Beyond Documents: Visualizing Informal Communication

Kurt Schneider, Kai Stapel, Eric Knauss
Software Engineering Group, Leibniz Universität Hannover
Welfengarten 1, 30167 Hanover, Germany
{Kurt.Schneider, Kai.Stapel, Eric.Knauss}@inf.uni-hannover.de

Abstract

Requirements engineering heavily relies on oral and informal communication like meetings, phone calls, and e-mails. Some requirements are traditionally communicated outside formal documents in many companies. Considering document-based communication only is, therefore, not sufficient. Communication beyond documents should be taken into account when a company wants to improve its requirements engineering practices.

We present a notation for visualizing the flow of requirements, explicitly including both informal and document-based communication. An example illustrates the notation by applying it to eliciting and validating security requirements. We compare our visualization approach for flows of requirements to related notations. The comparison shows how differences in notations affect the ability to model essential concepts of our approach.

Explicitly modeling flows beyond documents provides new opportunities for requirement awareness, process improvement and innovation of tools and techniques.

1. Introduction

Despite maturing processes and sophisticated tools, practitioners often feel frustration about requirements engineering: Perfection seems impossible, and mistakes can cause severe damage to a project. Apparently, tools and formal specifications are not sufficient to guarantee successful requirements engineering [1].

In practice, many requirements are represented and transferred outside formal documents: Practitioners feel the need for informal communication. Theoretical work [2] and case studies alike emphasize the importance of informal communication in software projects [3, 4]. All valid requirements should be considered in building a software system.

We propose visualizing communication about requirements, no matter whether it is based on documents or not. Customers and requirements engineers, developers and managers communicate in meetings, phone calls, using Wikis, slides, e-mails and informal notes. Those media establish flows of requirements beyond documents.

Agile methods, for example, emphasize direct exchange of information and requirements without using intermediate documents. Many companies working in a more traditional style also transfer requirements through rather informal communication. Our approach of *visualizing flow of requirements* is not restricted to agile or plan-driven approaches. Requirements flow in every style of development.

In section 2, we elaborate on concepts of modeling the flow of requirements. Section 3 sketches how models are built and used. Related work in section 4 further clarifies the concepts. Section 5 presents a simple notation for visualizing stores and flows of so-called *solid* and *fluid* information, which are essential concepts of our approach. Section 6 illustrates the notation by visualizing the flow of security requirements in an example. In section 7, our concepts are mapped onto existing modeling notations like UML or Data Flow Diagrams (DFDs). We compare notations with respect to our specific visualization purpose.

2. Fluid and Solid Flow of Requirements

The term "communication" refers to the exchange of information, but also includes phenomena that build on this exchange. Phenomena can be social, emotional, or economic in nature. Communication has been found to be a basis for trust and closeness between companies [5]. We focus on the flow of requirements and derived information, which we regard as a layer of communication. Emotional, social, and other layers that build on it are not discussed in this paper.

A main contribution of our work is explicitly modeling fluid information and flow together with solid representations. Fluid information is commonly used in most development organizations, but many

Third International Workshop on Requirements Engineering Visualization (REV 08)
at 16th IEEE International Requirements Engineering Conference, Barcelona, Spain

current process models and visualizations are restricted to solid information.

Solid representation refers to documents and stores with certain characteristics:

(1) information can be retrieved by others
(2) without the help of the author or source
(3) even after some time has expired and
(4) in a form that supports dissemination.

All other representations are called **fluid**. Fluid representations include meetings and oral communications, blogs, chats, informal Wikis, phone calls, and personal e-mails not accessible to others. Not every act of informal communication adds to requirements engineering (e.g., personal chat, random meetings at the coffee machine). However, ignoring fluid requirements leads to errors, misunderstandings, and parallel work.

There are trade-offs: Creating solid representations and retrieving information is often more effort-consuming than fluid exchange by human interaction. In turn, fluid information can be forgotten, and it is limited in access. Combining advantages of both styles is the basis for designing individual networks of flows. It is important to note that we do not assume each and every requirement or piece of information needs to be solidified in the end. For some purposes, fluid representations are more appropriate than solid.

The **visualization** of fluid and solid flows of requirements must be tailored to meet users' needs. In requirements engineering, a diverse group of people are supposed to understand it, reaching from customers and domain experts to developers, managers and software process improvement specialists.

3. Purpose and Focus of Modeling Flows

Concepts and models of information flow can be used for several purposes in requirements engineering. We sketch typical uses of information flow models and reference earlier publications to related application examples. Since the main focus of this paper is the visualization aspect of fluid information, we can only touch on methodology issues, like building and maintaining the models. In most real-world situations, several of the following modelling purposes are pursued.

Usually, information flow models are created in an interactive process of interviews, consistency checks and workshops. A moderator uses our visualization in order to make participants aware of fluid information, combined with traditional solid representations. The visualization helps to focus and document discussions. Depending on their purpose, some models need to be updated, while others are only needed to trigger improvements. Those models can be discarded afterwards.

Increasing Awareness and Overview. A model of requirements and information flow corresponds to a map of project communication. Stakeholders and participants can use it to find a relevant document or person to talk to. At a car manufacturing company, information flow models were applied for visualizing the OEM vs. subcontractor relationship in requirements engineering for electronic control units: Allmann [6] visualized and discussed different variants of that relationship.

Improving Requirements Processes and Practices. Requirements engineering processes are often based on informal communication. Choosing appropriate information flows can be essential for improving those processes effectively. For example, a slow and bureaucratic process can be sped up by using more fluid elements. In a financial institution, we evaluated a large process model with respect to information flow [8]. A custom-made search tool helped to identify over 100 findings; some were simple modeling flaws, while others uncovered potential for improving information flow. Appropriate metrics to assist in semi-automated information flow analysis include percentage of fluid vs. solid flows; fan-in and fan-out of fluid information stores. In addition, more complex patterns of flows and stores were used in all our industrial applications (e.g. Chinese Whisper or Dead Document [8]).

Defining and Tailoring Communication. Information flow can be used to model communication of a newly designed process in order to achieve specific communication characteristics. Fluid and fast interactions with customers can be specified, as well as the use of experience in requirements analysis tasks, to name just two examples. Section 6 provides an example of specifying the flow of security requirements. We also used a FLOW analysis in the automotive industry to plan for the introduction of a Wiki system [7]. It was supposed to improve flow of requirements and information.

Inventing Tools and Techniques. It is possible to build tools and techniques specifically for improving certain flow patterns [9]: *FlowCombiner*, for example, applies heuristic consistency checks on information flow models (in-out consistency, patterns of solid/fluid flows). *FastFeedback* provides feedback on use cases while they are elicited during stakeholder interviews. Both tools were built in response to analyzing concrete information flow models in the requirements elicitation phase of a large administrative project.

Extending the scope. For the above-mentioned purposes, the focus of our approach is on qualitative models of requirements flows in both *fluid* and *solid* represen-

tation. They were appropriate for the cited applications. However, we currently investigate opportunities for combining qualitative models of flow with goal models, company relationship aspects, and even quantitative models of flow. The latter would facilitate measuring requirements process properties [10]. Quantitative extensions led to a motivational requirements engineering game [11]. In this paper, we concentrate on core visualization aspects.

4. Related Work: Visualizing Communication in Requirements Engineering

Requirements must be elicited and communicated within a software project. Several researchers have investigated dependencies and dynamics of software projects including the impact of requirements flowing through projects:

Abdel-Hamid and Madnick [12] developed quantitative simulation models of software projects. Abdel-Hamid and Madnick were mostly interested in *policies* which they described by formulas. Informal communication or visualization was not considered.

Other approaches emphasize the social and psychological dimensions of project communication. Pikkareinen et al. [13], for example, investigate communication in agile projects and their impact on building trust. The focus of their work is the impact of agile communication patterns on project success.

Winkler [14] uses the term "information flow", too. He focuses on dependencies between artifacts. An artifact is a part of a document. His main interest is traceability of requirements. Winkler considers only document-based requirements and information flows. He uses different ad-hoc illustrations to discuss inter-document flow of requirements.

Berenbach and Borotto [10] present requirements project metrics. All metrics are based on solid information only. However, information flow within their "Formal Requirements Development Process" could be modeled with our approach. Starting from their solid perspective and then extending them by fluid aspects. Some of their metrics could be extended to refer to fluid information, too.

Kwan et al. [15] propose to create Requirements Centered Social Networks (RCSN) by collecting data on formal and informal flows of information. They plan to build RCSN by observing actual stakeholder communication on all available levels, including phone calls made, e-mails sent. They suggest collecting that information automatically and visualizing it (see section 7). Computer-based communication through documents or e-mails is easier to capture than off-line meetings and oral communication [16].

An RCSN visualizes *past observations* of information transfer, while our approach focuses on *future* and *recurring* flows of requirements. Kwan et al.'s approach might be used together with ours to compare intended flow and actual communication.

5. FLOW Notation

In the FLOW project at Leibniz Universität Hannover, we model, analyze and improve information and requirements flows in software projects. A notation for visualizing information flows was designed to meet the following criteria:

- The notation should be easy to use on whiteboards, in general drawing tools like PowerPoint, and in custom-built editors.
- Models should be self-explanatory to domain experts without any training in software modeling.
- The notation must explicitly visualize and distinguish solid and fluid flows of information.
- There should be visualizations of activities in order to link flow models to process steps.
- The notation should contain as few symbols and details as possible. It must not distract from the information flow focus.
- Existing and potentially familiar notation concepts should be reused when this does not conflict with any of the before-mentioned criteria.

The syntax as presented in Table 1 was designed to convey the concepts of information flow while observing the criteria mentioned above:

- Stores are depicted by easy-to-draw symbols that are provided in most drawing tools: A document symbol and a human face. These two symbols are self-explanatory in many cases: Humans are the most important *fluid stores*, while documents are the classical storing device of *solid information*.
- Identifiers refer to individuals or roles, depending on the purpose of a model.
- Multiple document or person symbols refer to one or more stores of the same document type or group of persons. Details are omitted intentionally.
- Flows are represented by arrows. All flows originating from a solid store are solid. All flows originating from a fluid store are fluid.
- Flows originating from an activity can be either solid or fluid: A model builder can express an intention or assumption by using fluid or solid style.
- Tags on arrows can be used to highlight specific types of information flowing. Requirements and derived information are the default type. It needs no tags, which simplifies models visually.

- Experience often acts as catalyst on other information flows. Despite its prominent role, experience is rarely considered explicitly. In FLOW, experience is depicted in a different color or in gray.
- The activity symbol is used as follows:
  - Rectangle in the style of SADT or IDEF0 (http://www.idef.com/idef0.html) with a descriptive label: Incoming flows are connected to the symbol on the left side. Outputs are connected to the right. Controls and tools are connected at the top and the bottom, respectively.
  - Activities are often controlled by experience. This case is illustrated in Table 1 by showing the control flow in the experience color (e.g., gray).

The activity symbol serves several purposes:

- Black-box to hide details of information flow and transformations. Different implementations must be consistent with all flows connected to the symbol the "flow interface" of the activity, according to the terminology used in [17].
- Connection to other process notations. Documents, stakeholders, and activities can be identified with respective elements of a process notation. Activities are connected with process steps. Connecting FLOW models to existing process models facilitates and speeds up building initial models. However, FLOW models will exceed the scope of traditional process models when they include visualizations of non-documented communication among humans.

Editing FLOW models: A graphical editor with semantics (ProFLOW) is currently co-evolving with the notation. It is implemented as an Eclipse plug-in. The editor is structured as a framework to accommodate fast updates in the definition of our graphical notation. We consider the concepts and manual use of the notation in industry our primary concerns. Therefore, a tool like ProFLOW is a convenient, yet secondary aspect of our work. In all current industrial applications, FLOW models were initially drawn by hand or in PowerPoint.

Other tools do not use the notation, but improve specific requirements engineering situations, as exemplified in section 3 [7, 9]. Information flow analysis led to specifying and building those tools.

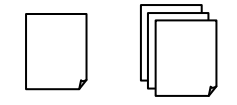
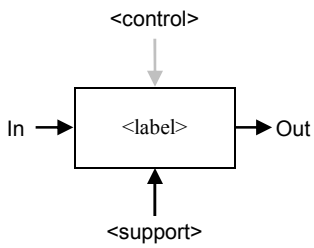
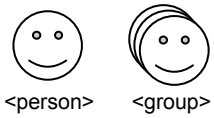
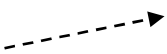
6. Application Example: Handling Security Requirements

During a research project, the flow of security-related requirements had to be defined. There were several ideas and decisions on how to get and to use those requirements. Using the FLOW notation, the flow of security requirements was visualized and discussed by security experts.

This example demonstrates the equal use of fluid and solid flows. Using the simple notation allowed participants with different backgrounds to discuss issues with a moderator. The goal pursued in this example was *Defining and Tailoring Communication* in order to specify the core of the new technique.

Information flow and data flow have many concepts in common (cf. Sec. 7). This resemblance allows us to reuse data flow techniques for information flow modeling: Hierarchical decomposition is a concept associated with data flow diagrams [18]. Refining activities is the corresponding concept in FLOW.

Table 1. FLOW Syntax

information state	store	information flow	experience flow	activity
solid				
fluid				

DeMarco [18] describes Context Diagrams as one way to start modeling a data flow system. In analogy, we used the FLOW notation to describe the context of a new technique of handling security requirements, called SecReq. Fig. 1 shows only one activity symbol representing the entire new technique being developed. Other symbols are linked to that activity. Being outside the activity box, those symbols represent sources and sinks of security requirements. They correspond to Externals in context diagrams.

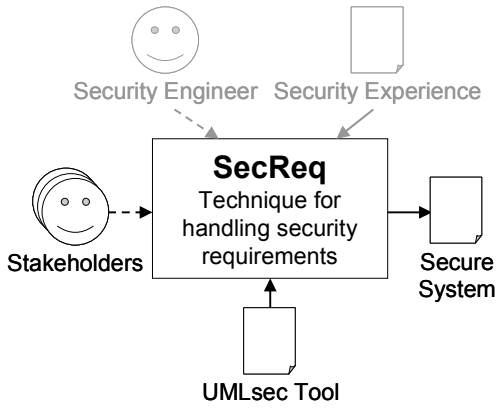


Fig. 1. Visualization of flow context of SecReq

The following decisions are visualized in Fig. 1:

- SecReq is a technique for handling security requirements. In FLOW terminology, it is represented as an activity.
- Using the input of several stakeholders, SecReq will lead to implement a secure system that meets those requirements.
- We do not expect stakeholders to write down their requirements, so SecReq will need to accommodate fluid input.
- We decided to use Jürjens UMLsec tool [19], and we want to reuse both fluid and solid experiences on handling security requirements.
- The UMLsec tool is visualized by a document symbol because it meets the criteria for solid stores.
- We know nothing else about the SecReq technique. It is treated as a black-box that can be implemented in various ways.

During later discussions, we adjusted the interface of SecReq to match with existing parts of the process. Since UMLsec [19] already supports UML specifications enriched with a security profile, design is outside our intended scope of SecReq.

As Fig. 2 (right) shows,

- System construction is supposed to receive a valid set of security requirements in a solid form as *Improved SecReqs*. It will be supported by UMLsec.
- We do not know how UMLsec transforms valid security requirements into a secure system, nor how Security Requirements Elicitation works in detail, which is encapsulated in two other activities.
- We *do* know that the requirements are supposed to be delivered to SecReq in a fluid form.

Information flow modeling continues by considering other aspects. For example, the input of requirements from stakeholders needs to be considered in detail. Fig. 3 shows alternative flows:

(1) Stakeholders could be asked to write down their own requirements each. Those solid documents can be merged by a security expert (Fig.3, top).

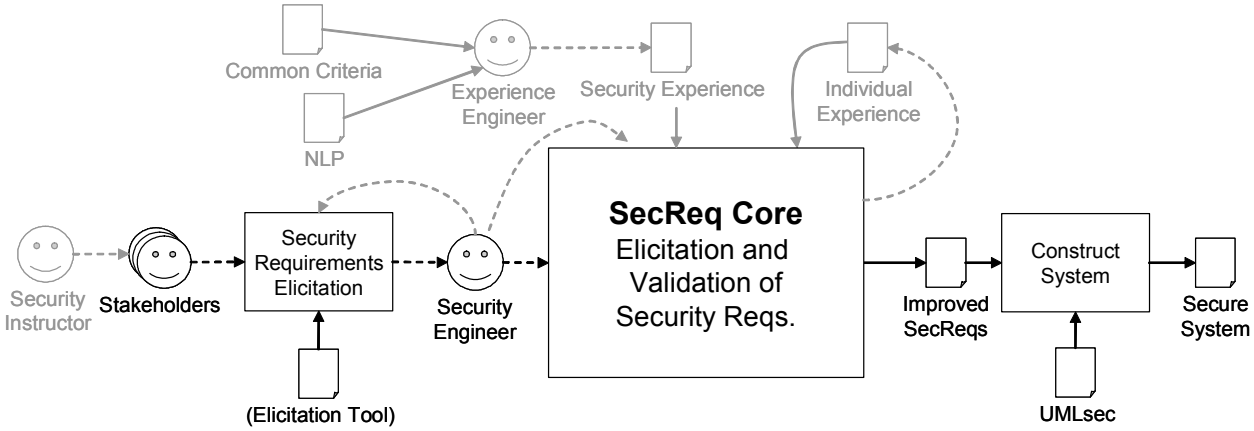


Fig. 2. Modeling Requirements and Experience Flows

(2) In many software projects, stakeholders will have little experience in documenting requirements, and in particular in identifying security requirements. For instance, customers of a bank will be neither willing nor able to provide complete and consistent requirements documents. In our case, a workshop (Fig. 3, center) was proposed to capture requirements. It consists of a clique (i.e., fully connected set) of stakeholders together with a moderator. By using a security expert as a moderator, communication between stakeholders can stay completely fluid. This will speed up communication and make it more enjoyable for stakeholders. The threshold for contributing requirements can be lowered; listening to each other facilitates elicitation and validation. The moderator takes notes and solidifies them afterwards.

(3) Experience in requirements elicitation shows how difficult it can be to assemble all stakeholders in one workshop. Fig. 3 (bottom) shows a third variant to avoid that need: Stakeholders receive an introduction from a security instructor. In a short fluid conversation, they learn about the nature and importance of security requirements.

Option (3) was chosen for SecReq (Fig. 2, left): In a typical elicitation situation a security engineer talks to the stakeholders. The model does not specify whether that happens in a workshop or in one-to-one interviews.

The expert uses his or her experience to guide the interaction. This is depicted by a gray dashed arrow. The outcome is a fluid flow of requirements – no document is being produced. *SecReq Core* at the center of Fig. 2 must, therefore, transform incoming fluid requirements into a solid document for UMLsec.

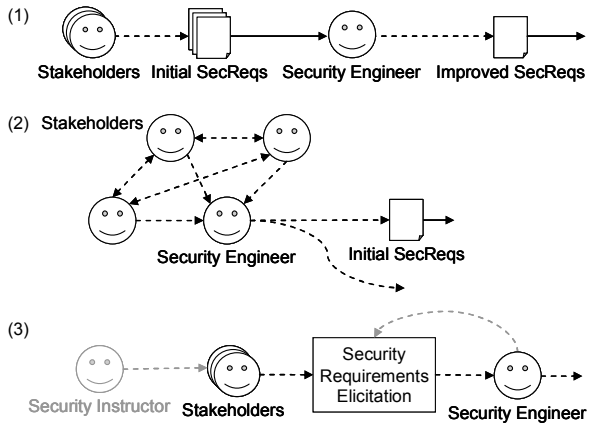


Fig. 3. Three Variants of Requirements Input

Along the same lines, experience flow was specified during discussions between security experts. As Fig. 2 (top) shows, written experiences are supposed to include documents well-known in the security community, such as Common Criteria, and a constantly updated list of lessons learned. Since new lessons learned will be applied immediately, there is a visible feedback loop.

During this entire application of FLOW to SecReq, security experts discussed issues of getting and passing on security requirements. We acted as facilitators with a background in requirements engineering. FLOW was used as a common visualization for focusing on solid vs. fluid flow of requirements and experiences. The notation was introduced in only a few minutes, and served as a tool for shared understanding.

To conclude, FLOW has been used in *this* example application for designing the future flow of security requirements in SecReq. In other company settings [6-8], FLOW has been used to analyze and visualize existing processes and flows of information.

Within the focus of this paper it is most interesting to note that the issue of informal communication can be visualized with very simple means. Visualization can facilitate decision making and process improvement.

7. Information Flow in Related Notations

The FLOW notation was designed to explicitly visualize the concepts of fluid and solid information and flows. According to the above-mentioned criteria, the notation is intended for a wide variety of users. Therefore, we wanted to reuse well-known concepts from existing notations. Due to this rationale, FLOW has much in common with some of those notations.

In this section we focus only on the essential concepts conveyed by FLOW. Although differences between FLOW and related notations may seem subtle at first, fluid flows of experience and requirements are obviously more difficult to represent in other notations which were not tailor-made for them.

This comparison does not intend to evaluate related notations as such. The value of a visualization or notation depends on the purpose it is used for. As the comparison shows, fluid requirements flow can be expressed in other notations, but not in a very straightforward way. Therefore, it is justified to define FLOW as a new notation for our new purpose of modelling informal communication.

We use a typical excerpt from a FLOW model as a benchmark for information flow modelling: the elicitation part of the SecReq example, see Fig. 4. We compare corresponding visualization in each of the other notations.

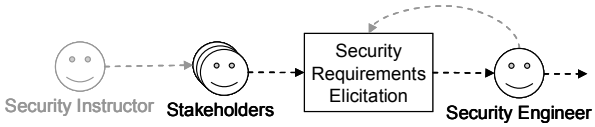


Fig. 4. Requirement Elicitation in FLOW notation

Data Flow Diagrams (DFD). Data Flow Diagrams were designed to model data flow within computer systems [18], but they can also be used to model data flow in a software process. It is characteristic of both DFDs and information flow models to ignore control flow. In principle, available inputs can be transformed into outputs at any time – there are no events, no triggers, and no timing in such a model.

From a semantic point of view, DFDs and FLOW mainly differ in the data vs. information flowing. In terms of visualization of concepts, there are several more differences. Visualizing fluid information flowing between stakeholders is difficult in DFDs and requires some violation of DFD modelling rules [18]:

- There is no notion of fluid information or flow. There is no way to emulate the distinction visually within the limits of the DFD notation.
- Experience is just another data or information type, no extra symbol or highlighting is available.
- DFD processes resemble FLOW activities, but persons also receive and send information.
- When mapping information stores to DFDs, storage symbols can be considered. However, data flows to and from storage symbols are defined to have exactly the same content structure as the store. This does not reflect the role of a person in a FLOW diagram: People work with information and transform it.
- Naming conventions call for unique labels on all data flows. Since flows have no tags in most FLOW models, DFDs are usually far more cluttered and take more time to read.

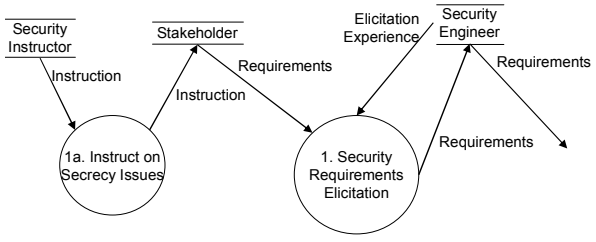


Fig. 5. Requirement Elicitation as DFD

Fig. 5 depicts the security elicitation example as a DFD. People are modelled as data stores. An extra activity had to be introduced since direct data flows are not permitted between storage symbols – while they occur frequently between persons in FLOW. The security engineer uses both information about requirements *and* experience about security systems engineering. This implies he cannot be modelled as a single data store in a DFD: A data store can only collect data of a *single* type. Therefore, the security expert could be drawn as process instead. However, a process is not supposed to store data, whereas persons are considered fluid stores of information. There is no straightforward way to solve all conflicts.

The main drawbacks of using a DFD to model information flows for our purposes are:

- No distinction between solid and fluid information
- No explicit notion for experience
- No direct flow between people
- People look like two lines, not like people or faces. No self-explanatory symbols, cluttered models.

Obviously, these drawbacks violate several of the criteria set out in section 5 for an appropriate notation.

UML. UML offers a wide range of diagram types. Communication Diagrams seem to be close to FLOW models. However, Communication Diagrams show flow of messages between object instances rather than flow of information in a RE process. UML Activity Diagrams can be used to model information flows. Activity diagrams were designed to model processes using control flows, which again is not our intention. But there are options to model data and object flows, which can be used to emulate information flow.

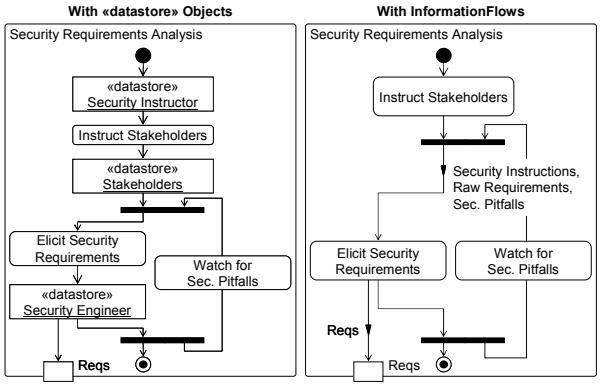


Fig. 6. Flow of Security Requirements: The Example as a UML Activity Diagram

Fig. 6 depicts the SecReq example as a UML Activity Diagram. In the version on the left, data store objects to model persons, using the auxiliary UML information flow construct [20] (right). Although

UML's and our information flows share the same name they are not the same thing. UML's information flows were designed to model data flow in a system on an abstract level in early development stages, they can be used for our purpose in Activity Diagrams.

In both cases it is difficult to model the experience flow from the security engineer to the elicitation activity. It may occur in parallel with requirements flow, but it does not need to be synchronized with the stakeholders' requirements.

The drawbacks of using Activity Diagrams to model information flows are:

- No distinction between solid and fluid
- No means of explicitly modelling experience
- No visual clues like faces, documents, broken lines
- Many details and labels irrelevant for information flow
- Activities forced in a sequence or order

To compensate for the drawbacks, UML profiling mechanism can be used. Profiles extend a UML diagram type by adding concepts and visualizations. Missing symbols and line styles can be added, which will lead to a quite different apparel of the activity diagram. An information flow profile will retain its undesired focus on control flow, but look much more like a FLOW model otherwise.

Little-JIL. Little-JIL [21] is a visual programming language for the coordination of agents in process programming. Agents are either human or a tool capable of performing a work task. Work tasks are similar to activities in other process languages. Thus, Little-JIL can shortly be characterized as a programming language for processes. In order to be executable Little-JIL diagrams need to be rich in detail. That makes even small diagrams difficult to read.

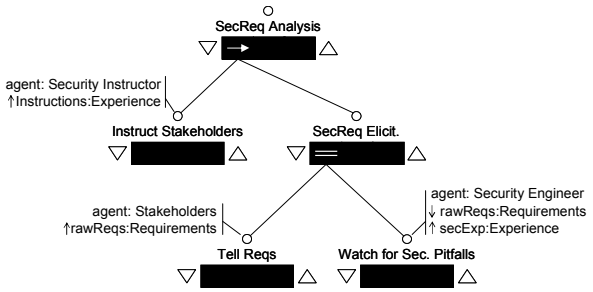


Fig. 7. Security Elicitation Example in Little-JIL

Fig. 7 shows the SecReq example in Little-JIL. Information flows are represented by in- and out-going parameters. Persons are represented as agents that perform certain steps that correspond to FLOW activities. Direct connections between the information stores and flows are not directly visible. Additional steps needed to be inserted that were not visible in the original FLOW diagram, namely Instruct Stakeholder, Tell Requirements, Watch for Security Pitfalls. Drawbacks for modelling information flows are:

- No distinction between solid and fluid
- No explicit notion for experience
- Many different symbols, difficult to distinguish for non-expert users
- No dedicated symbol for people, flow not visible
- Many details in labels and symbols

YAWL. Yet Another Workflow Language [22] is a representative of modelling languages for business processes based on Petri Nets. It resembles UML Activity Diagrams in mainly focusing on control flow. It resembles Little-JIL since it was designed to support direct enactment of processes. Both similarities lead to overwhelming distracting details when modelling information flows in YAWL.

RCSN. Requirements Centred Social Networks were specifically designed for requirements engineering [15]. They promote awareness in a development process of changing requirements. Each diagram shows the social network of a single requirement. Nodes are individuals in a certain role, like a customer or a requirements analyst. Edges represent communication flows including informal communication. Each edge shows the date, media and frequency of a communication flow related to that one requirement. RCSNs are supposed to be generated automatically, so that only a few types of informal communication can be captured, such as e-mails and chat protocols. However, the notation can easily be used to visualize all types of communication, including phone calls or meetings.

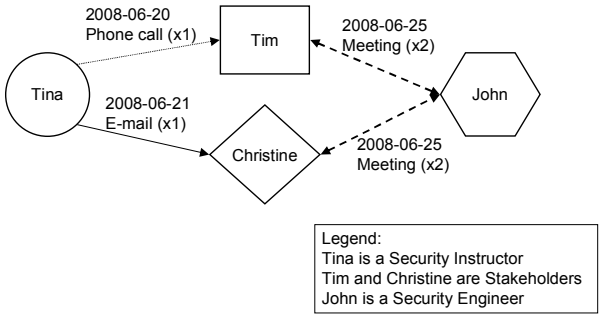


Fig. 8. Security Elicitation Example as a RCSN

In Fig. 8 the SecReq example is depicted as a RCSN. The roles of Security Instructor, Security Engineer, and Stakeholders are represented by individuals.

Table 2. Comparison of Notations: Purpose and Focus

Diagram type	FLOW	DFD	UML Activity Diagram	Little-JIL	RCSN
Main purpose	Process improvement by considering solid and fluid flows alike	System design	Process design	Process programming	RE awareness
Main object of interest	Information, in particular requirements	Data	Activities	Steps	Social network

Table 3. Symbols to Visualize Information Flow Aspects (Syntax)

Concept	FLOW	DFD	UML Activity Diagram	Little-JIL	RCSN
Information store	Person (fluid), Document (solid)	Data store, External	Data store stereotype	Parameters, Agents	Persons
Information flow	Dashed arrows (fluid) Solid arrows (solid)	Data flow arrow	Data/object flow edge	Parameters with control flow	Communication flow
Distinction of solid and fluid	Different symbols (see above)	No	Through Stereotypes	No	Style of arrow (color etc.)
Explicit experience	Explicit edge color	No	Association stereotypes	Parameter Type	No
Challenges when used for information flow modeling	No	Stakeholder as process, data store, or external? Labeling rules violated.	Requires stereotyping for symbols, extend-ed meaning etc. Will look like FLOW when stereotyped.	Very fine-grained symbols, very detailed and abstract models due to its roots as a *process programming* language	Single req. flowing as observed in reality. Notation not fully defined, many symbols.

Each individual in Fig. 8 is depicted by a different node shape. A legend had to be introduced to provide a connection between roles and individuals. Three different types of arrows show three different communication channels. A date, media, and occurrence frequency are attached. There are main drawbacks:

- RCSNs are designed to visualize the network for a single requirement
- No means to explicitly express experience
- Too many different node shapes are confusing.
- RCSN represent single concrete cases observed in the past, not recurring flows.

Table 2 summarizes the similarities and differences between the notations in general. Table 3 shows how certain model mechanisms can be used to address information flow aspects. This will require more or less severe violations of modelling rules or redefinition of symbols to capture information flow *syntactically*.

Experts familiar with the original notation may find it difficult to work with the new semantic *meaning* of such an adapted visualization, however. We recommend using the FLOW notation whenever flow of requirements and information is the main focus.

8. Conclusions

Communication and flow of requirements are essential for effective requirements engineering. While many requirements and related aspects are described in documents, many others are not. They are passed on during meetings, phone-calls, or e-mails.

We investigate information flow underlying communication. Requirements flow in both fluid and solid forms, and they do this on a regular basis. Therefore, it is wise to take non-documented flows of information into account when improving requirements practices and processes.

It is a main contribution of our work to explicitly model fluid representations together with solid ones.

Obviously, many extensions can be envisioned. Our own future work will focus more on our on-going

efforts of tool-building and attaching goals, responsibilities, and even quantitative aspects to our qualitative models of requirements flow. At this stage, we focus on raising awareness and increasing feedback on requirements practices. This can be achieved by qualitative models when they are visualized properly. This worked in our industrial applications.

Our proposed FLOW notation was designed to be simple and meet a number of criteria. We illustrate the use of that notation through an example. FLOW was designed for visualizing information flow. However, our concepts are not limited to the FLOW notation. To substantiate this claim, we compared how standard situations of information flow modelling could be represented in a number of related notations. It turns out, however, that the seemingly subtle differences affect the ease of modelling drastically.

Therefore, using the new FLOW notation rather than an existing visualization seems reasonable. This will enable requirement engineers to model issues of fluid and solid information flow in a concise and easy way. This is essential for reaching our goals. Additional research by others can build on our contribution in many ways; for example, combining it with other types of models or extending it towards new purposes.

Taking fluid information seriously is the core contribution of our approach. In this paper we have focused on visualization aspects.

9. References

[1] A. M. Davis and D. Zowghi, "Good requirements practices are neither necessary nor sufficient," *Requirements Eng*, vol. 11, pp. 1-3, 2005.

[2] J. S. Brown and P. Duguid, "Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning, and Innovation," *Organization Science*, vol. 2, pp. 40-57, 1991.

[3] J. D. Herbsleb and R. E. Grinter, "Architectures, Coordination, and Distance: Conway's Law and Beyond," *IEEE Software*, vol. 16, pp. 63-70, 1999.

[4] D. Damian, L. Izquierdo, J. Singer, and I. Kwan, "Awareness in the Wild: Why Communication Breakdowns Occur," presented at Second Intl Conference on Global Software Engineering, Munich, Germany, 2007.

[5] C. C. Nielson, "An empirical examination of the role of "closeness" in industrial buyer-seller relationships," *European J. of Marketing*, vol. 32, pp. 441-463, 1998.

[6] C. Allmann, L. Winkler, and T. Kölzow, "The Requirements Engineering Gap in the OEM-Supplier Relationship," LSO+RE 2006, Hannover, Germany.

[7] K. Stapel, E. Knauss, and C. Allmann, "Lightweight Process Documentation: Just Enough Structure in Automotive Pre-Development," EuroSPI², Dublin, Ireland, 2008.

[8] K. Stapel, K. Schneider, D. Lübke, and T. Flohr, "Improving an Industrial Reference Process by Information Flow Analysis: A Case Study," PROFES 2007, Riga, Latvia, 2007.

[9] K. Schneider, "Generating Fast Feedback in Requirements Elicitation," Requirements Engineering: Foundation for Software Quality (REFSQ), 2007.

[10] B. Berenbach and G. Borotto, "Metrics for Model Driven Requirements Development," 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, 2006.

[11] E. Knauss, K. Schneider, and K. Stapel, "A Game for Taking Requirements Engineering More Seriously," MERE'08: Third International Workshop on Multimedia and Enjoyable Requirements Engineering, Barcelona, Spain, 2008.

[12] T. Abdel-Hamid and S. E. Madnick, *Software Project Dynamics*. Englewood Cliffs, NJ: Prentice Hall, 1991.

[13] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still, "The impact of agile practices on communication in software projects," *Journal on Empirical Software Engineering*, vol. 13, pp. 303-337, 2008.

[14] S. Winkler, "Information Flow Between Requirement Artifacts," REFSQ 2007 International Working Conference on Requirements Engineering: Foundation for Software Quality, Trondheim, Norway, 2007.

[15] I. Kwan, D. Damian, and M.-A. Storey, "Visualizing a Requirements-centred Social Network to Maintain Awareness Within Development Teams," Workshop on Requirements Engineering Visualization (REV), Minneapolis-St. Paul, Minnesota, USA, 2006.

[16] D. Damian, S. Marczak, and I. Kwan, "Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks," 15th IEEE International Requirements Engineering Conference (RE 2007), New Delhi, India, 2007.

[17] K. Schneider and D. Lübke, "Systematic Tailoring of Quality Techniques," World Congress of Software Quality 2005, Munich, Germany, 2005.

[18] T. DeMarco, *Structured Analysis and System Specification*: Prentice-Hall, 1979.

[19] J. Jürjens, *Secure Systems Development with UML*. Berlin, Heidelberg, New York: Springer, 2004.

[20] OMG (2007). Unified Modeling Language (UML), version 2.1.2 (http://www.omg.org/technology/documents/formal/uml.htm). Object Management Group, 2007.

[21] A. Wise, "Little-JIL 1.5 Language Report," Department of Computer Science, Univ. of Massachusetts (http://laser.cs.umass.edu/techreports/06-51.pdf), Amherst 2006.

[22] N. Russell, A. H. M. t. Hofstede, and W. M. P. v. d. Aalst, "newYAWL: Specifying a Workflow Reference Language using Coloured Petri Nets.," Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, 2007.