SNMG: a social-level norm-based methodology for macro-governing service collaboration processes

Ji Gao1,2,*, Hexin Lv1, Zhiyong Jin1 and Ping Xu1
1Zhejiang Shuren University, Hangzhou 310015, China
2Zhejiang University, Hangzhou 310027, China
*Corresponding author, Email: gaoji1@zju.edu.cn, +86-571-8799-6934

Abstract. In order to adapt to the accelerative open tendency of collaborations between enterprises, this paper proposes a Social-level Norm-based methodology for Macro-Governing service collaboration processes, called SNMG, to regulate and control the social-level visible macro-behaviors of the social individuals participating in collaborations. SNMG not only can remove effectively the uncontrollability hindrance confronted by open social activities, but also enables across-management-domain collaborations to be implemented by uniting the centralized controls of social individuals for respective social activities. Therefore, this paper provides a brand-new system construction mode to promote the development and large-scale deployment of service collaborations.

1. Introduction
With progressing globalization and fiercely competitive market, the open tendency for collaborations between enterprises is accelerating. “Open” means that the enterprises participating in collaborations possess heterogeneous IT application systems which locate in independent management domains, and such across-management-domain collaborations often need to be created dynamically on demand. However, at present the application systems for open collaborations are still constructed according to the logically centralized control mode adopted by traditional software engineering. Moreover, open collaborations often involve the dynamical creation and change of social relationships between enterprises, and thereby belong to typical social application. However, this mode lacks the use of social-level abstract concepts, and therefore enables application systems only to be built by “low-level control flow abstraction” [1, 2]. As a result, this makes the reusability of system parts restricted seriously, and leads to unnecessarily complex software models and code, which in turn lead to expensive and painful software maintenance requirements.

In order to remove the predicament above, [3] proposed that the construction of social application should be grounded on basic social-level abstract concepts, and requires the social application languages and platforms for representing, computing, and reasoning those concepts. Since Social Dependences, called SDs, have been identified as basic social relationship, and popular social relationships, such as partner, friend, co-author, and so on, are only application-specific and are created on the top of SDs, SDs can be adopted as basic social-level abstract concepts. An SD means that some social individual (e.g., an enterprise participating in collaboration) depends on the assistance of another one to achieve an expectant goal, and the assistance is a social activity executed by the latter. For example, manufacturers depend on suppliers for parts, sellers depend on the bidders winning an auction for making the payment, and the invitees depend on the party sponsor for hosting the party. However, [3] have not specified how to use SDs to support the construction of those systems, and neither there is the subsequent achievements of this mode to be reported.
We propose that the key for innovation is to create the across-management-domain social-level macro-governance mode to replace the traditional centralized control mode limited in single management domains. Here, the macro-governance aims at regulating and controlling the social-level visible macro-behaviors of social individuals, which reflect how social sates are changed by social activities of those individuals, while the social-level norms for constraining those macro-behaviors are used as the governance basis. Because SDs arise in every social activity, formulating SD norms to constrain individual macro-behaviors enables macro-governance to be realized by the computation and reasoning of individuals for how to conform to those norms. Therefore, across-management-domain collaborations can be implemented by uniting the centralized controls of social individuals for respective social activities, and this enables the requirement for creating statically a physical or logical control center to be removed. Meanwhile, due to no need for concerning with the implementation detail of social activities, the macro-governance mode can reduce largely the difficulty for developing and maintaining open collaborations, and make the attention for developing open collaborations focused on designing architectures and solving challenging problems originated from openess.

This paper focuses the research of social-level macro-governance on open collaborations, and proposes a Social-level Norm-based methodology for Macro-Governing across-management-domain service collaboration processes, called SNMG, to support the open collaborations formed dynamically on demand. Here, “service collaboration” means that collaborations can be realized by creating the supply-demand relationship of business (i.e., software) services. Since SOC (Service-Oriented Computing) has grown into maturity, and the virtualization of services can remove the difficulty and complexity of the interaction between heterogeneous IT application systems, services have become the building blocks of collaborative business processes.

The layout of this paper is as follows. We first discuss the challenges and countermeasure for researching the macro-governance mode in section 2, and then the methodology SNMG is specified in section 3. In section 4 we introduce the implementation of SNMG. After discussing the relative work and comparison in section 5, the conclusions (in section 6) are provided.

2. Challenge and Countermeasure

Open collaborations are confronted with the uncontrollability hindrance to social activities because the openness makes social individuals not controlled directly by outer management domains. The social-level macro-governance mode we proposes above can just be used as the general way for removing this hindrance: as long as the macro-behaviors of individuals for executing social activities conform to relevant norms, those activities are transformed into predicable and controllable ones. However, how to make individuals conform to norms is also the challenging problem which must be solved. For this we propose that social individuals should be constructed into rational ones which have both ability and desire for conforming to norms in order to ensure that their social activities can comply with the macro-behavior constraints prescribed by relevant norms. Of course, in order to restrain possible norm-violating behaviors, social individuals should register into a community so that the supervision system located in the community can urge those individuals to conform to norms and sanction the ones violating norms.

Due to the requirement for computing and reasoning how to conform to norms, the rational individuals participating in collaborations should have high autonomy and intelligence. Thus it is advisable adopting software agents as the brokers for enterprises to provide / consume business services through Internet.

3. Social-level Norm-based Macro-Governance Methodology SNMG

In order to achieve the macro-governance for a cross-management-domain service collaboration processes, SNMG consists of two levels of both standalone and complementary technologies: the social-level norm system for collaboration processes and the rational agents whose collaboration behaviors conform to norms. Although the implementation of SNMG also requires the assistance of the supervision system located in the agent community, the relevant introduction is omitted due to the limited space.
3.1. Social-Level Collaboration Process Norm System

The core for achieving social-level macro-governance is to create the social-level norm system for collaboration processes published in the agent community. An intuitive way may be to formulate, for each service collaboration instance, the SD norms which are used to constrain the macro-behaviors of the two parties participating in the instance and can be adopted by them. Since the process for implementing a collaboration instance often includes multiple social activities executed in succession, the norms for constraining those activities form the norm set called collaboration process pattern. However, the instance needs to be created dynamically on demand, and this makes creating the pattern from scratch by negotiation face the predicament that computation complexity and space-time consumption are too large. Therefore, we propose that the more convenient and effective approach should be: formulating a parameterized collaboration process template first, then instantiating dynamically this template, by negotiating and signing, into the shared pattern adopted by the two parties.

Define the social-level norm system \( L \) for collaboration processes as the set of norm systems for application domains: \( L = (L_1, L_2, \ldots, L_w) \), and \( L_k \) (\( k=1, 2, \ldots, w \)) is the set of collaboration process templates: \( L_k = \{M^{S_1}, M^{S_2}, \ldots, M^{S_n}\} \). Every template \( M^{S_i} \) (\( i=1, 2, \ldots, n \)) includes a group of SDNs (SD Norms) corresponding to a specific business service \( S_i \): \( M^{S_i} = \{SDN_{A_1}^{S_i}, SDN_{A_2}^{S_i}, \ldots, SDN_{A_m}^{S_i}\} \). Wherein, \( SDN_{A_j}^{S_i} \) is used to constrain social activity \( A_j \) (\( j=1, 2, \ldots, m \)) and \( A_j \) may be an activity regular or for transacting abnormity. As an example, for “purchase” service (e.g. purchase a car), the collaboration process includes not only the regular activities executed alternately by the provider and consumer of this service: pay deposit, notice the pick-up of goods, pay for goods, provide goods, etc., but also the activities for transacting abnormity: complement deposit, defer the provision of goods, amerce, withdraw collaboration relationship, etc.

Define \( SDN_{A_j}^{S_i} = PSD_e(\rho \leq \delta | \sigma) \), which is used to represent the Parameterized SD (PSD), for social activity \( A_j \), expected to maintain in the process performing bilateral (providing-consuming) collaboration oriented to service \( S_i \), and is endowed with the following meanings: under the condition that the current collaboration state satisfies \( \sigma \) and before the deadline \( \delta \) is reached, party \( o \) depends on \( A_j' \) executer \( e \) (the other party) to transform this state into satisfying \( \rho \).

Here, \( \sigma \) and \( \rho \) are also called the precondition for activating a norm and the postcondition for executing the norm respectively, and thereby make the execution of the norm possess the explicit start-up occasion and success-determining standard. An SDN defined in the description language corresponding to PSD_e(\( \rho \leq \delta | \sigma \)) is shown in Figure 1.

```
(eio:SDNorm //The norm for social activity “PayDeposit”
   NormPurpose: “Backbone” //This is the norm for driving a regular social activity
   NormNo: 1
   Description: “Specify the performer, \( \sigma \), \( \delta \), and \( \rho \) of this norm”
   NormTransactionType: “dmo:DMNormTransaction”
   //The name of enumeration set of regular social activities
   NormTransaction: “dmo:DepositProviding” //The name of this activity in the set
   Performer: “InitiatingRole” //This norm is performed by the service consumer
   Trigger: (@eio:ContractInuring ContractRegisterNo: ?x Service: DataArrangement
     InuringDate: ?y) //?x and ?y are variables with “?” as their prefixes
   Deadline: (@eio:dateTimePeriod Type: RD BeginDate: ?y Period: ?neg_11)
   Postcondition: (@dmo:DepositStatus ContractRegisterNo: ?x DepositLowLimit: ?neg_12 Currency: “¥” DepositSatisfactionStatus: “Satisfying”)
)
```

Figure 1. The social dependence norm for social activity “PayDeposit” which can be activated by an event “ContractInuring”, and should make the activity- ended time satisfy the deadline and the activity post-effect the postcondition.

Since SDNs only constrain the social-level visible Macro-behaviors of broker agents for executing social activities: activity-activated occasion, activity-ended time, and activity post-effect, this enables
template $M^{Si}$ to be formulated in the highly abstract form in which agent mental models and internal micro-operators are invisible. Such a platform-independent description mode not only makes $M^{Si}$ adopted and conformed to easily by heterogeneous agents, but also enables $M^{Si}$ to be created on the basis of deep analysis and synthetic investigation. After created, $M^{Si}$ is published in the agent community, becoming the negotiation foundation for creating collaboration relationships. Once the negotiation (which belongs to the scope of self-organization, and is not discussed here) successes, $M^{Si}$ is instantiated into collaboration process pattern $M^{Y}$, which is used to govern the current service collaboration instance $Y$, and the SDNs included in $M^{Y}$ is also specialized into appropriate to $Y$.

3.2. Rational Agents
Rational agents are configured into the individuals which have both ability and desire to make their social activities in collaboration processes accept the constraints from $M^{Y}$, and therefore the outer observers (i.e., people or other agents) can believe those individuals reasonable and trustable. The key for achieving rationality is to install broker agents the local management policies for driving social activities in order to enable those agents to know how to execute those activities by using local micro-operators and decide how to conform to the constraints from $M^{Y}$, including the norm-violating in an unavoidable situation. Since the SDNs included in $M^{Y}$ have already prescribed accurately the constraints agent social-level macro-behaviors should conform to, those constraints, as the basis for agents to compute and reason how to conform to norms, can be incorporated into the social activities driven by policies. Therefore, we propose creating the policy-driven Collaboration Process Management mechanism, called CPM, as the core mechanism for the rational agent running platform.

For collaboration instance $Y$ oriented to business service $Si$, CPM is defined as the following multiplet:

$$CPM = (M^{Si}, Policy, T-Event, C-State, M-State, B-Instruction, M-Operator, triggering, activating)$$

- $M^{Si}$: the collaboration process pattern for constraining $Y$.
- Policy: the set of policies for managing the agent collaboration process oriented to $Si$. Those policies are divided into 4 types: $P_{es}$ (policies for examining collaboration state transition and activating norms), $P_{ec}$ (ones for resolving the conflicts of multi-norm activation), $P_{po}$ (ones for planning the micro-operators for executing social activities), and $P_{op}$ (ones for executing operator plans).
- $T-Event$: the set of events for triggering management policies. Those events are also divided into 4 kinds: $E_{at}$ (events indicating collaboration state transformation), $E_{an}$ (ones indicating the conflicts of multi-norm activation), $E_{ms}$ (ones indicating the single norm selected), $E_{pg}$ (ones indicating the operator plan generated).
- $C-State$: the set of collaboration states occurring possibly in the collaboration process oriented to $Y$.
- $M-State$: the set of agent local micro-states occurring possibly in the collaboration process oriented to $Y$.
- B-Instruction: the set of local business instructions and transaction principles, oriented to $Si$, sent out by the owner of the agent statically, dynamically, or in dynamical initiation, such as the way for resolving the conflicts of multi-norm activation, the tolerance degree for QoS deviation, the tolerance degree for norm execution delay, and the constraints for replacing collaboration partners, etc.
- $M-Operator$: the set of agent local micro-operators, oriented to $Si$, for executing collaboration activities.
- triggering: $T-Event \rightarrow Policy$, indicating that each $e \in T-Event$ triggers a relevant policy.
- activating: $C-State \rightarrow \mathcal{P}M^{Y} \times (E_{at}\cup E_{an})$, which means: once a collaboration process starts or a previous social activity finishes, the current collaboration state activates one to multiple norms contained in $M^{Y}$, by making the match examination with $\sigma$ (the precondition for activating a norm), and an event $e \in (E_{at}\cup E_{an}) \subset T-Event$ is generated (note, the kind of $e$ is decided according to the number of activated norms, and symbol $\mathcal{P}$ denotes power set).

A management policy $p\in\mathcal{P}$ is represented in BNF as follows:

$$<p> ::= \langle trigger\rangle \langle processing\rangle$$
$$<Processing> ::= \{ <m-operator> | <rule> | <rule-group> \}^*$$
$$<rule> ::= (\rightarrow <condition> \{ <m-operator> \})^*$$
<rule-group> ::= <selection-mode-for-activated-rules> { < rule > }*

The main parts of a policy are its trigger and the transaction driven by it. The former is set as a trigger pattern, enabling the policy to be triggered by an event \( e \in T\text{-}Event \). The latter is represented as a sequence of micro-operators, condition-action rules (conditional operators), and rule groups (for selecting activated rules), enabling an agent to make the decision analysis by detecting the activation of rules in order to drive appropriate local m-operators \( \subset M\text{-}Operator \) for executing a social activity. Here, the decision analysis includes to examine the current collaboration state \( cs \in C\text{-}State \) and / or local micro-state \( ms \in M\text{-}State \), to inspect the relevant business instructions and transaction principles \( bis \in B\text{-}Instruction \) from the agent owner, and to compute and reason how to conform to the current activated SD norm according to the constraints prescribed by this norm.

The policy-driven management for agent collaboration processes is illustrated in figure 2. The backbone of process management is divided into three stages: generating policy-triggered events, triggering the relevant management policies, driving appropriate local micro-operators. Note, the autonomic management for collaboration behaviors can be applied to drive both regular and abnormality transaction activities.

![Figure 2. The agent collaboration process management Driven by policies (which is divided into three stages)](image)

4. Implementation of SNMG

The implementation of methodology SNMG depends on the development of two-level technologies: social-level collaboration process norm system and rational agents. Because the key for creating the norm system is the formulation of SD norms, we have designed the norm description language corresponding to SDN\( A^\text{S}\subseteq\text{PSD}^\text{S}(\rho \leq \delta | \sigma) \) and defines SD norms as the structured description called SDNorm, including a group of attribute slots, and an example for SD norms has been shown in Figure 1. The rational agents can be developed with the agent modelling and running platform, which are one of authors’ previous research results: trusted autonomic service cooperation model IGTASC and application development framework [4].

We have established multiple simulation test instances oriented to service collaborations, such as small meeting arrangement, knowledge provision, data mining, and obtaining-employment management for graduates, to examine the feasibility and performance of SNMG. The test results have illustrated: formulating the collaboration process templates as the norms for constraining social-level macro-behaviors is the effective means for achieving autonomic coordination and maintenance of service collaboration processes.

5. Related Work and Comparison

Although [3] has proposed that the construction of social application should be grounded on basic social-level abstract concepts: SDs (social dependences), no feasible scheme and key implementation technology is specified. In contrast, we have proposed that the key for innovating the traditional software engineering is to create the across-management-domain social-level macro-governance to replace the logically centralized control limited in single management domains. In order to achieving the macro-governance, we have already created the norms for constraining the macro-behaviors of social individuals in the social level, and provided the support environment (rational agents plus the agent community supervision system) for promoting those individuals to comply with the norms. Besides, we have removed the limitation that the traditional theory for representing SDs doesn’t support the representation for state change and dependence evolution [5, 6]. By using parameterized representation
structures $PSD_{\alpha}(\rho \leq \delta | \sigma)$ to accurately defines $SDN_{A}^{S}$, agents are provided with the basis for computing and reasoning how to conform to norms.

Although NMASs (Normative Multi-Agent Systems) permits the sociology to be applied to MASs[7,8], the norms for constraining agent macro-behaviors visible in the social level are represented in the form of deontic logic [9], and thereby cannot represent the social relationships and their evolution intuitively. Besides, current NMASs are disjointed from conventional application software, making norms difficult to display their value for supporting real collaboration application. In contrast, the macro-governance methodology SNMG has not only established the solid foundation for developing high-performance social application of service collaborations, but also overcome the drawback of disjointing by incorporating SOC (Service-Oriented Computing) and policy-driven management technologies.

6. Conclusions
This paper has proposed the innovatory system construction mode for open collaborations: across-management-domain social-level macro-governance, and made macro-governance created on the foundation of basic social-level abstract concepts: SDs (social dependences). For implementing this mode, we have created the methodology for macro-governing collaboration processes with SNMG to remove effectively the uncontrollability hindrance confronted with by open social activities and enable across-management-domain collaboration processes to be implemented by uniting the centralized controls of social individuals for respective social activities, forming a distributed control mode which is created dynamically and accepts the macro-governance. Finally, developing real application and solving the real-life problems will be our further research work.

7. Acknowledgment
We gratefully acknowledge the support of the National Science Foundation of China under Grant No. 61375071, the Zhejiang Province Science Foundation of China under Grant No. LZ15F020001, and the Zhejiang Province Science Foundation of China under Grant No. LY14F020008.

8. References
[1] Garlan D, Cheng S W, Huang A C, Schmerl B, and P. Steenkiste P, 2004, Rainbow: Architecture-based self-adaptation with reusable infrastructure, IEEE Computer, 37 (10), pp. 46–54.
[2] Zinnikus I, Cao X, Fischer K, 2013, Agent-supported collaboration and interoperability for networked enterprises: Modeling interactions and service compositions, Computers in Industry, 64 (2013), pp. 938–950.
[3] Chopra A, 2011, social computing: Principles, platforms, and applications, RESC 2011, pp. 26-29.
[4] Gao J and Ly H, 2012, Institution-governed cross-domain agent service cooperation: a model for trusted and autonomic service cooperation, Applied Intelligence, 37(2), pp. 223-238.
[5] Lau B, Singh A, Tan T, 2015, a review on dependence graph in social reasoning mechanism, Artif Intell Rev, 43(2), pp. 229-242.
[6] Caire P, van der Torre L, and Villata S, 2013, Argumentation Theoretic Foundations for Abstract Dependence Networks, Agreement Technologies, Lecture Notes in Computer Science, Volume 8068, pp. 180-194.
[7] Criado N, Argente E, and Botti V, 2011, Open issues for normative multi-agent systems, AI COMMUNICATIONS, 24(3), pp. 233-264.
[8] Chopra A and Singh M, 2016, Custard: Computing Norm States over Information Stores, Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (AAMAS ‘16), and 1096-1105.
[9] Sergot M, 2010, Norms, Action and Agency in Multi-agent Systems, Deontic Logic in Computer, Lecture Notes in Artificial Intelligence, Volume 6181.