# Meta-design to Face Co-evolution and Communication Gaps Between Users and Designers

Maria Francesca Costabile[1], Daniela Fogli[2], Rosa Lanzilotti[1], Andrea Marcante[3], Piero Mussio[3], Loredana Parasiliti Provenza[3], and Antonio Piccinno[1]

[1] Università di Bari, Dip. Informatica, via Orabona 4, 70125 Bari, Italy
{costabile,piccinno,lanzilotti}@di.uniba.it
[2] Università di Brescia, DEA, via Branze 38, 25123 Brescia, Italy
fogli@ing.unibs.it
[3] Università di Milano, DICO, via Comelico 39/41, 20135 Milano, Italy
{marcante,mussio,parasiliti}@dico.unimi.it

**Abstract.** The paper presents a meta-design approach to the design and development of interactive systems, which bridges the communication gaps arising among the members of an interdisciplinary design team including different experts: software engineers, human-computer interaction experts, end users as domain experts. Each experts is a stakeholder that proposes design solutions from her/his perspective. The approach, which relies on a novel model of Interaction and Co-Evolution processes, also supports co-evolution of users and systems.

**Keywords:** Meta-design, Communication gap, Co-evolution, End-user development.

## 1 Introduction

In the last years we have been working to the design of visual interactive systems supporting collaborative human activities in different domains, such as medicine [6], [7], geology [15], mechanical engineering [9]. All these situations confirm Fischer's claim that "complex design problems require more knowledge than any one single person can possess, and the knowledge relevant to a problem is often distributed and controversial" [13]. Indeed, the design of a visual interactive system supporting the achievement of some activities in a domain of interest involves several stakeholders, each one bringing her/his experience and view of the problems at hand. End users, as the "owners of problems" [12], have their own knowledge and a domain-oriented view of the tasks they have to perform (with the support of a computer) coming from the practice in their working environment. Software engineers have the knowledge about computer science methods and tools; they bring into software design and development their own view of the activity to be supported, focusing on implementation and efficiency aspects. Human-Computer Interaction (HCI) experts give their contribution by guaranteeing system usability and accessibility.

In the collaboration among such stakeholders communication gaps arise because of their different cultures; users, software engineers and HCI experts adopt different approaches to abstraction and follow different reasoning strategies to modelling, performing and documenting the tasks to be carried out in a given application domain; additionally, they express and describe such tasks in their own language and jargon.

In this paper we describe a meta-design approach to overcome the communication gaps so that visual interactive systems are capable to support people activities by allowing users to act as designers, collaborating with HCI experts and software engineers. The approach also permits to cope with another important phenomenon, that is the co-evolution of users and systems [8]. After analysing communication gaps and the co-evolution process in the next three sections, Section 5 and 6 are about meta-design, first discussing how this concept is presented in literature, then illustrating our specific approach. Section 7 concludes the paper.

## 2 Competent Practitioners as Users

People facing real world problems act as *competent practitioners* in that "they exhibit a kind of knowing in practice, most of which is tacit" and they "reveal a capacity for reflection on their intuitive knowing in the midst of action and sometimes use this capacity to cope with the unique, uncertain, and conflicted situations of practice" [24]. Competent practitioners use their (tacit) knowledge to interpret documents that support their activity and to understand how to use their tools. In the case of VIS usage, a relevant part of the information conveyed by the system is '*implicit information'* [6], i.e., it is embedded in the actual shape of the displayed elements and in the visual organization of the overall screen image and can only be understood by users who possess domain (tacit) knowledge. For example, sequences of images illustrating sequences of actions to be performed are organized according to the reading habits of the expected user: from left to right for Western readers, from right to left for Eastern ones. Furthermore, some icons, textual words, or images may be meaningful only to experts in some discipline: icons representing cells in a liver simulation may have a specific meaning only for physicians, while a dept core would be meaningful only to geologists.

Moreover, complex activities must be carried out by experts characterized by different cultures, executing different tasks. For example, in the medical domain, neurologists cooperate with neuro-radiologists to interpret a Magnetic Resonance Image (MRI) and define a diagnosis; they are members of two different communities who must analyze and manage the same data set with different tools, on the basis of different knowledge they possess and from different points of view. However, in this activity, as in many others, members of different communities reach a common understanding and co-operate to achieve a common purpose [10].

In a team of competent practitioners collaborating to solve a problem, each practitioner is a stakeholder, owner of a specific knowledge that is crucial to the resolution of the problem, but not sufficient to solve it. This situation is defined as *symmetry of ignorance* [14], [23]; for overcoming it, the knowledge owned by every stakeholder must be shared and integrated with the knowledge of the other stakeholders. This holds for the design of interactive systems as well. The knowledge of software engineers, relative to design methods and technologies, must be integrated with knowledge about human factors, that HCI experts possess, and with knowledge about application domain, that domain experts (end users) possess.

## 3 Communication Gaps Among Stakeholders

The collaboration among different stakeholders in a design team presents problems, some due to *communication gaps* that exist among the team members, depending on their different cultural backgrounds, experience and view of the problems at hand.

When interacting with a VIS, end users, as competent practitioners, use their knowledge, both explicit and tacit, gained in their traditional and concrete work environments, to understand how to operate in the new virtual environment. In designing a VIS, software engineers bring into design their own tacit and explicit knowledge and their own views of the activity to be implemented, which are different from the knowledge and perspective of end users. Indeed, a gap exists between end users and software engineers: they adopt different approaches to abstraction, since, for instance, they have different notions about the details that can be abridged. Users reason heuristically rather than algorithmically, using examples and analogies rather than deductive abstract tools, documenting activities, prescriptions and results through their own developed notations, articulating their activities according to their traditional tools rather than computerized ones [18]. Moreover, end users and software engineers possess distinct types of knowledge and express such knowledge according to different languages and notations. As a consequence, end users do not understand software engineers jargon and, conversely, software engineers often do not understand user jargon [6]. On the other hand, HCI experts, often advocated to represent user views in the design, own a specific knowledge - on system usability and human factors, which is not the one of the users neither that of software engineers [17]. Communication gaps, thus, exist among HCI experts, end users and software engineers. HCI experts cannot take users' place and, vice versa, users cannot act for HCI experts: only users are able of reading the screen with user tacit knowledge, and understanding what is misleading or difficult to interpret for them, but they are not able to think as HCI experts and propose adequate HCI solutions [21]. Both end users and HCI experts cannot take software engineers place: they cannot evaluate the technical consequences of their proposals nor the influence on the adopted technologies, i.e., they are not able to think as software engineers [18], who know the technology but have difficulties in thinking as end users or HCI experts.

## 4 Co-evolution of Users and Systems

Many authors have pointed out an important phenomenon that must be considered in Human-Computer Interaction: the user evolution. Nielsen says [19] that "using the system changes the users, and as they change they will use the system in new ways". More recently, Norman says in [21] that "the individual is a moving target". This means that the design of an interactive system may be good today, but no longer appropriate tomorrow. Once people gain proficiency in system usage, they would like to use the system in different ways and need different interfaces than those they required when they were novice users. These new uses of the system make the working environment and organization evolve, and force the designers to adapt the system to meet the needs of the end-user organization and environment. Therefore, it is more appropriate to speak about *co-evolution of users and systems* [1], [4], [8]. In our experiments with end users, we found that several usability problems depended very much on the rigidity of the interactive systems, which are not able to take care of the changes occurring in users' activities and/or in their organizational context. Indeed, two processes must be considered. The first process - the interactive use of the system to perform activities in the application domain - occurs in a short time scale: every activity is the result of a sequence of interaction cycles in which the user applies her/his intuitive knowing and reflects on the obtained results, gaining new experience. The second process is the co-evolution of user and system, which occurs during the use of an interactive system in a longer time period. Co-evolution is also a cyclic process in which two cycles are identified. The first one is the task-artifact cycle, initially discussed in [5]: it refers to the fact that software artifacts are produced to support some user tasks. However, such artifacts suggest new possible tasks so that, to support these new tasks, new artifacts must be created. The second cycle refers to the fact that technology advances give computer scientists ways of improving interactive systems once they are already in use: this leads to new interaction possibilities that might change users working habits. For example, recently improved voice technology allows software engineers to add voice commands to their systems and this might provide an easier and more natural way of interaction. Also the user socio-organizational context is evolving during time, often requiring new ways of performing tasks. Therefore, technology and socio-organizational contexts repeatedly affect each other and this is modelled by a second co-evolution cycle in a model of *Interaction and Co-Evolution* process we have proposed [8]. Software engineers are required to produce the tools to support both interaction and co-evolution processes, i.e., they must not only produce interactive systems supporting user activities, but also the tools that permit to tailor [26] and evolve the system according to user and organization evolution.

## 5 Meta-design

Software engineers and HCI experts are aware of the gaps existing among them and of the need of communicating and sharing their different points of view during the VIS design process. Lauesen [17] proposes the *virtual window method*, an early graphical realization of the data presentation to bridge the gap between software engineers and HCI experts. Folmer et al. [16] propose *bridging patterns,* which describe a usability design solution and consist of a user interface part and an architecture/implementation part. Borchers [3] recognizes the necessity of capturing the knowledge of competent practitioners, together with HCI and software engineer expertise by forging a *lingua franca* that makes the design experience understandable by end users, HCI experts and software engineers.

However, the problem is how to embed the user implicit information in these languages and how to make the stakeholders express their tacit knowledge. In user-centered approaches, users are analyzed in order to acquire knowledge about work activities, procedures, standards, users' habits and needs [20]; they are also involved in system evaluation. Participatory approaches include representatives of users in the design team [25]. These approaches exploit techniques derived from social science, which support communication and collaboration within the interdisciplinary team, for prototyping [2].

Meta-design goes beyond, but includes the user-centered design approach and participatory design. As defined in [12]: "meta-design characterizes objectives, techniques, and processes for creating new media and environments allowing 'owners of problems' (that is, end users) to act as designers. A fundamental objective of meta-design is to create socio-technical environments that empower users to engage actively in the continuous development of systems rather than being restricted to the use of existing systems". In this perspective, meta-design underlines a novel vision of interactive systems that is at the basis of our approach. All stakeholders of an interactive system are "owners" of a part of the problem and therefore they must all contribute to system design. Moreover, co-evolution forces all stakeholders in a continuous development of the system. This is carried out, on one hand, by end users, who can perform tailoring activities to adapt the software environments they use to their evolved needs and habits. On the other hand, end users should collaborate with all other stakeholders not only in the design but also in the evolution of the interactive system. For these reasons, stakeholders need different software environments, specific to their culture, knowledge and abilities, through which each stakeholder can contribute to shape software artifacts. They can also exchange among them the results of these activities, to converge to a common design or evolve an existing system.

In light of these considerations, we view meta-design as *a design paradigm that includes end users as active members of the design team and provides all the stakeholders in the team with suitable languages and tools to foster their personal and common reasoning about the development of interactive software systems that support end users' work.*

To support a meta-design approach, Fischer et al. have developed the Seeding, Evolutionary growth, and Reseeding (SER) process model, which considers system design as a three-phase activity: 1) seed creation, 2) seed evolutionary growth, 3) reseeding phase [11]. The SER model is exploited in the development and evolution of the so-called DODEs (Domain-Oriented Design Environments), which are "software systems that support design activities within particular domains and that are built specifically to evolve" [11]. Our approach has some similarities with this work, but it emphasizes the need of providing personalized environments to all stakeholders, in terms of language, notation, layout, and interaction possibilities.

## 6 The Software Shaping Workshop Methodology

In this section we describe the Software Shaping Workshop (SSW) methodology [6], according to which an interactive system is developed as a network of different software environments customized to the different stakeholder communities involved in the use, design and evolution of the interactive system. We show how the SSW methodology helps bridging the communication gaps among the different stakeholders in the design team and supports the co-evolution of users and systems.

## 6.1 Software Shaping Workshops

In the SSW methodology, software environments are designed in analogy with artisan workshops, i.e., small establishments where artisans, such as blacksmiths and joiners, manipulate raw materials in order to manufacture their artifacts. At each step of their activity, artisans can extract from a repository the tools necessary for the current activity and set back those ones no more needed. In this way, every artisan adapts the environment to her/his needs and has available all and only the tools needed in the specific situation. By analogy, a software environment is designed as a *virtual workshop*, in which users find a set of (virtual) tools whose shape, behaviour and management are familiar to them. Such an environment allows users to carry out their activities and adapt environment and tools without the burden of using a traditional programming language, but using high level visual languages tailored to their needs. Users get the feeling of simply manipulating the objects of interest in a way similar to what they might do in the real world. Obviously, while traditional artisans shape real supplies, users shape software artifacts. For this reason we call these environments *Software Shaping Workshops* (SSWs) [6]. It is worth noting that virtual workshops and the tools they provide are required to evolve more quickly than real ones in the artisan workshops.

The SSW methodology provides each end user community with a workshop, called *application workshop*, used by that community in its daily work. An application workshop is customized to end users' culture, background and skills, and can possibly be tailored by the users themselves, also by creating new artifacts [6]. Application workshops are not directly created (and successively evolved) by software engineers, but their design, development and modification are carried out by an interdisciplinary team that, besides software engineers, includes HCI experts and end user representatives as domain experts. Each member of the team of experts use a type of workshop, called *system workshop,* customized to her/his culture, background and skills, in order to carry out the design, development and evolution of other workshops.

Overall, an interactive system to support the work practice in a given application domain is not a monolithic piece of software, but it is developed as a *network* of *system and application workshops*. The network allows the different stakeholders to communicate and collaborate to the system design, implementation, use and evolution by working with a workshop customized to them, they use their own languages and notations, so that they are not disoriented and may overcome the gaps existing among them. In general, a network is organized in levels. Fig. 1 presents a generic workshop network including three levels:

a) *Meta-design level.* Software engineers use a system workshop, called W-SE, to provide the software tools necessary to the development of the overall interactive system, and to participate in the design, maintenance, and validation of application and system workshops. More specifically, software engineers produce the initial programs, which generate the SSWs to be used and refined at lower levels, and participate in the maintenance of SSWs by modifying them to satisfy specific requests coming from lower levels.
b) *Design level.* HCI experts, and domain experts cooperate in design, maintenance, and validation of application workshops through their own system workshops: domain experts belonging to the user community X

participate in the design and maintenance of the application workshop, W-End-UserX, devoted to their community, using a system workshop, W-ReprX, created by the software engineers and customized to their own needs, culture and skills; they collaborate with HCI experts, who use their own system workshop, W-HCI, to check the functionalities and behavior of the application workshop, W-End-UserX, and adapt it.

c) *Use level.* End users belonging to the community X participate in task achievement using the application workshop, W-End-UserX, devoted to their community.

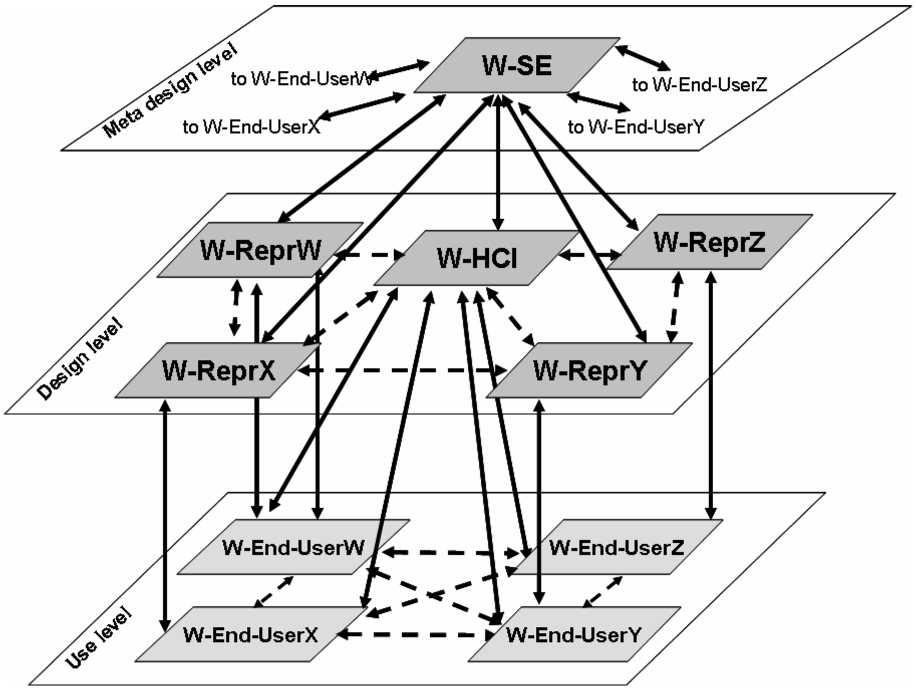

**Fig. 1.** A network of SSWs. Dashed arrows indicate exchange paths and plain arrows indicate request and generation paths.

On the whole, both meta-design and design levels include all the system workshops that support the design team in performing the activity of participatory design. Such workshops can be considered User Interface Development Environments (UIDEs) [22]. The novel idea is that the UIDEs used by domain experts are very much oriented to the application domain and have specific functionalities, so that they are easy to use by domain experts. On the other side, UIDE used by HCI experts are very much oriented to the HCI domain and have specific HCI functionalities, so that they are easy to use by HCI-experts; UIDE used by software engineers are very much oriented to the software engineering domain and have specific software engineering functionalities, so that they are easy to use by software engineers.

### 6.2 Communication Among SSW

We show in this section how the network organization supports co-evolution of users and systems and overcomes the communication gaps among the different stakeholders. In a SSW network, communication is guaranteed among workshops at the same level and from a lower level to the upper one, and vice versa, by communication paths. Through these paths, at the use level, end users exchange among them data related with their current task, to achieve a common goal. At the design level, HCI experts and domain experts exchange data and programs specifying workshops. HCI experts and domain experts also communicate with software engineers when it is necessary to forge new tools for their activities. Moreover, requests for workshop modification or extension can be sent to the design level or to the meta-design level from the lower one. Finally, when new tools or workshops are created at high levels, they are made available to the lower ones. Precisely, communication paths can be classified as:

1. *exchange* paths: they are the paths along which the exchanges of data and programs occur. Exchange paths are those existing among the workshops at the same level;
2. *request* paths: they are concerned with the communications going from low levels to higher levels; these communications trigger the co-evolution process, carrying on the feedback from end users that may include requests for workshop modification or extension;
3. *generation* paths: they represent the activity of using system workshops at a high level to generate, modify or extend workshops to be used at the lower level; new or evolved workshops are made available to lower levels along such generation paths.

The design team activity keeps going through the whole lifecycle of an interactive system due to co-evolution. In a first phase, called *design time,* the design team develops application workshops for the user communities addressed by the overall system. Co-evolution determines the adaptation of the workshops to the requests arising from new usage. Hence, at *co-evolution time,* thanks to the communication possibilities the network offers, the design team receives user complaints and suggestions about the workshops they interact with. Request paths are crucial to allow an end user or a designer to notify her/his problems or requests to the higher level. In particular, an end user or a designer finding problems during her/his interaction with a workshop, has the possibility to annotate such problems in the workshop itself. The problems might depend on either lack of functionalities or poor usability. Annotations can be made available to all the experts reachable in the network along the request paths. The experts analyze these annotations, communicate among them using exchange paths (or request paths, if they in turn refer to the higher level), and agree on a possible solution to the notified problems, thus updating the corresponding workshop. Co-evolution is thus the result of a combination of generation, request and exchange activities that are carried out throughout the lifecycle of the SSW network.

Moreover, the design team has the possibility to observe user activities, the new usages of the system, the new procedures induced by the evolving organization. Consequently, the design team updates the system and sometimes also the underlying software technologies. In these phases, HCI experts take the responsibility of usability and accessibility aspects, and software engineers take the responsibility of the efficiency and implementation aspects. Both application and system workshops must be maintained and co-evolved during the system lifecycle.

The fact that each stakeholder works with a customized workshop is the key to overcome communication gaps. Indeed, such workshops allow the stakeholders to interact by using languages, tools and working strategies that are familiar to them.

## 7 Conclusion

In real situations, interactive systems and their end users undergo a continuous co-evolution. The need of keeping the systems usable and fully fledged requires that the multidisciplinary team of designers remains active for the whole system lifecycle. Our approach conceives an interactive system as a network of software environments (the workshops), each customized to a specific community of stakeholders involved in design and use of the system. This approach permits to bridge the communication gaps existing among the different stakeholders and also to transfer as much as possible the responsibility of system design and evolution to domain experts.

## References

1. Arondi, S., Baroni, P., Fogli, D., Mussio, P.: Supporting co-evolution of users and systems by the recognition of Interaction Patterns. In: Proc. AVI, pp. 177–189. ACM Press, New York (2002)
2. Bødker, S., Grønbæk, K.: Cooperative prototyping: Users and designers in mutual activity. International Journal of Man.-Machine Studies 34(3), 453–478 (1991)
3. Borchers, J.: A pattern approach to interactive design. John Wiley & Sons Ltd, New York (2001)
4. Bourguin, G., Derycke, A., Tarby, J.C.: Beyond the Interface: Co-evolution inside Interactive Systems - A Proposal Founded on Activity Theory. In: Proc. IHM-HCI, pp. 297–310 (2001)
5. Carroll, J.M., Rosson, M.B.: Deliberated Evolution: Stalking the View Matcher in design space. Human-Computer Interaction 6(3 and 4), 281–318 (1992)
6. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: End user Development: the Software Shaping Workshop Approach. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 183–205. Springer, Heidelberg (2006)
7. Costabile, M.F., Fogli, D., Lanzilotti, R., Mussio, P., Piccinno, A.: Supporting Work Practice through End User Development Environments. Journal of Organizational and End User Computing 18(4), 43–65 (2006)
8. Costabile, M.F., Fogli, D., Marcante, A., Piccinno, A.: Supporting Interaction and Co-evolution of Users and Systems. In: Proc. AVI 2006, Venice, Italy, pp. 143–150. ACM Press, New York (2006)
9. Costabile, M.F., Fogli, D., Fresta, G., Mussio, P., Piccinno, A.: Building environments for End-User Development and Tailoring. Proc. HCC 03, Auckland, New Zealand 38, 28–31 (2003)
10. Dillon, A., Watson, C.: User analysis in HCI: the historical lesson from individual differences research. International Journal of Human-Computer Studies 1996 45(6), 619–637 (1996)

11. Fischer, G.: Seeding, Evolutionary Growth, and Reseeding: Constructing, Capturing, and Evolving Knowledge in Domain-Oriented Design Environments. Automated Software Engineering 5(4), 447–468 (1998)
12. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-Design: a Manifesto For End user Development. Communications of the ACM 47(9), 33–37 (2004)
13. Fischer, G., Giaccardi, E.: Meta-Design: A Framework for the Future of End User Development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 427–457. Springer, The Netherlands, Dordrecht (2006)
14. Fischer, G.: Symmetry of ignorance, social creativity, and meta-design. Knowledge.-Based Syst. 13(7-8), 527–537 (2000)
15. Fogli, D., Fresta, G., Mussio, P.: On Electronic Annotations and its implementation. In: Proc. AVI 2004, Gallipoli, Italy, pp. 98–102 May 25–28 (2004)
16. Folmer, E., van Welie, M., Bosch, J.: Bridging patterns: An approach to bridge gaps between SE and HCI. Journal of Information and Software Technology 48(2), 69–89 (2005)
17. Lauesen, S.: User Interface design - A software engineering perspective. Addison-Wesley, Reading (2005)
18. Majhew, D.J.: Principles and Guideline in Software User Interface Design. Prentice-Hall, Englewood Cliffs (1992)
19. Nielsen, J.: Usability Engineering. Academic Press, San Diego, CA (1993)
20. Norman, D.A., Draper, S.W.: User-Centered System Design: New perspectives on Human-Computer Interaction. Lawrence Erlbaum, Mahwah (1986)
21. Norman, D.A.: Human-centered design considered harmful. Interactions 12(4), 14–19 (2005)
22. Preece, J.: Human-Computer Interaction. Addison-Wesley, Reading (1994)
23. Rittel, H.: Second-Generation Design Methods. In: Cross, N. (ed.) Developments in Design Methodology, pp. 317–327. John Wiley and Sons, New York, NY (1984)
24. Schön, D.: The Reflective Practitioner – How Professionals Think in Action, Basic Books (1983)
25. Schuler, Namioka (eds.): Participatory Design - Principles and Practices. Lawrence Erlbaum Associates, Hillsday, N.J (1993)
26. Stiemerling, O., Kahler, H., Wulf, V.: How to Make Software Softer – Designing Tailorable Applications. In: Proc. ACM Symposium DIS, pp. 365–376. ACM Press, New York (1997)