# Social Threats and the New Challenges for Requirements Engineering

Fabiano Dalpiaz
Department of Information Engineering and Computer Science
Università degli Studi di Trento, Italy
dalpiaz@disi.unitn.it

***Abstract*—We understand social computing as a paradigm based on abstractions that describe how social actors—humans and organizations—interact to conduct their business. These abstractions are part of a social layer that lies on top of the technical layers that constitute software systems. This social layer is founded upon social concepts such as agent, role, commitment, delegation, trust, and reputation. Social computing is affected by many of the threats we successfully handle in our lives: business transactions with unknown parties, volatile environments, existence and enforcement of norms that regulate the marketplace, malicious actors, etc. These threats pose new challenges for Requirements Engineering (RE): the specification of artefacts able to deal with these threats. In this vision paper (i) we argue for the social nature of the threats; and (ii) we outline some of these new challenges for RE.**

***Index Terms*—requirements engineering; social computing; social threats.**

## I. Introduction

Layering simplifies the engineering of complex computing artefacts by hiding implementation details to other layers. Current computing artefacts (e.g. network stacks and software architectures) rely on purely technical layers. Our position is that the topmost layer of computing artefacts should be a *social layer*.

Such social layer does not refer to technological concepts (node, port, channel); rather, it consists of social concepts, such as actors and social relationships among them. This layer represents the business meaning (the rationale) behind the use of computing artefacts. Suppose you are using the eBay website to create an auction for an old book. This usage scenario creates a social relationship: your commitment [1] to deliver the book to the auction's winner, provided that you receive the payment beforehand.

We understand *social computing* as a paradigm founded upon the social layer. We conceive social computing in terms of social primitives, such as roles, agents[1], and commitments (like Chopra and Singh [2]), as well as trust, reputation, and identity. In social computing, for instance, a buyer agent may want to opt for a seller's commitment for a book only if he trusts the seller, perhaps because of his good reputation, and has proofs of the authenticity of the seller's website.

The term social computing has been previously discussed [3] [4] in a broader sense, referring to computing artefacts that facilitate collaboration between humans and their interaction. We go a step further, as we argue for treating the social layer as a first-class citizen in computing.

Service orientation—in its more general sense—provides an adequate foundation for the social layer. Service-oriented applications [5] consist of autonomous and heterogeneous agents that socially interact in order to achieve their own goals. Such interaction defines and evolves a network of commitments among these agents. Agents can be either service consumers, providers, or both. Services are captured by the commitments agents make (publishing a service) and discharge (service delivery).

In addition to the technical threats that exploit system vulnerabilities (e.g. denial-of-service, back-doors, cross-site scripting), social computing brings social threats that affect social actors, relationships, and artefacts. These threats derive from those we experience in our lives while interacting with others: lack of trust between actors, decreasing reputation, fake opinions about providers, new laws that regulate the marketplace. We investigate social threats in Section II.

Social threats pose new challenges for Software Engineering: new artefacts have to be designed to successfully cope with these threats. All the phases of software development are affected, since the early stages, i.e. Requirements Engineering (RE). Researchers in RE are already investigating these aspects, especially via goal-oriented approaches. Our purpose is to revisit these challenges in the spirit of social computing. In Section III, we outline some essential artefacts to effectively cope with social threats.

## II. The new wave of social threats

Social Computing carries along social threats—affecting the social layer—in addition to the technical threats that affect lower layers. While technical threats exploit vulnerabilities in the system—often due to software errors [6]—these new threats either concern social relationships and social artefacts (e.g. reputation, trust, delegation, laws), or are enacted via social mechanisms (e.g. assuming fake identities, user interaction with interfaces).

We adopt a service-oriented perspective to illustrate the following examples of social threats. We identified these threats by analysing the usage scenarios of the EU-funded project Aniketos (on secure and trustworthy service compositions); we are currently extending and refining such list.

[1] In this paper, we use the terms "actor" and "agent" interchangeably

978-1-4577-0950-0/11/$26.00 

T1. **Fake reporting**: reputation is one of the pillars of social interaction, as it is a main factor to determine whether an (unknown) actor is trustworthy. The eBay reputation computation mechanism is perhaps the most considered factor for customers to evaluate sellers trustworthiness. Being rooted in social interaction, social computing cannot help taking reputation into account. A related threat is fake reporting, i.e. actors that intentionally release fake ratings about service providers and consumers, possibly acting under false identity.

T2. **Decreasing reputation**: reputation is volatile, as it is updated whenever an actor submits a new rating. We often change partnerships based on recent experience, perhaps because our partners result less reliable than it they used to be. Decreasing reputation implies deciding whether to ignore the threat and stay with the current partner or switch to a different partner. Such threat is particularly severe in the case of composite services, as the possible failure of a service might result in a cascade failure that affects all services in the composition.

T3. **Lack of trust**: social computing is very different from paradigms based on method-invocation (object-orientation, software components, web-services). Each actor is autonomous and decides upon whom to interact with. An agent offering a service to the general audience is free to decide not to interact with consumers that he does not trust. In such setting, method invocation does not work. This is especially true for composite services, which cannot be assembled the same way one would orchestrate a set of software components.

T4. **Untrusted delegation**: it is often the case that you trust a specific actor to carry out a task, but you would not trust other actors, even though the trusted actor trusts them. As observed earlier, e.g. in [7], trust is not always transitive in real life. A service provider that delegates (part of) service provision to another provider—that the consumer does not trust—is a threat in social computing, as it violates the trust relation between the consumer and the original provider.

T5. **Dissolved redundancy**: redundancy is often devised to ensure availability. In component-oriented systems, redundancy means deploying at least two components providing the same functionality. In social computing, redundancy means relying on multiple providers for the same service type. However, each provider might further delegate the service, or part of it, to third parties. Redundancy might vanish if all redundant providers delegate the service to the same third party.

T6. **Incompatible laws**: laws are social artefacts that regulate relations in the society. You would not park your car in a no parking zone, since police might issue you with a parking ticket. Likewise, software systems should comply with laws. For example, the public disclosure of political views might be forbidden by law in some countries. Running services in these countries would infringe such law. Incompatible laws are a threat both at deployment-time (is the service law-compliant?) and at runtime (law changes over time).

T7. **Unauthorized data disclosure**: data confidentiality is a hot topic in computer security. Many mechanisms have been devised to preserve it, such as (role-based) access control, data anonymisation, etc. Social computing is an open and logically distributed setting. Due to openness, actors might adopt a fake identity (perhaps someone else's) to access data they have no authorization for. Logical distribution implies that each actor specifies its own access control policy; if these policies are not consistent, users can gain unauthorized access to data.

T8. **User interaction**: an orthogonal type of threat affects users interaction with the system. If not adequately designed, users might unintentionally open the way to threats. For example, having too many confirmation dialogues stimulates users to accept without knowing what they are actually committing to (perhaps a credit card payment or personal data disclosure).

## III. New challenges for RE

The social threats described in the Section II introduce new challenges for RE. In order to address these threats, new artefacts have to be engineered. Though some approaches have been already proposed to such extent, RE has to evolve to enable a systematic and effective specification of these artefacts. We list some challenges in the following sub-sections, then show how they relate to the social threats.

### *A. Trustworthiness management systems*

Social computing settings are open environments: the participating agents and their services are unknown at design-time, and agents join and leave as they wish at runtime. Due to such volatility, traditional requirements engineering methodologies are inadequate, as they make too many assumptions about the goals/requirements of participants. To overcome this limitation, RE has to support specifying systems that enable interaction among unknown agents.

To this extent, trustworthiness management systems will be essential. They have to consider a wide set of factors, such as opinions by peers, information about compliance, and certificates issued by trusted third parties. The mechanisms to monitor and compute trustworthiness shall be robust to bootstrapping (when little data is available) and malicious actors (who aim to corrupt trustworthiness metrics).

### *B. Service interface specification and monitoring*

In service orientation, service implementations are hidden to the consumers. Service interfaces are the unique artefact to express the functional and non-functional properties of a service. In terms of the social layer, a service interface implies a commitment by the service provider to deliver the service as specified. Current languages and monitoring frameworks do not support complex specifications of service interfaces (apart from Quality-of-Service properties).

In social computing, rich service interfaces are necessary to enable agents make a well-informed choice about partners. Interfaces can include (i) flexible parameters to enable negotiation with consumers; (ii) fine-grained access control policies that specify which information can be shared and with whom, as well as to which agents the service is offered; (iii) redundant provision of a service; (iv) whether the provider retains accountability in case of delegation; and (v) compensation rules that define the consequences (e.g. additional commitments) for a provider who violates its commitments. All these factors shall be monitored to identify violations and to enable compensation enforcement.

### C. Early warning and response mechanisms

Software adaptation in response to failures is a very active research area, which becomes crucial when considering the social layer, as failures directly affect the business of the agents. However, current mechanisms to respond to threats are mainly reactive: they are enacted after a problem is detected. This is a sub-optimal solution since failure effects are just mitigated, rather than being avoided.

Early warning and response mechanisms will be fundamental to to prevent bad events from occurring. The key is detecting threats before they result in a failure and pro-actively switching to an alternative configuration—involving different commitments—to avoid the failure. To specify systems including these mechanisms, risk assessment techniques could be exploited to relate risks to requirements and enable evidence-based identification of threats.

### D. Deployment-time certification

Agents deploy their services by manifesting service interfaces, and they commit to comply with such interface while delivering the service. When the service is deployed by an agent who recently joined the environment, other agents have very little information about the agent's trustworthiness. This might discourage other agents from interacting, especially if the mission of an agent is critical.

Certificates—issued by trusted third parties—can be released as an additional guarantee. Certification authorities commit to the trustworthiness of specific services (and their providers). Certification might involve checking the way the service is implemented, adding monitoring probes, ensuring that a specific service development methodology is followed. The certification authority becomes accountable in case the service is not delivered as certified. In turn, the authority will most likely make up for the non-compliant agent.

### E. Adaptation mechanisms

Current mechanisms for software adaptation adopt a centralized perspective, in which software consists of cooperating components (web-services are physically distributed components) that can be re-routed as needed. Such perspective is incompatible with the logical decentralization of social computing, wherein each agent is a locus of autonomy.

New adaptation mechanisms are needed to better take into account the characteristics of social computing settings:

- Adaptation should be conceived from the perspective of a single agent [8], and consists of changing the internal strategy as well as the commitments made to and taken from other agents (while preserving agents autonomy);
- Social threats are triggers for agent adaptation;
- In a volatile environment, it is hard to determine in a single step the best configuration—including which agents to interact with—to achieve an agent's requirements. An alternative strategy is to rely on incremental techniques (i.e. via partial planning): a partial configuration is generated and is then refined incrementally.

### F. Law representation and compliance

A key challenge in social computing is the representation of laws in a machine-understandable form. This will enable determining compliance and enforcing law. Despite of some promising efforts [9] [10], existing languages are still inadequate to capture laws in a comprehensive way.

One challenge will be devising languages that support temporal constraints, nested clauses, cross-references, compensations and penalties. Another challenge is to specify systems that enable enforcing compliance with laws. Law representation, monitoring, and enforcement are useful, for example, to represent data confidentiality restrictions that apply in certain countries, as well as to prevent unauthorized disclosure of data (via monitoring and enforcement functions).

### G. Identity management systems

In social computing, each agent is characterized by an identity, the same way each of us possesses an identity in the physical world. This is a radical shift from current computing paradigms, in which software does not always possess a legal identity, and is often released without guarantees.

The challenge will be the development of robust identity management systems that unequivocally bind a software system to its legal entity. Such entity is accountable for that software system. Identity management systems have to prevent malicious users from assuming fake identities, as well as using someone else's identity to spoil his reputation.

### H. Development methodologies

An orthogonal challenge to the previous ones is development methodologies to alleviate or prevent threats. Such methodologies should start from the early requirements phase, so to determine how the threats affect system requirements. Important aspects to tackle are (i) avoiding confidentiality violations, by taking into account the authorizations provided by data owners; (ii) designing adequate user interfaces so that users are aware of the social relationships they manipulate by interacting with the interface; (iii) ensuring that the specification of the system is law-compliant.

Table I shows how these challenges relate to the social threats of Section II. *Fake reporting* (T1) is addressed by building robust management systems for trustworthiness and identity, so to mitigate the effect and reduce the likelihood of fake reports, respectively. *Decreasing reputation* (T2) is

| Challenge for RE | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |
|---|---|---|---|---|---|---|---|---|
| Trustworthiness management systems | ✓ | ✓ | | | | | | |
| Service interface specification and monitoring | | | | ✓ | ✓ | | | |
| Early warning and response mechanisms | | ✓ | | | | | | |
| Deployment-time certification | | | ✓ | | | | | |
| Adaptation mechanisms | | ✓ | | | ✓ | | | |
| Law representation and compliance | | | | | | ✓ | | ✓ |
| Identity management systems | ✓ | | ✓ | | | | | |
| Development methodologies | | | | | | ✓ | ✓ | ✓ |

TABLE I
MAPPING BETWEEN SOCIAL THREATS AND CHALLENGES FOR RE

handled if trustworthiness metrics include such factor, and by developing systems able to early detect and respond to the threat. An effective method to avoid *lack of trust* (T3) is certification mechanisms, complemented by identity management to prevent agents from assuming fake identities. *Untrusted delegation* (T4) necessitates adequate service interfaces and monitoring functionalities to keep track of delegations. To prevent *dissolved redundancy* (T5), service interfaces shall allow specifying such property; also, adaptation patterns might introduce additional providers before redundancy vanishes. Frameworks to represent, check, and enforce law, as well as companion development methodologies are useful to deal with *incompatible laws* (T6) and *unauthorized data disclosure* (T7). *User interaction* threats require specific development methodologies (T8).

## IV. CONCLUSIONS

We presented our perspective on social computing, described a set of new threats such paradigm introduces, and outlined some challenges Requirements Engineering will have to address to successfully handle these threats.

We argue that social computing will be centred around a *social layer*, including social actors, relationships, and artefacts. Unlike other computing paradigms, social computing is about how social actors interact to conduct their business, and treats software as a means to support their interaction.

The threats we introduced are social by nature, and derive from the threats we encounter in our lives. Consequently, RE is likely to address the new challenges by adapting pragmatic solutions that demonstrate to be effective in the real world.

## ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos).

## REFERENCES

[1] M. P. Singh, "Agent Communication Languages: Rethinking the Principles," *IEEE Computer*, vol. 31, no. 12, pp. 40–47, Dec. 1998.

[2] A. K. Chopra and M. P. Singh, "Elements of a Business-Level Architecture for Multiagent Systems," in *Proceedings of the 7th International Workshop on Programming Multi-Agent Systems (ProMAS 09)*, ser. LNCS, vol. 5919. Springer, 2009, pp. 15–30.

[3] D. Schuler, "Social computing," *Communications of the ACM*, vol. 37, no. 1, p. 28, 1994.

[4] F.-Y. Wang, K. M. Carley, D. Zeng, and W. Mao, "Social Computing: From Social Informatics to Social Intelligence," *IEEE Intelligent Systems*, vol. 22, no. 2, pp. 79–83, 2007.

[5] A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments," in *Proceedings of 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10)*, ser. LNCS, vol. 6051. Springer, 2010, pp. 113–128.

[6] K. Tsipenyuk, B. Chess, and G. McGraw, "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors," *Security & Privacy, IEEE*, vol. 3, no. 6, pp. 81–84, 2005.

[7] B. Christianson and W. S. Harbison, "Why Isn't Trust Transitive?" in *Proceedings of the International Workshop on Security Protocols*. Springer-Verlag, 1996, pp. 171–176.

[8] F. Dalpiaz, A. K. Chopra, P. Giorgini, and J. Mylopoulos, "Adaptation in Open Systems: Giving Interaction its Rightful Place," in *Proceedings of the 29th International Conference on Conceptual Modeling (ER 2010)*, ser. LNCS, vol. 6412. Springer, 2010, pp. 31–45.

[9] E. Marengo, M. Baldoni, C. Baroglio, A. K. Chopra, V. Patti, and M. P. Singh, "Commitments with Regulations: Reasoning about Safety and Control in REGULA," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, 2011, pp. 467–474.

[10] A. Siena, J. Mylopoulos, A. Perini, and A. Susi, "Designing Law-Compliant Software Requirements," in *Proceedings of the 28th International Conference on Conceptual Modeling (ER 2009)*, ser. LNCS, vol. 5829. Springer, 2009, pp. 472–486.