipótesis específica* que abordará el becario es que el nivel de planificación en sistemas adaptativos puede y debe ser diseñada como una jerarquía de problemas de control de eventos discretos. Dicho diseño permitiría tolerar distintos niveles de riesgo y garantías funcionales, además de proveer mecanismos de degradación y mejora gradual en tiempo de ejecución.

2 Antecedentes

Síntesis de Controladores Este plan está fuertemente vinculado a técnicas que construyen automáticamente estrategias operativas de comportamiento. Esto representa un campo amplio que toma de áreas como síntesis

de programas (e.g. [29]), desarrollo basado en modelos (e.g [18]), planning (e.g. [38, 2, 41]), control-supervisor (e.g. [23]) y síntesis de controladores (e.g. [33]).

La síntesis de controladores se origina en el trabajo de Büchi, Landweber, and Rabin (ej. [5]) y su actualización a formas más modernas [34]. El problema general de síntesis es computacionalmente muy caro, sin embargo trabajo reciente a apuntado a encontrar sub-lógicas que permitan tratar especificaciones mas restringidas con complejidad polinomial (e.g. [33, 26]). La comunidad ha puesto foco originalmente en sistemas embebidos y circuitos digitales lo cual ha llevado a el estudio de síntesis en el contexto de comunicación vía memoria compartida y formalismos de tipo Kripke Sin embargo, en áreas como ingeniería de requerimientos, diseño arquitectónico y sistemas adaptativos, un modelo de comunicación vía mensajes es mas apropiada y más recientemente se ha explorado síntesis para modelos de comportamiento basados en eventos [11].

Planning es un campo de la inteligencia artificial [37] que está fuertemente vinculado con síntesis de controladores pero donde el foco está en sacar provecho de la estructura de la especificación del problema. Es decir, en vez de trabajar en una representación semántica se hace un trabajo de manipulación sintáctica en donde mediante heurísticas se busca lograr una eficiencia mayor a costa de completitud. La génesis de este área no es en sistemas reactivos, con lo cual inicialmente se asumía un entorno completamente controlable y determinístico [31]. Estas presunciones han sido relajadas a través de los años y de hecho planning está convergiendo con síntesis en muchos aspectos (e.g. [20]).

Tanto planning como síntesis de controladores han ampliado su campo de acción a modelos cuantitativos para razonar sobre el costo y beneficio de distintas estrategias. En robótica, por ejemplo, políticas de control basadas en Partial Observable Markov Decision Processes es muy común (ej. [43]). La gran mayoría de estas técnicas no proveen garantías duras de lograr los objetivos y suelen basarse en estrategias "mejor esfuerzo". Algunos trabajos en combinar preferencias cuantitativas con objetivos duros de seguridad han comenzado a surgir ([44]) pero son computacionalmente muy complejas.

Sistemas Adaptativos Hay abundantes antecedentes en el campo de sistemas adaptativos. La robótica es el área de donde surgen, no siempre de manera explícita, las primeras arquitecturas para sistemas adaptativos. Aunque existen muchos enfoques distintos (ej. [16, 22, 8]) las nociones de multi-capa, loops de control y un ordenamiento de estos en estratégico/alta-latencia a táctico/baja-latencia son comunes a la mayoría. En capas más tácticas, aprendizaje bayesiano, teoría de control continua y predicción estadística son técnicas claves. En niveles más estratégicos o de planificación, las técnicas tienden a ser en la práctica más ad-hoc. Trabajo inicial en reconfiguración dinámica a nivel arquitectónico (ej. [28]) provee muchos de los fundamentos para estos niveles. Mientras que el interés en técnicas de generación de estrategias automáticamente ha crecido más recientemente.

Síntesis de código "glue" para componentes y conectores ha sido estudiado a nivel arquitectónico (ej. [1]) y en particular para arquitecturas orientadas a servicios [3, 30] Estos trabajos y otros [38, 2, 41, 42] se han concentrado principalmente en propiedades de "safety". Pocos trabajos han sido desarrollados en el contexto de sistemas adaptativos que puedan dar soporte para propiedades de "liveness" [20, 2, 21]. Sin embargo, el tratamiento de liveness requiere de presunciones de liveness que deben ser representadas explícitamente y pueden, si están mal enunciadas, llevar a controladores vacuos. Estas problemáticas han sido estudiadas y resueltas en [10].

Se han desarrollado implementaciones de referencia para adaptación de alto nivel como por ejemplo Rainbow [15]. Sin embargo el foco es en integrar con actuadores y sensores y no en técnicas de razonamiento y generación automática de estrategias.

La provisión de degradación y mejora gradual requiere del monitoreo de presunciones del ambiente, área que es de intensa actividad [45]. La degradación y mejora gradual de aspectos cuantitativos ha recibido mucha atención (e.g. [17]). Estos trabajos tienen el mismo espíritu que el presente plan, pero la posibilidad de ajustar parámetros numéricos para degradar aspectos no funcionales no se traduce a mecanismos de degradación para propiedades funcionales.

Algunos trabajos apuntan a intercambiar implementaciones alternativas de componentes o aplicar secuencias alternativas de acciones o incluso de roll-backs [6] Sin embargo, en estos casos hay un sólo requerimiento que se está tratando de lograr y diferentes estrategias para lograrlo son exploradas. La necesidad de un rango de requerimientos funcionales de distinto tipo que puedan ser intercambiados según la certeza sobre el comportamiento del ambiente es todavía una cuenta pendiente y claramente individualizada en el SEAMS roadmap[7].

El concepto de replanificación como una forma de tratar incertidumbre ha sido estudiado, por ejemplo, en [4], pero sin dar garantías de que los objetivos serán logrados en la medida que las presunciones sean válidas.

El problema de identificar presunciones válidas en tiempo de ejecución es ortogonal a esta propuesta y

es objeto de estudio en si mismo (ej. [12, 40, 13]. A su vez esta propuesta también es ortogonal a el trabajo realizado en temas de control continuo que suele ubicarse en capas más bajas de adaptación y donde las estrategias suelen estar basadas en sensado del estado del entorno y no la historia de interacciones. En las capas mas altas el control es de tipo "feed-forward" donde el controlador puede anticipar el resultado de acciones controladas y entonces puede planificar de manera adelantada como lograr objetivos de largo plazo (e.g. [19, 43]). Esto es también el principio detrás del modelo de referencia de tres capas para sistemas adaptativos[25] en el cual nuestra propuesta caería en la capa de administración de objetivos. En el modelo de referencia MAPE-K[8] nuestro trabajo encaja en los componentes de análisis y planificación.

Nuestro antecedente mas relevante para este proyecto es un trabajo presentado recientemente [9] donde se propone un diseño multi-capa para adaptación. Este trabajo es un primer paso, pero limitado, en la dirección propuesta para esta beca doctoral. El enfoque tiene múltiples limitaciones teóricas y prácticas que se esperan resolver. En particular el enfoque requiere una jerarquía lineal de controladores que no permite alternativas en la degradación y mejora gradual. Además, se requiere una relación de simulación muy restrictiva entre los distintos problemas de control, lo cual fuerza, por ejemplo a ubicar objetivos de safety en niveles de menor riesgo y propiedades de liveness en niveles de mayor riesgo. Esto puede ser una limitación importante ya que es común sacrificar propiedades de safety en pos de liveness cuando estas últimas no pueden ser garantizadas. El debilitamiento de restricciones topológicas y de dependencia entre-capas requerirá resolver problemas de actualización dinámica de controladores [32] que está íntimamente relacionado con la problemática general de actualización dinámica de software y "quiescence" [24].

3 Actividades y Metodología

Las actividades a desarrollar pueden agruparse en tres grupos. Desde una perspectiva teórica *(1)*, se estudiará el uso de jerarquías de problemas de control para lidiar con diferentes niveles de riesgo y servicio funcional, y también para permitir degradación y mejora gradual. Se trabajará en un propuesta integral desde el punto de vista sintáctico y semántico así como también aspectos algorítmicos y resultados teóricos que incluyen correctitud, completitud y cotas de complejidad.

Desde una perspectiva más práctica *(2)* se desarrollará un nivel de planificación en una arquitectura de adaptabilidad basado en jerarquías de problemas de control. Se deberán resolver problemas de sincronización entre controladores así como aspectos de meta-control como ser la detección de cuando desactivar un controlador (a causa de presunciones inválidas) y la decisión de qué controlador activar (ya que puede haber múltiples opciones de degradación/mejora). También deberá estudiarse la integración con frameworks para sistemas adaptativos existentes. Finalmente *(3)*, se deberá instanciar la infraestructura desarrollada en casos de estudio realistas. Se apuntará a casos de estudio dentro del dominio de la robótica/sistemas "ciber-físicos" así como dominios puramente de software como ser sistemas orientados a servicios.

El método a utilizar busca desarrollar resultados teóricos (fundamentos, algoritmos probadamente correctos, etc) pero también implementaciones concretas integradas en herramientas que tengan potencial para convertirse en tecnología transferible. Ciertamente, una de las características de este dominio es la necesidad de complementar resultados teóricos, validados analíticamente, con validación experimental basada en casos de estudio [36].

Nuestra hipótesis general está vinculada con la provisión de comportamiento correcto por construcción para adaptar sistemas a cambios en el entorno. Los problemas de control garantizan esto de manera individual, pero una jerarquía de problemas de control con un meta-controlador que activa y desactiva controladores no necesariamente es correcto y su correctitud no se deduce de manera directa a través de razonamiento composicional. Por lo tanto elicitar y probar las propiedades emergentes de la jerarquía de control será una validación analítica a desarrollar.

Por otro lado, la flexibilidad del diseño para poder utilizarse en una variedad de casos de estudio será otro aspecto clave a validar. El riesgo es que las restricciones entre niveles de la jerarquía de controladores que deberán establecerse para poder probar propiedades emergentes del diseño impidan su aplicación a casos reales. En consecuencia, analizaremos por un lado desde una perspectiva teórica el espacio de equilibrios entre mayores garantías globales y mayor flexibilidad entre capas de la jerarquía. Además, estudiaremos casos de estudio de la literatura para ver si pueden ser abordados por nuestro enfoque. Se espera que todos los casos puedan abordarse con una jerarquía de un sólo nivel, la clave será cuanta robustez adicional podrá proveerse mediante una descomposición, en la medida que sea posible, en capas con distintos grados de presunciones y riesgos.

Finalmente, para analizar rigurosamente el balance entre robustez y flexibilidad desarrollaremos modelos de fallas para modificar los modelos de entorno sobre los que se computaron estrategias para ver cómo el sistema adaptativo se comporta cuando es instalado en un entorno que diverge del entorno supuesto. Modelos de arboles de falla [27] serán utilizados. Compararemos estrategias construidas por técnicas existentes contra

estrategias multi-capa para mostrar el nivel de robustez logrado por nuestro enfoque. Estas comparaciones se harán de manera exhaustiva usando tecnología de model checking.

Escalabilidad también será objeto de estudio y validación. Por un lado se tomará un enfoque analítico para entender si las restricciones interpuestas por la jerarquía de problemas de control aporta complejidad adicional por sobre la complejidad de cada problema de control en aislamiento. También estudiaremos escalabilidad desde lo práctico adaptando casos de estudio para hacerlos paramétricos en términos de tamaño (por ejemplo en un caso de estudio que utiliza servicios web, parametrizaremos el número de servicios disponibles).

Finalmente, buscaremos realizar una validación punta a punta usando un caso de estudio en el dominio de robótica. Aquí buscaremos mostrar factibilidad del enfoque mediante la resolución de un problema real en un entorno de ejecución real.

4 Factibilidad

El plan de trabajo será desarrollado en el Departamento de Computación (DC) de a Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires. Más específicamente dentro de Laboratorio de Fundamentos y Herramientas para la Ingeniería de Software (LaFHIS). Entre el Departamento y el Laboratorio se cubrirán los gastos correspondientes al equipamiento necesario para el desarrollo de las tareas incluyendo computadora, conectividad, servicios de impresión y resguardo de datos, además de una oficina, lugar de trabajo propio e insumos básicos de oficina. LaFHIS, brindará equipamiento robótico (un cuadróptero y un lego mindstorm) para realizar pruebas de concepto. Además, se utilizará un hexacóptero de especificaciones industriales, construido en la Facultad de Ingeniería de la UBA por el Dr. Giribet, para los casos de estudio finales. LaFHIS proveerá de fondos para que el becario pueda viajar a conferencias relevantes a través de sus proyectos de investigación activos (PICT 2011 N°1774 y PICT 2012 N°0724).

References

[1] M. Autili, P. Inverardi, M. Tivoli, and D. Garlan. Synthesis of "correct" adaptors for protocol enhancement in component-based systems. In *Specification and Verification of Component-Based Systems*, page 79. ACM, 2004.

[2] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. MBP: a model based planner. In *Proceedings of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information*, 2001.

[3] A. Bertolino, P. Inverardi, P. Pelliccione, and M. Tivoli. Automatic synthesis of behavior protocols for composable web-services. In *Proceedings of 7th ESEC/FSE*, pages 141–150, New York, NY, USA, 2009. ACM.

[4] R. I. Brafman and G. Shani. Replanning in domains with partial information and sensing actions. *J. Artif. Intell. Res. (JAIR)*, 45:565–600, 2012.

[5] J. Buchi and L. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, pages 295–311, 1969.

[6] A. Carzaniga, A. Gorla, A. Mattavelli, N. Perino, and M. Pezze. Automatic recovery from runtime failures. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 782–791. IEEE Press, 2013.

[7] B. H. Cheng et al. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *LNCS*, pages 1–26. Springer Berlin Heidelberg, 2009.

[8] A. Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 2006.

[9] N. D'Ippolito, V. Braberman, J. Kramer, J. Magee, D. Sykes, and S. Uchitel. Hope for the best, prepare for the worst: Multi-tier control for adaptive systems. In *International Conference on Software Engineering*, 2014.

[10] N. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesising non-anomalous event-based controllers for liveness goals. *ACM Tran. Softw. Eng. Methodol.*, 22, 2013.

[11] N. R. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesis of live behaviour models. In *Proceedings of the Int. Symp. on Foundations of Soft. Eng.*, FSE '10, pages 77–86. ACM, 2010.

[12] A. Elkhodary, N. Esfahani, and S. Malek. Fusion: a framework for engineering self-tuning self-adaptive software systems. In *Proceedings of the 18th SIGSOFT Int. Symp. on Foundations of Soft. Eng.*, pages 7–16. ACM, 2010.

[13] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time parameter adaptation. In *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*, pages 111–121. IEEE, 2009.

[14] N. Esfahani and S. Malek. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems*, volume 7475 of *Lecture Notes in Computer Science*, pages 214–238. Springer, 2010.

[15] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, Oct. 2004.

[16] E. Gat. *Three-layer Architectures, Artificial Intelligence and Mobile Robots*. MIT/AAAI Press, 1997.

[17] C. Ghezzi, L. S. Pinto, P. Spoletini, and G. Tamburrelli. Managing non-functional uncertainty via model-driven adaptivity. In *Proceedings of the Intl. Conf. on Software Engineering*, pages 33–42. IEEE Press, 2013.

[18] H. Giese and W. Schaefer. Model-driven development of safe self-optimizing mechatronic systems with mechatronicUML. Technical Report tr-ri-12-322, Heinz Nixdorf Institute, University of Paderborn, Apr. 2012.

[19] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design and analysis of hybrid systems, with applications to robotic aerial vehicles. In *ISRR*, volume 70 of *Tracts in Advanced Robotics*, pages 139–149. Springer, 2009.
[20] F. Giunchiglia and P. Traverso. Planning as model checking. In S. Biundo and M. Fox, editors, *ECP*, volume 1809 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 1999.
[21] W. Heaven, D. Sykes, J. Magee, and J. Kramer. Software engineering for self-adaptive systems. chapter A Case Study in Goal-Driven Architectural Adaptation, pages 109–127. Springer-Verlag, Berlin, Heidelberg, 2009.
[22] A.-C. Huang, D. Garlan, and B. Schmerl. Rainbow: Architecture-based self-adaptation with reusable infrastructure. In *Proceedings of the Intl. Conf. on Autonomic Computing*, pages 276–277. IEEE CS, 2004.
[23] S. Jiang and R. Kumar. Supervisory control of discrete event systems with ctl* temporal logic specifications. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 5, pages 4122–4127, 2001.
[24] J. Kramer and J. Magee. The evolving philosophers problem: Dynamic change management. *IEEE Trans. Softw. Eng.*, 16(11):1293–1306, Nov. 1990.
[25] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In *2007 Future of Software Engineering*, FOSE '07, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society.
[26] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *Proceedings of the IEEE Symp. on Found. of Computer Science*, FOCS '05, pages 531–542, Washington, DC, USA, 2005. IEEE CS.
[27] N. G. Leveson. *Safeware: System Safety and Computers*. ACM, New York, NY, USA, 1995.
[28] J. Magee and J. Kramer. Dynamic structure in software architectures. In *Proceedings of the Symp. on Found. of Soft. Eng.*, SIGSOFT '96, pages 3–14, New York, NY, USA, 1996. ACM.
[29] Z. Manna and R. Waldinger. A deductive approach to program synthesis. *ACM Trans. Program. Lang. Syst.*, 2(1):90–121, Jan. 1980.
[30] A. Mukhija, D. S. Rosenblum, H. Foster, and S. Uchitel. Runtime support for dynamic and adaptive service composition. In *Results of the SENSORIA Project*, volume 6582 of *LNCS*, pages 585–603. Springer, 2011.
[31] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[32] V. Panzica La Manna, J. Greenyer, C. Ghezzi, and C. Brenner. Formalizing correctness criteria of dynamic updates derived from specification changes. In *Proceedings of the Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems*, pages 63–72. IEEE Press, 2013.
[33] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. *Lecture notes in computer science*, 3855:364–380, 2006.
[34] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the Symp. on Principles of Programming Languages*, pages 179–190, New York, NY, USA, 1989. ACM.
[35] P. Ramadge and W. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, 1989.
[36] P. Runeson, M. Host, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley, 2012.
[37] S. Russell and P. Norvig. Artificial intelligence: a modern approach. *New Jersey*, 1995.
[38] M. Schoppers. Universal plans for reactive robots in unpredictable environments. In *IJCAI*, volume 87, pages 1039–1046. Citeseer, 1987.
[39] *ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS*. ACM/IEEE, 2006-2013.
[40] D. Sykes, D. Corapi, J. Magee, J. Kramer, A. Russo, and K. Inoue. Learning revised models for planning in adaptive systems. In *Proceedings of ICSE*, 2013.
[41] D. Sykes, W. Heaven, J. Magee, and J. Kramer. From Goals to Components: A Combined Approach to Self-Management. In *Proc. of the Workshop on Soft. Eng. for Adaptive and Self-Managing Systems*, 2008.
[42] H. Tajalli, J. Garcia, G. Edwards, and N. Medvidovic. Plasma: a plan-based layered architecture for software model-driven adaptation. In *Proceedings of the Int. Conf. on Automated Soft. Eng.*, pages 467–476. ACM, 2010.
[43] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.
[44] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus. Optimality and robustness in multi-robot path planning with temporal logic constraints. *International Journal of Robotics Research*, 32(8):889–911, 2013.
[45] *Workshop on Models at Runtime Series*. ACM/IEEE, 2006–2013.