Towards Emergency Software Development with iTasks

Bas Lijnse
HAN University of Applied Sciences /
Radboud University Nijmegen
bas.lijnse@han.nl

***Abstract*—Software development is not an activity that immediately comes to mind during, or immediately after, a disaster. But when existing systems are disrupted, new (temporary) processes with new information needs emerge. Makeshift systems based on a combination of general purpose tools and creative new uses of existing systems are "developed" all the time. In this paper we take the position that implicitly *emergency software development* is already happening and explore a more explicit approach in the context of a concrete software development framework: iTasks. iTasks is a task-oriented programming framework that was designed for rapid specification of information systems with crisis management applications in mind. Other constraints (rapid deployment, reliable operation and decommissioning) on development of temporary systems during, or after, disasters were not explicitly considered. We argue why we consider a framework like iTasks a good starting point for emergency software development, but also present a research agenda of topics that still need to be addressed to enable task-oriented development in this context.**

***Index Terms*—Emergency Software Development, Temporary Information Systems, Recovery, TOP, iTasks**

I. Introduction

When a disaster leads to a crisis, we generally do not think of software development as a possible activity to mitigate that crisis. Software, despite the emphasis on agility that's been a major trend in recent years, is something that takes time to create, and often relies on complex infrastructure to operate. And although there are many different types of disasters and crises, a lack of necessary infrastructure and a need to resolve the situation as quickly as possible are common attributes.

However, if we broaden the definition of software development to include all activities in which a new solution to a problem is devised by composing a configuration of components that store and automatically process information, we can say that to some extent, software development, or at least system development, is already common in crisis responses. Makeshift information systems consisting of a combination of general purpose tools like spreadsheets, and creative new use of existing systems are developed all the time. Information collection and processing is an integral part of crisis management, and without integrated solutions, one has to work with what is available.

One could argue that the development of new systems during or after serious crises is practically impossible or that the status quo of using of general purpose tools is good enough. In this paper we take the more optimistic position that there is room for improvement: Crises can be contained, and recovered from, faster when custom software tools to support unforeseen tasks could be developed quickly and deployed reliably. We'll develop this idea in the context of a concrete software development framework: iTasks.

The remainder of this paper is organized as follows: In section II we introduce the notion of *Emergency Software Development* by making an analogy with emergency shelter. In section III we discuss the iTask framework as a concrete starting point for facilitating emergency software development, and continue with a discussion of research and development topics that still need to be addressed in section IV. In section V we take a look at related work and conclude with final remarks in section VI.

II. Emergency Software Development

To the best of our knowledge, no software development tools exists that are explicitly designed to address emergency software development. By emergency software development we mean the creation of temporary software systems after a disaster has disrupted existing systems. We consider this to be a software development niche in its own right, with specific constraints and considerations.

To illustrate the difference between emergency software development and regular software development we'll use the common comparison between real estate development and software development. Let's compare the way we approach the basic human need for shelter from our environment in normal (in our case Western European) conditions to how this need is approached in crisis situations.

When we create houses or other buildings to live in, we consider more than just our basic need for shelter. We also want houses that are comfortable, good looking and provide privacy. A consequence of these additional wishes is that construction of such buildings takes more time and effort, and requires costly materials. To justify this investment, we have to recoup the costs over many years. This creates an additional constraint on the durability of materials used, and means that design choices cannot be made lightly because we have to live with their implications for a long time.

When we have a sudden need for shelter, for example because a natural disaster destroyed existing houses, we make different choices. A minimum level of safety is more important than comfort and aesthetics. Fast provisioning of materials and construction are more important than durability. Because of these different priorities, the construction materials and methods that are used to build permanent structures are not suitable to use to create temporary shelters. And although, many materials can be, and are, used to create improvised temporary shelters, solutions specifically designed for these situations exist. These solutions range from simple tents, to fully equipped shipping containers. In these solutions the constraints of emergency situations have been considered in their design. For example, a deliberate trade-off is made between the speed of deployment and the level of comfort such temporary structures provide. Simply put, when compared to a house, a tent is only a very crude type of shelter, but when compared to an improvised shelter of re-purposed materials it can make a big difference.

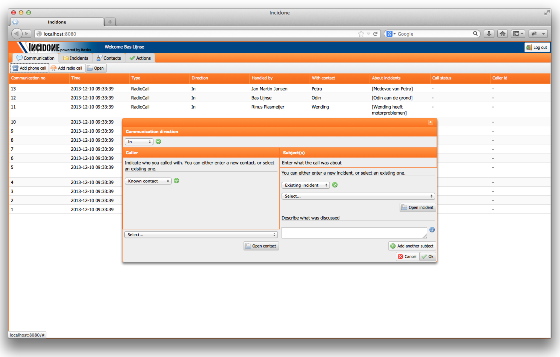
Information collection, processing and distribution plays an important role in our society. To facilitate these activities, we invest in complex software systems and infrastructures. Just like buildings, these are often costly investments that take time to design and build. Also just like buildings, non-functional considerations such as aesthetics and durability are taken into account.

In crisis situations, there is also a need to collect and process information. But unlike the need for shelter, no tools exists to quickly and efficiently create temporary emergency information systems. Sub-optimal solutions based on a combination of generic software (spreadsheets, word processors and communication tools) with pen and paper are still common.

The software development tools to create such emergency information systems would have to have some key properties: First, the specification of the information system must take little time. Second, the operational system should be deployed quickly and work reliably during its limited lifetime and third, it should be easy to decommission the system and transition to a better alternative or more permanent solution. These are challenging requirements, but just as with temporary shelter, it is acceptable to make compromises. In this case on non-functional properties of the information systems.

III. TOP Software Development with iTasks

The observation that no software development tools to create emergency information systems exist only, identifies a problem. A design problem for which no single correct solution exists. Different technologies can be valuable to some extent. In this paper we look at one specific software development framework as a starting point. Although we argue why we expect



Communication no	Time	Type	Direction	Handled by	With contact	About incidents	Call status	Caller id
13	2013-12-10 09:33:39	RadioCall	In	Jan Martin Jansen	Petra	[Medevac van Petra]	-	-
12	2013-12-10 09:33:39	RadioCall	In	Bas Lijnse	Odin	[Odin aan de grond]	-	-
11	2013-12-10 09:33:39	RadioCall	In	Rinus Plasmeijer	Wending	[Wending heeft motorproblemen]	-	-
10	2013-12-10 09:33:39							-
9	2013-12-10 09:33:39							-
8	2013-12-10 09:33:39							-
7	2013-12-10 09:33:39							-
6	2013-12-10 09:33:39							-
5	2013-12-10 09:33:39							-
4	2013-12-10 09:33:39							-
3	2013-12-10 09:33:39							-
2	2013-12-10 09:33:39							-
1	2013-12-10 09:33:39							-

Communication direction

In

Caller

Indicate who you called with. You can either enter a new contact, or select an existing one.

Known contact

Select...

Open contact

Subject(s)

Enter what the call was about

You can either enter a new incident, or select an existing one.

Existing incident

Select...

Open incident

Describe what was discussed

Add another subject

Cancel Ok

localhost:8080/#

Fig. 1. Incidone: An example of a web-based information system made with iTasks

this framework to be useful for this purpose, we don't claim it to be the only, or best solution.

The framework we consider is the iTask framework (iTasks for short). It is a framework for Task-Oriented Programming (TOP) which has been the focus of most of our prior work on information systems for crisis management. TOP is a programming paradigm for developing interactive multi-user systems. As implied by the name, it is about writing programs using "Task" as central concept. The main idea is that computer programs support people in accomplishing certain tasks, or autonomously perform tasks. If we can precisely define the task that needs to be accomplished, or can decompose a complex task into smaller simpler tasks, we should have enough information to generate executable systems that support those tasks. Technical infrastructure of an interactive information system (handling events, storing data, encoding/decoding etc.) can be factored out and solved generically.

Applications in the domains of command and control and crisis management have played an important role in the development of the TOP paradigm and the specification language of the iTask framework ([5, 7]). For example, search and rescue operations of the Netherlands Coastguard were used to test early versions of the specification language ([8]). This case later served as inspiration for the Incidone demonstrator project ([6]) in which a basic incident management system was developed with iTasks (see Figure 1).

There are two reasons why iTasks could be good starting point to consider as a platform for developing emergency information systems. The first reason is that the main concern when writing TOP specifications is determining what tasks need to be done, what information is needed to do those tasks and who can do them. These activities align with processes like sense-making and coordination which need to happen anyway when existing processes have been disrupted. The formal structure that TOP specifications provide to describe and interrelate tasks may even be valuable without its normal purpose of defining an executable information system.

The second reason to consider iTasks is its focus on minimal specification. Concise specification has always been part of the functional and task-oriented paradigms and is therefore a central aspect of TOP programming with iTasks. Compact specification of main concerns is traded for ease of control over details. With iTasks, complete executable information systems are generated from compact specifications of tasks, their relations and the information that is used by them. This is only possible if most of the underlying infrastructure is realized using high-level generic algorithms. For example, user interfaces are (by default) generated automatically based on the structure of information and the structure of tasks in which it is used.

This choice of priorities aligns with the constraints of emergency situations. Although a UI designed specifically for a task by an experienced designer will in most cases be better than what can be generated automatically, a generic algorithm will always be faster than any human designer. In a crisis situation it is preferable to have something that's good enough quickly, than to have some better too late.

IV. Towards Disaster Readiness

In the previous section we have introduced the iTask framework, and argued why we consider it a suitable starting point for developing a suite of emergency software development tools. However, if it would be used in a response today, it would still be of limited value. There is more to software than specification and programming alone. To make iTasks work as an emergency software development platform, we also need to consider the constraints on deployment, operation, and eventually decommissioning of the temporary systems we specify. In this section we discuss those areas in which work can, or should, be done to define a roadmap towards disaster readiness.

Although many of the topics covered in this section will end up on the backlog of the iTasks development team eventually, we present them in the this paper early and publicly as an open invitation for collaboration, to invite criticism and allow our assumptions to be challenged.

A. Specification

The specification language used in iTasks was designed with crisis management use cases in mind. It is expressive and flexible enough to model the complex combinations of tasks that are performed during response operations.

The specification of tasks, and thereby developing the system to support those tasks, was considered to be part of the preparation phase. If we want to be able to specify additional emergency systems *during* a crisis there are more concerns to consider.

1) Third-Party Integration: Although the key idea of this paper is the creation of new temporary information systems in the unstable phase after a disaster or crisis, this does not imply that you should not make use of other tools when they are capable of supporting some the tasks that need to be done. Being able to integrate with other tools and information systems is a way to reduce the scope of the temporary system and therefore the amount of time and resources needed to create it.

At the time of writing, the iTask framework supports different methods for integrating with other systems. There are built-in task definitions for running external programs, calling web-services, and perform low-level communication over TCP/IP. There are also libraries for accessing some popular relational databases.

Support for such integrations has been added demand-driven however. The API's provide only those functions that were needed for specific applications were created. To increase the chances of being able to integrate with a third party system quickly, would require a library of pre-made adapters and utilities. A systematic survey of integration methods to create a library of utilities for easy integration with arbitrary third-party systems has never been done. The design and implementation of such a library, or suite of tools, would be a valuable addition to the iTasks ecosystem.

2) Improvement of User Interface Generation Algorithms: The iTask framework contains algorithms that can generate a user interface for *any* task that processes *any* type of data. However, the usability of these generated interfaces is not always satisfactory. The framework provides a user interface layout language that enables user experience (UX) designers to specify customized interactions for specific tasks. For temporary information systems, such investments will likely be too time consuming in many cases.

If custom UX designs are not possible, the only other option is to find out if improvement of the usability of the generated user interfaces is possible. To find out, two questions need to be answered. First, what are common patterns found in user interfaces of information systems? And second, is it possible to express these patterns using iTasks' user interface layout language?

B. Deployment

The iTask framework has been developed first and foremost as a research platform and as a reference implementation to experiment with the TOP paradigm. For that purpose, the practicalities of deploying applications were never that important. If we want to further explore the development of emergency information systems with iTasks, deployment of applications becomes important. For systems that are intended to be used for a short time, development time is relatively large when compared to the time that the system is used. Therefore the combined overhead of *all* development tasks is also costly and worth looking at.

1) Deployment to Cloud Platforms: If we first consider cases where internet infrastructure is unaffected, the first strategy to reduce the effort of deploying iTask applications during an emergency is to make use of common cloud providers. The main idea would be to prepare automated workflows for deployment to such platforms in advance. When all tasks, from building, to packaging an iTask application as VM or Docker image, to the provisioning of a VM, are automated in advance, the task of deploying an application would be reduced to simply providing credentials for a cloud service. Given that popular platform like Amazon's AWS or Microsoft's Azure have public API's this approach is theoretically trivial, but given the amount of third party tools and services involved could be a challenging software development project. Especially if you also want to consider multiple platforms to increase the chances of at least one of the services being unaffected by the disaster.

2) Deployment to Linux Live Distributions: The other end of the spectrum is the case where no infrastructure is available at all. In those case we have to rely on deployment to local physical machines. The standard approach would be to facilitate building installers or packages for popular operating systems. However, this would mean that multiple versions for different operating systems would have to be created and without infrastructure you can't rely on automatic downloading of necessary dependencies.

A deployment strategy that can circumvents these issues is deployment as customized Linux live distributions preloaded with the system you want to deploy. By this we mean the creation of DVD's or USB thumb drives that contain a complete Linux operating system with the emergency information system already installed and ready to use. This allows you to use any salvaged computer to run your information system.

We have some experience with creating live Linux systems with iTask applications before, but the process of creating the images was never automated. It would be interesting to explore to what extent this process could be automated and integrated with other development tools. Additionally it would be interesting to investigate with what range of hardware this approach can work and what kind of resource constraints apply.

3) Deployment to Mobile Platforms: In the previous sections we have implicitly assumed deployment to average PC or server hardware, but a deployment target we should not ignore is deployment to mobile platforms. Originally iTasks was designed as a framework to create web-applications [10]. For deployment this means that applications have a central server which is accessed using web browser software. This architecture implies that a network with a central server that can be reliably accessed is available. In a situation where networks are unreliable, this architecture is not very robust. An experimental fork of iTasks exists that uses a fully distributed architecture and allows the traditional server-side of iTask applications to be embedded inside Android applications on various mobile devices. This approach enables deployment configurations consisting only of mobile devices.

At the time of writing, this fork is still a proof-of-concept. To make it a viable option for deploying iTasks applications, several non-trivial problems will have to be solved. For example, the common way of deploying mobile applications is through centralized "App Store" infrastructure. Additionally, very specific tool-chains are often needed to compile and package mobile applications. The challenge is therefore to integrate these tools with the iTask framework in such a way that no additional overhead tasks are required from the developer and that no dependencies on "App store" infrastructure are created.

4) Learning and Training: Another topic that requires extra attention in the development of emergency systems is learning and training. This is not a technical deployment problem, but equally important. Learning how to use a new system is necessary if the system is to have any value. In the case of temporary information systems, the overhead of learning to use the system is relatively large. The (perceived) lack of time to learn may be a reason by itself to not use any new system. This means that making iTasks based information systems easy to learn could be critical to the usefulness of iTasks as a framework or temporary information systems, and has not yet been investigated.

C. Operation

Emergency information systems are created to be used in an environment where reliability is a top priority. If the tools can't be trusted to work, the risk they impose is greater than their added value.

1) General Quality of Service: All of the points raised in this paper will, when addressed, result in changes to the iTask framework. An issue that every software project has to deal with, but that should be stated explicitly, is that these changes should not negatively impact the quality of the framework.

iTasks originated as a research prototype, and has always prioritized innovation over robustness. As a result there is technical debt that makes it suitable for demonstrations and experiments, but constitutes a risk for reliable deployments. Even for temporary deployments.

To be able to explore emergency development with iTasks based temporary systems, this priority has to change. At least for the known temporary lifespan of these information systems they should run reliably. If not, the tool that you intended to help you recover from a crisis, will cause more trouble than its worth.

As the focus has shifted from pure exploratory work towards questions of applicability in specific areas, the need for a reliable core iTask system has grown. The development team has therefore adopted industry best practices such as continuous integration. With nearly a decade of evolution of the software in an academic setting though, there is still work to be done.

2) Modification During Operation: Software development projects have a history of taking longer than planned. It is therefore not unlikely that a system that was intended to be only temporary, will be used beyond its planned lifespan and may have to be modified. The standard solution would be to create a new version from the original sources and migrate persistent data. However if we consider the situation where temporary systems are rapidly created in an unstable environment it is not unlikely that original sources and development tools are lost, but the deployed system remains. A solution would be to automatically bundle all sources and development tools that are needed to modify an information system when building and deploying it. How to do this is not immediately obvious. Exploring this approach is an interesting technical challenge and could lead to systems that are more easily patched when there use extends their expected lifetime.

D. Decommissioning

While vendor lock-in may be a common business strategy in commercial software development, it should be actively prevented in emergency systems. Data-entry tasks require valuable resources, and when the data cannot be easily transferred from the system they have to be repeated later. When a system is intended to be temporary, transition to its inevitable successor should be made as painless as possible. The question of how to create information systems that are easy to decommission without loss of data is one worth investigating further.

1) Extend Generic Data Exchange: The simplest way to prevent lock-in is to make data exporting a standard feature of every emergency information system. In iTasks, the high level TOP specifications are turned into executable systems with the help of type driven generic programming. This technique is used to generate user interfaces, provide persistence and to visualize data. It would be interesting to see if the type of algorithms could be used to make it easier to export data in iTasks based information systems to a common data exchange formats. This approach could provide a fail-safe way to export data with little work. The primary question would be which formats to target.

2) Generation of Semantic Context Information for Data Exports: While the ability to export data from temporary information systems to convenient is a necessary condition for migration to their more permanent successors, it is not sufficient. To migrate the exported data to another system, you need to know the origin and meaning of the data. This information can be learned from development documentation, or when that is not available, by examining the information system from which the data was exported. Good documentation about the data makes migration easier, but is costly to produce. With iTasks applications it may be possible to generate parts of this documentation automatically. TOP specifications contain information about tasks, their relation and the information they produce and consume. Therefore it should be possible to analyze where for which tasks data in an information system is used, and which tasks produced certain information. Prior work on the visualization of TOP specifications ([13, 12]) has shown that it is possible to extract visual representations of iTasks specifications similar to BPMN (Business Process Modeling and Notation) that illustrate the composition of tasks and their dependencies. If would be interesting to see if this work could be extended to automatically produce the necessary context information when data is exported from an information system.

3) Automatic Migration: Emergency information systems are created for temporary use. The eventual migration of information to a successor system should therefore be considered a normal and expected part of the system's use. Being able to export all information is only half of the solution. The exported data will have to be imported into the replacement of the emergency system. If this is a system that has no capabilities for importing data (for example when it is a system that was temporary offline, but has been restored), getting the information into new system can be challenge.

There are tools, like for example Selenium or PhantomJS, that allow arbitrary user interfaces to be accessed programmatically. Such tools are commonly used to automate testing of programs. It would be interesting to explore to what extent such tools could be used to automate the migration of the exported data to systems that do not support data importing directly.

V. Related Work

As already mentioned in II, we are not aware of any tools or studies that focus on emergency software development as an explicit activity. This does not mean that there are no research results or technologies that address some of the same or similar issues as those discussed in this paper. Most of the challenges we foresee exist to some extent in all types of software development, but with less priority or urgency. Additionally, to understand the different context that sets emergency software development apart from other software development types, we can build upon studies that have examined past disaster and crisis responses. Especially those that focus on information systems.

To identify recent work that could be of relevance, we have reviewed papers presented at the past three editions of both the ICT-DM [1] and ISCRAM [2] conferences. Both conferences attract research on the intersection of crisis and disaster management with information technology. In the remainder of this section we discuss a selection of those papers in relation to the topics in this paper.

[1]International Conference on Information and Communication Technologies for Disaster Management

[2]Information Systems for Crisis Response and Management

A basic premise underlying the idea of emergency software development, is that during disasters, temporary organizations and ways of working are established. This idea is supported by studies such as [2] that studied the use of artefacts by temporary police "staff" organization to create a common operational picture. Although the main topic of this study is unrelated, it helps to understand the context of ad-hoc information sharing in temporary organizations. Even more directly relevant is a study that looks at the role of both human and technical infrastructure during emergency response[11]. In this study that features interviews with emergency responders, the implicit "development" of customized makeshift information systems by repurposing existing off-the-shelf tools, that leads us to consider a structured approach, is reported as part of emergency response work.

Additionally, studies on crowdtasking such as [1] and [9] are helpful in identifying the types of tasks that occur during disasters that could be supported by temporary information systems. Similarly, papers that showcase the designs of existing software for disaster management can help to identify reusable components, or building blocks. A good example of this is the CBRN command information system in [4].

The last paper to explicitly mention is [3] that compares the development of emergency plans to software development. This paper highlighs the similarity between emergency plans and software as an opportunity to apply software engineering principles to emergency plan development. The paper mentions a "plan execution system" that executes formalized emergency plans. The work is still in an early stage, so there are not enough results yet to draw definitive conclusions. It appears however that the combination of formalized plans with a robust execution system could be viewed as a way of creating emergency information systems, and thus as a form of emergency software development.

VI. Conclusion

When disasters disrupt existing systems, temporary solutions have to be created to manage information while the situation is being resolved. Without dedicated software development tools for the creating of temporary information systems these information systems are commonly created by combining generic software with pen and paper solutions. In this paper we position the rapid creation of such temporary information systems as a specific type of software development, *Emergency software development*, that has its own design considerations and priorities. We have discussed Task-Oriented Programming with the iTask framework as one potential starting point to facilitate this type of software development, but have also identified a set of challenges that need further investigation.

References

[1] Daniel Auferbauer et al. "Crowdtasking: Field Study on a Crowdsourcing Alternative". In: *ISCRAM 2016 Conference Proceedings – 13th International Conference on Information Systems for Crisis Response and Management*. Ed. by A. Tapia et al. 2016.

[2] Erik A.M. Borglund. "The Role of Artefacts in Creating a Common Operational Picture During Large Crises". In: *Proceedings of the 14th International Conference on Information Systems for Crisis Response And Management*. Ed. by Tina Comes et al. 2017, pp. 191–203.

[3] José H. Canós and Diego Piedrahita. "Emergency Plans are Software, too". In: ed. by Tina Comes et al. 2017, pp. 374–379.

[4] B. Jandl-Scherf et al. "Software engineering in the light of evolving standards in CBRN disaster management". In: *2016 3rd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*. Dec. 2016, pp. 1–8. DOI: 10.1109/ICT-DM.2016.7857217.

[5] J.M. Jansen et al. "Web based dynamic workflow systems for C2 of military operations". In: *15th ICCRTS 'The Evolution of C2'*. 2010, pp. 1–19. URL: http://www.dodccrp.org/events/15th_iccrts_2010/papers/094.pdf.

[6] Bas Lijnse. *Incidone Projectbrochure*. http://www.cs.ru.nl/~baslijns/incidone/Incidone-projectbrochure.pdf. 2014.

[7] Bas Lijnse. "TOP to the Rescue – Task-Oriented Programming for Incident Response Applications". ISBN 978-90-820259-0-3. PhD thesis. Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands, 2013.

[8] B. Lijnse et al. "Capturing the Netherlands Coast Guard's SAR Workflow with iTasks". In: *Proceedings of the 8th International ISCRAM Conference - Lisbon, Portugal, May 2011*. Lisbon: National Civil Engineering Laboratory, 2011, pp. 1–10. URL: http://www.iscramlive.org/ISCRAM2011/proceedings/papers/139.pdf.

[9] M. Middelhoff et al. "Crowdsourcing and crowdtasking in crisis management: Lessons learned from a field experiment simulating a flooding in the city of the Hague". In: *2016 3rd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*. Dec. 2016, pp. 1–8. DOI: 10.1109/ICT-DM.2016.7857212.

[10] M. J. Plasmeijer, P. M. Achten, and P. W. M. Koopman. "iTasks: executable specifications of

interactive work flow systems for the web". In: *Proceedings of the 12th international conference on functional programming, ICFP'07*. Freiburg, Germany: ACM Press, Oct. 2007, pp. 141–152. ISBN: 978-1-59593-815-2.

[11] John J. Robinson, Jim Maddock, and Kate Starbird. "Examining the Role of Human and Technical Infrastructure during Emergency Response". In: *ISCRAM 2015 Conference Proceedings – 12th International Conference on Information Systems for Crisis Response and Management*. Ed. by L. Palen et al. 2015.

[12] Jurriën Stutterheim, Rinus Plasmeijer, and Peter Achten. "Static and Dynamic Visualisations of Monadic Programs". English. In: *Implementation and Application of Functional Languages*. ACM, 2016. ISBN: 978-1-4503-4273-5. DOI: 10.1145/2897336.2897337. URL: http://dx.doi.org/10.1145/2897336.2897337.

[13] Jurriën Stutterheim, Rinus Plasmeijer, and Peter Achten. "Tonic: An Infrastructure to Graphically Represent the Definition and Behaviour of Tasks". English. In: *Trends in Functional Programming*. Ed. by Jurriaan Hage and Jay McCarthy. Vol. 8843. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 122–141. ISBN: 978-3-319-14674-4. DOI: 10.1007/978-3-319-14675-1_8. URL: http://dx.doi.org/10.1007/978-3-319-14675-1%5C_8.