Responsabilidade, Responsabilidade e Robustez
em Organizações de Agentes

Matteo Baldoni[0000-0002-9294-0408](B),Cristina Baroglio[0000-0002-2070-0616], e
Roberto Micalizio[0000-0001-9336-0651]

Università degli Studi di Torino — Dipartimento di Informatica
c.so Svizzera 185, I-10149 Torino
(Itália) nome.sobrenome@unito.it

Abstrato.As organizações de agentes são amplamente utilizadas para o projeto e desenvolvimento de sistemas distribuídos. As organizações de agentes, no entanto, carecem de uma maneira sistemática de tratar exceções. Neste artigo, apresentamos as organizações ARFIN como uma nova maneira de conceituar sistemas multiagentes. Caracterizamos as organizações ARFIN em termos de*responsabilidade*e*responsabilidade*noções que complementam o sistema normativo de uma organização padrão. O ARFIN melhora a robustez do sistema resultante, pois permite um tratamento sistemático de exceções, detectando-as e reportando-as ao agente com as capacidades adequadas para tratá-las.

Palavras-chave:Responsabilidade·Responsabilidade·Organizações de agentes·Engenharia MAS.

1. Introdução

As organizações de agentes [16, 19, 11] são uma abstração bem conhecida para modularizar o software decompondo funcionalmente o objetivo organizacional em subobjetivos. Estes são, então, atribuídos aos agentes pela organização por meio de obrigações. A obediência, no entanto, não pode ser dada como garantida. Os agentes cumprirão uma obrigação somente quando esta for funcional para a realização de seus próprios objetivos, podendo ocorrer falhas. Isso não é um problema em si, o problema é que as organizações não fornecem meios para lidar sistematicamente com as informações que fizeram a execução desviar do esperado. Uma perspectiva interessante é traçada por trabalhos, como [8, 6, 3], que defendem que o software baseado em agentes deve ser modularizado em termos das responsabilidades que cada agente assume autonomamente, em relação aos objetivos que se espera que persiga.

O problema tem implicações na engenharia de software em geral e torna-se mais relevante pelo facto de assistirmos a uma crescente disseminação de software “inteligente e autónomo” em muitos aspetos da nossa vida quotidiana. De fato, já confiamos ao software, às vezes até sem perceber, decisões críticas em diversos cenários. Os aviões modernos, por exemplo, são equipados com pilotos automáticos (software) que não apenas auxiliam os pilotos a manter a rota, mas também se sobrepõem aos pilotos humanos quando suas decisões podem colocar em risco a segurança da aeronave. É evidente, portanto, que, quando um software inteligente tem o poder de tomar decisões de forma autônoma, temos um sério problema de tornar esse software *responsável*, ou seja, sob determinadas condições, ele fornece contas do que foi alcançado ou do que deu errado.

Tentar tornar um software existente responsável pode ser uma tarefa árdua. É conveniente pensar na responsabilidade desde o projeto ao longo das fases de desenvolvimento do software. Neste artigo, apresentamos alguns resultados preliminares (ver [3, 4]) e algumas perspectivas interessantes sobre uma nova maneira de conceituar software responsável. Especificamente, levamos em consideração o software que é distribuído na natureza e que pode ser visto como uma organização de sistema multiagente (MAS) (por exemplo, [1]): componentes independentes (ou seja, agentes) atuando simultaneamente e que cooperam para alcançar um objetivo global . As organizações de agentes representam estratégias de decomposição de metas organizacionais complexas em subtarefas mais simples e alocando-as em funções. Ao adotar papéis na organização, os agentes assumem responsabilidades e executam as tarefas correspondentes de forma distribuída,

Os modelos actuais, apesar de visarem os sistemas abertos através da atribuição e imposição de direitos e deveres aos agentes sobre as tarefas a realizar, carecem de uma representação explícita de relações dirigidas entre os agentes que permita aos agentes identificar quem deve restituir a quem por um determinado estado do organização. Tal representação supriria uma carência atual das organizações de agentes, ou seja, mesmo que os agentes que ingressam na organização estejam regulamentados por normas, que estipulam seus direitos e deveres, a organização não dispõe de meios para solicitar feedback dos agentes sobre sua conduta. Feedback que, comunicado a quem tem o direito de o obter, pode ser utilizado para fins de certificação, ou para fazer face a situações excecionais.

Afirmamos que a realização do software responsável deve ser fundamentada na representação explícita de dois conceitos fundamentais na base das organizações humanas: prestação de contas e responsabilidade.

2 Organizações ARFIN

Pensamos na organização do MAS como um processo executado coletivamente por diversos agentes envolvidos, que geralmente dependem uns dos outros para a realização de suas atividades. Em sua interação, os agentes produzem e respondem a eventos institucionais, e precisam se coordenar para atingir o objetivo organizacional. Agora apresentamos os conceitos-chave no núcleo de um ARFIN MAS, que são*responsabilidade*,*responsabilidade*,*fitting*, e*normas*.

De acordo com Grant e Keohane [13], a responsabilidade implica que alguns atores têm o direito de exigir que outros atores cumpram um conjunto de padrões, julgar se eles cumpriram suas responsabilidades à luz desses padrões e impor sanções se determinarem que esses padrões foram cumpridos. responsabilidades não foram cumpridas. Eles explicam que a responsabilidade pressupõe uma relação entre os detentores do poder e aqueles que os responsabilizam, onde há um reconhecimento geral da legitimidade de (1) os padrões operativos de responsabilidade e (2) a autoridade das partes no relacionamento (um para exercer poderes particulares e o outro para responsabilizá-los). Ainda em [13], os mecanismos de responsabilização operam sempre após o fato (ex post), expondo as ações à vista, julgando-as e sancionando-as. No entanto, devido à antecipação de sanções, os mecanismos de responsabilização podem exercer efeitos também ex ante, direcionando o comportamento para a norma [2]. À luz desse entendimento, e com base em [3], representamos explicitamente as responsabilidades comoA(*x, y, r, u*), significa que*x*, o prestador de contas, é responsável perante *y*, o tomador da conta, pela condição*você*quando a condição*r*(*contexto*) detém. Ambos*r*

e *você* são expressões temporais, dadas em lógica de precedência [20]. Cada prestação de contas produz tanto uma *permissão* e um *obrigação*: a permissão para *y* pedir uma conta de *você* quando a condição *r* é verdadeiro, e a obrigação de *x* fornecer tal conta quando *y* tem o direito de reclamar por isso. Através da prestação de contas *x* declara estar ciente de tal permissão e obrigação, e isso é suficiente para *y* esperar e ligar *x* comportar-se de acordo com tal padrão. Isso, por sua vez, produz uma relação direta entre *x* e *y*, onde *x* é responsável perante *y* de fornecer uma conta sobre *você* quando a condição *r* detém. Observe que a prestação de contas não cria uma obrigação sobre *x* realizar *você*.

O segundo componente chave é *responsabilidade*, um termo poliédrico; em particular, [22] propõe uma ontologia relacionando seis diferentes conceitos de responsabilidade (capacidade, causal, papel, resultado, virtude e responsabilidade), decorrente da parábola de "Smith, o capitão" do filósofo jurídico Herbert LA Hart. Grosso modo, equivalem respectivamente a: fazer a coisa certa, ter deveres, resultado imputável a alguém, condição que produziu algo, capacidade de compreender e decidir o que fazer, algo juridicamente imputável. Vemos a assunção de responsabilidade como uma declaração, por um agente, de ser destinatário de (e ser movido por) algum evento institucional. No entanto, a responsabilidade não implica que se espera que o agente forneça qualquer feedback,

A responsabilidade também não implica que o agente tenha capacidade para realizar uma determinada tarefa. Ainda assim, quando o agente os possui, pode nem sempre estar disposto a agir conforme o esperado, ou pode falhar na tentativa.

Os eventos institucionais são gerados por um *sistema normativo*, que é capaz de gerar obrigações, permissões etc., dependendo da ocorrência de eventos no mundo físico e no mundo institucional. As obrigações, entretanto, não são suficientes, veja [8, 6], porque os agentes não necessariamente irão aceitá-las. Sempre que os objetivos do agente estiverem em conflito com os objetivos da organização, e a sanção esperada for aceitável, o agente ignorará a obrigação. Por meio de pressupostos de responsabilidade (que são fornecidos pelos próprios agentes) identificamos aqueles agentes receptivos à obrigação, mas é somente por meio da prestação de contas (que também é fornecida pelos agentes) que se torna possível responsabilizar os agentes por sua execução. A ligação entre responsabilidades e responsabilidades é realizada através da *ajuste de responsabilidade* (*apropriado* para abreviar, [3]). Assim, na emissão de alguma obrigação, não apenas um agente estará receptivo, mas em determinadas condições, também estará sujeito a um segundo agente que, por meio das permissões concedidas pela prestação de contas, tem autoridade para solicitar ao primeiro agente uma conta. Por meio da conta, será possível identificar falhas no sistema – por exemplo, um agente que não tem capacidade para lidar com um caso ou dano devido à exposição a condições exógenas não gerenciadas.

Por brevidade, nos referimos a uma organização MAS que inclui todos os elementos como um *organização ARFIN*. As organizações ARFIN são particularmente interessantes por dois motivos principais. Em primeiro lugar, porque cumprem os requisitos que se impõem à coordenação dos agentes pela autonomia dos agentes. Na ausência de introspecção, a autonomia do agente, de fato, exige uma forma de conceituar a coordenação onde os agentes são claramente constrangidos em termos de responsabilidades (que assumem explicitamente) e onde os agentes estabelecem relações dirigidas

de um para outro, que refletem as expectativas legítimas que o segundo principal tem do primeiro. Em segundo lugar, porque suportam a robustez do sistema. Em [12], robustez em sistemas de software é definida como “a capacidade de um software manter um comportamento 'aceitável', expresso em termos de *requisitos de robustez*, apesar de *excepcional* ou *imprevisto* condições de execução (como indisponibilidade de recursos do sistema, falhas de comunicação, entradas inválidas ou estressantes, etc.)”. Lançando tal definição no contexto das organizações, uma organização é *robusto* quando seus agentes podem reagir adequadamente a eventos inesperados. O driver de tal processo é a tentativa de executar até os padrões pré-estabelecidos, possivelmente através de auto-regulação, adaptando a execução ou a própria organização a casos inicialmente não previstos. Este processo depende fortemente das contas que se espera que os agentes envolvidos produzam.

3 Prestação de Contas e Responsabilidade pelo Tratamento de Exceções

As organizações de agentes são geralmente especificadas em termos de *funções*, que devem ser adotados pelos agentes, *tarefas* (por exemplo, ações, metas, interações) atribuídos a papéis por meio de *normas* [9, 7]. As tarefas são organizadas de acordo com uma decomposição funcional que especifica como uma meta organizacional complexa pode ser alcançada pela distribuição e coordenação do trabalho. Os modelos organizacionais atuais não captam diretamente a noção de prestação de contas e responsabilidade conforme descrito acima. Em nossa opinião, isso dificulta o desenvolvimento de uma organização robusta.

Especificamente, nossa afirmação é que o papel fundamental da responsabilidade no desenvolvimento de software (juntamente com a responsabilidade) é a realização de sistemas robustos. Seguindo [18], sobre projeto de software, uma pergunta útil é “Quais eventos estão entrando em nosso sistema? Por que? Porque temos que projetar o software para lidar com esses eventos [...] Basicamente, um sistema de software reage a três coisas: 1) eventos externos de atores (humanos ou computadores), 2) eventos de timer e 3) falhas ou exceções” . A robustez é alcançada garantindo, por design, que eventos excepcionais, quando ocorrem, são relatados aos agentes que podem tratá-los adequadamente. A prestação de contas cumpre esse propósito porque, por natureza, impõe ao doador a obrigação de prestar contas do que faz. A conta, então, pode ser utilizada pelo a-taker para recuperar, quando possível, da situação excepcional. Potencialmente, o tomador poderia passar a conta para outros agentes para os quais é um doador. Propomos, assim, complementar a decomposição funcional do objetivo organizacional, que é a espinha dorsal do sistema normativo subjacente, com um conjunto de especificações de prestação de contas e responsabilidades.

Considerando uma prestação de contas $A(x, y, r, u)$, se pensarmos em um processo sendo executado coletivamente, podemos dizer que quando o r parte do processo é feito, então x torna-se responsável pelo *você* papel. Quando *você* é verdade, x considera-se que construiu uma prova que pode ser fornecida ao contabilista. Uma prova, aqui, é o que se entende como um conjunto de fatos registrados, que demonstram o alcance da condição especificada. Quando, em vez disso, *você* for falso, espera-se novamente que o agente forneça uma prova a pedido do que aconteceu. Quando r for falsa, ao contrário, a condição sob a qual o a-taker tem o direito de exigir uma prova não mais se sustenta, e o a-doador não terá a obrigação de fornecer uma prova a pedido. Em vez disso, intuitivamente com $R(x,q)$, x declara aceitar ser considerado na condição de causador q. Isso pode implicar que x tem capacidade de produzir

q diretamente, ou que *x* pode exercer algum controle sobre algum outro agente que trará *q*. denotamos por *A* um conjunto de responsabilidades, chamando-o de *especificação de responsabilidade*, e por *R* a *distribuição de responsabilidade*, ou seja, um conjunto de pressupostos de responsabilidade que complementam a especificação de uma organização agente.

Em geral, o fato de um grupo de agentes ser compatível com um determinado ajuste confere robustez ao sistema de agentes interagentes. Do ponto de vista da implementação, é conveniente focar no conjunto de possíveis eventos relevantes para o propósito de prestação de contas. Dizemos que o a-taker é robusto em relação a determinados eventos quando, na ocorrência deles, presta conta proativamente ao a-doador, sem esperar que seja solicitado. Apenas para dar uma ideia da configuração técnica, usamos a residuação [5] para calcular o progresso das responsabilidades e das suposições de responsabilidade. Deixar *você* ser o universo do discurso (o conjunto de eventos sobre os quais as expressões temporais são formadas). Deixar *e* ser uma sequência de eventos em *você*, então $A(x, y, r/e, você/e)$ denota o resíduo de $A(x, y, r, u)$ em relação a *e* (intuitivamente o que resta após a ocorrência do sequência de eventos *e*). Residuação é o mecanismo pelo qual a permissão e obrigações que estão naturalmente envolvidas em uma prestação de contas são acionadas. Quando $r/e = 0$, a responsabilidade não é mais significativa; quando $r/e = \top$, *y* tem permissão para perguntar *x* uma conta sobre *você*, e *x*, em caso de solicitação, é obrigado a fornecer tal conta, em particular quando a condição é alcançada ($você/e = \top$) e quando algo dá errado ($você/e = 0$). No primeiro caso, a conta é *você* em si. No segundo caso, em vez disso, sequência *e* contém pelo menos um evento *e* que é complementar a algum outro evento em *você*: a ocorrência de *e* impede a satisfação de *você*. A pedido de *y*, *x* será obrigado a prestar contas sobre o descumprimento de *você*, e, portanto, será obrigado a divulgar a ocorrência do evento excepcional *e*.

vamos ligar *realização* de uma expressão temporal *q* qualquer sequência *e* de eventos em *você* de tal modo que $\dot{q}/e = \top$; denotamos tal atualização como $q\hat{}$.

Definição 1 (Encaixe de responsabilidade [3]). *Dada uma especificação de responsabilidade A e uma distribuição de responsabilidade R, dizemos que R fits A, denotado por RA, se para cada responsabilidade* $A(x, y, r, u) \in A$*, há uma responsabilidade* $R(x, q) \in R$ *de tal modo que* $(u/r)/q\hat{} \cong \top$*, para alguma atualização* $q\hat{}$.

Intuitivamente, o encaixe *R* *A* relaciona uma declaração de responsabilidade $R(x, q) \in R$ para uma (ou até mais) relação de responsabilidade $A(x, y, r, u)$ que é suportado por essa responsabilidade através de uma de suas atualizações. O encaixe tem uma consequência importante no desenho da organização: quando a condição *RA* sustenta, existe pelo menos um caminho, ao longo da decomposição funcional do objetivo, que é coberto por responsabilidades e pressupostos de responsabilidade [3]. Isso significa que, nesse caminho, há sempre um papel que assume a responsabilidade por uma tarefa e responde por ela. A adequação garante que uma organização seja especificada de forma coerente. Além disso, é também uma especificação de uma organização robusta no sentido de que eventos excepcionais são reportados.

Propomos, então, caracterizar o encaixe com o *eventos excepcionais* que, por razões dependentes do domínio, são consideradas críticas. Deixar *E* ser um conjunto de eventos excepcionais, ou seja, $E \cap U = \emptyset$ e cada evento $e \in E$ é complementar a algum evento em *você*. Em geral, o mesmo evento *e* poderia ser considerado complementar a muitos eventos em *você*. Relação $F \subseteq U \times E$ mapeia eventos em *você* aos seus correspondentes complementares em *E*. Nós

dizer que uma expressão *você é tocado* por uma exceção $e \in E$ se para pelo menos um evento c ocorrendo em *você*, (*nós*) $\in F$. Por extensão, uma relação de responsabilidade $A(x, y, r, u)$ é tocado pela ocorrência do evento e quando c ocorre em *você* e (*nós*) $\in F$. Deixar $[RA]_F$ ser um ajuste de responsabilidade caracterizado por F. Uma organização ARFIN está em conformidade com $[RA]_F$ se, sempre que $A(x, y, r, u) \in A$ é tocado por um evento $e \in E$, uma conta sobre *você* é solicitado a x por padrão.

4. Discussão

Sistemas multiagentes normativos (NorMAS) [17, 7] têm sido amplamente estudados na área de pesquisa em SAM. Segundo [7] uma NorMAS é: "um sistema multiagente juntamente com sistemas normativos em que os agentes, por um lado, podem decidir se seguem as normas explicitamente representadas e, por outro, os sistemas normativos especificam como e em que medida os agentes podem modificar as normas". Curiosamente, as normas especificam ações institucionais ou as condições para o uso de tais ações, consequentemente regulando o comportamento aceitável dos agentes em um sistema ("fazendo a coisa certa" em vez de "fazer o que leva a um objetivo" [21]). . Organizações de agentes, como aquelas modeladas em *M*OISE [15], são um caso especial de NorMAS que se baseiam basicamente em obrigações para o andamento da execução. Intuitivamente, o sistema normativo se encarrega de emitir obrigações aos agentes nos momentos certos. Se os agentes cumprirem suas tarefas, a execução progride em direção ao objetivo organizacional. Quando os agentes falham em cumprir uma obrigação, não há mecanismo para lidar com a falha. Na maioria dos casos, uma violação leva à sanção do agente responsável, mas as sanções são dissuasivas, que devem impedir um agente de agir fora das normas e não suportam nenhum mecanismo de recuperação. As soluções adotadas na prática nem sempre são eficazes. Em *M*OISE, por exemplo, a organização continua emitindo para o mesmo agente a obrigação não satisfeita. Se as causas da falha estiverem fora do escopo do agente, porém, a reemissão da obrigação não resolverá o problema.

Protocolos baseados em compromisso, por exemplo [23], fornecem uma alternativa para modelagem de coordenação. Grosso modo, um compromisso é uma promessa que um devedor faz em favor de um credor de que caso alguma condição antecedente seja satisfeita, o devedor acarretará uma condição consequente. Quando o antecedente é válido, o compromisso é destacado e equivale a uma obrigação do devedor de provocar o consequente. Quando o conseqüente não é mais atingível, o compromisso é violado. Nesse caso, o credor tem o direito de reclamar contra o devedor, no entanto, o credor não pode exigir que o devedor forneça uma explicação. Essa falta de informação dificulta tanto o entendimento do que de fato ocorreu, quanto qualquer tentativa de recuperação da falha. A introdução da accountability nesse quadro ajudaria a superar tais limites,

Tanto as organizações de agentes quanto os protocolos baseados em compromisso, portanto, tendem a ser frágeis, pois as exceções não podem ser tratadas sistematicamente. Em vez disso, o ajuste de responsabilidade que discutimos aqui oferece suporte à robustez, permitindo um gerenciamento sistemático de exceções. Notadamente, o encaixe não substitui o sistema normativo, mas sim o encaixe acompanha as normas de organização e a decomposição funcional de seu objetivo, as-

garantindo que, quando necessário, seja possível obter informações (por exemplo, explicações) dos agentes. De fato, as responsabilidades criam obrigações para fornecer contas para as condições de interesse, desde que algumas condições contextuais sejam válidas. A conta é captada em primeiro lugar pelo a-taker, que pode tentar se recuperar da falha ou, como segunda possibilidade, encaminhar a conta ao seu a-taker (se houver). As responsabilidades desempenham seu papel no quadro: quando um agente assume uma função e assume todas as responsabilidades que lhe são impostas, o agente declara estar ciente das funções que pode ser incumbido de cumprir dentro da organização. Em termos de software, o agente declara ter um plano que é desencadeado por um evento institucional (por exemplo, a emissão de uma obrigação), e que irá desencadear qualquer tarefa que esteja sob sua responsabilidade. Claro,

Também [10] reconhece o valor da responsabilidade no desenvolvimento de software e faz uma proposta complementar à nossa. Especificamente, os autores focam a questão da produção de respostas na presença de uma relação de accountability, problema que envolve: como definir adequadamente a janela temporal a ser considerada? Quais informações são relevantes e, portanto, devem ser mantidas nesse intervalo temporal? Quais perguntas são adequadas para serem feitas nesse cenário? O agente prestador de contas produz uma resposta em termos de seus mecanismos internos. O que essa proposta não oferece é a visão organizacional do sistema de agentes interagentes e não contempla robustez e exceções.

As organizações ARFIN estabelecem as bases para o desenvolvimento de um mecanismo para lidar com exceções em sistemas baseados em agentes. Uma estratégia de referência é aquela explorada no modelo de ator (por exemplo, [14]): um ator geralmente cria outros atores-filhos atribuindo-lhes tarefas específicas. Quando um ator-filho não consegue lidar com uma situação e recebe uma exceção, ele geralmente reporta a exceção ao seu ator-pai. A lógica é que o ator pai é o único interessado em realizar a tarefa e possui as informações corretas para lidar com a exceção adequadamente. Em um sistema baseado em agente, tal mecanismo não é diretamente aplicável, pois os agentes são entidades independentes e não estão relacionados por um relacionamento pai-filho. As responsabilidades podem preencher essa lacuna.

Reconhecimentos

Os autores gostariam de agradecer a Olivier Boissier pelas discussões estimulantes.

Referências

1. Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., Padget, J. (eds.): Estruturas de Coordenação Social para Sistemas Técnicos Sociais, Law, Governance and Technology Series, vol. 30. Springer International Publishing (agosto de 2016). https://doi.org/10.1007/978-3-319-33570-4, https://hal-emse.ccsd.cnrs.fr/emse-01355372
2. Anderson, PA: Justificativas e precedentes como condicionantes na tomada de decisões em política externa. Jornal Americano de Ciência Política25(4) (1981)
3. Baldoni, M., Baroglio, C., Boissier, O., May, KM, Micalizio, R., Tedeschi, S.: accountability e responsabilidade em organizações de agentes

4. Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., Tedeschi, S.: Responsabilidade e Agentes para Processos de Negócios de Engenharia. In: Bordini, RH, Dennis, LA, Lesperance, Y. (eds.) Proc. do 7º International Workshop on Engineering Multi-Agent Systems, EMAS 2019, realizado em conjunto com o AAMAS 2019. Montreal, Canadá (13 a 14 de maio de 2019), http://cgi.csc.liv.ac.uk/ lad/emas2019 /
5. Baldoni, M., Baroglio, C., Chopra, AK, Singh, MP: Composição e verificação de protocolos multiagentes baseados em compromisso. Em: Wooldridge, M., Yang, Q. (eds.) Proc. da 24ª Conferência Conjunta Internacional sobre Inteligência Artificial, IJCAI 2015. Buenos Aires, Argentina (25 a 31 de julho de 2015), http://ijcai-15.org/
6. Baldoni, M., Baroglio, C., May, KM, Micalizio, R., Tedeschi, S.: Responsabilidade Computacional em Organizações MAS com ADOPT. Ciências Aplicadas8(4) (2018)
7. Boella, G., van der Torre, LWN, Verhagen, H.: Introdução aos sistemas multiagentes normativos. In: Sistemas Multiagentes Normativos. Dagstuhl Seminar Proceedings, vol. 07122 (2007)
8. Chopra, AK, Singh, MP: De máquinas sociais a protocolos sociais: Fundamentos de engenharia de software para sistemas sociotécnicos. Em: Proc. do 25º Int. conf. na WWW (2016)
9. Coutinho, LR, Sichman, JS, Boissier, O.: Dimensões de modelação para organizações de agentes. In: Manual de pesquisa em sistemas multiagentes: Semântica e dinâmica de modelos organizacionais, pp. 18–50. IGI Global (2009)
10. Cranefield, S., Oren, N., Vasconcelos, W.: Responsabilização por agentes de raciocínio prático. In: AT 2018: 6ª Conferência Internacional sobre Tecnologias de Acordo. LNCS (2018)
11. Craven, R., Sergot, MJ: vertentes do agente na linguagem de ação nc+. J. Lógica Aplicada6(2), 172–191 (2008)
12. Fernandez, JC, Mounier, L., Pachon, C.: Uma abordagem baseada em modelo para testes de robustez. In: Proceedings of the 17th IFIP TC6/WG 6.1 International Conference on Testing of Communicating Systems. pp. 333–348. TestCom'05 (2005)
13. Grant, RW, Keohane, RO: Responsabilidade e abusos de poder na política mundial. A Revisão de Ciência Política Americana99(1) (2005)
14. Haller, P., Sommers, F.: Atores em Scala - programação simultânea para a era multi-core. Ártima (2011)
15. Hübner, JF, Sichman, JS, Boissier, O.: Desenvolvendo sistemas multiagentes organizados usando o MOISE. IJAOSE1(3/4), 370–395 (2007). https://doi.org/10.1504/IJAOSE.2007.016266, https://doi.org/10.1504/IJAOSE.2007.016266
16. Jones, AJI, Sergot, MJ: Uma caracterização formal do poder institucionalizado. Revista Lógica do IGPL4(3), 427–443 (1996)
17. Jones, AJ, Carmo, J.: Lógica deôntica e contra-deveres. In: Gabbay, D. (ed.) Handbook of Philosophical Logic, pp. 203–279. Kluwer (2001)
18. Larman, C.: Aplicando UML e Padrões: Uma Introdução à Análise e Projeto Orientados a Objetos e Desenvolvimento Iterativo (3ª Edição). Prentice Hall PTR, Upper Saddle River, NJ, EUA (2004)
19. Sergot, MJ: Uma teoria computacional de posições normativas. ACM Trans. Comput. Registro. 2(4), 581–622 (2001)
20. Singh, MP: Implementação Distribuída de Fluxos de Trabalho Multiagentes: Lógica Temporal para Composição de Serviços Web. In: Segunda Conferência Conjunta Internacional sobre Agentes Autônomos e Sistemas Multiagentes, AAMAS 2003, 14 a 18 de julho de 2003, Melbourne, Victoria, Austrália, Anais. pp. 907–914. ACM (2003)
21. Therborn, G.: De volta às normas! sobre o alcance e a dinâmica das normas e da ação normativa. Sociologia Atual50,863–880 (2002)
22. Vincent, NA: Responsabilidade Moral, Biblioteca de Ética e Filosofia Aplicada, vol. 27, cap. Uma Taxonomia Estruturada de Conceitos de Responsabilidade. Primavera (2011)
23. Yolum, P., Singh, MP: Commitment Machines. In: Agentes Inteligentes VIII, 8ª Int. WS, ATAL 2001. LNCS, vol. 2333, pp. 235–247. Primavera (2002)