



Department of Informatics
University of Fribourg
eXascale Infolab Group

Master Thesis

Crowd-Flow Designer

An Open-Source Toolkit to Design and Run
Complex Crowd-Sourced Tasks

Dani ROTZETTER
dani.rotzetter@unifr.ch
07-208-507
Weiermattweg 10
3186 Düringen

Supervisors:

Prof. Dr. Philippe Cudré-Mauroux
pcm@unifr.ch

Dr. Gianluca Demartini
gianluca.demartini@unifr.ch

Fribourg, February 10, 2014

Contents

Contents	v
List of Figures	v
1. Introduction	1
1.1. Contents of the Thesis	1
1.2. The Problem	1
1.3. The Solution	2
1.4. Thesis Structure	2
1.5. Technical Terms	2
1.6. Related Work	4
1.6.1. CrowdWeaver	4
1.6.2. CrowdForge	4
1.6.3. TurKit	5
1.6.4. Qurk	6
2. Crowd-Sourcing	7
2.1. Introduction	7
2.2. Actors	9
2.2.1. Requesters	9
2.2.2. Workers	9
2.3. Tasks	10
2.4. Incentives and Rewards	11
2.5. Quality Control	12
2.6. Crowd-Sourcing Providers	13
2.6.1. Amazon M-Turk	13
2.6.2. CrowdFlower	15
2.6.3. MobileWorks	16
3. Implementation	19
3.1. Application Scope	19
3.1.1. Usage by Requester	19
3.1.2. Usage by the Crowd	20
3.2. Technology	22
3.2.1. CrowdFlow Designer's Web-Service Approach	23
3.2.2. Back-End	25
3.2.3. Front-End	27

3.3. Architecture	29
3.4. Use of a Crowd-Sourcing Platform	31
3.5. Component Description	31
3.5.1. MacroTasks	32
3.5.2. Workspaces	32
3.5.3. Project Analysis	33
3.5.4. Tasks	35
3.5.5. Post-Processors	36
3.5.6. Splitters	36
3.5.7. Mergers	38
3.6. Database Structure	38
3.7. Flow Implementation	38
3.8. Deployment	41
3.9. Extension	46
3.9.1. Platform Modules	46
3.9.2. Component Modules	48
4. Experiment Evaluation	52
4.1. Experiment One: Traditional versus WebApplication Approach	53
4.1.1. Goal	53
4.1.2. Measures	53
4.1.3. Macro-Task A: Translation	54
4.1.4. Macro-Task B: Artwork Assignment	59
4.1.5. Conclusion	64
4.2. Experiment Two: Crowd-SourcedMacroTask Composition	64
4.2.1. Goal	64
4.2.2. Measures	64
4.2.3. Use Cases	65
4.2.4. Experiment Results	65
4.2.5. Quality	66
4.2.6. Conclusion	67
4.3. General Experiments Feedback and Findings	67
5. Conclusions	70
5.1. Lessons Learned	70
5.2. Limitations	71
5.3. Future Work	72
Glossary	75
A. Bibliography	78
B. Referenced Web Resources	81

C. Experiment Results	83
C.1. Experiment One: Traditional versus WebApplication Approach	83
C.1.1. Macro-Task A: Translation	83
C.1.2. Macro-Task B: Artwork Assignment	103
C.2. Experiment Two: Crowd-Sourced MacroTask Composition	110
C.2.1. Experiment Two A: Multiple Workers	110
C.2.2. Experiment Two B: Single Worker	111

List of Figures

2.1. Crowd-courcing concept	8
2.2. User interface for requesters on M-Turk	14
2.3. User interface for CrowdFlower requesters	16
2.4. User interface for creating a MobileWorks project	18
3.1. Architecture of CrowdFlow Designer	20
3.2. Application scope: flow design by the requester	21
3.3. Application scope: flow design by the crowd	22
3.4. Model-View-Control concept	26
3.5. Model-View-ViewModel concept	28
3.6. Screenshot of the application's flow designer	29
3.7. CrowdFlow Designer application architecture	30
3.8. CrowdFlow Designer platform accounting	32
3.9. CrowdFlow Designer components with their relations	33
3.10. CrowdFlow Designer: visual execution state	34
3.11. CrowdFlow Designer: textual execution state	34
3.12. Database schema of CrowdFlow Designer	39
3.13. Depiction of the items flow	40
4.1. Translation run 1	58
4.2. Translation run 2	59
4.3. Artwork assignments run 1	62
4.4. Artwork assignments run 2	63
4.5. Crowd-source macro-task design: ground truth	66
4.6. Crowd-source macro-task design: result with one worker	67
4.7. Crowd-source macro-task design: result with multiple workers	68
4.8. The influence of the design phase in CrowdFlow Designer	69

Abstract

In science and industry, a new approach for collecting, generating, and treating data has appeared: crowd-sourcing. Instead of running costly and time-consuming tasks by oneself, more and more such jobs are outsourced to workers all over the world, thereby improving effectiveness and efficiency. However, the scope of crowd-sourcing at the current state is mostly limited to small, self-contained and easy assignments.

To address such limitations, in this thesis I propose a new open source toolkit called “CrowdFlow Designer”. Thanks to this system, more complex work flows can be designed and run, allowing requesters to start a new and innovative way of executing and managing large crowd-sourced tasks. They can organize projects by themselves or even let workers elaborate the composition of such “macro-tasks”.

The open and loosely-coupled structure of CrowdFlow Designer enables custom implementations and extensions. It is for this reason ready to support a wide range of tasks, while various modules can be added to work with different crowd-sourcing platforms.

Experiments run on the crowd-sourcing platform Amazon M-Turk demonstrate the substantial benefits of applying CrowdFlow Designer for complex tasks compared to manually managing the flow of data and assignments, revealing the power of this new system.

Keywords: Crowd-sourcing, M-Turk, Task Composition

1. Introduction

1.1. Contents of the Thesis

In the recent years, the term *crowd-sourcing* has emerged more and more. It describes the approach of outsourcing tasks that are impossible to be solved by a machine. *External people* work on these problems in order to gather both quantitative and qualitative data, which would be too expensive or too cumbersome and time-consuming to achieve internally. Numerous projects have already been crowd-sourced, and it is nowadays a common practice to publish such small jobs or “*micro-tasks*” to the public. Typical examples for such assignments are identifying and tagging images or transcribing a video or audio file. On the other hand, it is less usual to rely on unknown *workers* when it comes to managing composite or more complex “*macro-tasks*” like creating a website or translating an entire book.

By implementing an open-source prototype of a web-application called “*CrowdFlow Designer*”, the procedure of composing and managing complex tasks and flows of data between crowd-sourced jobs is simplified. A study will then prove or refute the assertion that the toolkit can be used to crowd-source and organize complex task flows and that this approach improves efficacy and effectiveness compared to manual handling.

1.2. The Problem

Often, crowd-sourced jobs are self-contained: they can be executed independently of any preceding action, and no following task depends on it. Thus, they can easily be scaled up and are suitable for parallelization.

Larger tasks are usually *split* into smaller ones.: the divided jobs can easier or quicker be solved and are thus more suitable to be crowd-sourced. The requester, however, might not know, how to divide the task meaningfully. Furthermore, creating all the defined micro-tasks and publishing them on a crowd-sourcing platform is a tedious, repetitive work.

Another issue is the task *composition* and *flow*. Even if eventually many people are involved in the work, crowd-sourcing is usually organized top-down [Brabham, 2011]. This means that one entity (usually the requester) designs the entire work flow and publishes the task all by himself. Coming back to the previously mentioned example of creating a website, the client would create independent assignments for elaborating or evaluating the web-site design, for translating pages into different languages and for tagging products of an online shop.

1.3. The Solution

CrowdFlow Designer automatizes the work flow of crowd-sourcing and managing macro-tasks. It also offers means that allow the requester to compose and design the flow required to achieve a more sophisticated goal. He can define the parameters needed to publish the tasks on the crowd-sourcing platform. CrowdFlow Designer handles the interaction with the platform and the crowd. Submitted assignments can either be accepted automatically and used in the subsequent step, or they are validated first. But rather than demanding the requester's opinion to verify a result, this step is either accomplished automatically or again *crowd-sourced*, thereby even further reducing his workload and time needs.

At all time, the requester has insight into the current state of execution.

Finally, the overall result is collected and made available as a download.

This new approach saves time and reduces the risk of producing errors when publishing and managing tasks on a crowd-sourcing platform.

The software is composed by a front-end and a back-end. The back-end of the application is running independently on a server and is accessed through internet requests. Hence, the two parts remain highly decoupled. This offers the opportunity to utilize any desired implementation of a user interface that consumes the services of CrowdFlow Designer. On the other hand, a prototype of an end-user program is provided, which acts as proof of concept and reveals the features of CrowdFlow Designer.

1.4. Thesis Structure

At the beginning of the thesis, the idea of crowd-sourcing is presented, and its actors, qualities and issues are introduced. Also, some common crowd-sourcing *platforms* are presented.

The subsequent chapter approaches the proposed system. Here, the technology and architecture of the CrowdFlow Designer are discussed.

An experiment was conducted to verify or disprove the benefits of CrowdFlow Designer. In the third part of the document, these tests are presented, and results are discussed.

The thesis then concludes with a summary and final findings of the development of CrowdFlow Designer.

1.5. Technical Terms

Throughout the document, some abbreviations and technical terms applied in CrowdFlow Designer are mentioned frequently. For this reason, they shall be defined briefly in this section.

In order to make use of web services like a crowd-sourcing platform, third-party software needs a “bridge” that mediates among the involved programs. The most

widely used web technology in this case is the Hypertext Transfer Protocol (HTTP). [Allamaraju, 2010] defines HTTP as follows:

“HTTP is an application-level protocol that defines operations for transferring representations between clients and servers.” [Allamaraju, 2010]

In the discussed environment, the application like CrowdFlow Designer acts as a *client*, whereas the crowd-sourcing platform’s *server* responds to the client’s queries. A *representation* can be seen as information about a *resource*, like a task or a user.

On the server side, “hooks” or entry points provide access to some functions and services. The set of these operations externally available is called “Application Programming Interface (API)”. More precisely, the term API refers to

“[the] set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service” [14]

Besides the API, a service provider must also define *how*, or with which application *protocol* its functions are called. As of today, two technologies are of common practice: Simple Object Access Protocol (SOAP) and Representational State Transfer (REST).

“SOAP is a[n] XML based messaging framework used to exchange encoded information (e.g. web service request and response) over a variety of protocols (e.g. HTTP, SMTP, MIME)” [Rahaman et al., 2006]

With its a rather heavy-weight approach, SOAP targets more complex information flows with sophisticated security mechanisms and verification steps. Typically, complex work flows involve several messages that are sent back and forth between client and server, implying that both are aware of the current *state* of a resource. On the other hand, the protocol can also be applied in a *stateless* environment, in which every request can be sent independently of each other.

SOAP was more widespread in the earlier years of web services and is likely to be replaced by the simpler and more compact REST protocol [2].

REST was introduced in 2000 by the student Roy Fielding and provides means to work with remote *resources*. The idea behind it is that each object or entity can be addressed through a unique Uniform Resource Identifier (URI), thereby making use of the default HTTP requests and methods. A URI is a series of characters with which an entity in the web can be found unambiguously. Fielding describes the approach as follows:

“The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system” [Fielding, 2000]

Using REST, one can interact with underlying objects without need to know the concrete implementation of the components and their functions. Furthermore, REST is a pure *stateless* protocol. For these reasons, server and client remain *loosely coupled*, which simplifies publication and integration of such web services.

1.6. Related Work

In this section, some existing tools and applications are presented that follow a similar approach to CrowdFlow Designer.

1.6.1. CrowdWeaver

CrowdWeaver is a tool that, like CrowdFlow Designer, has the overall goal of creating “integrated task flows” [Kittur et al., 2012a]. The system manages the data flow between tasks. There is also support for tracking and even real-time notification.

The application has a web-based user interface and makes use of templates. Users can vary *crowd factors* like the price or the maximum time to solve a specific task.

What Is Different to CrowdFlow Designer

CrowdWeaver is built on top of Crowd Flower. CrowdFlow Designer, on the other hand, is built on top of an abstract layer and remains open for any crowd-sourcing platform (see section 3.4). CrowdWeaver is thereby also restricted to the API and task types of Crowd Flower. CrowdFlow Designer, however, can be extended to support any kind of task and other component implementations. For example, CrowdFlow Designer makes it possible to let the *crowd* verify assignments of workers, and there is an in-built support for majority voting.

Furthermore, there are some technical differences. Whilst CrowdWeaver uses Ruby as a back-end, CrowdFlow Designer is based on PHP that possibly facilitates the application deployment.

Another point is the strict front end - back end separation that is seen as a strength in CrowdFlow Designer, allowing any custom consuming user interface through a standardized REST API. Finally, CrowdFlow Designer is designed to be run not only by the *requester*. Instead, also *workers* may use the system and partition macro-tasks, as opposed to CrowdWeaver, where the requester controls and designs the entire data flow all by himself.

Despite all this, CrowdWeaver is a system that most resembles CrowdFlow Designer.

1.6.2. CrowdForge

Similar to CrowdWeaver, CrowdForge is a web-based tool that enables the composition of a series of small micro-tasks. The goal is to “support the coordination dependencies involved in complex work done using micro-task markets” [Kittur et al., 2011a]. However, CrowdForge is rather seen as a framework than an application.

CrowdForge builds on the model of MapReduce. In MapReduce, a heavy task is divided into smaller sub-tasks. These sub-tasks are spread among available task-solving components that work in parallel. The thereby resulting outputs are later collected and aggregated into a single result. With this abstraction and due to the fact, that each of

these steps can further be divided and executed recursively, a complex task can endlessly be partitioned¹.

CrowdForge aims at letting the crowd decide, how the macro-task should be partitioned. It is also foreseen to support a variety of quality control features. Another common point is that an aggregation can be done using the crowd or by machine computation. Furthermore, like CrowdFlow Designer, the implemented prototype of CrowdForge is interacting with M-Turk.

What Is Different to CrowdFlow Designer

CrowdFlow Designer allows even more freedom in task definition and data processing mechanisms. For example, mergers and data sources are implemented that allow users to treat data externally by making use of web services.

CrowdForge is written in the Python framework Django - CrowdFlow Designer is built on the JavaScript framework Angular JS.

The implementation of CrowdForge uses pre-defined task templates that are crowd-sourced on M-Turk. CrowdFlow Designer on the other hand remains more open by giving the freedom to create tasks of any type described in section 3.5.4.

While MapReduce is a powerful approach to divide complex tasks, it is possible that this method limits the scope of application. CrowdFlow Designer is less restricted in that any combination of mergers, splitters, micro-tasks and data sources is possible.

1.6.3. TurKit

TurKit is a toolkit that allows M-Turk requesters to define *programmatically*, how tasks are crowd-sourced. This means, that a work flow can be created in a programming language - as soon as a task requires interaction from the crowd, they are automatically published on the platform. Like the previous solutions, this tool is an aid for designing more complex projects than simple, self-contained tasks.

TurKit is implemented as a JavaScript extension that serves as a “wrapper” around the M-Turk API. A programmer may then use the TurKit functions that support asynchronous Human Intelligence Task (HIT) calls and task flow operations. As an example, the *fork* and *join* operations should be mentioned. Requesters can implement an imperative program that include M-Turk task definitions and deployments. TurKit will automatically perform the necessary operations - requesters do not need to interact with the platform directly.

The main feature of using TurKit is that it supports the programming model “crash-and-rerun” [Little et al., 2010a]. This means, that the code that defines the crowd-sourced project or the series of tasks, can be re-run without the cumbersome and time-consuming set-up operations needed to deploy them on the platform.

¹This is valid for the framework - the implemented prototype does not support iteration yet.

What Is Different to CrowdFlow Designer

TurKit was written in Java and makes use of Rhino, an open-source JavaScript interpreter - the back-end of CrowdFlow Designer is written in PHP.

There is no user interface shipped with TurKit, which is just a scripting language toolkit. Hence, the target end users are professionals with a deep knowledge of the API. CrowdFlow Designer is an all-encompassing solution with both front- and back-end support. The user interface prototype can be used without programming skills.

Again, the presented tool is based on M-Turk, whereas CrowdFlow Designer remains crowd-sourcing platform independent.

1.6.4. Qurk

Qurk is a “declarative query processing system designed to run queries over a crowd of workers” [Marcus et al., 2011b]. Its purpose is to use the crowd within database queries for data *filtering* and *sorting*. The authors claim that there are no re-usable implementations available so far that define crowd-sourced relational operations like joins and sorts. Removing duplicates, for instance, is a common crowd-sourced task. Using Qurk, requesters can define such an operation declaratively. The database-related query will then automatically crowd-source the sort operation.

Qurk features batch operations and supports quality control by relying on majority voting. The system helps requesters to reduce costs by optimizing the coordination between database-related operations and crowd-sourced tasks.

For example, assume the goal is to join two sets of images where the same person appears on a picture of each set. In this case, the system applies a crowd-sourced filter that works as follows. The crowd is first asked to indicate the gender of the persons appearing on the pictures. Only then, the join operation is started. Hence, not the cross product of the picture sets is presented, but only images with same-sex persons. This reduces the number of generated micro-tasks.

The user interface for the generated micro-tasks is defined in the Qurk language - for example by coding a small Hyper-Text Markup Language (HTML) table that contains a form field with a Qurk directive.

Qurk can automatically deploy micro-tasks and collect the generated data and is platform provider independent. The program has some similarities with TurKit, but differs in the mode of use. Whilst TurKit is an *imperative programming language toolkit*, Qurk introduces a *declarative query language*.

What Is Different to CrowdFlow Designer

Qurk follows another approach compared to CrowdFlow Designer: the requester must program a query that will then be executed. CrowdFlow Designer, on the other hand, also contains a front-end user interface for non-programmers.

Moreover, Qurk optimizes database operations. CrowdFlow Designer offers a more general solution, allowing to compose entire workflows.

2. Crowd-Sourcing

Since CrowdFlow Designer is a toolkit for crowd-sourcing, a dedicated chapter describes this concept in more detail.

2.1. Introduction

The Oxford dictionary defines the term “to crowdsource” as follows:

“Obtain (information or input into a particular task or project) by enlisting the services of a number of people, either paid or unpaid, typically via the Internet.” [13]

There are some tasks that cannot be solved by machines or artificial intelligence (yet), and hence, people must be hired to carry them out. And this is what we call *crowd-sourcing*. A definition of Jeff Howe, an editor for the computer magazine “Wired”¹, is more expressive:

“Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers.” [8]

Here, an “open call” means that the task is “published” and may be executed by any interests (“potential laborers”). “Peer-production” refers to the act where a self-organizing community produces a common output [Benkler and Nissenbaum, 2006] - it is like a synonym for mass collaboration.

For example, a manufacturer might be interested in the reputation of its products. It is very difficult for a machine to interpret feelings and “extract” an attitude out of a product review. Thus, a crowd-sourced research could be initiated by asking people to read the corresponding product reviews. The answered questionnaire can then be evaluated by the company.

Another common crowd-sourced task is to transcribe a video or audio file. For instance, a recorded political debate or extracts of it are presented to the workers, who

¹www.wired.com

then transcribe the conversation. In the next step, the typed texts - possibly after being *verified* - are merged, forming a complete capture of the discussion. While this task can be crowd-sourced on any platform, there seems to be such a high demand for this service that a company called “CastingWords”² has specialized in this subject and operates a custom crowd-sourced platform for this specific purpose.

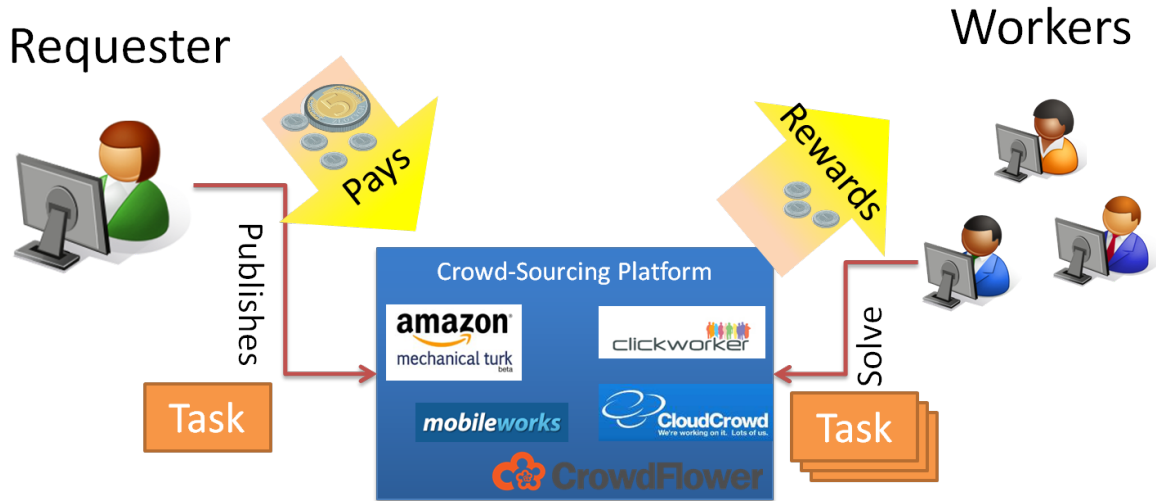


Figure 2.1.: Crowd-courcing concept

As a third example, translation of text shall be mentioned. Traditionally, some employees are hired or an external organization is contacted for the complete translation of an article. Using a crowd-sourcing platform, companies can not only achieve the same goal with fewer funds and maybe quicker. They also have the possibility to *verify* the translations and possibly to let the crowd *improve* the texts iteratively.

An interesting approach on this topic is taken by “Duolingo”³. The platform offers free language learning “courses” to the public. By participating, users learn a foreign language and at the same time translate texts - the latter being a service that is sold to customers.

Sometimes, the jobs are simple and can be accomplished very quickly. In these situations, it is often the large number of executions needed that drives requesters to make use of a crowd-sourcing platform. An interested party might not be able to achieve the goal all by itself in the given amount of time or with the available funds. A typical use case for this situation is tagging or classifying items like products or images.

²www.castingwords.com

³www.duolingo.com

2.2. Actors

To run a crowd-sourced work, there are two parties involved: the *requesters* and the *workers*. Their roles are described in this section - a graphical overview over the crowd-sourcing concept with the actors is presented in figure 2.1.

2.2.1. Requesters

A requester is an entity - which can be a person, a company or an organization - that has the goal to let people execute one or a series of tasks. [Brabham, 2011] emphasizes that the crowd-sourcing procedure is “sponsored by an organization”. “Organization”, however, is rather to be understood as party, since not only enterprises but also private individuals have interests to rely on the crowd in order to hand over tedious jobs.

Requesters are usually kept away from doing the actual work and are only interested in gathering the final output. A crowd-sourcing online application can help to achieve this goal, whereby an efficient, inexpensive and straightforward platform is demanded. Other requesters build their own, custom-tailored tool that more precisely meets their requirements.

2.2.2. Workers

We refer to persons that are actually asked to carry out the crowd-sourced tasks as *workers*. [Ross et al., 2010] has conducted research to find out the demographics of a “typical” worker⁴. The findings are that the affected people have a very diversified background. Whilst in past, there were way more female workers (e.g. 66% in November 2008), the gender gap seems to be closing in the recent years.

A typical worker is about 30 years old. There are some variations between different nationalities, but the vast majority of about 80% range in age from 18 to 44. Big differences can be observed in the annual household income. While US inhabitants working on M-Turk originate from all income levels, most Indian workers have a low income (64% of the surveyed people earned 10'000\$ or less per year).

We can also conclude from the survey that education has no or little impact on the willingness to participate in crowd-sourcing. About two out of five worker are highly educated (Bachelor degree) and nevertheless spend a part of their leisure on the platform.

A further question has revealed that most of the workers (40%) spend between one to five hours carrying out crowd-sourcing tasks, followed by one third of the interviewees investing the entire day (between five and fifteen hours).

The payment is an essential incentive on M-Turk. Only about every tenth worker indicated that the crowdsourcing salary is irrelevant. Depending on the country, for up to a third of the questionnaires, the salary from the crowd-sourcing task was even

⁴This research has been made on M-Turk, see section 2.6.1. With M-Turk’s workers spread all over the world, we assume that the outcome of these studies also applies to crowd-sourcing workers in general.

essential to make their basic ends meet (India). This conclusion cannot be applied to crowd-sourcing in general, since the survey was made on a paid platform.

2.3. Tasks

Often, crowd-sourced tasks are self-contained and short. This makes it easier to fulfill the job quickly and is more attractive to workers, as a long-period commitment might keep interests off from accepting an offer. It is also favorable to have tasks of little complexity, such that workers understand them quickly and do not need to sift through a long lasting learning phase. It has been proven that participants are rather reluctant to open questions or to complex tasks [of Waterloo, 2012].

The crowd-sourcing platform M-Turk discussed in section 2.6.1 defines six types of crowd-sourcing task types [1]. This classification captures pretty well the field of application for crowd-sourcing and is thus described subsequently.

Data Cleansing *Data Cleansing* HITs are useful to improve the *quality* of data. The aim is to verify existing data and to validate results, which may be needed in quality control. Further applications are finding duplicates in a data set and identifying errors like spelling mistakes.

Categorization In a *categorization* task, the workers help to collect metadata for an item, such as classifying an item by choosing the most appropriate option among a list of alternatives. An example HIT would be to assign a gender to a person shown in a picture, or to select a category that best describes the content of an external website. Another possible case is that a worker is asked to rate how appealing a website layout is.

Sentiment When a requester wants to capture the feelings and mood of the public with respect to a product or company, he can conduct a *sentiments* evaluation. In such a survey, users try to find out the general opinions about an object by scanning, for example, a series of blog posts or Twitter⁵ messages. A beverage selling company could for instance make use of this method to analyze, whether a new TV advertisement campaign was well received or not.

Tagging Another way of gathering metadata is to let the crowd *tag* an object. Hereby, keywords are generated that describe the item precisely.

For instance, a company that sells high quality photos wants to provide an online search function, such that potential customers can find an image by keyword or by browsing items by category. It would consume a lot of time for a company to ask its employees to manually type in suitable keywords for every picture. When using a crowd-sourcing

⁵Twitter (www.twitter.com) is a web service to publish very short, mostly news messages.

platform (see 2.6), maybe hundreds of workers tag some dozen pictures each. This way, a vast number of photos is processed in a short time, and costs are reduced⁶.

Create and Moderate Content This type of task is applied when content has to be commented, or if data is generated by the crowd. Such a task can be as simple as finding a website that offers a specific product. A more complex example would be to write a product review, based on specifications spread over different review websites.

A further kind of moderation is *transcription*, in which non-textual content is made machine- and human-readable. A television company could for instance let the crowd transcribe a news program, such that it can be published on the internet with subtitles.

Business Feedback Search engines and results can be tested in the *search relevancy* task. This may help to work on and improve a search algorithm such that it yields more accurate query hits.

Product research can be conducted by evaluating and rating items in *product testing*, allowing manufacturers to receive valuable information regarding their client's opinions about an item. This correlates with the above discussed sentiment analysis.

The market situation, on the other hand, can be evaluated by crowd-sourcing *research* tasks. This can be a handy tool and powerful method when it comes to judging whether a product should be launched or not. A food company, for example, could ask the crowd of a country whether people prefer as a refreshment rather ice creams or cold drinks, which may vary depending on the geography and the country's culture.

2.4. Incentives and Rewards

People join in crowd-sourcing projects for various reasons - different persons work for different reasons.

One type of incentive is financial compensation. The big challenge for the requester here is to define an appropriate wage. If it is too low, it is likely that only few workers accept the task, leading to increased overall execution time to complete the crowd-sourced project. On the other hand, too high salaries attract spammers, which might result in an increased abuse of the system with higher portion of fake or machine-generated and thus unusable answers. However, if the reward remains within a "reasonable" limit, [Buhrmester et al., 2011] has observed no loss in quality for higher or lower paid questionnaires.

Besides this very common motivation, people also participate in crowd-sourcing by altruism. For instance, they fill in questionnaires of research studies for friends or they participate in surveys of governmental organizations. Sometimes, the feeling of being part of a community can also be a motivation.

⁶Usually, such micro-tasks are paid very little money - tagging one single image would perhaps be remunerated by as little as 0.01 USD.

A very handy way of attracting people is to *gamify* pending jobs. When implemented smartly, participators have fun and feel like playing a game rather than working. Similar to the gamification of the task is the case where workers profit from learning effects gained in the course of execution. An example was already discussed in section 2.1.

2.5. Quality Control

While the quantity of gathered data might play a role, a significant aspect in crowd-sourcing is the *quality* of the assignments. There are several mechanisms that can be applied to make sure that the submitted tasks conform to a certain standard. We refer to this process as *quality control*. This is also a way to eliminate spam and machine-generated answers.

Some of the quality control approaches are discussed subsequently.

Qualification Test In this method, workers are only allowed to execute the tasks if they have proven themselves. Hence, before accepting a job, a *qualification test* must be passed. The test consists of pre-defined questions with a known set of correct answers. Qualification tests have the advantage that only domain experts are consulted, and spammers or auto-generated replies are eliminated. However, writing such tests can be tedious. Depending on the type of task, such an assessment can even be impossible to write. Furthermore, when the test is passed, the user might assume that he is “safe” now and fudge the actual job.

Intermediate Test Questions This concept is similar to the qualification test but differs in that the questions are inserted in between the regular flow of execution. The system knows the correct answer referred to as the *golden set*. Again, this reveals untrustworthy assignments but requires precisely elaborated validation checks. And not all jobs are suitable for this type of quality control. For example, it is hardly imaginable how an opinion questionnaire can be tested and verified. Also, a user is not guaranteed to answer the test questions correctly - the question might simply be too difficult. This leads to the user being classified as a machine (“false positive”). He, on the other hand, possibly feels humbled when these checks are too simple.

Honey Pots Requesters may intrude assignments that provoke a specific user reaction that machines do not mimic. When the answer to such a trap question or *honey pot* is incorrect or not usual, the system can assume that the user does not work reliably or that the form was filled out by a robot. In response, the task is either aborted, or the results from that particular user are simply ignored.

This approach seems similar to the previously described intermediate test questions, and the difference is indeed subtle. Honey Pots can be designed in a way that the worker is forced to a specific action instead of giving a straightforward reply. Assuming a website categorization task, requesters could indicate a Uniform Resource Locator (URL) that is

not valid. Or a series of dumb, senseless or empty categories are suggested in a tagging task. While spammers would just choose *any* option, a reliable worker would report an error in this case. This is in contrast to test questions, where a “true” answer does indeed exist.

By applying this quality check, requesters make sure that users are attentive all the time. Moreover, falsifying results can be filtered. But again, it might be difficult or costly to develop a work flow with intermediate trap questions. For example, a text summarization task can hardly be interrupted by a such a test.

Reputation Many crowd-sourcing platforms implement a *reputation* system, in which both requesters and workers are evaluated after each completed assignment. By doing so, before accepting an offer, workers can verify that the requester is reliable and fair, by relying on the feedback of previous workers. On the other hand, requesters may limit their offer to users that are proven trustworthy. While the idea behind is simple and such a system can be set up easily, people can bypass the system with little effort. For example, a company could create some “fake” users that assess the firm with perfect ratings, just to attract workers that then possibly are deceived.

Agreement Between Workers Depending on the type of a job, it is likely that most workers come up with the same result. A typical example is categorization - when for instance an image of a dog is shown, it is expected that the vast majority of users classify the subject as an animal rather than a vehicle. This fact can be harnessed to evaluate a worker by comparing the answer with the results gathered so far: if the reply is in accord with e.g. 90% of all cases, the work is probably serious. Thus the *agreement between workers* is measured. This approach obviates the need for manual assessments. But questions where a majority answer can be expected are possibly very rare. And especially shortly after publishing the task, the set of available answers to compare with might be too small. Furthermore, for free-text or open questions, it is impossible to find a “most common” answer.

2.6. Crowd-Sourcing Providers

A requester can either develop his own crowd-sourcing system, or he can search for an existing and proven crowd-sourcing platform. There are currently several established providers available - three of them are presented briefly in this section.

2.6.1. Amazon M-Turk

Mechanical Turk or M-Turk⁷ is probably the best-known crowd-sourcing platform and is developed and maintained by the online shop and service provider Amazon⁸. By the

⁷www.mturk.com

⁸www.amazon.com

time of writing, Amazon still declares the application as Beta version. There are about 150'000 so-called “HITs” (Amazon’s notion of crowd-sourced tasks) available.

CrowdFlow Designer is built on top of M-Turk (see section 3.4). On that account, this platform shall be presented in more detail.

The system works as follows: a *requester* defines a set of tasks that he or she wishes to crowd-source. Amazon provides a web user interface to design the tasks. Then, the requester specifies some task-related parameters like limiting the maximum allowed time to perform a HIT. He also specifies the reward for an accomplished task and defines restrictions with regard to the worker’s reputation and demographic profile. Once all parameters are set, the task is published.

Workers can browse the list of tasks that are available and which they are qualified for. The list can be filtered and ordered as desired. Once the user is interested in a HIT, he can preview the task. Upon accepting, the user follows the instructions given and accomplishes the task.

A reputation system allows for detecting unfair requesters and workers. Both can deny a HIT if the opposite party does not meet their requirements. Figure 2.2 displays the user interface for requesters when creating a new crowd-source project.

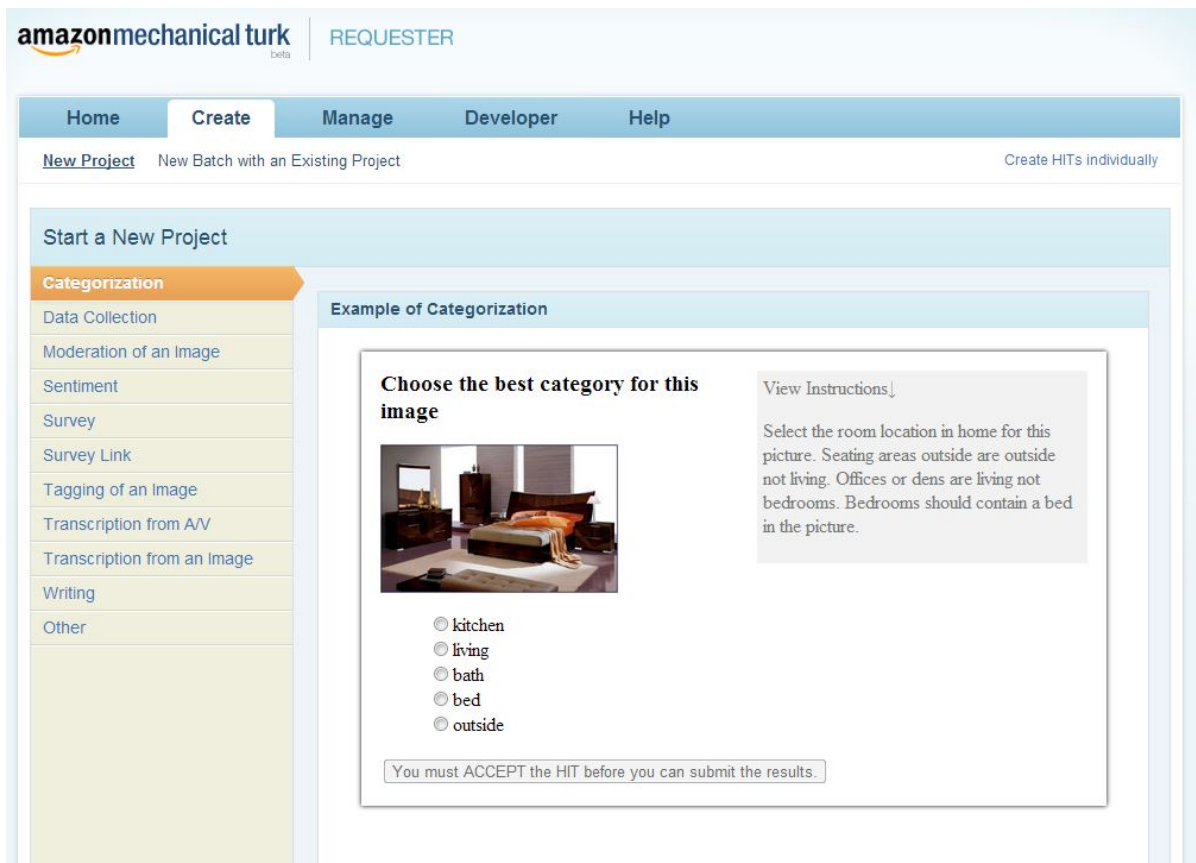


Figure 2.2.: User interface for requesters on M-Turk

In order to participate in M-Turk as a *requester*, one has to be citizen of the United States. There is no possibility for foreigners to directly crowd-source work on the M-Turk platform. Foreigners, however, may instead fall back on a “proxy” platform (see section 2.6.2).

In M-Turk, typical tasks can be solved with little effort and a short amount of time [Kittur et al., 2011a]. They are also self-contained, such that it is easy to “jump” from one task to another.

Besides the above mentioned tasks, M-Turk also provides an online tool to generate work by batch. This method facilitates and simplifies the procedure of publishing a series of tasks by supplying a list of inputs that are transformed into HITs.

M-Turk defines two “super-types” of batch tasks - the *categorization* and the *sentiment project*. Whilst the former is applied to let the users assign an input to one or more answers, the latter is applied to express personal feelings for a specific topic. An example could be to specify how positive or negative a person perceives a citation of a politician.

2.6.2. CrowdFlower

Crowd Flower⁹ describes itself as “the world’s leading crowdsourcing service”. At the time of writing, there were five million contributors from 90 countries, and over one billion tasks had been completed.

On Crowd Flower, requesters define a series of tasks. These tasks are then distributed among their over 50 crowd-sourcing platform partners all over the world. The list of partners, among others, includes

- InboxDollars (www.inboxdollars.com)
- Swagbucks (www.swagbucks.com)
- ClixSense (www.clixsense.com)
- Amazon M-Turk (www.mturk.com)

Here, Crowd Flower acts as a “proxy”, linking the requester with the publishing websites. This has the advantage that requesters only have to deal with administrative issues on this single site. Publishing and service fees, for example, only have to be paid to this platform - the bills from M-Turk resulting from a crowd-sourced assignment, for instance, will be settled by Crowd Flower. The platform is open to any crowdsourcing provider by offering an API that coordinates the communication between the involved parties¹⁰. This powerful approaches not only facilitates the distribution of tasks for requesters, but also bypasses obstacles of the providers like M-Turk participants’ origin restriction(see section 2.6.1).

Screenshot 2.3 shows the user interface for defining a task.

⁹[urlwww.crowdfunder.com](http://www.crowdfunder.com)

¹⁰See [Inc., 2012]

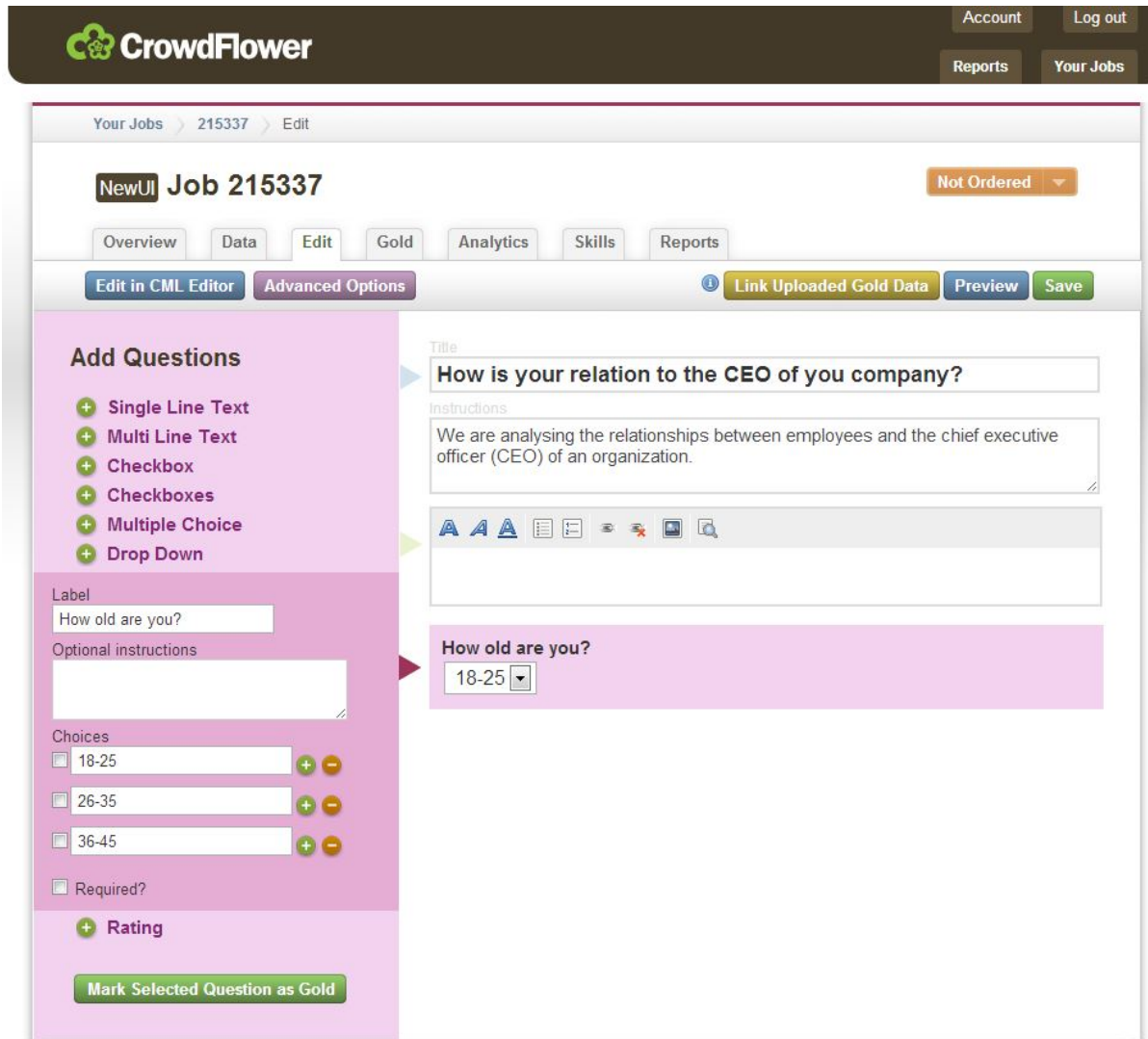


Figure 2.3.: User interface for CrowdFlower requesters

Another feature is the so-called CrowdFlower Markup Language (CML). CML is an Extensible Markup Language (XML)-coded language which can be used to layout tasks. Together with the support of Cascading Style Sheets (CSS) and JavaScript, designing the template is facilitated and allows for the creation of appealing interfaces.

A *dashboard* constantly monitors the progress of the currently published jobs.

As for quality control, CrowdFlower supports the notion of *intermediate test questions* with a golden set (see section 2.5).

2.6.3. MobileWorks

MobileWorks was developed at the University of California Berkeley with the goal to “reinvent” crowd-sourcing [11]. Similar to CrowdFlow Designer, MobileWorks aims at

simplifying complex tasks by breaking bigger projects into smaller jobs which everybody can work on.

The organization emphasizes running a fair and social system by addressing also unemployed people and persons from development countries or with little income - it claims to pay better than the average marketplace¹¹.

MobileWorks focuses on its API called “BrainPower APITM”. A user interface for requesters called “APEye” is available. However, it seems like requesters are encouraged to access the system through internet queries, with APEye only wrapping calls to the internal API. This approach is visible in figure 2.4, where the direct API calls issued by the tool are immediately displayed on the right-hand side of the screen. A PHP and Python library is provided, which simplifies the interaction with the application.

In MobileWorks, quality control is managed by four types of *work flows*. In the **parallel** work flow, the most likely answer is selected among a series of submitted assignments. The approach of the **iterative** work flow aims at achieving high quality by asking workers to improve the answer from an assignment submitted by another user. Finally, the work flows **survey** and **manual** do not feature any *direct* quality control and differ in that the former produces multiple results, whereas the latter results in exactly one assignment that is submitted by a single worker.


On the other hand, a reputation system is maintained, and workers with insufficient scores are imposed a training session [Kulkarni et al., 2012].

Besides this, high reputation members are assigned *manager* status that supervise assignments of other users - a rather new concept in crowd-sourcing.

¹¹MobileWorks suggests 5\$ per worker per hour [12].

APEye

Visual Tool for the BrainPower API™



What's this tool all about?
Using the APEye, you can visually test the MobileWorks BrainPower™ API and generate code that you can use in your own crowd application. Click on the different parameters below to link directly to the documentation.

Post a Project ?

Project ID ?

Instructions (required) ?

The goal is to get statistical data about each state in the USA

Workflow ?

Parallel

Redundancy ?

2

Priority ?

2

Webhooks URL ?

Resource Type ?

Image

Tasks (required) ?

- each task should be a resource (on separate lines)

Enter demographic data

Fields (required) ?

What is your age? t

What is your current job? m Unemployed, Employee, : remove

Add Field

Post a Task

List of POSTs ? [Clear List]

[30/4/2013 22:3:8] Failed: project_0

[30/4/2013 22:3:9] Failed: project_1

[30/4/2013 22:3:53] Failed: project_2

[1/5/2013 17:33:20] Failed: project_3

[1/5/2013 18:26:48] Failed: project_4

POST Details

Data

Click on an item above to see the details here.

Response

Click on an item above to see the response here.

Done trying out the BrainPower™ API? Install the library and start making your own crowd app.

Start building now

Figure 2.4.: User interface for creating a MobileWorks project

3. Implementation

In this chapter, the structure and the technology of CrowdFlow Designer are discussed. Then, the extensibility of the application is presented.

Figure 3.1 aims at providing an insight into the architecture of CrowdFlow Designer. The system is composed of two main parts. Users interact with the application through the *front-end*. This is where the task decomposition takes place and where the dataflow is defined.

The *back-end* is in charge of communicating with crowd-sourcing providers and of managing the database. At the same time, a web component handles the internet requests that come either from the web interface or from third-party applications.

Moreover, a *scheduler* is always active in the background, checking in regular intervals whether assignments from crowd-sourcing providers can be downloaded, processed and a workflow update can be carried out.

As described subsequently, CrowdFlow Designer is composed of several elements. These range from an implemented prototype of the user interface to server-side tools that can be consumed by third parties. This prevents a strict application classification, and thus the terms “toolkit” and “framework” can be used interchangeably in this case.

3.1. Application Scope

It is intended to have a twofold field of application with CrowdFlow Designer. On one side, the *requester* should be able to design a complex work flow using the Graphical User Interface (GUI). On the other side, the same task can also be accomplished by the *crowd*.

The two graphics 3.2 and 3.3 illustrate the two application scopes of CrowdFlow Designer.

3.1.1. Usage by Requester

As described in section 1.2, the goal of CrowdFlow Designer is to facilitate the flow management of complex crowd-sourced tasks. The setup of this flow is essential for the successful deployment. Therefore, an easy-to-use user interface is needed that allows the requester to quickly and precisely design the macro-task.

The user running the application also profits from the overview of both the current execution state and the cost breakdown.

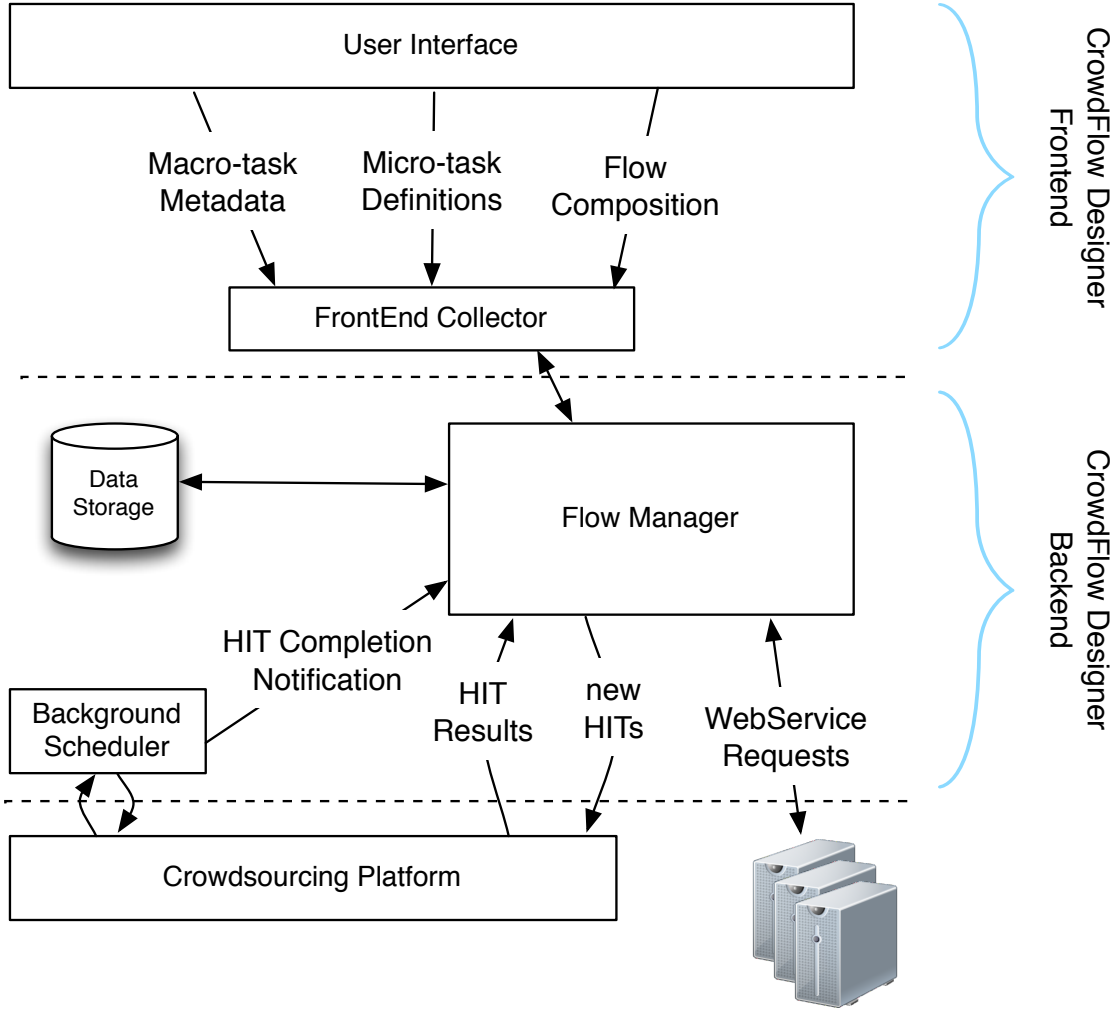


Figure 3.1.: Architecture of CrowdFlow Designer

3.1.2. Usage by the Crowd

Requesters usually are not able or willing to spend a lot of time in the work flow setup. This is where the second software usage can bring relief. Instead of designing the macro-task all by himself, an expert can *crowd-source* this step by letting workers elaborate the macro-task composition. Another not negligible effect that arises hereby is that new ideas emerge. This is due to the fact that other people presumably have a different vision about how to split a job. Hence, it is also imaginable that a requester publishes the macro-task definition multiple times and can then choose the most appropriate version from all submissions. Or, even better and if the budget and time permit, all alternatives are run in an evaluation period. Afterwards, the results are analyzed, and the most effective approach is used for the actual job.

Another imaginable situation is that the requester might not have enough knowledge

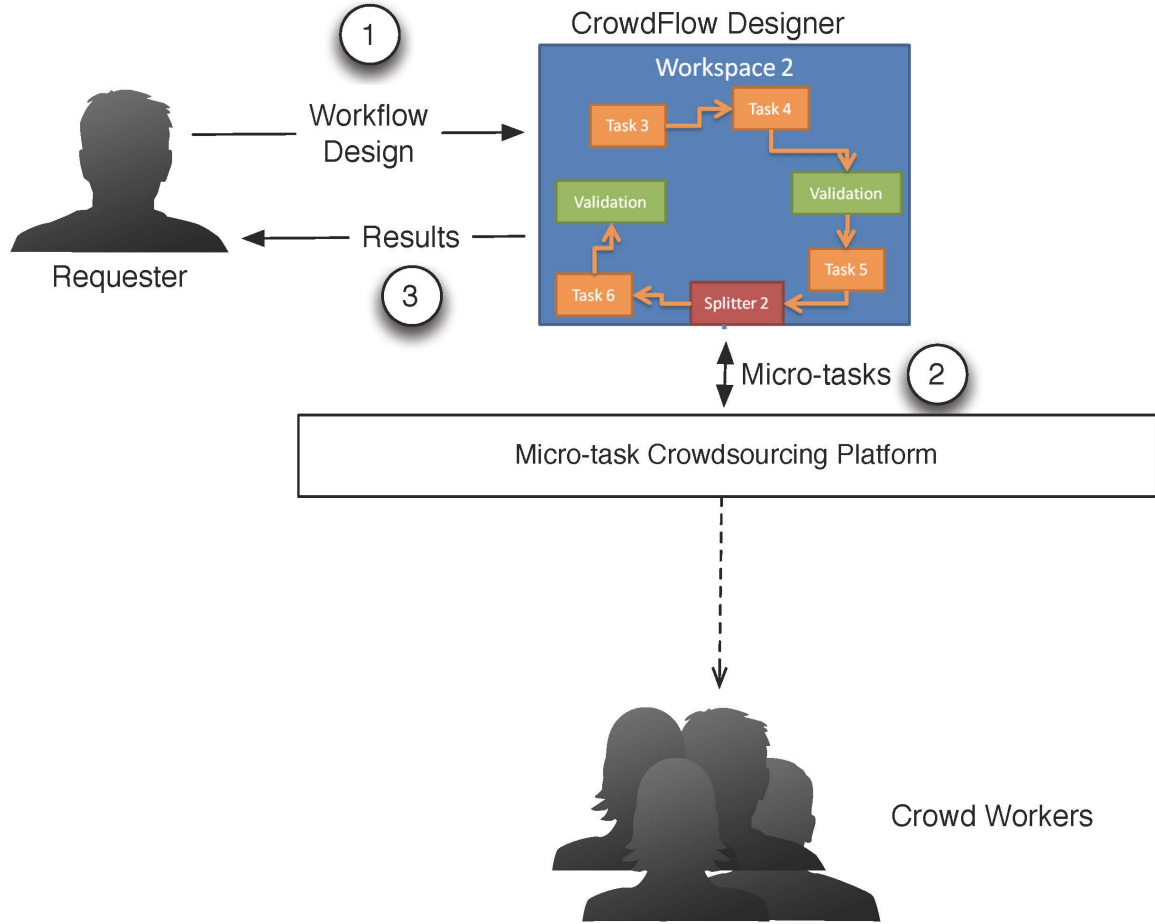


Figure 3.2.: Application scope: flow design by the requester

to split the macro-task meaningfully and instead leave this to a more experienced user from the crowd.

A slightly different variant is that not only one but multiple workers are asked to design the flow *together* sequentially. For example, there could be an expert for micro-task design, whilst another person is acquainted with web services. They thus will be responsible to design their part, whereas a third participant who is experienced in crowdsourcing will finally define appropriate parameters like reward and publication duration. A simultaneous interaction with the system by multiple users is also possible but requires a careful planning.

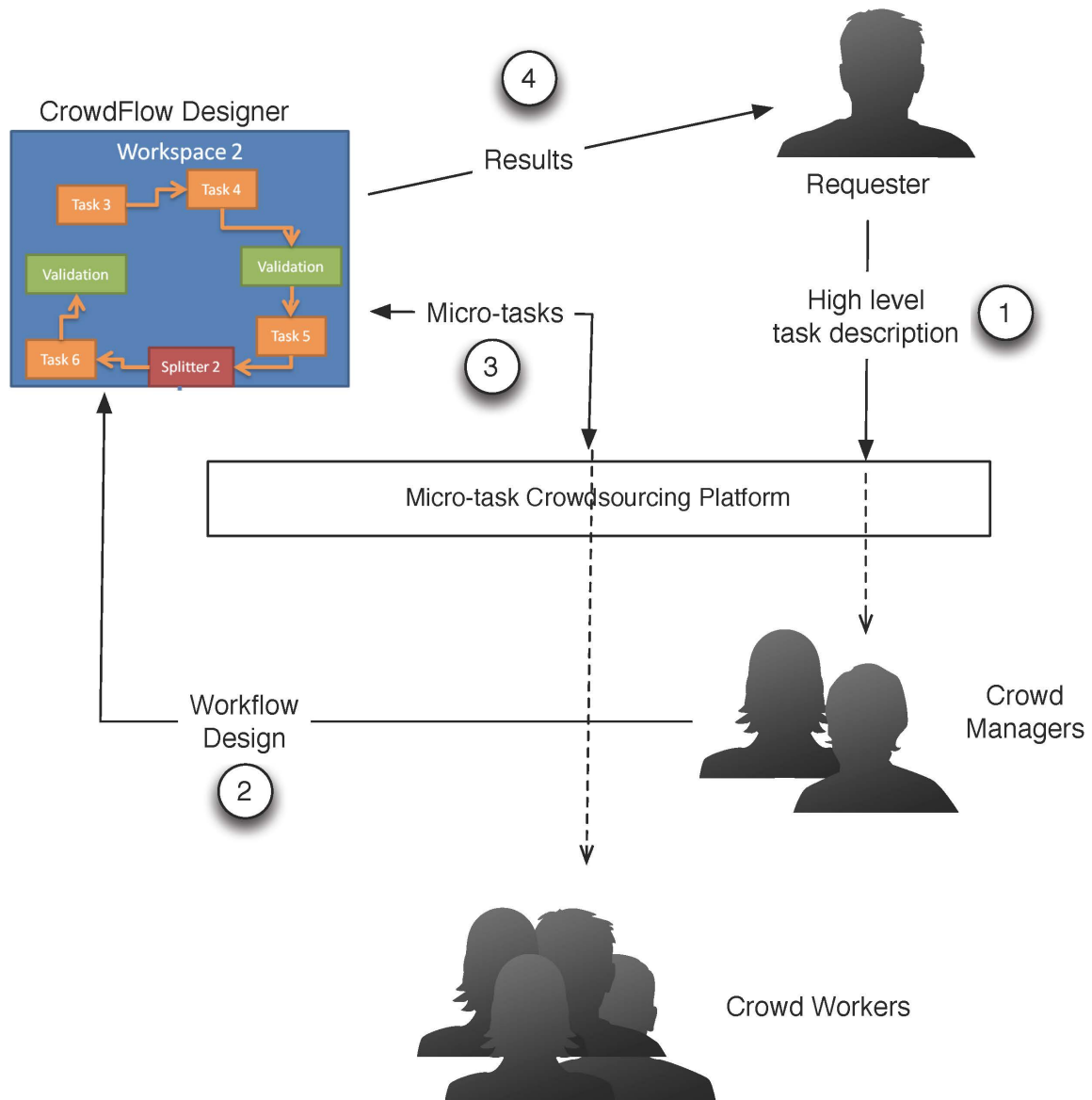


Figure 3.3.: Application scope: flow design by the crowd

3.2. Technology

The clear separation of the client and server side enables to apply two completely different programming technologies, and the front-end version can be implemented anyhow and as desired.

3.2.1. CrowdFlow Designer's Web-Service Approach

In order to present a web-based user interface that can handle the user interaction, it needs access to many functions from the API. Moreover, the API itself is accessed by any client implementation, independent of the technology the latter is made of. The straightforward way to accomplish this is to base upon a standardized protocol that functions over the web channel.

Protocol Among the two most common and established web-service protocols REST and SOAP, it has been decided to rely on REST. This light-weight approach can easier be implemented on both client and server side. It enables a convenient way of communication and is heavily used throughout web applications¹. The CrowdFlow Designer API is thus programmed according to the REST standards. For example, when a macro-task is created, an HTTP POST request is sent from the client to the back-end. Or when an entity is updated (e.g. when a task's description is modified), a PUT message is transmitted.

Data Format To provide a RESTful web service, a data representation format must be selected. Again, a reliable standard was sought. One possibility would have been the XML format. This format features validation, allowing to check data for its correctness before processing it. But XML has a rather inefficient way to represent information - a lot of text is "wasted" to maintain a valid structure. With a lot of objects being stored in the application's database, it seemed that XML would blow up storage². Instead, the more lightweight JavaScript Object Notation (JSON) format was selected. JSON is an open standard text exchange format and popular among web applications [15]. Its advantage is not only that it is parsed and generated with little effort. But it can also more straightforwardly be integrated in JavaScript client applications. A JSON data packet can contain an associative array (i.e. key-value pairs) or a set of arrays. Each of the arrays can again be composed of unlimited, nested sub-arrays, which makes the format very powerful.

To demonstrate the implication of this approach, we consider the case where a new crowd-sourced task is created. For this purpose, the client sends an HTTP POST request, which, in its payload, contains a JSON file with task-related data such as a task description, lifetime and reward. The reply of the web service has always the same pattern - the JSON contains an associative array of three tagged elements:

success A Boolean value specifying whether the request could successfully be answered or not on the server side.

¹See <http://www.programmableweb.com>, a dictionary with a list of public web services. E.g. on 28.09.2013, there were 6368 REST services available, versus 2067 services in SOAP.

²Compression algorithms could reduce the payload of each XML object. However, this would cause a lot of computations. Furthermore, the service consumer would also have to implement an intermediate "translation" module that compresses and decompresses data. This is a contrast to the envisaged goal of providing an easy to use API.

data Optional. Additional data that is sent along with a successful request reply.

errors Optional. May contain errors that have occurred while processing the web request. This element can contain error information even if the request was successfully processed: for example, when processing a series of tasks, some might cause failures, whereas all other objects were unproblematic. On the other hand, the request may fail without reporting explicit errors, although this situation obviously should be avoided.

To demonstrate the CrowdFlow Designer format, another, textual example shall be given. Assume a user wants to create a new workspace, which is the notion of an “approach” to compose a macro-task. The front-end implementation then sends a POST request with the payload as shown in listing 3.2.1.

```
1 {
2   "name": "Translate a book",
3   "description": "A book is given in English. In this
      attempt, all translations should be verified by
      the crowd",
4   "macrotask": {
5     "id": "11",
6     "name": "Translate a book",
7     "description": "translate the book",
8     "date_created": "2013-08-31 19:32:34",
9     "user_id": "0",
10    "type": "macrotask"
11  },
12  "macrotask_id": "11"
13 }
```

Listing 3.1: Create a workspace: JSON request from the client

On the outermost level, the workspace’s metadata are listed. An inner element is used to describe the macro-task, which is on its part a self-contained object with name, description etc.

Once the server has completely initialized the workspace and stored it in the database, the response will look like listing 3.2.1. The *success* value indicates that the request of creating a new workspace could be processed. An identifier was assigned to the workspace (61). The final results of the entire macro-task will be available in the “result” key. In the user interface, each component is visualized with an element that can be dragged freely within the workspace area. Its position must be stored such that when the user re-loads the workspace, he encounters the same layout as when he left. The relative offset of an element is stored in the “pos_x” and “pos_y” key.

```
1 {
2   "success": true,
3   "data": {
```

```
4      "id": "61",
5      "pos_x": "0",
6      "pos_y": "0",
7      "name": "Translate a book",
8      "result": null,
9      "macrotask": {
10         "id": "11",
11         "name": "Translate a book",
12         "description": "translate the book",
13         "date_created": "2013-08-31 19:32:34",
14         "user_id": "0",
15         "type": "macrotask"
16     },
17     "description": "A book is given in English. In this
        attempt, all translations should be verified by
        the crowd",
18     "date_created": "2013-09-29 06:31:49",
19     "type": "workspace"
20 }
21 }
```

Listing 3.2: Create a workspace: JSON response from the server

3.2.2. Back-End

In order to maintain a well-structured back-end, a framework was searched which supports the concept of the Model-View-Control (MVC). MVC is a pattern in software architecture that separates the concerns which an application is made of.

The model represents the structure of the data that is being treated and worked with. Typically, models are objects that are stored in the database or that contain information that is being processed.

Secondly, the view is the rendered part that will be shown to the user. In web programming, the view is normally a web page (or a part of it).

Finally, it is in the controllers where the application logic takes place. A controller coordinates the communication between application parts, “feeds” and prepares the view as desired, mostly by making use of models. It also processes user input.

Figure 3.4 depicts the interaction between the MVC components.

Server Side Framework: Yii framework The technology should also be well supported and widely used. Therefore, the PHP was chosen as server side programming language. PHP is a scripted language that can run on both Linux and Windows[®] servers.

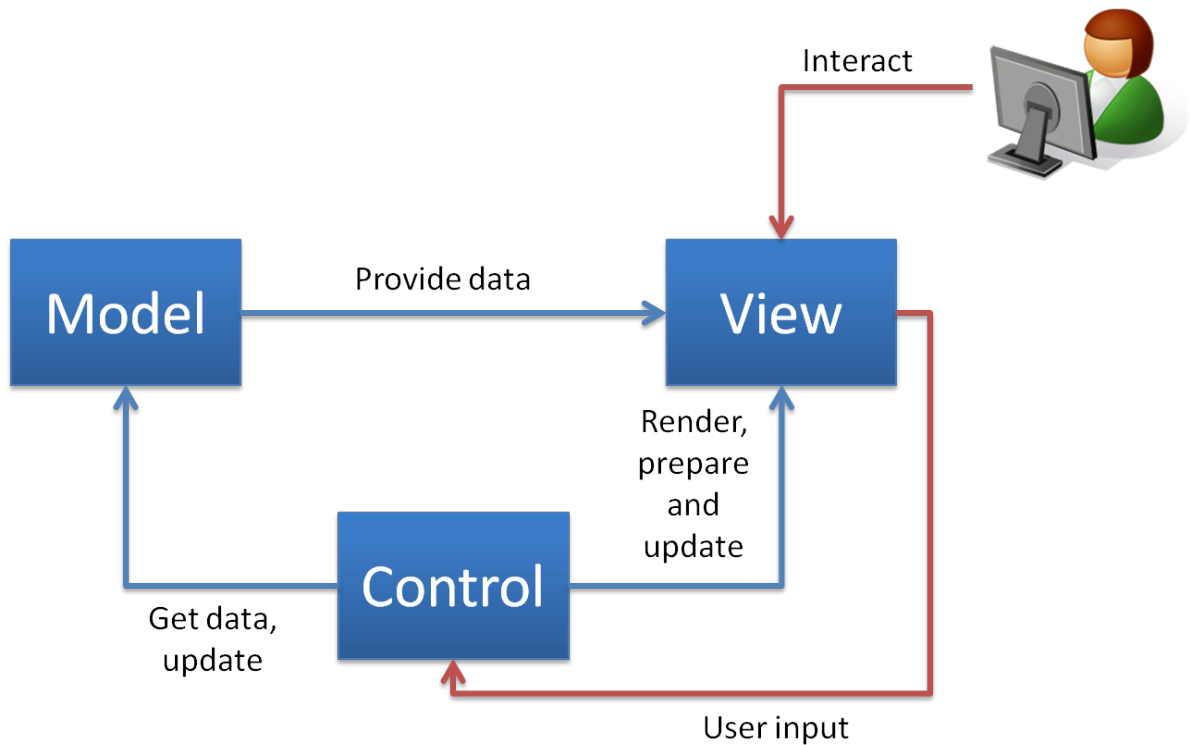


Figure 3.4.: Model-View-Control concept

The code was built on PHP MVC framework Yii framework³. Yii framework is open-source and available for free in version 1 - a second version is currently being developed [10]. It is published under the Berkeley Software Distribution (BSD) license.

Even though based on MVC, the view is not actually used in CrowdFlow Designer's back-end. Instead, this invisible part is supposed to be run on a server, independently from the client implementation (see section 3.3). Yet, a clear structure can be designed with the aid of controllers and models, allowing a clean separation of concerns and a modular application development.

Database Yii framework uses the unified data access format "PHP Data Objects (PDO)"⁴, with which the entire framework remains independent of the database system [9]. As a consequence, any Relational Database Management System (RDBMS) can be employed in CrowdFlow Designer, too. Initially, CrowdFlow Designer was run on the open-source database system PostgreSQL. In the course of coding, however, this system was replaced by the RDBMS MySQL. This substitution during the development phase underlines the loose coupling.

For the database schema, please refer to section 3.6 on page 38.

³<http://www.yiiframework.com/>

⁴<http://php.net/manual/en/book.pdo.php>

3.2.3. Front-End

One goal was to build a smooth web application that supports interactive and responsive interaction with the user. With the desire for immediate action response, web applications tend towards client-side code processing. This means that software instructions and logic are executed on the client's machine. JavaScript is a de facto standard for programming such applications [Gardner et al., 2012]. Besides this, the language is very powerful and supported by all major browsers [18], and allows for asynchronous programming - an essential feature when it comes to responsive applications. On account of this, it was decided to build the client side part of CrowdFlow Designer with JavaScript.

User Interface Framework: Angular JS Angular JS is an open-source JavaScript framework that is being developed by Google Inc. Even though the fairly young project [7] holds a relatively small market share, its usage has been growing in the recent years [16]. A very powerful property of Angular JS is the support for *data binding*. This feature allows to program declarative HTML code that is *bound* to a JavaScript model. This means that as soon as the JavaScript model changes, the modification will automatically be propagated to the web page or the *view* and vice versa. In fact, there is a listener waiting for events triggered by the browsers. The framework then modifies the JavaScript model accordingly. Model changes, on the other hand, are intercepted and again trigger an update of the view [5] [6].

Furthermore, the framework provides a dependency injection facility. Dependency injection is a design pattern that helps to maintain a clean code and application setup by resolving code dependencies, thereby avoiding circular references.

With Angular JS, the Model-View-ViewModel (MVVM) concept can be applied⁵. This design pattern has a similar structure to MVC discussed in the back-end section 3.2.2, but has an important distinction in the model-view link. Whilst in MVC, a controller is responsible for initializing and preparing the view, the view in MVVM remains completely passive: it just *assumes* that the required model data will somehow be present, once the view is rendered. Hence, there is no controller directly interacting with the view. Instead, the view *consumes* or is *bound* to data that is provided. This very data is available through the *view-model*, which acts as a “base” behind the view. The idea behind this is that the view can easily be modified or even replaced by other alternative versions, requiring no modification or update of the other components like the controller.

Figure 3.5 depicts the structure of the MVVM concept.

Even though there are components called “controller” in Angular JS, their task is typically limited to configure the models and functions of the viewmodel. Accordingly, functions that are triggered by user actions are also stored in the *scope*, Angular JS's term for the viewmodel. HTML elements are then bound to scope objects.

It is common to have one controller per application view in Angular JS. The view, however, can be composed of different sub-pages or partial views, for which their own

⁵In fact, Angular JS also supports other coding patterns like MVC, depending on how the web application is being programmed.

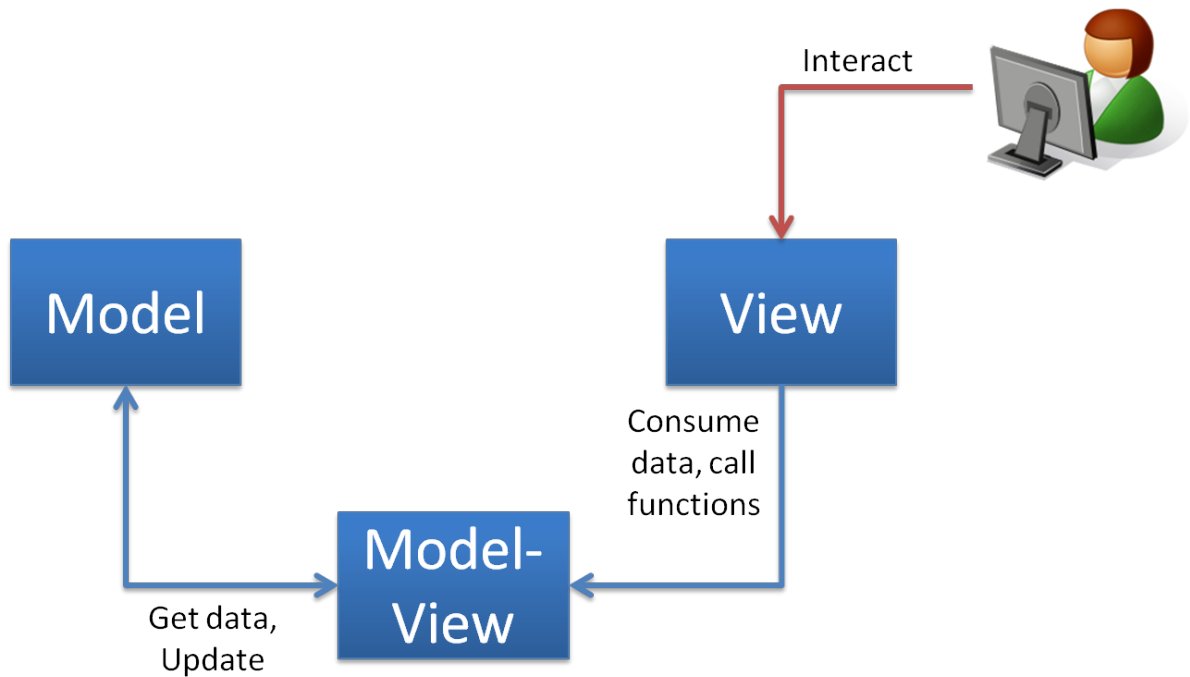


Figure 3.5.: Model-View-ViewModel concept

viewmodels are created.

The JavaScript code files are separated into views, services and controllers. Angular JS has the notion of *services*, which are components that provide “general” functions that are always available and context-insensitive. A typical example of such a service is a “proxy” that handles database interaction for application objects. Likewise, the CrowdFlow Designer models’ Create, Read, Update and Delete (CRUD) functions are programmed as services. Each time a model is retrieved from or updated in the database, the corresponding service sends an asynchronous REST request to the appropriate back-end web service.

In contrast to the widespread usage of jQuery⁶, Angular JS enables a much clearer separation of concerns by keeping all view-related instructions in declarative code. In jQuery, instructions in the JavaScript code, e.g. a controller, modify HTML elements directly. With Angular JS, model-related information is kept in the viewmodel, and the HTML element is adapted “automagically” by the framework.

A typical Angular JS application contains just one “main” HTML page into which different rendered views and sub-views are injected (which on their side are again coded in HTML). This means that a web address does never have to be reloaded throughout the application. Instead, the framework updates or replaces the affected HTML elements, inducing a more desktop-like feeling.

⁶www.jquery.com

User Interface Library: JsPlumb In a web application, when users feel familiar with the controls and patterns they know from the traditional applications, they can act more intuitively and straightforwardly. With JavaScript, interactive interfaces can be implemented to provide desktop-like patterns like drag-and-drop. While coding such elements is simplified by relying on the Angular JS framework, the handling of drawing connections to represent an information flow still is relatively complex and can become tricky.

JsPlumb⁷ is a toolkit that provides exactly these functions. It allows programmers to quicker implement interactive sketching areas that support drawing connections. It leans on patterns known from traditional programs like drag-and-drop. The tool supports many browsers and relies on the vector drawing format “Scalable Vector Graphics (SVG)”.

CrowdFlow Designer makes use of JsPlumb to represent and design the flow of the crowd-sourced tasks. 3.6 gives an insight into this very use case.

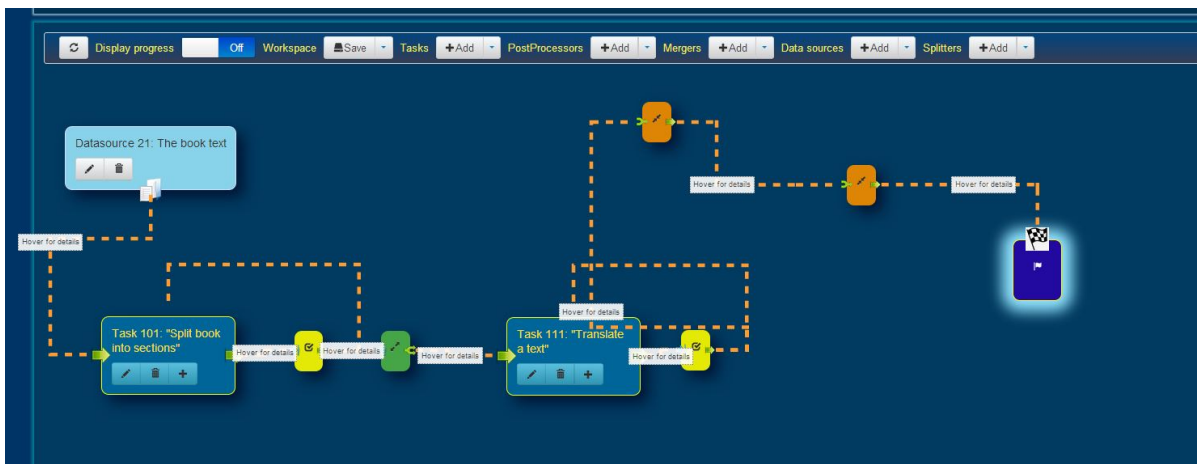


Figure 3.6.: Screenshot of the application’s flow designer

JsPlumb is open-source⁸ and can be used in two license versions: GNU General Public License (GPL)v2 and Massachusetts Institute of Technology (MIT).

3.3. Architecture

With the goal of building a modular project, CrowdFlow Designer is split into a front- and a back-end. Diagram 3.7 depicts the architecture of CrowdFlow Designer.

Deployed Application This is the top-most layer that consists of the front- and back-end implementations.

⁷<http://jsplumbtoolkit.com/>

⁸<https://github.com/sporritt/jsplumb/>

3 Implementation

Implementations For both the front- and the back-end, an implementation is provided. Further clients can be added as desired.

Operations The user interaction takes place in the front-end, consisting of the task definition and the design of the data flow. Finally, the tasks are crowd-sourced. The back-end, in turn, handles data storage and serves as a mediator between the front-end, the database and the crowd-sourcing platform.

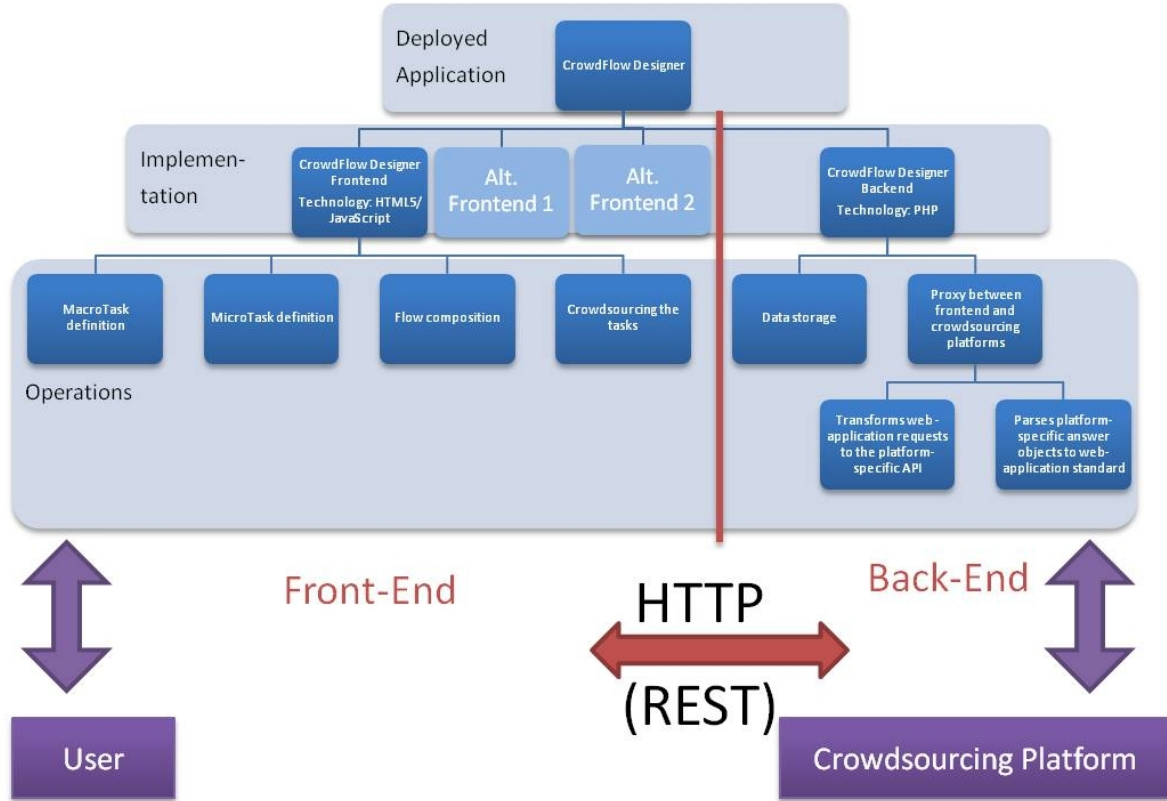


Figure 3.7.: CrowdFlow Designer application architecture

The back-end of the application serves as a library that can be consumed by any programmed application or web service. Therefore, an API is provided. According to the external web requests, the framework interacts with the database and communicates with the crowd-sourcing platforms. Thus, it operates as a “black box”, and the programmer only has to know the basic commands.

On the other hand, a front-end prototype was implemented, which allows end-users to interact with the application. It remains completely decoupled. The provided version serves as proof of concept that demonstrates the link to and interaction with the server side.

3.4. Use of a Crowd-Sourcing Platform

As seen in the previous chapter, there are different crowd-sourcing platforms established and running successfully. CrowdFlow Designer was constructed in a way that any platform from now or in the future is supported by running on an abstraction layer. Instead of directly issuing commands targeting one specific platform like M-Turk or MobileWorks, a more generic set of instructions was defined. In CrowdFlow Designer's back-end, these instructions are converted into appropriate HTTP requests, depending on the currently utilized platform. By doing so, not only a wide range of crowd-sourcing systems is accepted⁹. Also, the application is prepared for potential shifts in the crowd-sourcing business, like if the traditional providers suddenly disappear or are replaced by new companies.

This approach also facilitates accounting. It is foreseen that CrowdFlow Designer maintains one account on each crowd-sourcing platform. Members utilizing the web application will only need a CrowdFlow Designer account. Any fees arising from publications to the crowd-source providers are paid by CrowdFlow Designer in the first place. In order to be compensated for infrastructure and maintenance costs, a slightly higher amount will then be charged to the requester that has created the task on CrowdFlow Designer. Figure 3.8 aims at clarifying the payment flow.

This bookkeeping features the same advantages and avoids the limitations as revealed in section 2.6.2, where the system of Crowd Flower is introduced.

This provider openness comes with a small additional effort: each attachable platform module must be programmed separately. It also has to conform to the CrowdFlow Designer standards and implement all functions that are required to define and manage crowd-sourcing tasks¹⁰.

The platform currently in use is specified in the back-end configuration (see section 3.8).

With the first version of CrowdFlow Designer, only the module for the crowd-sourcing provider M-Turk is supplied. It has been decided to work primarily with Amazon's service because it is one of the most common and widespread provider. Moreover, its API is well-documented and offers a wide range of operations for third-party software.

3.5. Component Description

In order to support a big variety of macro-task compositions, a CrowdFlow Designer project is composed of several *components* that interact with each other. These components are described subsequently.

⁹the restriction being that the platform must support external requests through an HTTP API

¹⁰The methods referred to are listed in the extensibility section 3.9.1.

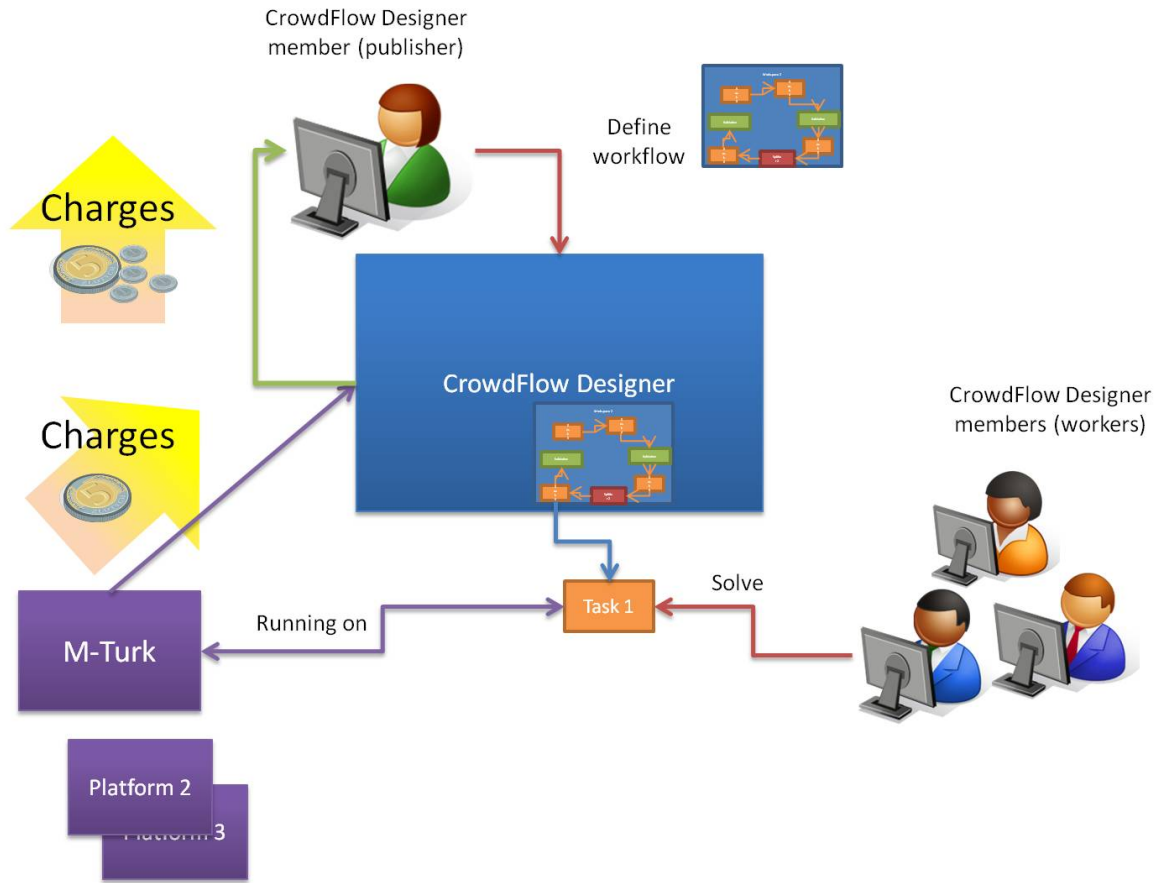


Figure 3.8.: CrowdFlow Designer platform accounting

3.5.1. MacroTasks

A Macro-task is a main goal or job that is crowd-sourced. Its difficulty can range from easy to very complex. But since there is some administration overhead to set up a crowd-sourcing project, straightforward and simple jobs are not suitable as macro task. Rather, these should be crowd-sourced “directly” on a platform.

An example of a macro task would be a book translation. It would cause too much work to process the entire content all at once by one crowd-sourcing participant. Preferable to this is to split it up such that workers can work on individual parts of the book, e.g. on single chapters. The next steps might then consist of verifying each translation, selecting the best among them, and finally joining the translated chapters again. This task was also used in the experiment presented in chapter 4.

3.5.2. Workspaces

A workspace contains the information about how to achieve a given goal. It thus represents one approach of solving a specific macro-task. It is possible to have one macro-task

assigned to multiple workspaces, in which case a macro task is being solved in alternative flow compositions. Items of a flow composition are associated with exactly one workspace.

In figure 3.9, the relations among the components, and the concept of workspace and macro-task are depicted.

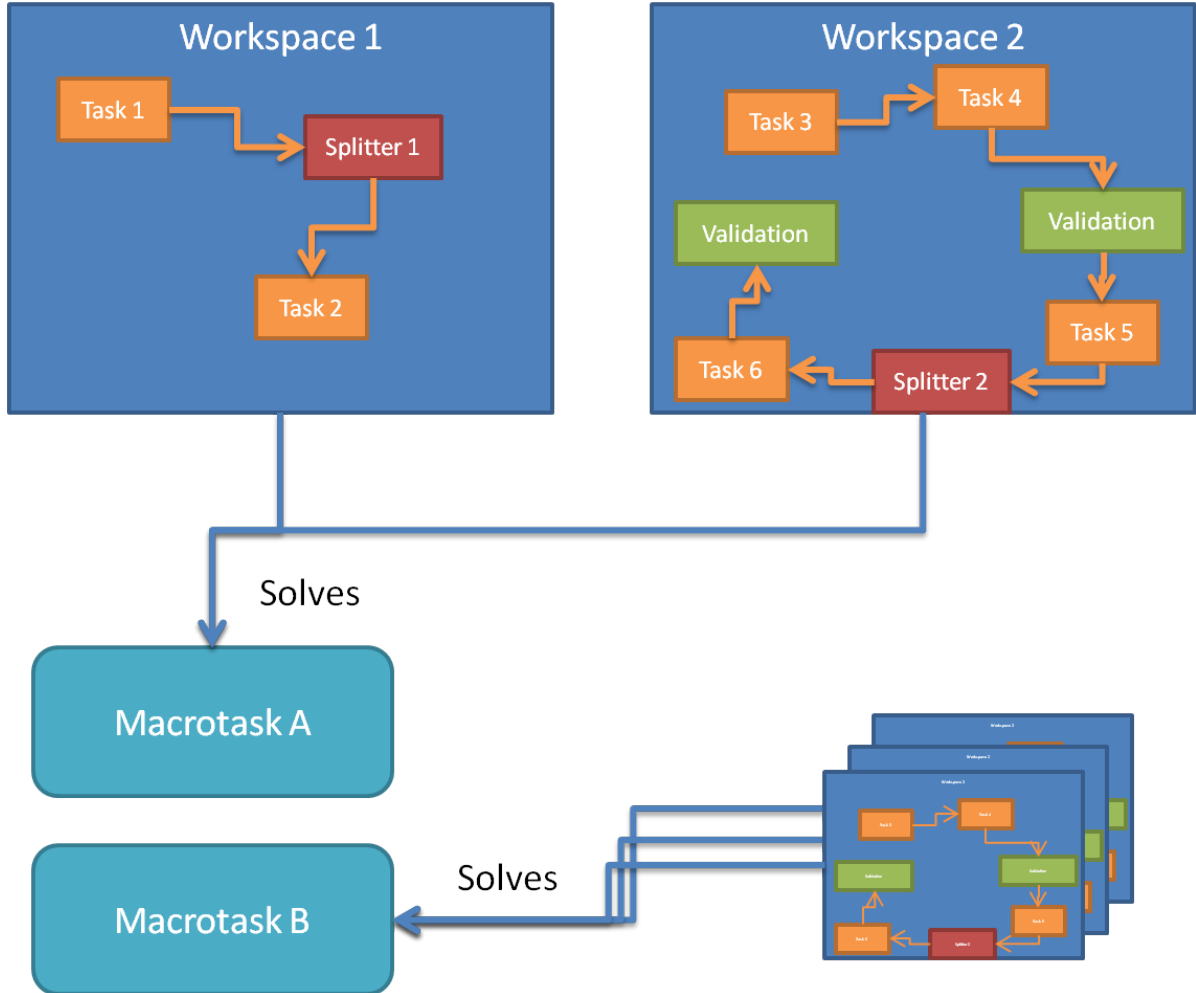


Figure 3.9.: CrowdFlow Designer components with their relations

3.5.3. Project Analysis

Keeping track of the currently published and pending tasks as well as the number of accepted and rejected assignments can be essential for financial analysis and to get an overview of the execution state. For this purpose, CrowdFlow Designer features both a visual outline and an API entry point to access the corresponding data. Therefore, the breakdown can be accomplished through one click or one REST request. Pictures 3.10

3 Implementation

and 3.11 reveal the visual execution state display and the textual breakdown, respectively.

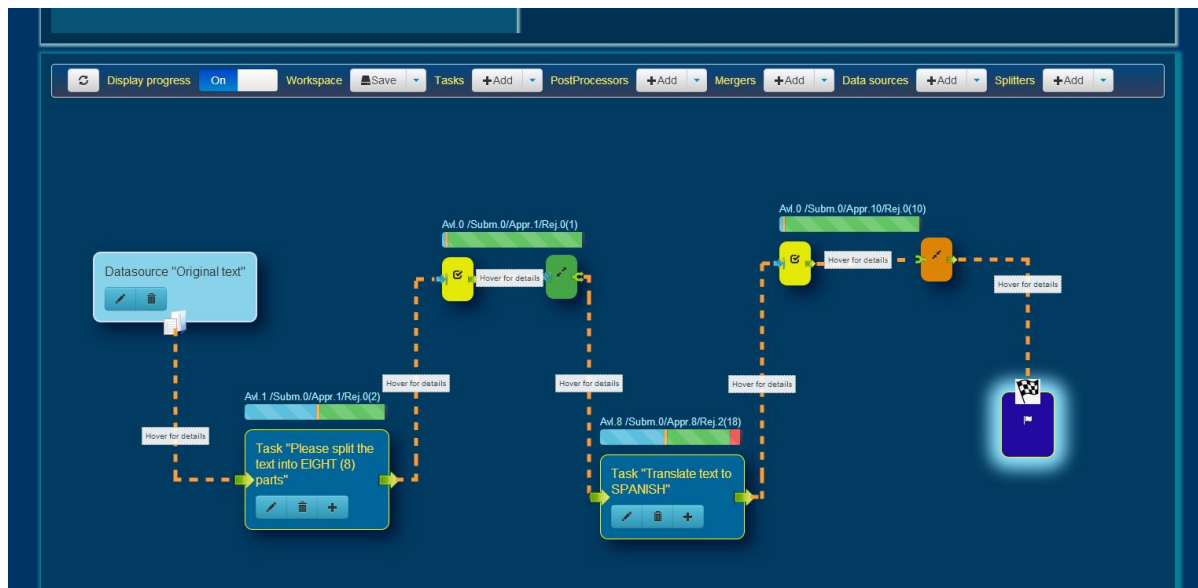


Figure 3.10.: CrowdFlow Designer: visual execution state

Final results of this workspace	
Data about assignments of the crowd-sourcing platform	
Property	Value
Total number of assignments that were approved on the crowd-sourcing platform	20
Total number of assignments that will be published on the crowd	6
Total number of assignments that are pending on the crowd-sourcing platform	0
Total number of assignments that were submitted on the crowd-sourcing platform	22
Data about rewards and payments	
Property	Value
Total amount paid for assignments when all expected assignments are published and approved	4.1
Average amount paid per assignment so far when all planned assignments are executed	0.68
Total amount paid for assignments	11.5

Figure 3.11.: CrowdFlow Designer: textual execution state

3.5.4. Tasks

A task is an instruction that has to be executed by a user. CrowdFlow Designer features a series of available task type modules. Each module requires a crowd-sourcing form page and some processing logic in the back-end. Any type that is not supported in CrowdFlow Designer's initial release can easily be added by implementing the corresponding extension (see section 3.9.2).

To define these modules, a task classification was elaborated. Inspired by the previously seen crowd-sourcing task types of section 2.3, a more coarse categorization has been applied that fits better the CrowdFlow Designer logic.

Input and output properties help to determine a type's characteristics. The result of this classification is a table giving an overview of the possible attribute combinations, shown in table 3.1.

Task Content Types Each possible task is assigned to one of four main types, specifying the content or purpose of the task:

Categorization A categorization task consists of adding metadata to some existing input. This includes a big variety of possible jobs and ranges from simple multiple choice selection up to free text description of an image.

Data Transformation Whenever the crowd is *working on*, rather than *using* the input, a task belongs to the data transformation category.

Data Collection & Research For tasks in which users do research and acquire *new* knowledge, we speak of data collection.

External Task Work that has to be done externally, possibly on another website, is assigned this category. In this case, the platform simply serves as an entry point to a more complex, "out-sourced" operation. Once the external job is accomplished, the user is presented a confirmation code that is copied in CrowdFlow Designer's form, serving as a proof that the work has been done.

To distinct more clearly: *Categorization* means to enrich the content with additional information. *Data Transformation*, on the other hand, means modification or improvement of the input.

The big difference between *Data Categorization* and *Data Transformation* is that, in the former, the input is not changed.

We then further differentiate between the various input and output forms for each task type.

Input Properties The form that is being crowd-sourced can be in any *media type*, like text, image or video.

Furthermore, we distinguish static from dynamic input. An example for a *static* input is a questionnaire, in which always the same questions appear. On the other hand, when

categorizing websites, the input is a URL that varies for each worker - thus the displayed page is *dynamic*.

When the form consists of different options (like a series of categories or keywords), then these are either *ordered* or *unordered*. A typical case for the former is rating - there is a straightforward order for the options “very good” to “very bad”. On the other hand, if an image’s content must be assigned to the category “animal”, “building” or “person”, then these options cannot be ordered in a natural way.

Output Properties As for the input, the output can equally be of various *media types*: whilst a translation task results in a simple text output, the result for a picture search task will contain images.

A distinction can also be made for the *determination* of the result. *Static* form fields contain pre-defined values, like for instance a list of options. Free-form answers, on the other hand, generate *dynamic* output, with each assignment containing another result.

Similar to the input, the worker’s answer can be *ordered* or *unordered*. If he or she has to rate news websites according to the currentness of their headlines, the inspected URLs are ordered (from most up to date to never updated). But if keywords are assigned to an image, these are unordered (when a brown dog is shown, the submitted keyword “brown” has no precedence over “animal”).

Finally, to each form, the worker may produce one or multiple results, depending on the kind of output *mapping*. Referring to the website categorization task, either exactly one answer/ category is accepted, or the requester might allow multiple categories to be selected.

3.5.5. Post-Processors

Sometimes, further treatment of a task’s output is desired, like data *transformation* or result *verification*. A **transformation** might be necessary for the upcoming steps. For example, if users are asked about their salaries in local currency, a post-processor could transform the value to USD. Instead of a list of heterogeneous and incomparable information for each assignment, the work flow can be continued with consistent data.

Alternatively or additionally, it might be desirable to control the quality of the crowd-sourced task and thus to review and **verify** the output. This approval can be obtained either by the requester, or again by the crowd (crowd-sourced validation).

3.5.6. Splitters

Throughout the work flow, splitters might be necessary to divide items or results. A splitter breaks up the incoming data and produces a new series of items that then can be processed individually. This component is required as soon as a single task generates multiple outputs. For instance, a micro-task might ask workers to submit a list of people they spot on a photo. Then, a splitter must be appended that consumes this list and generates a series of new items, i.e. the names of the persons listed. Only now, each person can be processed individually in a subsequent component.

Type	Input Media	Determination	Order	Output Media	Determination	Order	Mapping
Categorization	<ul style="list-style-type: none">TextImageVideoAudio	<ul style="list-style-type: none">Fixed/ non variable (e.g. form fields: always the same question)Varying (may vary for each task, e.g. for each task, a different image has to be tagged)	<ul style="list-style-type: none">Ordered (e.g. when a meal is linked with a country, the countries are listed by continent)Unordered (e.g. website classification: the category options have no particular order)	<ul style="list-style-type: none">Text field (e.g. custom category suggestion)Text area (e.g. multi-line description of an item)Number/ identifier (e.g. id of a category in a list of answers)Binary answer (yes/ no)	<ul style="list-style-type: none">Pre-defined (e.g. select among a list of options)Free text (e.g. custom text to label an image)	<ul style="list-style-type: none">Ordered (e.g. rating - order websites from “good” to “bad”)Unordered (e.g. select three categories for a website: there is no natural order among the categories)	<ul style="list-style-type: none">One-To-Many (e.g. define one label)One-To-OneBinary (e.g. confirm a sentence)Non-Binary (e.g. select one category)
Data Transformation	<ul style="list-style-type: none">TextImageVideoAudio	Varying	-	Text <ul style="list-style-type: none">Text field (e.g. single word translation)Text area (e.g. summarize a text)	<ul style="list-style-type: none">Pre-defined (e.g. list of suggested translations)Free text (e.g. custom translation)	-	One-To-Many (e.g. suggest different translations)
Data Collection & Research	<ul style="list-style-type: none">TextImageVideoAudio	Varying	-	<ul style="list-style-type: none">TextText field (e.g. URL)Text area (e.g. description, definition)Binary answer (yes/ no)	Free text	-	One-To-Many
External Task	URL	Varying	-	Text field (confirmation code)	-	-	One-To-Many

Table 3.1.: CrowdFlow Designer Task Classification

3.5.7. Mergers

The splitter's opposite component is the *merger*. There are three types of merger: *selection*, *input aggregation* and *external processing*.

In a *selection*, among a series of inputs, only one item is picked. An imaginable use case is when a sentence is translated by multiple crowd workers. The different versions could then be delegated to the merger, where the most appropriate translation is selected. A selection can be made by both, the requester or the crowd itself.

For the second type, **aggregation**, all items from the input list are taken into account. Using an algorithm, a single output item will be “constructed” out of this list. Example aggregation algorithms are sum and average. This merging is done by machine.

One aggregation variant is majority voting. This is a common pattern in crowd-sourcing. The idea behind this method is that the more workers give the same answer, the more likely this is the correct one. Obviously, the scope of application is constraint to discrete answer types like a selection, identifiers or precise URLs. On the other hand, there is a very little chance that two workers for instance translate a sentence in exactly the same way, such that it is impractical to apply majority voting in this case.

For **external processing**, data is sent to an external web services that perform the merging in any desired manner. The server's answer can then be used and further processed.

3.6. Database Structure

As mentioned earlier, a crowd-sourcing project is composed of a workspace and a macro-task. Thus, basis of all flow designs are the two tables **Workspace** and **Macrotask**.

There is one table for each item type, like **Task**, **Merger** and **Splitter**. Furthermore, by database design nature, a dedicated table is present for every many-to-many relationship. As an example, a record in the table **Task_Task** indicates that the output of a micro-task is forwarded to and serves as input of another micro task.

Besides the core features, the work flow has to be managed and organized. For example, a **User** table is foreseen for authentication and contribution overview of each participating person¹¹. Diagram 3.12 illustrates the database schema.

3.7. Flow Implementation

As CrowdFlow Designer was built to process crowd-generated data, the information flow is an asynchronous procedure. Once a task is published on a crowd-sourcing platform, the remaining processing must not be interrupted. Instead, the system should continue without requiring an immediate answer from the crowd - it cannot be predicted how soon a worker might accept and execute the task. In this status, the subsequent components are stuck, waiting for submitted assignments that can be processed.

¹¹This table is not used in the first CrowdFlow Designer version that does not support user management.

3 Implementation

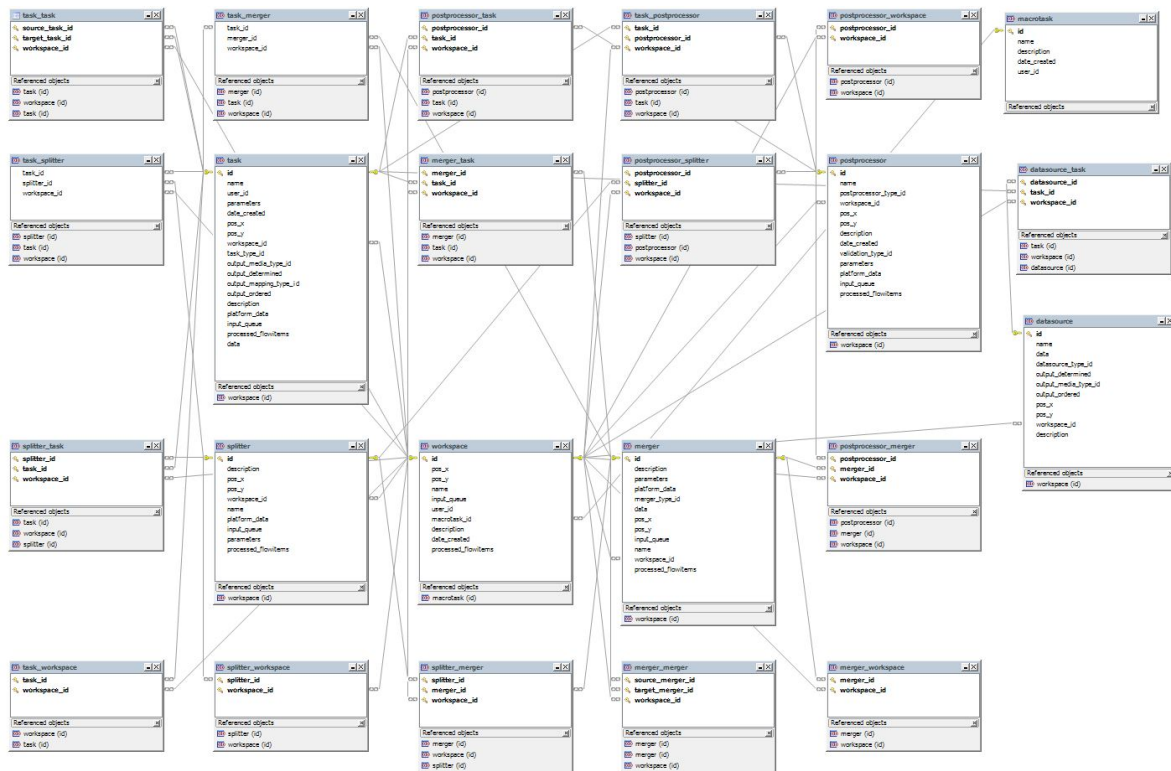


Figure 3.12.: Database schema of CrowdFlow Designer

It has been found that the best way to deal with this situation is to have the notion of an *input queue*, with each item owning a “personal” queue. As soon as its predecessor produces a new result (for example if a worker has completed a crowd-sourced task), it is appended to the component’s input queue. In the next step, the item is called by the application to process its list. What happens then is solely based on the component’s implementation.

To describe the procedure, we shall have a look at a splitter. The splitter’s input queue is filled with the assignments of the preceding crowd-sourced task. Let us assume that the assignment contains a list of keywords describing an image. When a job is completed on the crowd-sourcing platform, it is appended to the splitter’s input queue. In the next iteration step, the splitter processes its input queue. Hence, it removes the topmost element and parses it. The subsequent component is a crowd-sourced validation. Thus, when the keywords have been extracted from the crowd-sourced tagging task, each keyword is added individually to the input queue of the validator. The validator, on its part, is then published on the platform. This flow of items is depicted in 3.13.

We refer to an element in the input queue of a component as a *flow item*. Each item is assigned a unique, randomly generated identifier (ID). Along with the input queue, each component contains a list of processed flow items as well as a list of accepted and rejected assignments. By doing so, CrowdFlow Designer can keep track of every individual item.

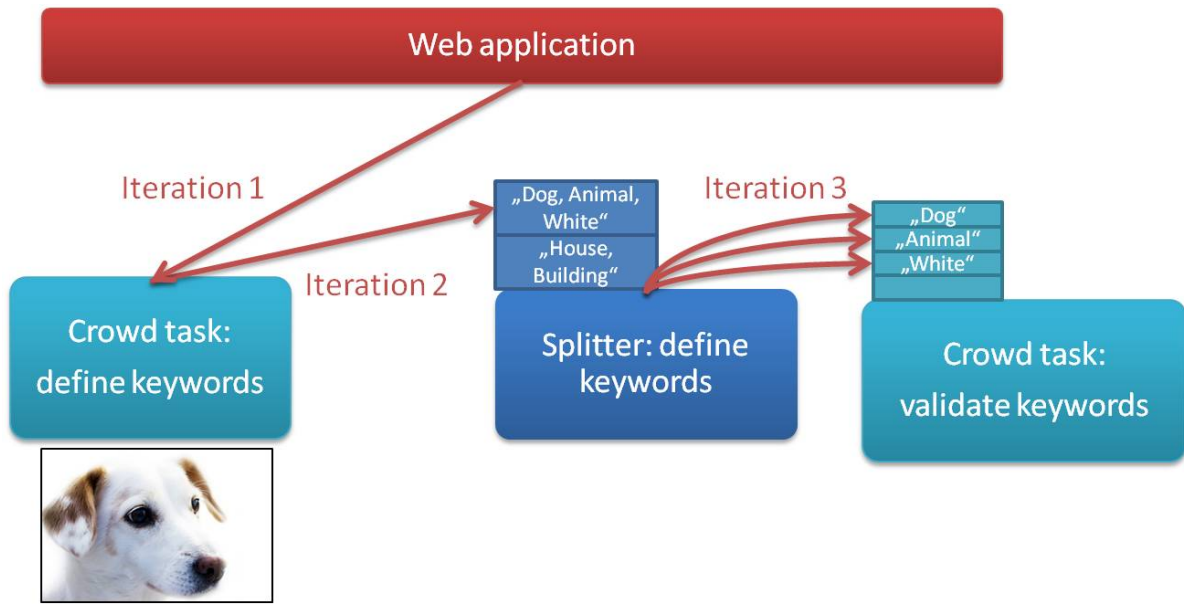


Figure 3.13.: Depiction of the items flow

This helps to prevent items from being processed multiple times or being “lost” in case of a program failure while the input queue is being processed.

Also, this system makes it possible to set crowd-sourced components into “stand-by” mode: if a crowd’s assignment is accepted, the item will be removed from the task’s input queue and forwarded to the subsequent component, where it appears in the list of “processed items”. If the assignment is rejected, the same task can be published again.

As described, the item flow is updated iteratively. For this reason, a background thread is started that regularly refreshes the flow. This action consists of several steps:

- **Accept non-validated components** Each crowd-sourced item that does not undergo validation is accepted automatically.
- **Forward assignments to validators** Each crowd-sourced item’s result that does require validation is appended to the input queue of the validator.
- **Process assignment validations** According to the “judgment” of a validator, the assignments from the crowd are accepted or rejected on the crowd-sourcing platform.
- **Forward accepted assignments** Each assignment that was submitted by the crowd is forwarded. If it was accepted, it is forwarded to the input queue of the subsequent component. If not, the rejected task is re-published.
- **Enable publishable tasks** Each component that has a non-empty input queue is enabled on the crowd-sourcing platform. Each component that has an empty input queue is disabled on the crowd-sourcing platform.

- **Process input queues** Each component processes its input queue. This allows custom data treatment, depending on the item's implementation.

A pseudo-code implementation of this procedure is presented in algorithm 1.

Algorithm 1 Items Flow Implementation

```

1: acceptedAssignments = new List();
2: rejectedAssignments = new List();
3:
4:   ▷ Accept submitted assignment on the crowd-sourcing platform automatically, if
   they do not have a validator
5: for all assignment in crowdAssignmentsWithoutValidator do
6:   acceptOnCsPlatform(assignment);
7:   acceptedAssignments.add(assignment);
8: end for
9:
10:  ▷ If submitted assignments on the crowd-sourcing platform do have a validator,
   forward them to the corresponding validator
11: for all assignment in crowdAssignmentsWithValidators do
12:   forwardToValidator(assignment);
13: end for
14:
15:  ▷ Accept the crowd-sourced validations, i.e. the crowd's answer whether or not an
   assignment is approved
16: for all validator in validators do
17:   for all csValidation in validator.getCrowdsourcedValidations() do
18:     acceptOnCsPlatform(csValidation);      ▷ This only accepts the validation
   assignment, but NOT the assignment that is being validated
19:   end for
20: end for

```

3.8. Deployment

This section gives a short instruction on how to deploy CrowdFlow Designer. The entire code of this open-source toolkit is available for free on GitHub¹²

Pre-Requisites As stated above, the server-side version CrowdFlow Designer is written in PHP. Thus, the server on which the back-end is deployed must support PHP. The back-end is serving HTTP requests from client implementations. Thus, one has to

¹²To get the source code, clone the repository using the URL `git@github.com:danirotzetter/crowdflow_designer.git`.

Algorithm 1 Items Flow Implementation (continued)

```

21:           ▷ Approve or reject assignments that were validated by the crowd
22: for all validator in validators do
23:   for all validatorResult in validator.getResults() do
24:     validatedAssignment = validatorResult.getValidatedAssignment();
25:     if validatorAssignment.isAccepted() then
26:       acceptOnCsPlatform(validatedAssignment);
27:       acceptedAssignments.add(validatedAssignment);
28:     else
29:       rejectOnCsPlatform(validatedAssignment);
30:       rejectedAssignments.add(validatedAssignment);
31:     end if
32:   end for
33: end for
34:
35:           ▷ Forward/ re-publish accepted and rejected assignments
36: for all acceptedAssignment in acceptedAssignments do
37:   forwardToSubsequentItem(acceptedAssignment);
38: end for
39:
40: for all rejectedAssignment in rejectedAssignments do
41:   originalTask = rejectedAssignment.getOriginalTask();
42:   republish(originalTask);
43: end for
44:
45: ▷ Create tasks for each item in the flow that can be published, i.e. that does have a
   non-empty input queue
46: for all item in allItems do
47:   if !item.getInputQueue().isEmpty() then
48:     publish(item);
49:   end if
50: end for
51:           ▷ Process the input queue for each item (i.e. perform item-specific operations, if
   any)
52: for all item in allItems do
53:   item.processInputQueue();
54: end for

```

make sure that the requests pass through any firewall.

As for the database, any RDBMS may be used that supports PDO¹³ (see 3.2.2).

If the prototype of the front-end included in the CrowdFlow Designer distribution is used, then the client internet browser requires enabled JavaScript. Angular JS is compatible with most of the common browsers like Chrome¹⁴, Firefox¹⁵, Internet Explorer¹⁶ and some mobile browsers [3]. However, with the complex user interface and employment of CSS v. 3¹⁷, the testing of the web application was limited to the most recent versions of the Chrome (\geq v. 30) and Firefox (\geq v. 24) browser. Note that Angular JS offers limited support for legacy Internet Explorer versions [4].

Back-End Configuration The CrowdFlow Designer back-end must be configured in the designated configuration file which is located in `<basedir>/backend/app/protected/config/main.php`

This file contains an array containing different *configuration sections*.

The database server details and access credentials are defined in the `db` section. One can also connect to two different databases, depending on whether the access originates from the same machine or from within the same network. This approach might be useful either for testing or to separate (and secure) the internal from a public web application.

Any crowd-sourcing platform module that is used has to be declared in the `components` section. It behaves like the `import` or `using` directives in the Java or C# programming language by instructing the interpreter to also include the specified components when the application is run.

In the section `log`, the logging level and path for log files can be indicated.

Finally, some application logic specific settings can be set in the `params` section:

platform, string At this place, the utilized crowd-sourcing provider is indicated. Currently, one platform, M-Turk, is supported. A partial module for MobileWorks is also implemented. With only the current user balance that can be requested, this module only serves as a proof-of-concept to demonstrate the platform openness (see section 3.4).

payRejectedAnswers, Boolean One can decide to also pay submitted crowd-sourcing assignments that were actually *rejected*. This may be desired in order to prevent bad ratings from users and/ or to still give thanks to the accomplished job of a worker.

sandboxMode, Boolean When enabled, the sandbox mode is used in the attached crowd-sourcing platform modules. For example, M-Turk's API has two different URLs - one for the sandbox mode and one for the "real" mode.

Listing 3.3 depicts the configuration file with the main application parameters.

¹³<http://php.net/manual/en/book.pdo.php>

¹⁴<https://www.google.com/intl/en/chrome/browser/>

¹⁵<http://www.mozilla.org/en-US/firefox/new/>

¹⁶<http://windows.microsoft.com/en-us/internet-explorer/download-ie>

¹⁷CSS is a language used in the web to format and layout web pages.

```

1  ...
2  /* Crowd-Sourcing Platform Modules */
3  'components' => array(
4      'MTurk' => array(
5          'class' => 'application.components.platforms.mturk.MTurk',
6      ),
7  ...
8  /* Database Parameters */
9  'db' =>
10     array(
11         'connectionString' => 'mysql:host=<TODEFINE:URL_TO_DATABASE
12                                >;dbname=<TODEFINE:DATABASE_NAME>',
13         'username' => '<TODEFINE:USER_NAME>',
14         'password' => '<TODEFINE:PASSWORD>',
15     ),
16     ...
17     /* Logging Parameters */
18     'log' => array(
19         'class' => 'CLogRouter',
20         'routes' => array(
21             array(
22                 'class' => 'CFileLogRoute',
23                 'levels' => 'error, warning, trace, info, debug',
24                 'enabled' => true,
25             ),
26             array(
27                 // Log special analysis events to a custom file
28                 'class' => 'CsAnalysisFileLogRoute',
29                 'levels' => 'error, warning, trace, info, debug',
30                 'enabled' => true,
31                 'categories'=>'Analysis',
32                 'logFile'=>'analysis.log'
33             ),
34         ),
35     ),
36     ...
37     'params' => array(
38         'platform'=> 'MTurk', // Specifies the to be used crowd-
39                               sourcing platform. Possible values: 'MTurk', 'MobileWorks',
40                               custom modules etc. This value case-sensitive!
41         'payRejectedAnswers' => true, // Whether the answers from the
42                                       crowd should be paid even if the answer is not accepted
43         'prefix' => '', // A prefix that will be added to each
44                           published task
45         'addWebAppData' => false, // Whether web-application specific
46                                   item data should be added to the title of the crowd-
47                                   sourced task (i.e. item type and ID as well as the
48                                   publication date)
49         'sandboxMode' => true, // Whether instead of using the 'real'
50                                 platform with payments, the sandbox mode should be
51                                 enabled for development & testing

```

```

43     'backendBaseUrl' => '<TODERFINE:URL_TO_BACKEND>/index.php', //
    'The absolute URL to the web application back-end'
44 ),

```

Listing 3.3: Extract of the main application configuration file

Front-End Configuration If the shipped GUI of CrowdFlow Designer is deployed for the front-end, some minor parametrization have to be carried out. As the crowd-sourced tasks require an URL to which the forms are sent, an absolute REST service address has to be set. This is accomplished in the file `<basedir>/frontend/js/services/ConfigService.js`.

The corresponding address must be replaced in the lines listed in listing 3.4.

```

1  var ConfigService = function (\$http, \$q) {
2      this.BACKEND\_SERVICE\_HEROKU = '<TODERFINE:PATH\_TO\_SITE\_
    _BACKEND>/app/index.php/';
3      ...
4  };

```

Listing 3.4: Front-end configuration

Crowd-Sourcing Platform Configuration: M-Turk As stated in 3.4, the module for running CrowdFlow Designer on M-Turk is implemented. In order to operate on M-Turk's API, the requester's credentials must be available. These are defined in `<basedir>/backend/app/protected/components/platforms/mturk/MTurkApiTools.php`.

On M-Turk, the credentials are not of the form user name - password, but instead make use of a so-called "Secret Access Key" and "Access Key ID". These strings are to be set in the lines shown in 3.5.

```

1  private function getSecretAccessKey()
2  {
3      return '<TODERFINE:SECRET_ACCESS_KEY>';
4  }
5
6  private function getAccessKeyId()
7  {
8      return '<TODERFINE:ACCESS_KEY_ID>';
9  }

```

Listing 3.5: M-Turk configuration

Background Scheduler To ensure proper working, the items that are being processed in the application must be updated regularly. There are several steps involved in this procedure (see section 3.7). Thus, one has to make sure that the application's state is refreshed frequently. This can be achieved by availing oneself of the supplied *update script*. This component automatically triggers the required steps. Thus, on the server, a

Cron Job (Linux) or *Scheduler Task* (Windows Server) must be installed to launch the file.

On a Linux machine, the cron job editor is opened by issuing the following command

```
1 $ crontab -e
```

Listing 3.6: Launch Linux cron job editor

Then, a new entry specifies that the PHP file is run every minute:

```
1 #min hour day month weekday command
2 $ */1 * * * * /usr/local/bin/php /<basedir>/cron/cronrun.php
```

Listing 3.7: Instruction to run a Linux cron job each minute

Documentation on how to modify this command according to the publisher's needs can be found online (e.g. <https://help.ubuntu.com/community/CronHowto>).

On a Windows server machine, a scheduler can be created in the terminal with the command

```
1 schtasks /create /sc minute /mo 1 /tn "WebApp" /tr <basedir>\cron
   \cronrun.bat
```

Listing 3.8: Instruction to run a Windows task each minute

Please refer to the Microsoft help pages for further customization of the scheduler (e.g. <http://technet.microsoft.com/en-us/library/cc725744.aspx>).

The PHP file that is launched by the scheduler¹⁸ contains the instructions to run the controller that is responsible for the flow refresh. Depending on the machine setup, the path to the bootstrap and application configuration files (`yii.php` and `cron.php`) might also have to be adjusted. Besides, the same configuration as stated in paragraph 3.8 must also be performed in the cron job's configuration file¹⁹.

3.9. Extension

3.9.1. Platform Modules

In this paragraph, it is shortly explained, how an additional crowd-sourcing platform can be integrated into CrowdFlow Designer.

A custom platform module must comply with the standards defined for CrowdFlow Designer. This means that the JSON data format presented in section 3.2.1 must be parsed.

Throughout the application flow, several times, the crowd-sourcing platform is addressed with information requests or task publication orders. The methods that the platform module has to provide are listed subsequently.

Note that parameters with the name `$modelOrId` can be either a task object from the database or a platform-specific identifier. This openness is needed since, in some

¹⁸<basedir>/cron/cronrun.php

¹⁹<basedir>/backend/app/protected/config/cron.php

situations, CrowdFlow Designer's functions are called over REST with a platform task identifier (e.g. with the HIT ID in the case of M-Turk), while the system internally works with data objects.

A parameter named `$model` represents a crowd-sourced CrowdFlow Designer item that is stored in the database. It contains platform-related attributes like creation date, the task expiration date and the running status.

The running status describes the task's execution state on the platform. The possible values are "running", "unpublished" and "disabled".

Assignment statuses indicate whether an assignment by a crowd-sourcing worker was already handled or not. Assignments can be "submitted", "approved" or "rejected". If an assignment is not found on the crowd-sourcing platform, it is of status "notfound".

The return type describes the `data` part of the JSON-encoded answer object described in section 3.2.1.

`public function getFormBaseUrl(), returns string` In CrowdFlow Designer, the forms that are presented to the workers on the crowd-sourcing platform are HTML-encoded, containing the `form` element. This method returns the URL that must appear in the form's `action` parameter. This means that the URL represents the address to which the completed crowd-sourcing form must be sent.

`public function getAccountBalance(), returns string` Returns the balance that is available on the crowd-sourcing platform to publish further tasks.

`public function getAllTasks(), returns array` Returns an array of all CrowdFlow Designer tasks that are currently published on the crowd-sourcing platform.

`public function getExecutionInformation($platformTaskId), returns array` Returns an array of crowd-sourcing related information for the specified platform identifier. The values returned contain "status", "CreationDate", "ExpirationDate", "ReviewStatus" and "MaxAssignments".

`public function publishTask($model, $maxAssignmentsOverride)` Publishes the task on the crowd-sourcing platform. Optionally, the number of requested assignments can be overridden with the `submitted` parameter.

`public function validateAssignment($assignmentId, $approve, $message)` The specified assignment with the platform-specific identifier is validated according to the `$approve` (Boolean) parameter. If a message is given, then this text should be used inform the worker why an assignment was rejected or accepted.

`public function getTaskResults($modelOrId, $assignmentStatus), returns array` Retrieves all assignments that were submitted for the given crowd-sourced task. The

`$assignmentStatus` (string) parameter may be applied to restrict the assignments by their status.

`public function getAssignmentStatus($platformResultId), returns string` Return the status of the assignment supplied as a parameter.

`public function parseResult($model, $platformResult), returns object` All results of a platform's task are represented in a special, platform-specific way. Throughout the application, CrowdFlow Designer may encounter platform-encoded assignments. In order to process these, they have to be converted into CrowdFlow Designer's specific assignment format. This transformation is done in this method. The `$model` is the crowd-sourced task, whereas the `$platformResult` is a string that is encoded on the platform-specific way. The parsed assignment is returned.

3.9.2. Component Modules

The web application CrowdFlow Designer is a rather complex system. Each component like a task type or a merger type must be programmed individually. Each component should furthermore support as many kinds of input as possible, and its output must be consumable by the subsequent component. This leads to a vast amount of possible compositions and data flows, such that it was not possible to produce an all-encompassing solution for the first CrowdFlow Designer release.

Hereafter, the currently allowed input for each component type is specified, and the list of implemented modules is given.

Common As discussed in section 3.7, the flow of data is handled by making use of input queues. Each component implementation must thus be able to process and consume its input queue (except for the datasource, which - serving as a flow origin - does not have an input connection).

As a hook, the web application will call a component's `processInputQueue()` method. It is then the component's task to fetch the input queue, parse it and thereby taking into account the various data formats and structures that the queue item might have. For example, an implementation of a merger type must be able to process data coming from any of the so-far implemented task, post-processor, splitter and merger types, since all these components might "feed" the merger.

Tasks Almost any kind of task is imaginable to be crowd-sourced. But the big amount of property combinations discussed in section 3.5.4, which define a task, indicates that all types of tasks can hardly be implemented or presented in the same way. For example, a form published on a crowd-sourcing platform for a text processing micro-task requires one or multiple text input form elements, whereas a video item can be a URL or an upload form.

This also applies to the form used in the micro-task definition (which is filled out by

the requester). Let us consider the case where not one worker answer is accepted, but a series of answers, like in the task “describe the image with keywords”. Then, besides the regular crowd-sourcing parameters like amount paid and lifetime of the task, the form must have an input field for an additional parameter, i.e. the number of keyword fields that a crowd-sourced form shows.

Thus, each micro-task type not only requires a queue processing method but also needs an implementation for the *crowd-sourced form* and one for the *task definition form*.

Accepted Input Components

- Task
- Splitter
- Merger
- Datasource
- Postprocessor

Currently Implemented Types

- Transformation tasks with text as input and text as output, having results that have no natural order. The result contains multiple answers (one-to-many mapping).
- Transformation tasks with text as input and text as output, having results that have no natural order. The result is a non-binary selection (one-to-one mapping, but with multiple options).
- Categorization tasks with images as input and an image identifier as output with no natural order. The result contains exactly one answer (one-to-one mapping).

Post-Processors Post-processors perform operations on their incoming data. Some post-processor types are crowd-sourced, in which case the corresponding web form is required.

Accepted Input Components

- Task

Currently Implemented Types

- Post-Processor for crowd-sourced validation. When this module is used, a web form is generated and published on the crowd-sourcing platform for each flow item of the post-processor’s input queue. The form allows the worker to accept or reject the assignment. The flow item’s original input is also displayed, such that the worker can see what input another worker has transformed into which output.

Splitters Splitters divide the input data into multiple output data items. There are no splitter types - only one splitter implementation exists.

Accepted Input Components

- Task
- Postprocessor

Currently Implemented Types

- Input queue splitting. This module splits each flow item from the input queue into multiple output elements. It assumes that the input queue item has an array key “data”, composed of an array of sub-data that form the split parts.

Mergers Mergers transform items of their input queue into fewer output data items. A crowd-sourcing form also exists for the implemented module “selection by the crowd”.

Accepted Input Components

- Task
- Splitter
- Merger
- Postprocessor

Currently Implemented Types

- Web service call. The input queue items are sent to an external web service that performs a custom merging procedure. The result of the service can contain both, the list of items that are forwarded to the subsequent queue and the list of items that should remain in the input queue²⁰.
- Selection by the crowd. The requester defines a number as a threshold. Once such a number of assignments were submitted from the merger’s preceding task, the merger is crowd-sourced. This newly published task then asks workers to select the best among the submitted assignments. The requester also indicates with another parameter, after how many “merger” votes the most often selected assignment should be processed.

This module comes close to, but must not be confused with the concept of majority voting. In majority voting, the most often submitted answer is accepted, whereas in this merger type, the crowd explicitly selects the best option among a series of submitted results.

Note that this module, in the first CrowdFlow Designer version, is unstable and should not yet be used in production mode.

²⁰For example, in an aggregation function, the web service might require a minimum of items in the input queue. If this number is not yet reached, the items are not processed but remain in the input queue.

- majority voting. In this type of merger, the requester is asked to define a threshold in percents. This number specifies the minimum agreement among workers to accept a result.

The algorithm takes into account the potential future replies. Assuming, for example, a task that is published five times, and three equal assignments were already submitted. If the threshold is set to 50%, the system recognizes that, in any case, the minimum agreement will be at least 60% and that therefore the merger can already be processed.

By contrast, it might happen that the threshold is not reached (e.g. if in the previous example, only two assignments correspond with each other). In this particular case, the most often returned result is accepted instead.

majority voting has some limitations. More precisely,

“Majority voting assumes all experts are equally good”

[Raykar et al., 2010]. CrowdFlow Designer handles this issue by evaluating the results with respect to the *entire* or *envisaged* number of answers, and thus not only computing the assignments submitted so far.

Data Sources A datasource does not process any input. Instead, it is the origin of flow items. It thus only produces data.

Currently Implemented Types

- Text. For this datasource type, a simple input field is given, where a text can be pasted.
- Web service for images. A URL of a web service must be typed in. The web service must return an image URL.

4. Experiment Evaluation

After having described the features and the setup of CrowdFlow Designer, the conducted experiment is discussed to outline the impacts of using CrowdFlow Designer. For this purpose, two tests had been elaborated, both of them focusing on a specific property that is analyzed. They cover both scopes of application discussed in section 3.1. The measured numbers can be viewed in the appendix section C.

On each task, a note was displayed, asking participants to send any feedbacks by e-mail.

The experiments were executed at different times of day and on both week-ends and workdays, diminishing the risk of attracting the same workers over and over again. Since it is not a measure of interest, the time difference between HIT publication and acceptance was neglected. Instead, the test focused on the effort required to process, handle and analyze the assignments.

The *quality* of an individual, crowd-sourced micro-task is not examined in depth. The reason for this is that for the execution flow analysis, a single, self-contained assignment as part of a bigger macro-task is not substantial: for the worker, it makes no difference whether the task was created by the web application or by hand. And as the jobs themselves are the same in both approaches, the quality of the worker's assignments is just mentioned shortly.

As M-Turk is the primarily supported crowd-sourcing provider (see section 3.4), all tests were run on Amazon's service.

Research Questions

Experiment One In this experiment, the traditional crowd-sourcing approach is compared with the web application approach. The requester at first creates and publishes the crowd-sourced tasks manually. Furthermore, the assignments are retrieved from the crowd-sourcing provider and then processed by hand.

The same macro-task is then also solved on CrowdFlow Designer, and the results are checked against each other.

The research hypothesis is formulated as follows:

“Using CrowdFlow Designer results in a speedup and gain of efficacy compared to crowd-source macro-tasks manually”

Experiment Two The second experiment focuses on letting a *worker* define the work flow. The goal is to find out whether this essential part can also be crowd-sourced, or whether it is necessary to let an expert accomplish this step.

The hypothesis to verify or refute is worded as follows:

“CrowdFlow Designer is suitable to crowd-source the design and composition of macro-tasks”

4.1. Experiment One: Traditional versus WebApplication Approach

The two elaborated macro-tasks are presented hereafter - their work flows are listed in detail in section 4.1.3 and 4.1.4, respectively.

macro-task A: Translation An English story is given. The goal is first to cut the long text into smaller pieces. This task is accomplished by a worker of the crowd. Then, a validation task makes sure that the splitting is meaningful and appropriate. Thereafter, each part of the text is translated into Spanish by the crowd. Finally, after validating the translation, the output parts are merged again in order to build a Spanish copy of the original lines.
The first chapter of Charles Dicken’s short story “A Tale of Two Cities” [Dickens, 1859] serves as the original text.

macro-task B: Artwork Assignment The second experiment consists of tagging images. A web service supplies at random multiple pictures of famous painters. The requester then provides a list of painters that can be selected. Next, the crowd’s workers specify which artist they believe has created the picture. A majority voting is executed on these assignments, such that only the most frequent answer is considered. The final result is a list of assignments artwork → artist.

4.1.1. Goal

The goal of this experiment is to analyze speed and quality of using CrowdFlow Designer compared to carrying out all steps by hand.

4.1.2. Measures

The two dimensions efficiency and effectiveness are evaluated. For a statistical analysis, the following measures are applied:

Efficiency Time required to set up, execute and retrieve the final results for the defined macro-task.

Effectiveness A qualitative judgment of the result.

macro-task A: Translation For the translation experiment, it is checked whether the text splitting is meaningful and whether the paragraph translations are meaningful. In addition, the crowd’s validation is evaluated, i.e. whether translations are accepted or rejected correctly.

macro-task B: Artwork Assignment In this experiment, it is checked if the correct artist was assigned to the image.

4.1.3. Macro-Task A: Translation

Work Flow

In order to obtain comparable experiment results, the macro-task was composed in a concrete, prescribed way:

1. The requester prepares the text that is to be translated.
2. One task is published on the crowd-sourcing platform. This micro-task consists of splitting the entire text into sections that can easily be translated into Spanish.
3. As soon as the divided parts are available, the crowd is asked again to verify that the splitting is appropriate and useful.
 - If the splitting is appropriate, the text passages are used as input for the subsequent translation task.
 - If the divided parts are impractical (like having a text split in the middle of a sentence), the text passages are discarded, and the partitioning task is crowd-sourced again.
4. Workers have then to translate a paragraph that originates from the previous assignment.
5. This task is then verified by other workers. The original, English text is displayed together with the Spanish translation. This step was only executed in the second experiment run.
 - If the Spanish version is discarded, the translation task of the rejected paragraph is crowd-sourced again.
6. The validated lines are merged to produce the complete Spanish copy of the English original. For the automated approach, this step is accomplished by sending the translated parts to a web service which returns the concatenated pieces.

The corresponding pseudo-code is listed in algorithm 2.

Experiment Results

There were two runs for this use case.

In the first pass, the workers were asked to split the text into exactly five sections, and the translations were not validated but just merged. As for the back-end, the application was run on a local machine, i.e. a laptop with limited performance.

Executing this macro-task manually took 14:59 min. By using the user interface of CrowdFlow Designer, this duration could almost be halved (reduced by 45%) to 08:15

Algorithm 2 Translation Experiment Work Flow

```

1: publish(textSplittingTask);
2: while true do
3:                                     ▷ Publish a task to validate the splitting
4:   if textSplittingTask.getResults()!=null then
5:     splitValidationTask = createValidationTask(textSplittingTask.getResults()[0]);
6:     publish(splitValidationTask);
7:   end if
8:                                     ▷ Process text splitting task
9:   if splitValidationTask.getResults()!=null then
10:                                     ▷ The split task is validated
11:     result = splitValidationTask.getResults()[0];
12:     validatedAssignment = result.getValidatedAssignment();
13:     if result.isAccepted() then
14:       acceptOnCsPlatform(validatedAssignment);
15:       ▷ Can now publish the translation task for each of the text parts
16:       publishTranslationTasks(textSplittingTask.getResults()[0]);
17:     else
18:       rejectOnCsPlatform(validatedAssignment);
19:       republish(textSplittingTask);
20:     end if
21:   end if

```

Algorithm 2 Translation Experiment Workflow (continued)

```

22:                                     ▷ Process translation tasks
23:   for all translationTask in translationTasks do
24:     if translationTask.getResults()!=null then
25:                                     ▷ Translation is available
26:       assignment = translationTask.getResults()[0];
27:       forwardToValidator(assignment);
28:     end if
29:   end for
30:                                     ▷ Process translation validation tasks
31:   for all translationValidationTask in translationValidationTasks do
32:     if translationValidationTask.getResults()!=null then
33:                                     ▷ Translation validation is available
34:       result = translationValidationTask.getResults()[0];
35:       validatedAssignment = result.getValidatedAssignment();
36:       if validatedAssignment.isAccepted() then
37:                                     ▷ Translation was accepted - use the translated text
38:         acceptOnCsPlatform(validatedAssignment);
39:         forwardToSubsequentItem(validatedAssignment);
40:       else
41:                                     ▷ Translation was rejected - must re-publish the translation task
42:         rejectOnCsPlatform(validatedAssignment);
43:         republish(validatedAssignment.getOriginalTask());
44:       end if
45:     end if
46:   end for
47:                                     ▷ Process the text merging
48:   if textMerger.getInputQueue().length==NUMBER_OF_SPLITTINGS then
49:                                     ▷ All translated texts have been submitted to the merger
50:     textMerger.processInputQueue();
51:     ▷ The merged text will automatically be sent to the workspace's result
52:   end if
53: end while

```

min. This shows that even for a relative small task, the effort of the traditional handling exceeds the web application approach by far.

The difference is even greater for the more complex setup. This time, the translated texts were not accepted immediately but instead were validated by the crowd. Furthermore, the web application was this time run on a “real” server (instead of the local machine). Therefore, despite the additional steps, the total execution time remained almost the same (08:16 min) compared to the first run.

On the other side, the new and more complex work flow has a big impact on carrying out the tasks by hand. Particularly inconvenient is the fact that there are no custom-tailored forms available for several tasks. For example, one cannot provide five text input boxes for the splitting task on M-Turk. Instead, the workers had to be asked to visually separate the text pieces by leaving a horizontal gap between them. One worker failed to do so, which required the split task to be re-published - a problem that would not occur in the web application approach.

The influence of the manual handling can also be seen when publishing the split verification task: CrowdFlow Designer is able to generate a simple “accept” form by displaying the original assignment as submitted by a worker, an operation that takes thirteen seconds. If this step is accomplished by the requester himself, he needs 4:40 to download the task result, analyze the splitting and prepare an “accept” form.

With 4:48 minutes, it is also very time-consuming to create a new translation HIT for each of the text parts. The CrowdFlow Designer batch processing would cut this down to thirty seconds.

Another essential step is to publish the translation validation task, containing both the original and the Spanish phrases. This part is completely automated in the web application approach and takes exactly thirty seconds, whereas doing so by hand consumes 6:45 min, or more than twelve times longer.

Up to this point (before analyzing the translation validation), the CrowdFlow Designer application took 7:45 minutes, compared to 28:10 in the traditional version. In both techniques, some of the Spanish translations were rejected by workers, requiring a re-publication of the rejected HITs. On M-Turk, a HIT can be re-published with just a few clicks in the user interface. The problem hereby is that neither the payment can be increased, nor can the expiration date be extended - this limitation was neglected in the experiment. Furthermore, especially if many tasks had already been published, it is difficult to find the particular task that must be re-enabled. In the test, it took 03:26 min to extract the rejected translations and 1:50 min to re-publish them. CrowdFlow Designer carries out these steps altogether within 14 seconds.

The entire manual execution flow takes 51:07 minutes, or 514% longer than the CrowdFlow Designer approach (8:16 min).

Quality

In general, the quality of the translations was considered okay. Two times, however, a worker simply copied the original text and submitted it as Spanish translation. One of these two assignments even passed the subsequent validation conducted by another worker.

Furthermore, it has been observed that users were not reluctant to reject translated passages. Two out of eight translations were rejected by the crowd in the web application test. As for the manual run, three Spanish texts were refused and re-published. Among the three revised translations, one was dismissed again.

General Experiment Observations

As the translation and their validation are published separately on the crowd-sourcing platform, there is a risk that users are confronted with their proper assignments, allowing them to approve their very own result. Especially if the time gap between publishing the tasks is short, this situation may happen. Throughout the 22 translations that were validated in the experiment, three times an M-Turk worker was asked to evaluate his own answer. In all those cases, the assignment was approved.

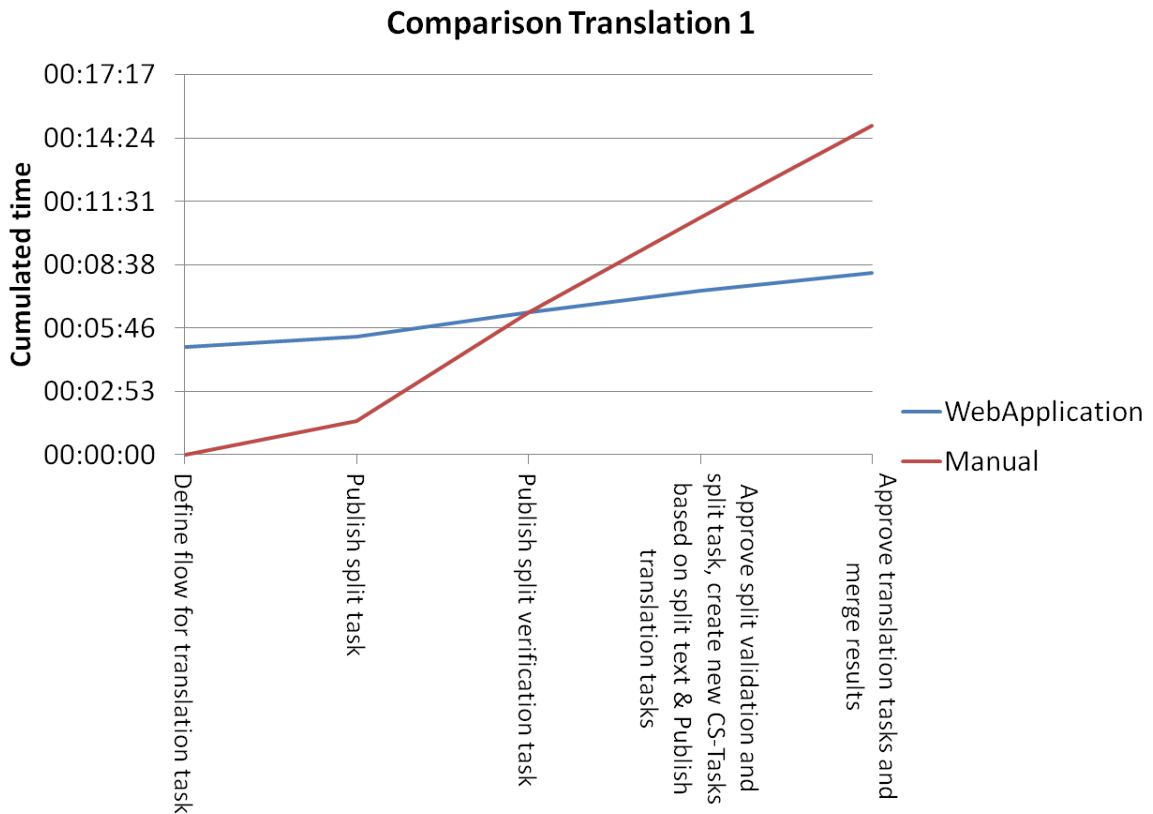


Figure 4.1.: Translation run 1

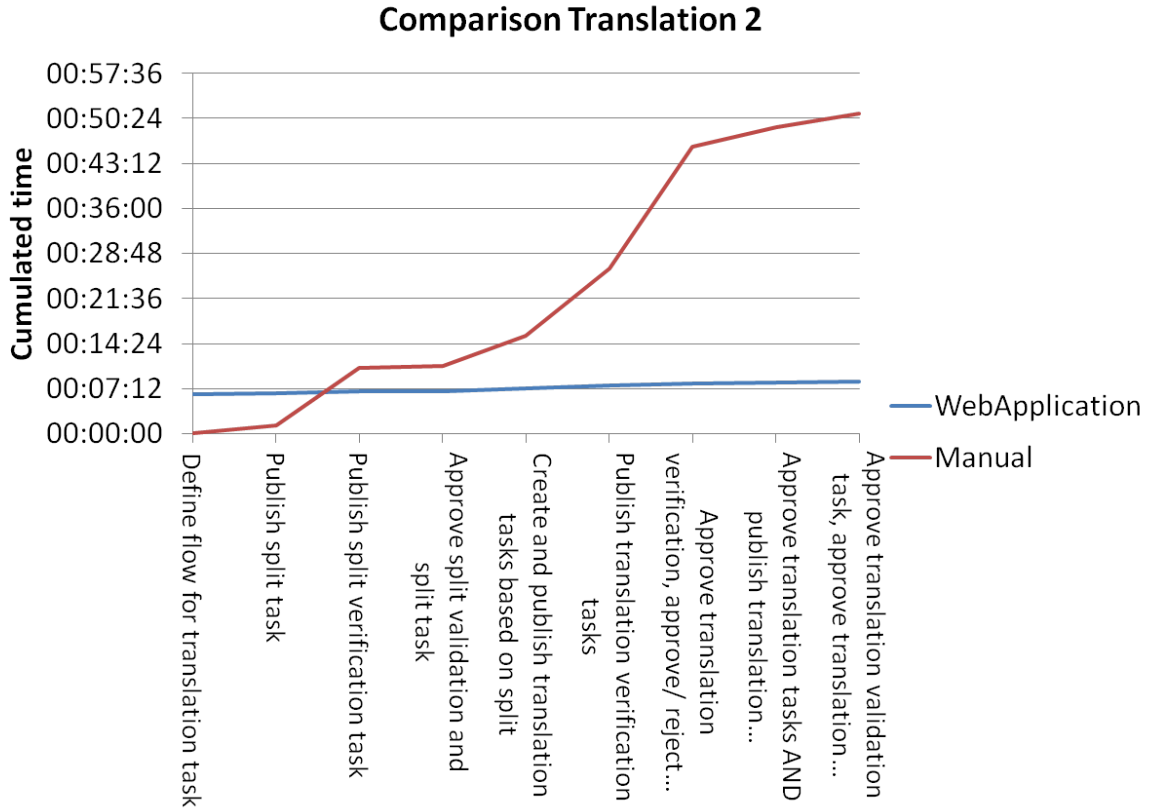


Figure 4.2.: Translation run 2

It is self-evident that the design phase is required only when using the CrowdFlow Designer interface. All subsequent tasks must be executed in both approaches, and this is where CrowdFlow Designer outperforms the manual flow. In figures 4.1 and 4.2, the cumulated durations of the individual sub-tasks are compared between the two techniques. Here, the influence of the design phase becomes obvious, since this is the only place where the manual approach takes advantage. Hence, it is essential to build a GUI that is as straightforward and as easy to use as possible.

Workers did like the tasks and were sometimes even explicitly asking for more jobs of the same kind: ¹.

4.1.4. Macro-Task B: Artwork Assignment

Work Flow

1. A web service that provides pictures is accessed. With each HTTP request, a URL of a painting image from one of the below mentioned artists is returned. This step

¹“[...]if possible assign all hits to me means to my worker id i’m very much interested to work in this kind of work.”, “i like to do more translation jobs from you..can you allocate some jobs for my id..so i can give my best all time”

is automated in the web application. However, in the manual flow execution, the same URLs are used again in order to be able to compare the results of both approaches.

In the first run, five artworks are presented to the crowd. Three HITs are generated for each of them. The second, larger test is constituted of ten paintings, each of them being tagged five times.

2. The crowd is asked to select a painter among the following options:
 - Unknown
 - Leonardo Da Vinci
 - Vincent Van Gogh
 - Rembrandt
 - Michelangelo
 - Claude Monet
 - Pablo Picasso
3. When all assignments are submitted, the most often selected painter is determined by conducting a majority voting. A web application component executes this step automatically². As for the manual run, the requester has to fetch all results and figure out the top answer by hand.

The pseudo-code 3 is used to describe this procedure.

Algorithm 3 Artwork Assignments Work Flow

```
1: images = getAllImages();
2: classificationTasks = new List();
3:                                     ▷ Create a crowd-sourced task for each painting
4: for all image in images do
5:   classificationTask = createClassificationTask(image);
6:   classificationTasks.add(classificationTask);
7: end for
8:                                     ▷ Publish each task on the crowd-sourcing platform
9: for all assignmentTask in assignmentTasks do
10:   publish(assignmentTask);
11: end for
```

²See component's description on page 51.

Algorithm 3 Artwork Assignments Work Flow (continued)

```

12: classificationAssignments = new List();
13: while true do
14:                                     ▷ Process painter classification assignments from the crowd
15:   for all classificationTask in classificationTasks do
16:     if classificationTask.getResults().length==ASSIGNMENTS_PER_TASK
17:       then
18:                                     ▷ All results have been submitted
19:         assignments = classificationTask.getResults();
20:         for all assignment in assignments do
21:           acceptOnCsPlatform(assignment);
22:           forwardToSubsequentItem(assignment);
23:         end for
24:       end if
25:     end for
26:                                     ▷ Execute the majority voting
27:   if merger.getInputQueue()==images.length then
28:     ▷ All painting assignments have been sent to the merger for majority voting
29:     ▷ We have to get a list of all assignments for each of the paintings. Each
30:     painting has its own flowItemId
31:     Dictionary<int, assignment> assignmentsPerPaintings =
32:     merger.getAssignmentsPerFlowItemId();
33:     for all KeyValuePair<int, assignment> assignmentsPerPainting in assign-
34:     mentsPerPaintings do
35:       mostChosenClassification = merger.executeMajorityVoting(assignmentsPerPainting);
36:       forwardAsFinalResult(assignmentsPerPainting.key, mostChosenClassifi-
37:       cation);
38:       ▷ The method's arguments contain the information about which painting
39:       was assigned which painter
40:     end for
41:   end if
42: end while

```

Experiment Results

Publishing the five tagging tasks by hand takes 283 seconds, and 340 for ten, which makes an average of 42 seconds per HIT. Once the first task has been created, the same settings can be used as a template to crowd-source the subsequent ones - a handy feature of M-Turk. Despite this function, this step is substantially faster in the automated approach, where the interaction with the platform takes 54 and 14 seconds, respectively³.

Since the entire flow processing on CrowdFlow Designer is carried out in one step, the submitted assignments are approved, analyzed and undergone the majority voting at the same time in 64 and 30 seconds, respectively. On the other side, just approving the crowd's results takes 1:23 min and 2:36 when doing by hand. Much more time is needed to gather the results and to manually detect the most often submitted answers. Since this also includes some calculations and extensive web page navigation, the effort is rather high with 3:52 minutes for the first and 9:05 minutes for the second run.

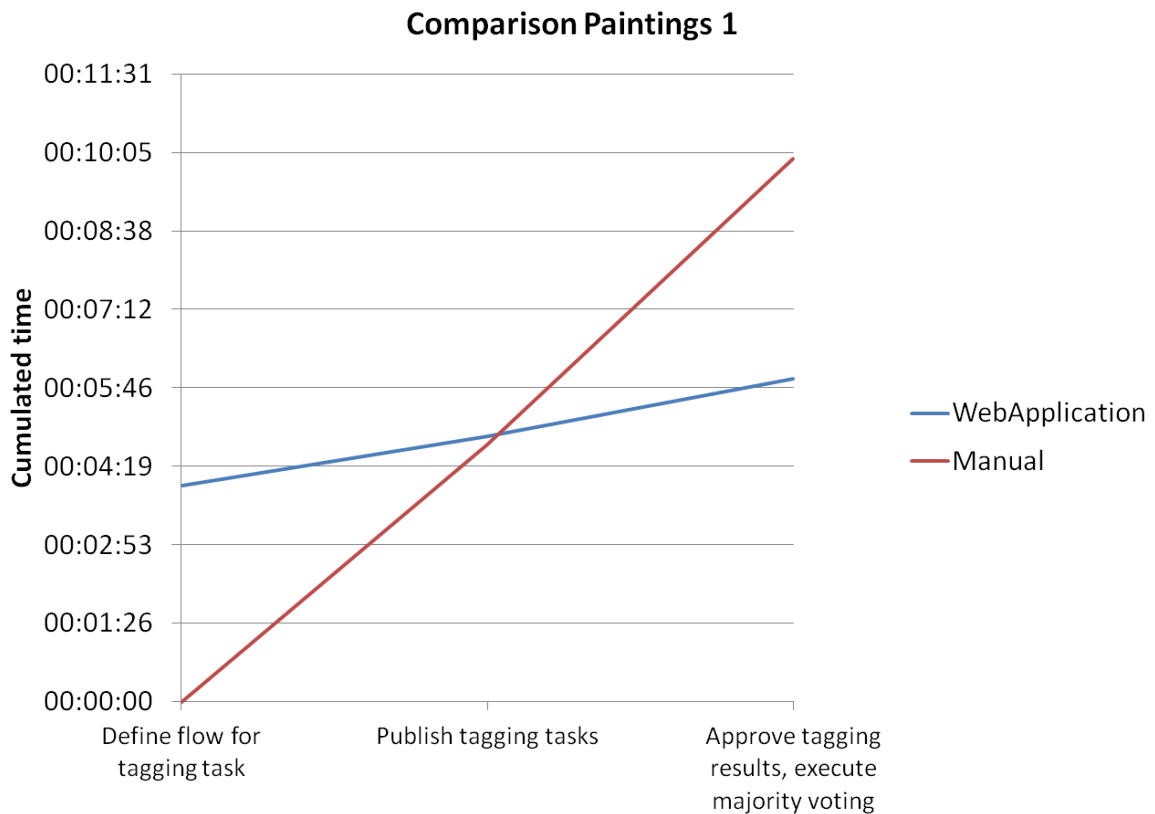


Figure 4.3.: Artwork assignments run 1

As visible in figures 4.3 and 4.4, the situation is similar to the one with the translation macro-task. The individual sub-tasks require by far more time in the manual approach.

³The first run was executed locally, whilst the second test was executed on a web server - see also section 4.1.3.

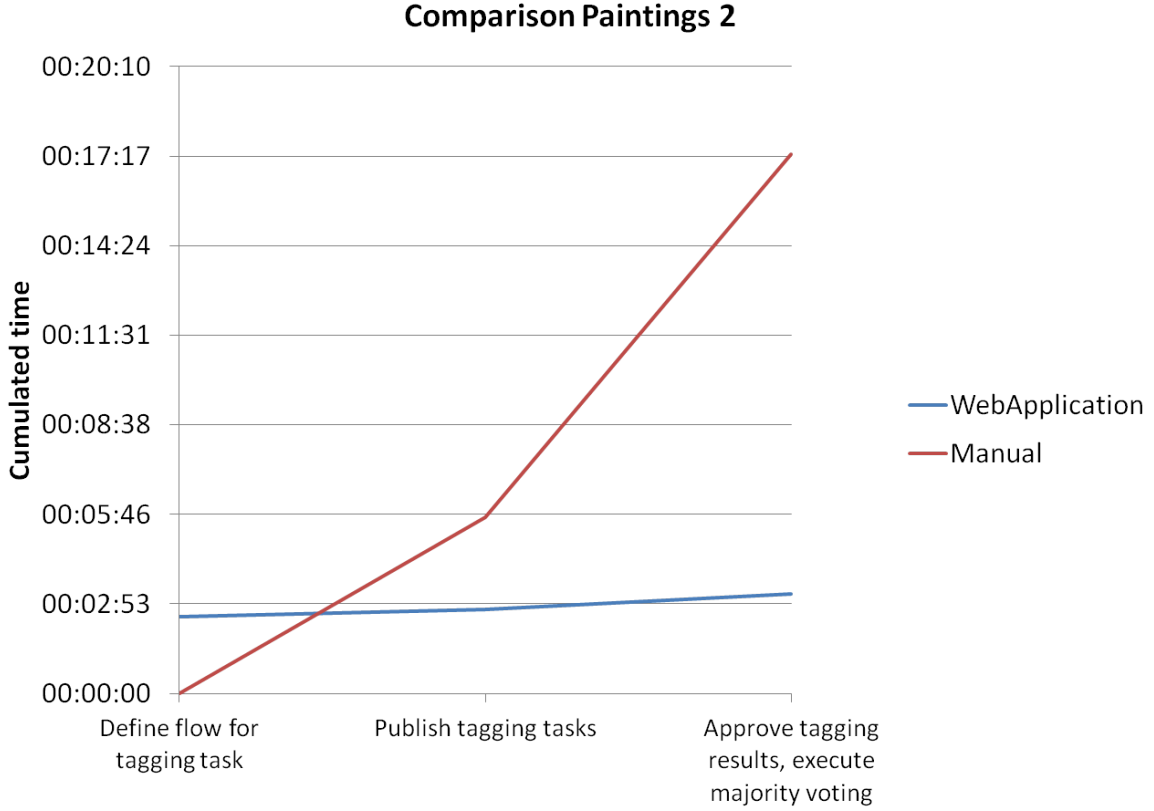


Figure 4.4.: Artwork assignments run 2

On the other hand, once the design phase is completed in CrowdFlow Designer, the subsequent steps are executed very quickly.

In summary, with a duration of 9:58 min, this experiment is executed 1.67 faster on CrowdFlow Designer (5:56), compared to a speedup of 5.42 in the second run (3:12 versus to 17:21). Again, for the reason of the relatively “heavy” design phase on CrowdFlow Designer, the speedup is expected to increase with even larger numbers of tasks.

Quality

In overall, the quality of the results with many votes was good. The only problem was with the first run, where the majority voting was carried out after only three assignments: three of the five painters could not be identified correctly in the CrowdFlow Designer test. In these cases, one of the three workers submitted the correct answer. But since three different answers had the same agreement of 33%, the majority voting algorithm picked a result at random, and that answer was incorrect.

The authors of all ten paintings in the second execution were detected correctly in both the manual and automated flow, with agreements ranging from 60% to 100%.

General Experiment Observations

The tasks of this experiment were relatively short and well paid (0.2 USD per HIT). Sometimes, all artworks assignments were submitted in just a few minutes after publishing them.

4.1.5. Conclusion

A substantial gain of efficiency has been observed throughout the experiments conducted. On these grounds, we can **approve** the hypothesis defined on page 52.

4.2. Experiment Two: Crowd-Sourced MacroTask Composition

Experiment Two was again divided into two parts, each of them analyzing one of the intended crowd-based application scopes described in section 3.1.2.

Setup A: Multiple Workers In the first run, *multiple* workers were asked to design the flow of the macro-task *sequentially*. Each of them received specific instructions on how to execute a single step of the macro-task composition.

Setup B: Single Worker For the second test, the entire intended composition was explained to a *single* worker. He was then solely responsible to create the *entire* workspace all by himself.

Before starting the experiment, the intended work flow was set up by the researchers. Thus, the output of the experiment can be compared with the version designed by the experts. We refer to this reference variant as *ground truth*.

4.2.1. Goal

This test aims at determining whether CrowdFlow Designer can be used to let the *crowd* compose the macro-tasks, either by dividing it into sub-steps or by publishing the complete job as an all-encompassing assignment.

These two versions are compared against each other to evaluate which setup produces a better result.

4.2.2. Measures

Since the workers are interacting with the CrowdFlow Designer directly, any modifications and work flow designs are immediately stored to the database. Therefore, one can easily verify whether the desired composition was achieved or not by comparing the results with the ground truth. In other words, the same elements and connections between them must be present as in the reference macro-task composition.

4.2.3. Use Cases

A tutorial on how to use CrowdFlow Designer had been created by recording the computer screen and commenting the steps that are required to execute various functions on the CrowdFlow Designer GUI.

The crowd-sourced task composition was presented along with a detailed set of instructions as listed below. Furthermore, the links to video tutorials for the individual steps were supplied.

1. Here <http://unifr-cs.herokuapp.com/frontend/#/home> Create a new Workspace for Painting Classification. Select the existing macro task “Assign painters to artwork”. Video Tutorial on Workspace creation: <https://www.youtube.com/watch?v=hqoPlZ31k1g>
2. Create a Datasource using a web service generating five items from URL: <WEB_SERVICE_URL> Video Tutorial on Datasource Creation: <https://www.youtube.com/watch?v=tThiXWVKl2w>
3. Add a Categorization task to categorize images into the following categories: Unknown, Leonardo Da Vinci, Vincent Van Gogh, Rembrandt, Michelangelo, Claude Monet, Pablo Picasso. Please decide for the appropriate reward for this task in USD. Video Tutorial on task creation: https://www.youtube.com/watch?v=G-wqhQp1s_0
4. Add a Majority Vote Aggregation and conclude the work flow. Video Tutorial on Majority Vote Aggregation: https://www.youtube.com/watch?v=f_NjqX92Aja

Thus, the work flow resembles the one created in experiment One B, but differs in that an additional *validation* is attached to the tagging task. This aimed-for composition is presented in figure 4.5.

4.2.4. Experiment Results

It can be observed that the version executed by multiple users iteratively differs from the ground truth version. In the crowd-sourced version, the validation step was omitted. And instead of merging the author assignments, a parallel work flow was created that joins the original categorization task flow. In the end, the macro-task is composed of two author classification tasks, one of which was lacking an input (datasource). Therefore, the hereby developed work flow, as visible in figure 4.6, would not function correctly.

On the other hand, the single-worker experiment matches exactly the envisaged work flow. The final design can be viewed in figure 4.7.

The total execution time is also substantially lower in the test with just one worker, where the entire job took 09:17 minutes compared to the 1:14:31 used in the divided conception⁴.

⁴The single sub-task of creating a categorization task took an unexpected long time (54:48 min). Though, even if this possibly not representative data is omitted: the remaining steps lasted 19:43 min, which is still more than the double of the single-worker version.

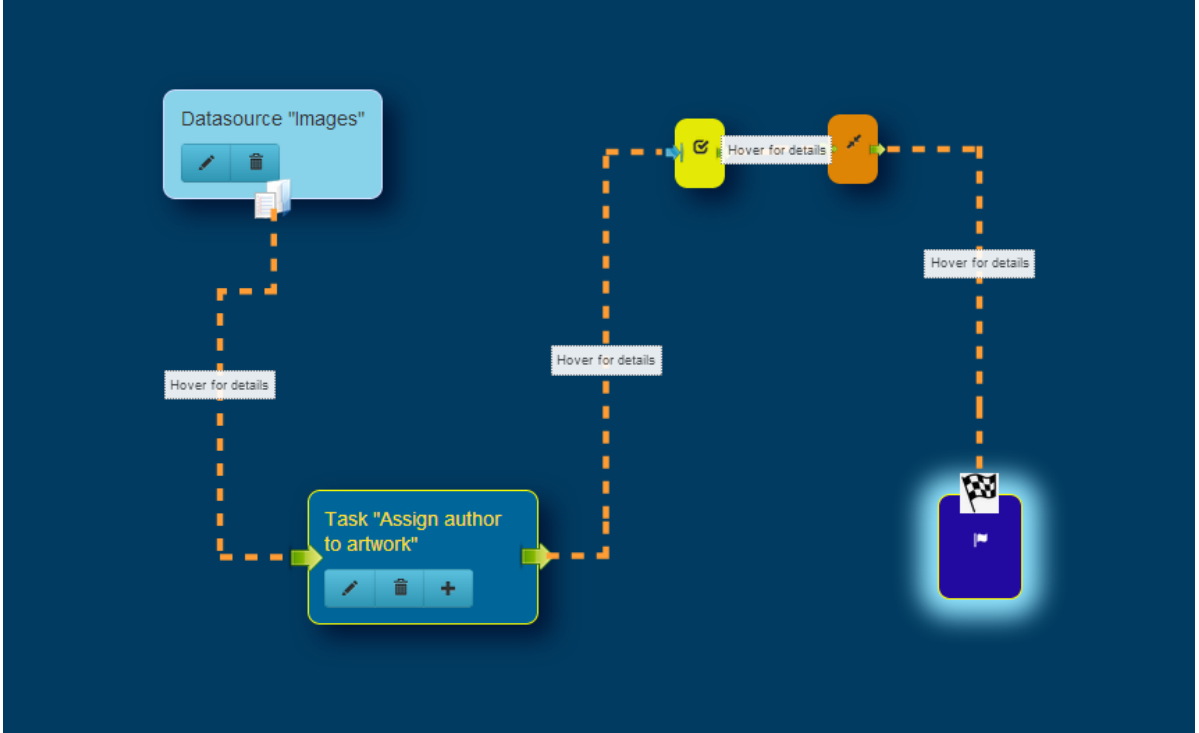


Figure 4.5.: Crowd-source macro-task design: ground truth

In addition to that, workers that do not know the overall goal and purpose of the macro-task have more difficulties to correctly implement their part. This fact can not only be observed when comparing the results. Rather, it was also explicitly reported by some users⁵. Hence, it seems essential to delegate the macro-task decomposition as one single task. If multiple workers are involved, they must be informed verbosely about the job.

When the design phase was crowd-sourced, workers could decide on the salary of the classification task. And this is where the two versions differ: while in the reference version, 0.2\$ were paid, crowd users awarded each HIT with as little as 0.01\$ to 0.02\$, thus 10 to 20 times less.

4.2.5. Quality

As described above, there is a non-negligible difference in quality between the two tests. On one hand, the single-worker version is usable outright for further processing, whereas the approach with sequential crowd-sourced steps revealed some shortcomings. For a more detailed and solid analysis, the same experiment should be executed multiple times to either prove or refute the assertion that shorter crowd-sourced tasks *in general* lead to poorer results.

⁵“It also is not entirely clear from the project description what needs to be done”

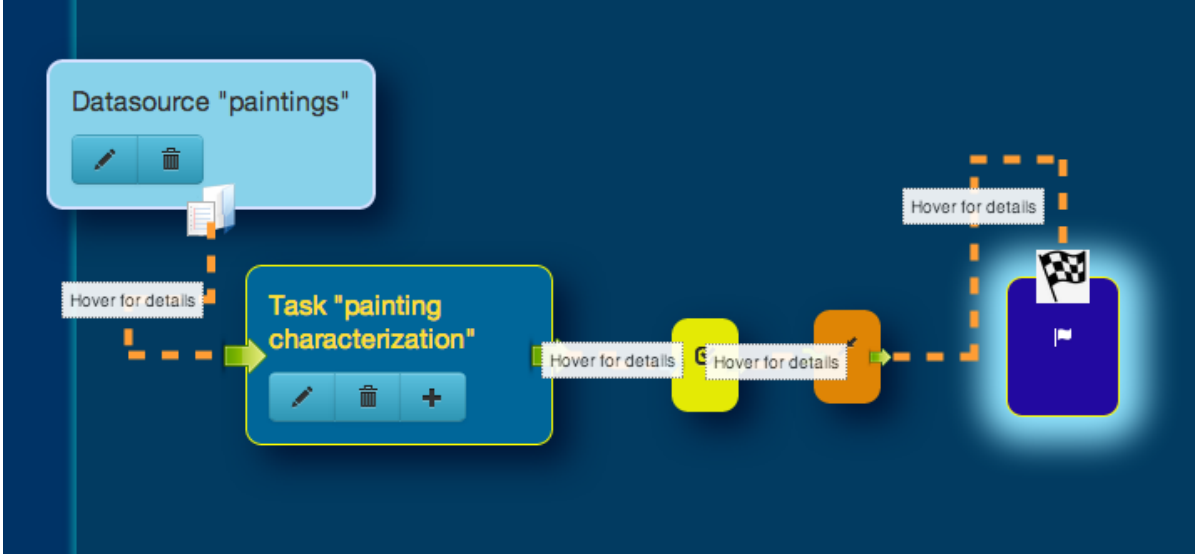


Figure 4.6.: Crowd-source macro-task design: result with one worker

4.2.6. Conclusion

At the beginning of this section, the assumption was made that CrowdFlow Designer can be applied to let workers define and construct the setup of macro-tasks. The approach consisting of sequential design steps did not result in a usable output, whereas a single worker was able to compose the macro-task meaningfully. Because of that, we can only **partially approve** the hypothesis.

4.3. General Experiments Feedback and Findings

As discussed in section 3.5.3, CrowdFlow Designer provides both visual and API tools to get insight in the current execution state. On the other hand, it is very hard to accomplish this breakdown on the crowd-sourcing platform⁶ and would require to inspect the HTML source code in order to identify the tasks⁷.

One can observe that the biggest part of the web application approach falls onto the design phase. In other words: once the macro-task has been defined, all subsequent steps are executed very quickly. Therefore, the more complex the macro-task and the more individual steps involved, the greater is the speed-up compared to the manual approach. This also explains why a larger advantage can be noted in longer and more sophisticated experiments (for example the translation experiment's second run that includes a validation).

⁶In fact, it is almost impossible if the requester does not write down the HIT identifiers after each publication.

⁷Neither the HIT nor the assignment identifiers are displayed in the online application. But they can be inferred from the hyperlinks of the web page or retrieved from the downloaded results.

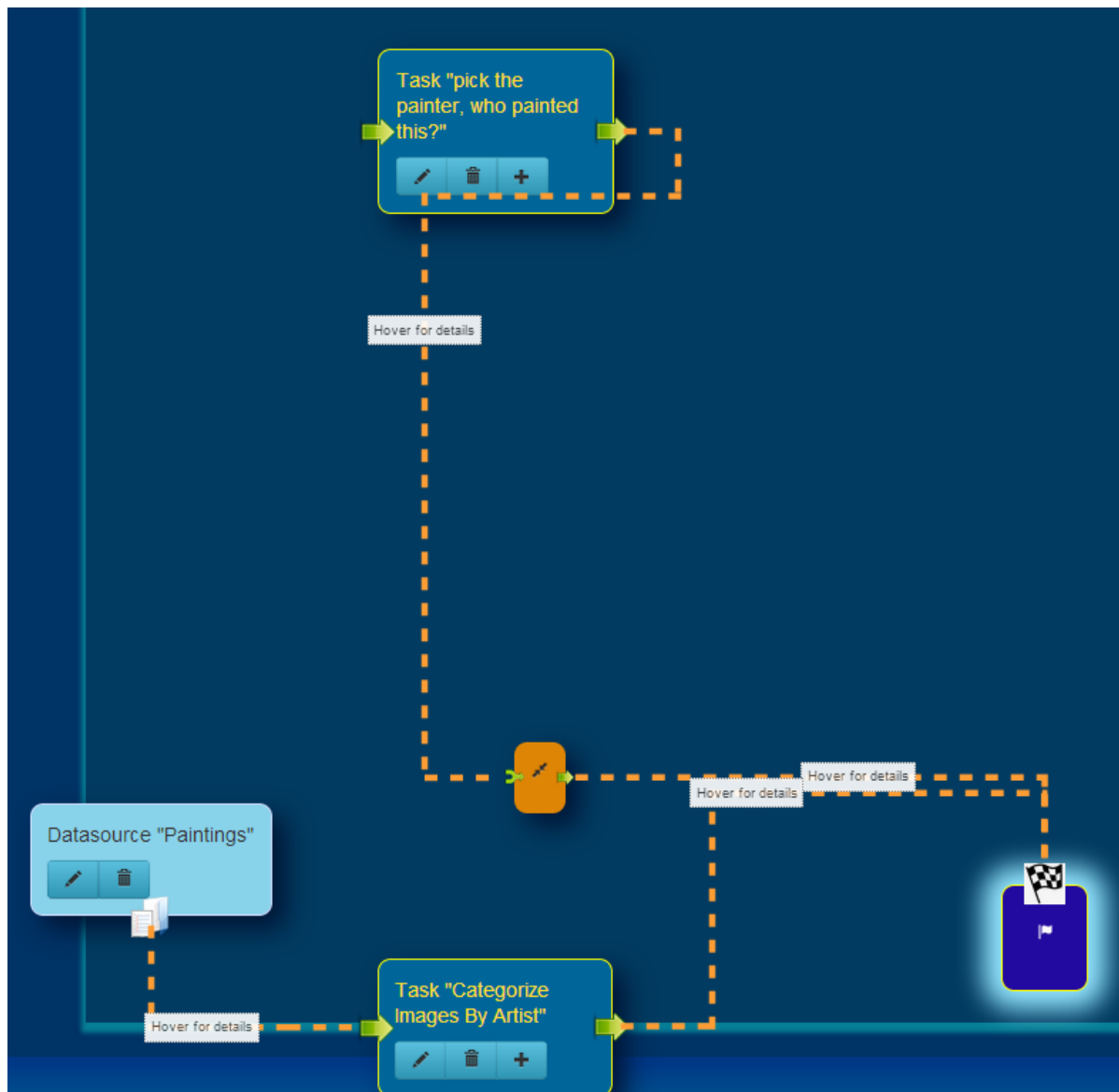


Figure 4.7.: Crowd-source macro-task design: result with multiple workers

Figure 4.8 displays the proportion of the design phase compared to the total execution time for experiments executed on CrowdFlow Designer.

Regarding the interface, the feedback was predominantly positive. For this purpose, a selection of comments submitted when using the interface shall be listed:

- “Nice design and everything is understandable, easy to use.create task very simple here.” *Worker asked to create a workspace*
- “I really like the user interface. It was a little unclear at first as I was learning the system, but I liked it once I got used to it.” *Worker asked to create a datasource*

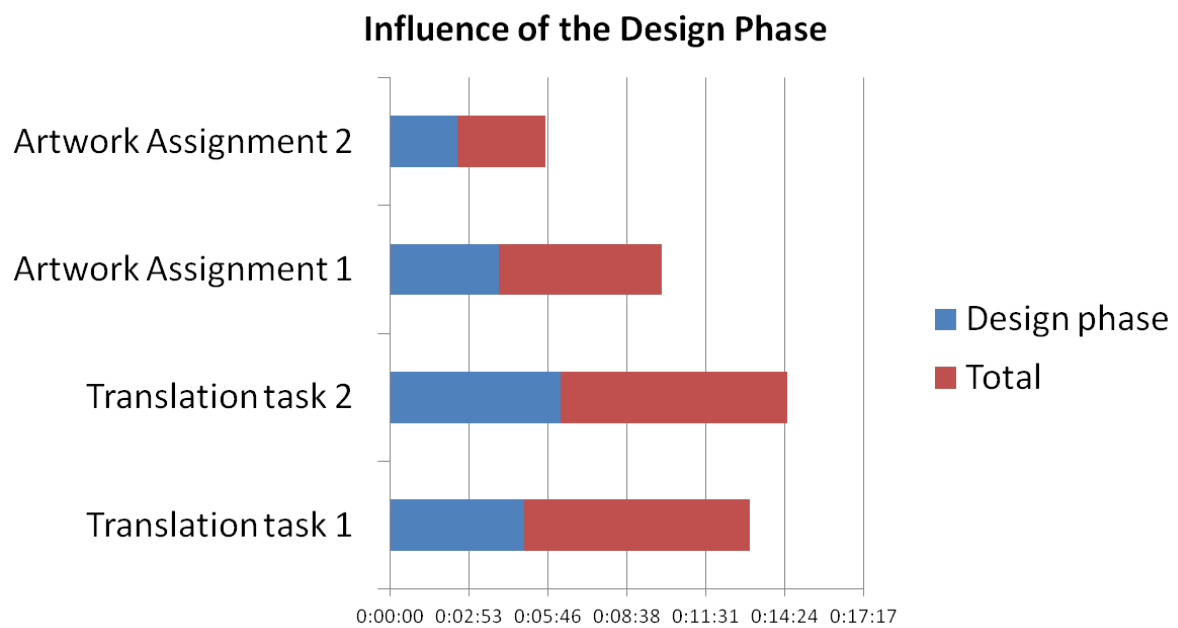


Figure 4.8.: The influence of the design phase in CrowdFlow Designer

5. Conclusions

Throughout this document, we have covered the new way of executing jobs that cannot be accomplished by machines. *Crowd-sourcing* is increasingly applied in industry and research, and an even growing scope of application is imaginable as this technique evolves.

In the first chapter, the notion, actors and ideas behind crowd-sourcing was discussed.

Subsequently, the toolkit “CrowdFlow Designer” has been introduced. This new application allows the composition and organization of complex macro-tasks. It offers a handy way of managing task flows that up to now had to be arranged inconveniently one by one. A big advantage of the application is the support for various types of use: the software is not only aimed at supporting the *requester* in designing sophisticated jobs, but makes it also possible to *crowd-source* this operation.

The components, structure and technique behind CrowdFlow Designer were presented, along with a guide on how to deploy, run and extend the program.

Finally, multiple experiments were run for different use cases and with various macro-tasks. The results and analysis of these tests have demonstrated that CrowdFlow Designer exhibits a huge advantage with respect to execution time, compared to the traditional approach where micro-tasks are managed manually. Whilst certain setups have revealed some shortcomings, the overall quality of the results point out the benefits and usefulness of the program.

Moreover, the entire organization, overview and breakdown of a large crowd-sourced project is quicker and more straightforward when applying CrowdFlow Designer.

In addition to that, feedbacks from crowd workers have shown that the GUI is in general appealing and easy to use.

To summarize, we can conclude that CrowdFlow Designer enables a new way of designing, managing and organizing big crowd-sourcing projects. Its modular structure helps to extend and customize the application. Any current or future crowd-sourcing platform can be integrated, and custom third-party applications can make use of the CrowdFlow Designer infrastructure and functions through the built-in API.

The experiments have demonstrated that the toolkit is working reliably; applying CrowdFlow Designer to manage complex macro-tasks as opposed to manual operation brings enormous gains in comfort and time effectiveness.

5.1. Lessons Learned

As soon as the software requirements of CrowdFlow Designer were worked up, the user interface could be developed. Soon, it became clear that there is a considerable difference

between programming a desktop application and designing a web tool. In the world of the internet, one has to work with weakly typed programming languages like JavaScript and deal with browser compatibility issues. In this context, it was a big aid to rely on libraries. On this occasion, the framework Angular JS was discovered and learned. This toolkit brings a clean structure to in-browser software by clearly separating business logic from visualization in a way rarely achieved before. With the soaring deployment of web applications, the hereby acquired knowledge will prove very valuable.

The elaboration of the application's back-end was started soon - almost too soon. The complexity of accepting any combination of a workflow has lead to problems that have not been thought of in time. For instance, the discussed approach of relying on input queues for each single component was not adapted from the beginning but has proven indispensable.

In other areas, the architecture was at first complicated unnecessarily. A telling example is how series of the same crowd-sourced tasks were handled. In the first draft, one single platform task was created, and an internal "tracker" memorized the number of published jobs. For each worker the task form was rendered for, the appropriate input was then figured out in a cumbersome way *just in time*, yielding problems like duplicate task publication and workers being faced with a corrupt or invalid form. Thus, in a future similar project, the entire workflow should be elaborated more thoroughly in advance.

Even though the application was run multiple times in a safe "sandbox" mode, not all problems could be prevented. For inexplicable reasons, for example, a feedback form that was included in crowd-sourced tasks interfered with M-Turk's web interface, such that workers could see and accept, but not submit assignments.

Moreover, the time that is required to run and analyze crowd-sourcing experiments must not be underestimated. It can be difficult to predict, how quick jobs are accepted on a crowd-sourcing platform. And even with payments above average, the project may have to deal with low quality or copy-paste assignments. One thus has to expect and plan for a portion of unusable user contributions.

In total, the experience that could be gained by programming and testing CrowdFlow Designer is of big importance for future projects - not only in the area of crowd-sourcing, but in software architecture, user interaction and evaluation projects in general.

5.2. Limitations

By targeting a broad variety of flow compositions and crowd-sourcing platforms, CrowdFlow Designer has grown in complexity. In the first delivery, not all aspects of a web application could be considered. Therefore, some limitations arise when applying CrowdFlow Designer.

Quality Control In section 2.5, the significance of *quality control* was mentioned. It would be wise to include some of the measures discussed in order to obtain reliable data

from the crowd. For this purpose, it is imaginable that a future version of CrowdFlow Designer handles the *minimum reputation* check for workers who participate. On the other side, the method of *intermediate test questions* could be implemented.

Security and User Error Even though some basic security precautions have been taken like dealing with invalid parameters or with an illegal parameter format, these measures are still below the level required to run CrowdFlow Designer in production mode. For example, a user could easily analyze the JavaScript code executed in the client's browser. Then, he could alter REST requests that are sent to the server and thereby, for example, maliciously access or modify objects he is not currently authorized to operate on.

On the server side, some of the user faults are captured and handled by returning an explanatory error message. But again, the provisions may not be comprehensive enough. These limitations are also associated with the currently missing *user management*.

User Management The interaction with the crowd-sourcing platform accounts was already introduced in section 3.4. It implies that CrowdFlow Designer requesters only need one account, handled in the web application, and do not have to be registered on the crowd-sourcing providers.

On the other hand, *workers* run jobs application-independently on their respective platforms - no user is executing tasks directly on CrowdFlow Designer.

By supporting various platforms, CrowdFlow Designer is confronted with heterogeneous reputation systems. This may cause some issues when implementing internal ratings, requiring some kind of reputation score translator to deal with different grading units.

Furthermore, by following an approach with completely de-coupled client and server side implementations (see section 3.3), each HTTP request handled by the back-end must contain an identification of the sender. This could come in the form of pretty insecure basic HTTP authentication, providing the credentials for each request, or by implementing a *signature* technique with the notion of public and private keys [Richardson and Ruby, 2007]. Another option would be to follow the OAuth protocol [Allamaraju, 2010]. This authentication method is based on so-called *access tokens* that are required to gain access to a *shared resource*, e.g. a CrowdFlow Designer workspace.

5.3. Future Work

To be applicable in an operating environment, the shortcomings of CrowdFlow Designer discussed in the previous section must first be eliminated. Moreover, it is imaginable that more than the currently supported components described in section 3.9.2 may be required to cover a wider range of application. For example, by the time of writing, there is no support for video or audio media types - an area in which crowd-sourcing is heavily used nowadays.

On the other hand, the base system is ready for use. In overall, the issues pending can be elaborated in a reasonable amount of time, making it conceivable to run CrowdFlow Designer in production mode in the near future. Besides the already implemented M-Turk module, any crowd-sourcing platform can be integrated in the system. The open structure of the CrowdFlow Designer enables in particular a wide range of use by providing an API, such that the application can be consumed by and attached to various third-party software of any kind.

Acknowledgments

Throughout the elaboration of the master thesis, I was supported by scientific researchers without whom this project would not have come to the stage it is now.

I would like to especially give thanks to the members of the research group “eXascale Infolab”¹ from the University of Fribourg². Their knowledge and experience have helped me a lot in programming and writing the master thesis. Most notably, I could always count on the support of Prof. Dr. Philippe Cudré-Mauroux³ and Dr. Gianluca Demartini⁴. Furthermore, Djellel Eddine Difallah⁵ assisted by bringing in new ideas with his reflections.

Finally, further acknowledgments are towards the University of Fribourg, at which I could write the master thesis and realize the CrowdFlow Designer project. It has always been helpful to rely on a sophisticated study environment and infrastructure which is run and maintained by skilled research teams and university staff.

¹www.exascale.info

²www.unifr.ch

³pcm@unifr.ch

⁴gianluca.demartini@unifr.ch

⁵djelleddine.difallah@unifr.ch

Glossary

Angular JS A JavaScript framework that allows to code a web application using the Model-View-Control paradigm. 5, 26, 27, 42

API Application Programming Interface. 3–6, 16, 17, 21, 22, 29–32, 43, 44, 66, 69, 71

Beta A pre-version of a final software product. 14

BSD Berkeley Software Distribution. 24

CML CrowdFlower Markup Language. 16

Crowd Flower A crowd-sourcing platform, see <http://www.crowdfLOWER.com/>. 4, 16, 30

CrowdFlow Designer A modular crowd-sourcing toolkit to define and manage composed macro-tasks on crowd-sourcing platforms. vi, 1–7, 14, 16, 19, 21–34, 37, 38, 42, 44–47, 49, 51–53, 56, 58, 61–64, 66, 67, 69–72

CrowdForge A web application that manages the flow of crowd-sourced tasks. 4, 5

CrowdWeaver A web application that manages the flow of crowd-sourced tasks. 4

CRUD Create, Read, Update and Delete. 27

CSS Cascading Style Sheets. 16, 42

Django A framework to generate web applications. 5

GPL GNU General Public License. 28

GUI Graphical User Interface. 19, 44, 58, 64, 69

HIT Human Intelligence Task. 5, 11, 14, 15, 51, 56, 59, 63, 65, 66

HTML Hyper-Text Markup Language. 6, 26, 27, 46, 66

HTTP Hypertext Transfer Protocol. 3, 22, 29, 30, 42, 58, 70

Java An object-oriented desktop programming language. 6

- JavaScript* A client-side programming language which is interpreted on the client side. 5, 6, 16, 22, 25, 27, 42, 70
- jQuery* A JavaScript library to provide simple commands for event handling communication and HTML manipulation. 27
- JSON* JavaScript Object Notation. 22, 45, 46
- JsPlumb* A JavaScript toolkit for drawing connections and flows on a canvas. 27, 28
- M-Turk* M-Turk or Mechanical Turk is the crowd-sourcing platform of Amazon, see <http://www.mturk.com/>. vi, 5, 6, 10, 11, 14, 16, 29, 31, 42–44, 51, 56, 57, 61, 71
- macro-task* A complex job that is composed of multiple smaller micro-tasks that are linked in a specific way. vi, 1, 4, 5, 19, 22, 23, 31, 37, 51–53, 61, 63–66, 69
- Majority Voting* In *majority voting*, a group of workers can select one item among a list. The item which is picked most often by the crowd will then be used for further processing. 4, 6, 37, 49, 50, 52, 59, 61, 101, 104
- MapReduce* An algorithm that distributes a possibly calculation demanding tasks and aggregates the results of those tasks. 4, 5
- Metadata* Metadata is information about data, i.e. a description of the data itself. 11, 23
- micro-task* A simple/ short task that is crowd-sourced as part of a bigger, encompassing task. 1, 4–6, 9, 11, 20, 35, 37, 48, 51, 53, 69
- MIT* Massachusetts Institute of Technology. 28
- MobileWorks* A crowd-sourcing platform founded by the University of California Berkeley US. 16–18, 29, 42
- MVC* Model-View-Control. 24, 26
- MVVM* Model-View-ViewModel. 26
- MySQL* An open-source relational database system. 25
- OAuth* An open protocol used for user authentication in web services. 70
- PDO* PHP Data Objects. 25, 42
- Perl* An interpreted scripted programming language. 75
- PHP* A server-side programming language which is interpreted on the server side. 4, 6, 17, 24, 42, 45

Glossary

PostgreSQL An open-source relational database system. 25

Python An open-source object-oriented scripting programming language. 5, 17

Qurk A system that processes declarative queries with crowd-sourced join and sort operations. 6

RDBMS Relational Database Management System. 25, 42

REST Representational State Transfer. 3, 4, 22, 27, 32, 44, 70

Rhino An open-source implementation of a JavaScript interpreter <https://developer.mozilla.org/en-US/docs/Rhino>. 6

Ruby An object-oriented programming language inspired by Perl. 4

SOAP Simple Object Access Protocol. 3, 22

SVG Scalable Vector Graphics. 28

TurKit A toolkit to support programming a series of M-Turk tasks. 5, 6

Twitter A internet-based service to publish short messages that can be read and followed by the public, see www.twitter.com. 11

URI Uniform Resource Identifier. 3

URL Uniform Resource Locator. 13, 34, 35, 37, 42–44, 46, 48, 50, 58, 59

XML Extensible Markup Language. 16, 22

Yii framework An MVC framework written in php that allows to build structured web applications. 24, 25

A. Bibliography

- [Allamaraju, 2010] Allamaraju, S. (2010). *RESTful Web Services Cookbook*. O'Reilly Media, Inc.
- [Benkler and Nissenbaum, 2006] Benkler, Y. and Nissenbaum, H. (2006). *Commons-based Peer Production and Virtue*. Blackwell Publishing.
- [Bernstein et al., 2010] Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. (2010). Soylent: A word processor with a crowd inside. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, New York, NY, USA. ACM.
- [Brabham, 2011] Brabham, D. C. (2011). *Crowdsourcing: A Model for Leveraging On-line Communities*. University of North Carolina at CHapel Hill.
- [Buhrmester et al., 2011] Buhrmester, M., Kwang, T., and Gosling, S. D. (2011). *Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality Data?* University of Texas at Austin.
- [Cudré-Mauroux and Demartini, 2012] Cudré-Mauroux, P. and Demartini, G. (2012). Social computing. Lecture “Social Computing”, University of Fribourg.
- [Demartini et al., 2012] Demartini, G., Difallah, D. E., and Cudré-Mauroux, P. (2012). Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 469–478, New York, NY, USA. ACM.
- [Dickens, 1859] Dickens, C. (1859). *A Tale of Two Cities*. London: Chapman & Hall.
- [Difallah et al., 2013] Difallah, D. E., Demartini, G., and Cudré-Mauroux, P. (2013). Pick-a-crowd: Tell me what you like, and i'll tell you what to do. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 367–374, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
- [Fielding, 2000] Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. UNIVERSITY OF CALIFORNIA, IRVINE.
- [Gardner et al., 2012] Gardner, P., Maffeis, S., and Smith, G. (2012). Towards a program logic for javascript. -.

- [Inc., 2012] Inc., C. (2012). CrowdFlower Handbook. Handbook for developers.
- [Kittur et al., 2012a] Kittur, A., Khamkar, S., André, P., and Kraut, R. E. (2012a). *CrowdWeaver: Visually Managing Complex Crowd Work*. Carnegie Mellon University.
- [Kittur et al., 2012b] Kittur, A., Khamkar, S., André, P., and Kraut, R. (2012b). CrowdWeaver: Visually Managing Complex Crowd Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW '12*, pages 1033–1036, New York, NY, USA. ACM.
- [Kittur et al., 2013a] Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. (2013a). The future of crowd work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work, CSCW '13*, pages 1301–1318, New York, NY, USA. ACM.
- [Kittur et al., 2013b] Kittur, A., Nickerson, J. V., Bernstein, M. S., Gerber, E. M., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. J. (2013b). *The Future of Crowd Work*. Proceeding CSCW '13, San Antonio, Texas, USA.
- [Kittur et al., 2011a] Kittur, A., Smus, B., Khamkar, S., and Kraut, R. E. (2011a). *CrowdForge: Crowdsourcing Complex Work*. Carnegie Mellon University.
- [Kittur et al., 2011b] Kittur, A., Smus, B., Khamkar, S., and Kraut, R. E. (2011b). CrowdForge: Crowdsourcing Complex Work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11*, pages 43–52, New York, NY, USA. ACM.
- [Kulkarni et al., 2012] Kulkarni, A., Gutheim, P., Narula, P., Rolnitzky, D., Parikh, T., and Hartmann, B. (2012). *MobileWorks: Designing for Quality in a Managed Crowdsourcing Architecture*. University of California, Berkeley.
- [Law and Zhang, 2011] Law, E. and Zhang, H. (2011). Towards large-scale collaborative planning: Answering high-level search queries using human computation. In *AAAI*.
- [Little et al., 2010a] Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. (2010a). *TurKit: Human Computation Algorithms on Mechanical Turk*. Proceedings of the 23rd annual ACM symposium on User interface software and technology.
- [Little et al., 2010b] Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. (2010b). Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology, UIST '10*, pages 57–66, New York, NY, USA. ACM.
- [Marcus et al., 2011a] Marcus, A., Wu, E., Karger, D., Madden, S., and Miller, R. (2011a). Human-powered sorts and joins. *Proc. VLDB Endow.*, 5(1):13–24.

- [Marcus et al., 2011b] Marcus, A., Wu, E., Karger, D., Madden, S., and Miller, R. (2011b). *Human-powered Sorts and Joins*. MIT Computer Science and Artificial Intelligence Laboratory.
- [of Waterloo, 2012] of Waterloo, U. (2012). *Human Participant Research Guidelines*. University of Waterloo.
- [Quinn and Bederson, 2011] Quinn, A. J. and Bederson, B. B. (2011). Human computation: A survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, New York, NY, USA. ACM.
- [Rahaman et al., 2006] Rahaman, M. A., Schaad, A., and Rits, M. (2006). Towards secure soap message exchange in a soa. In *Proceedings of the 3rd ACM Workshop on Secure Web Services*, SWS '06, pages 77–84, New York, NY, USA. ACM.
- [Raykar et al., 2010] Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322.
- [Richardson and Ruby, 2007] Richardson, L. and Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media, Inc.
- [Ross et al., 2010] Ross, J., Zaldivar, A., Irani, L., Tomlinson, B., and Silberman, M. S. (2010). *Who are the Crowdworkers? Shifting Demographics in Mechanical Turk*. Proceeding CHI EA '10 CHI '10 Extended Abstracts on Human Factors in Computing Systems.
- [Zhu et al., 2012] Zhu, S., Kane, S. K., Feng, J., and Sears, A. (2012). *A Crowdsourcing Quality Control Model for Tasks Distributed in Parallel*. Conference on Human Factory in Computing Systems.

B. Referenced Web Resources

- [1] Amazon. Introduction to MTurk for requesters, 2013. <https://requester.mturk.com/tour> (accessed: 26. July 2013).
- [2] Adam DuVander. The Next Wave? Enterprises Moving SOAP to REST, 2012. <http://blog.programmableweb.com/2012/03/22/the-next-wave-enterprises-moving-soap-to-rest/> (accessed: 16. January 2014).
- [3] Google. Browser support of AngularJS, 2013. <http://docs.angularjs.org/misc/faq> (accessed: 12. October 2013).
- [4] Google. Internet Explorer support of AngularJS, 2013. <http://docs.angularjs.org/guide/ie> (accessed: 12. October 2013).
- [5] AngularJs/ Google. Documentation about data binding in AngularJs, 2013. <http://docs.angularjs.org/guide/concepts#runtime> (accessed: 28. September 2013).
- [6] AngularJs/ Google. Implementation of data binding in AngularJs, 2013. <https://github.com/angular/angular.js/blob/ca30fce28ca13284bfa1c926e810ed75cdcde499/src/ng/directive/input.js#L380> (accessed: 28. September 2013).
- [7] AngularJs/ Google. Source code of AngularJs, 2013. <https://github.com/angular/angular> (accessed: 28. September 2013).
- [8] Jeff Howe. Crowdsourcing: A Definition, 2006. http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html (accessed: 28. July 2013).
- [9] Yii Software LLC. Database support of the Yii framework, 2013. <http://www.yiiframework.com/doc/guide/1.1/en/database.dao> (accessed: 12. October 2013).
- [10] Yii Software LLC. Website of the Yii 2 framework, 2013. <https://github.com/yiisoft/yii2> (accessed: 12. October 2013).
- [11] MobileWorks. About Mobile Works, 2013. <https://www.mobileworks.com/company/> (accessed: 25. October 2013).
- [12] MobileWorks. Developers Guide for MobileWorks, 2013. <https://www.mobileworks.com/developers/concepts/> (accessed: 25. October 2013).

B Referenced Web Resources

- [13] Oxford University Press. Oxford Dictionary, 2013. <http://www.oxforddictionaries.com/definition/english/crowdsource> (accessed: 23. October 2013).
- [14] Oxford University Press. Oxford Dictionary, 2014. <http://www.oxforddictionaries.com/definition/english/API?q=api> (accessed: 16. January 2014).
- [15] David Romeo. Presentation of the JSON format, 2013. <http://www.json.com/> (accessed: 29. September 2013).
- [16] W3 Techs. Statistics about AngularJs, 2013. <http://w3techs.com/technologies/details/js-angularjs/all/all> (accessed: 28. September 2013).
- [17] W3C. ASP.NET MVC tutorial, 2013. http://www.w3schools.com/aspnet/mvc_intro.asp (accessed: 27. September 2013).
- [18] W3C. JavaScript supporting browser, 2013. <http://www.w3resource.com/javascript/introduction/javascript-version.php> (accessed: 27. September 2013).

C. Experiment Results

C.1. Experiment One: Traditional versus WebApplication Approach

C.1.1. Macro-Task A: Translation

Original Text Charles Dickens’ “A Tale of Two Cities”, Chapter One: The Period

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way – in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever.

It was the year of Our Lord one thousand seven hundred and seventy-five. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster. Even the Cock-lane ghost had been laid only a round dozen of years, after rapping out its messages, as the spirits of this very year last past (supernaturally deficient in originality) rapped out theirs. Mere messages in the earthly order of events had lately come to the English Crown and People, from a congress of British subjects in America: which, strange to relate, have proved more important to the human race than any communications yet received through any of the chickens of the Cock-lane brood.

France, less favoured on the whole as to matters spiritual than her sister of the shield and trident, rolled with exceeding smoothness down hill, making paper money and spending it. Under the guidance of her Christian pastors, she entertained herself, besides, with such humane achievements as sentencing a youth to have his hands cut off, his tongue torn out with pincers, and his body burned alive, because he had not kneeled down in the rain to do honour to a dirty procession of monks which passed within his view, at a distance of some fifty or sixty yards. It is likely enough that, rooted in the woods

of France and Norway, there were growing trees, when that sufferer was put to death, already marked by the Woodman, Fate, to come down and be sawn into boards, to make a certain movable framework with a sack and a knife in it, terrible in history. It is likely enough that in the rough outhouses of some tillers of the heavy lands adjacent to Paris, there were sheltered from the weather that very day, rude carts, bespattered with rustic mire, snuffed about by pigs, and roosted in by poultry, which the Farmer, Death, had already set apart to be his tumbrils of the Revolution. But, that Woodman and that Farmer, though they work unceasingly, work silently, and no one heard them as they went about with muffled tread: the rather, forasmuch as to entertain any suspicion that they were awake, was to be atheistical and traitorous.

In England, there was scarcely an amount of order and protection to justify much national boasting. Daring burglaries by armed men, and highway robberies, took place in the capital itself every night; families were publicly cautioned not to go out of town without removing their furniture to upholsterers' warehouses for security; the highwayman in the dark was a City tradesman in the light, and, being recognised and challenged by his fellow-tradesman whom he stopped in his character of 'the Captain', gallantly shot him through the head and rode away; the mail was waylaid by seven robbers, and the guard shot three dead, and then got shot dead himself by the other four, 'in consequence of the failure of his ammunition': after which the mail was robbed in peace; that magnificent potentate, the Lord Mayor of London, was made to stand and deliver on Turnham Green, by one highwayman, who despoiled the illustrious creature in sight of all his retinue; prisoners in London gaols fought battles with their turnkeys, and the majesty of the law fired blunderbusses in among them, loaded with rounds of shot and ball; thieves snipped off diamond crosses from the necks of noble lords at Court drawing-rooms; musketeers went into St Giles's, to search for contraband goods, and the mob fired on the musketeers, and the musketeers fired on the mob; and nobody thought any of these occurrences much out of the common way. In the midst of them, the hangman, ever busy and ever worse than useless, was in constant requisition; now, stringing up long rows of miscellaneous criminals; now, hanging a housebreaker on Saturday who had been taken on Tuesday; now, burning people in the hand at Newgate by the dozen, and now burning pamphlets at the door of Westminster Hall; to-day, taking the life of an atrocious murderer, and to-morrow of a wretched pilferer who had robbed a farmer's boy of sixpence.

All these things, and a thousand like them, came to pass in and close upon the dear old year one thousand seven hundred and seventy-five. Environed by them, while the Woodman and the Farmer worked unheeded, those two of the large jaws, and those other two of the plain and the fair faces, trod with stir enough, and carried their divine rights with a high hand. Thus did the year one thousand seven hundred and seventy-five conduct their Greatnesses, and myriads of small creatures – the creatures of this chronicle among the rest – along the roads that lay before them.

Run 1

Split the original text into five parts. Excluding translation validation.

Step	Time	Total time
Define the macro-task and design the flow composition	04:53	04:53
Publish split task	00:29	05:22
Approve split task and published verification task	01:07	06:29
Approve split validation task, approved split task, create new CS-Tasks based on split text	00:27	06:56
Publish translation tasks	00:32	07:28
Approve translation tasks	00:47	08:15

Table C.1.: Exp. 1 A I: execution time - WebApplication Approach

WebApplication Approach Split text

1. It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way – in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only. There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever. It was the year of Our Lord one thousand seven hundred and seventy-five.
2. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster. Even the Cock-lane ghost had been laid only a round dozen of years, after rapping out its messages, as the spirits of this very year last past (supernaturally deficient in originality) rapped out theirs. Mere messages in the earthly order of events had lately come to the English Crown and People, from a congress of British subjects in America: which, strange to relate, have proved more important to the human race than any communications yet received through any of the chickens of the Cock-lane brood. France, less favoured on the whole as to matters spiritual than her sister of the shield and trident, rolled with exceeding smoothness down hill, making paper money and spending it. Under the guidance of her Christian pastors, she entertained herself, besides, with such humane achievements as sentencing a youth to have his hands cut off, his tongue torn out with pincers, and his body burned alive, because he had not kneeled down in the

rain to do honour to a dirty procession of monks which passed within his view, at a distance of some fifty or sixty yards.

3. It is likely enough that, rooted in the woods of France and Norway, there were growing trees, when that sufferer was put to death, already marked by the Woodman, Fate, to come down and be sawn into boards, to make a certain movable framework with a sack and a knife in it, terrible in history. It is likely enough that in the rough outhouses of some tillers of the heavy lands adjacent to Paris, there were sheltered from the weather that very day, rude carts, bespattered with rustic mire, snuffed about by pigs, and roosted in by poultry, which the Farmer, Death, had already set apart to be his tumbrils of the Revolution. But, that Woodman and that Farmer, though they work unceasingly, work silently, and no one heard them as they went about with muffled tread: the rather, forasmuch as to entertain any suspicion that they were awake, was to be atheistical and traitorous. In England, there was scarcely an amount of order and protection to justify much national boasting.
4. Daring burglaries by armed men, and highway robberies, took place in the capital itself every night; families were publicly cautioned not to go out of town without removing their furniture to upholsterers' warehouses for security; the highwayman in the dark was a City tradesman in the light, and, being recognised and challenged by his fellow-tradesman whom he stopped in his character of 'the Captain', gallantly shot him through the head and rode away; the mail was waylaid by seven robbers, and the guard shot three dead, and then got shot dead himself by the other four, 'in consequence of the failure of his ammunition': after which the mail was robbed in peace; that magnificent potentate, the Lord Mayor of London, was made to stand and deliver on Turnham Green, by one highwayman, who despoiled the illustrious creature in sight of all his retinue; prisoners in London gaols fought battles with their turnkeys, and the majesty of the law fired blunderbusses in among them, loaded with rounds of shot and ball; thieves snipped off diamond crosses from the necks of noble lords at Court drawing-rooms; musketeers went into St Giles's, to search for contraband goods, and the mob fired on the musketeers, and the musketeers fired on the mob; and nobody thought any of these occurrences much out of the common way.
5. . In the midst of them, the hangman, ever busy and ever worse than useless, was in constant requisition; now, stringing up long rows of miscellaneous criminals; now, hanging a housebreaker on Saturday who had been taken on Tuesday; now, burning people in the hand at Newgate by the dozen, and now burning pamphlets at the door of Westminster Hall; to-day, taking the life of an atrocious murderer, and to-morrow of a wretched pilferer who had robbed a farmer's boy of sixpence. All these things, and a thousand like them, came to pass in and close upon the dear old year one thousand seven hundred and seventy-five. Environed by them, while the Woodman and the Farmer worked unheeded, those two of the large jaws, and those other two of the plain and the fair faces, trod with stir enough, and

carried their divine rights with a high hand. Thus did the year one thousand seven hundred and seventy-five conduct their Greatnesses, and myriads of small creatures – the creatures of this chronicle among the rest – along the roads that lay before them.

Translations

1. Fue el mejor de los tiempos, era el peor de los tiempos, era la edad de la sabiduría, era la edad de la estupidez, era la época de la fe, era la época de la incredulidad, era la estación de la Luz, que fue la época de la oscuridad, era la primavera de la esperanza, era el invierno de la desesperación, teníamos todo ante nosotros, no teníamos nada ante nosotros, todos íbamos directo al cielo, todos íbamos directo a la inversa - en resumen, el período fue tan lejos como el actual período, que algunos de sus autoridades más ruidosos insistieron en que se recibieron, para bien o para mal, en el grado superlativo de la única comparación. Había un rey con una gran quijada y una reina con una cara plana, en el trono de Inglaterra, había un rey con una gran quijada y una reina con un bello rostro, en el trono de Francia. En ambos países era más claro que el cristal para los señores de los cotos estatales de los panes y los peces, que las cosas, en general, se establecieron para siempre. Era el año de Nuestro Señor un mil setecientos setenta y cinco.
2. Revelaciones espirituales fueron concedidas a Inglaterra en ese período favorecida, como en este. Sra. Southcott había alcanzado recientemente su cumpleaños bendito cinco y veinte, de los cuales un profética privado en la Guardia Real se había anunciado la aparición sublime con el anuncio de que se hicieron los arreglos para la deglución de Londres y Westminster. Incluso el fantasma Cock-lane se había establecido sólo una docena ronda del año, después de rapear sus mensajes, como los espíritus de este mismo año pasado pasado (sobrenaturalmente deficiente en la originalidad) rapped fuera suyo. Mensajes Mere en el orden terrenal de los acontecimientos habían llegado últimamente a la Corona y el pueblo Inglés, a partir de un congreso de súbditos británicos en América: que, por extraño que parezca, han demostrado ser más importante para la raza humana que cualquier comunicación ha recibido a través de cualquiera de los pollos de la nidada Cock-lane. Francia, menos favorecida en general como a los asuntos espirituales que su hermana del escudo y tridente, rodó con gran suavidad cuesta abajo, por lo que el papel moneda y gastarlo. Bajo la guía de sus pastores cristianos, se entretenía, además, con este tipo de logros humanos como sentenciar a un joven para tener las manos cortadas, con la lengua arrancada con tenazas, y su cuerpo fue quemado vivo, porque no se había puesto de rodillas en el lluvia para honrar a una procesión de monjes sucia que pasó a su juicio, a una distancia de unos cincuenta o sesenta metros.
3. Es probable que, arraigada en los bosques de Francia y Noruega, que cada vez los árboles, que sufre cuando se puso a la muerte, ya marcada por el Woodman, suerte, y se cortan en las placas, a realizar un cierto marco móvil con un saco y un cuchillo en el mismo, terrible en la historia. Es probable que en el áspero dependencias de

algunos labradores de las grandes tierras al lado de París, había protegido de la intemperie, a partir del mismo día, grosero, carros rústicos bespattered con lodo, sofocadas por los cerdos, y por las aves de corral en roosted, que el agricultor, la muerte, ya había puesto aparte de ser su rodado carretas de la Revolución. Pero, que Woodman y agricultor, a pesar de que trabajar sin cesar, trabajan en forma silenciosa, sin que nadie les oyó como lo fueron acalladas con ancho de vía: el más grande y a entretener a cualquier sospecha de que se despierta, se atheistical y traidor. En Inglaterra, hubo apenas una cantidad del orden y la protección para justificar tanto nacional ofreciendo.

4. Audaces robos por hombres armados, y los robos en la carretera, se llevó a cabo en la propia capital todas las noches; las familias se advirtió públicamente para que no salgan de la ciudad, sin eliminar sus muebles para tapiceros los depósitos de seguridad; el bandido en la oscuridad era una ciudad comerciante a la luz, y que, al ser reconocido y fue cuestionado por sus compañeros de comerciante quien se detuvo en su carácter de "el Capitán", con nobleza le disparó en la cabeza y rode, el correo fue asaltado por siete ladrones, y los de la guardia disparó tres muertos, y, a continuación, se ha muerto de un disparo por los otros cuatro, "como consecuencia de la falta de sus municiones": después de lo cual el correo fue robado en paz; ese magnífico potentado, el Alcalde de Londres, Se hizo entrega a ponerse de pie y en Turnham Green, por un bandido, que despojados del ilustre criatura a la vista de toda su comitiva, y los presos en las cárceles Londres batallas con sus turnkeys, y la majestad de la ley dispararon trabucazosâ en entre ellos, cargados con rondas de disparos y la bola; los ladrones de diamantes retesta cruza de los cuellos de nobles señores en el Tribunal de las habitaciones; mosqueteros entró en St Giles's, para la búsqueda de mercancías de contrabando, y dispararon a la multitud los mosqueteros, y los mosqueteros dispararon sobre la multitud; y a nadie se le ocurrió alguna de estas apariciones de la manera habitual.
5. . En medio de ellos, el verdugo, siempre ocupado y cada vez es peor que inútil, se mantenía en constante solicitud; ahora, el tendido de largas filas de varios criminales; ahora, colgando un housebreaker el sábado que habían sido tomadas el martes; ahora, quema de personas en la mano en Newgate por decenas, y quema actualmente folletos a la puerta de Westminster Hall; a día, tomando la vida de una atroz asesino, y después de un desafortunado golpe que le había robado de un agricultor joven de apenas unos centímetros. Todas estas cosas, y mil al igual que ellos, vino a pasar en y cerca de la vieja y querida año de mil setecientos setenta y cinco. Colamente por ellos, mientras que el Woodman y el agricultor ha trabajado sin tregua los dos de los grandes mandíbulas, y los otros dos de la llanura, y enfrenta la feria, pisaron con agitar lo suficiente, y llevaron sus derechos divinos con una mano alta. Por lo que el año de mil setecientos setenta y cinco realizar sus Greatnesses, y multitud de pequeñas criaturas, las criaturas de esta crónica entre el resto, a lo largo de los caminos que les.

Step	Time	Total time
Define and publish the split task	01:31	01:31
Approve split task	00:25	01:56
Download and store the split texts	01:04	03:00
Prepare and publish the validation task	03:54	06:54
Approve text splitting validation task	00:16	07:10
Publish the translation tasks	03:37	10:47
Approve the translation tasks	01:09	11:56
Fetch the translated texts and merge them	03:03	14:59

Table C.2.: Exp. 1 A I: execution time - Manual Approach

Manual Approach Split text

1. It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way – in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only. There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever. It was the year of Our Lord one thousand seven hundred and seventy-five.
2. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster. Even the Cock-lane ghost had been laid only a round dozen of years, after rapping out its messages, as the spirits of this very year last past (supernaturally deficient in originality) rapped out theirs. Mere messages in the earthly order of events had lately come to the English Crown and People, from a congress of British subjects in America: which, strange to relate, have proved more important to the human race than any communications yet received through any of the chickens of the Cock-lane brood. France, less favoured on the whole as to matters spiritual than her sister of the shield and trident, rolled with exceeding smoothness down hill, making paper money and spending it. Under the guidance of her Christian pastors, she entertained herself, besides, with such humane achievements as sentencing a youth to have his hands cut off, his tongue torn out

with pincers, and his body burned alive, because he had not kneeled down in the rain to do honour to a dirty procession of monks which passed within his view, at a distance of some fifty or sixty yards.

3. It is likely enough that, rooted in the woods of France and Norway, there were growing trees, when that sufferer was put to death, already marked by the Woodman, Fate, to come down and be sawn into boards, to make a certain movable framework with a sack and a knife in it, terrible in history. It is likely enough that in the rough outhouses of some tillers of the heavy lands adjacent to Paris, there were sheltered from the weather that very day, rude carts, bespattered with rustic mire, snuffed about by pigs, and roosted in by poultry, which the Farmer, Death, had already set apart to be his tumbrils of the Revolution. But, that Woodman and that Farmer, though they work unceasingly, work silently, and no one heard them as they went about with muffled tread: the rather, forasmuch as to entertain any suspicion that they were awake, was to be atheistical and traitorous. In England, there was scarcely an amount of order and protection to justify much national boasting.
4. Daring burglaries by armed men, and highway robberies, took place in the capital itself every night; families were publicly cautioned not to go out of town without removing their furniture to upholsterers' warehouses for security; the highwayman in the dark was a City tradesman in the light, and, being recognised and challenged by his fellow-tradesman whom he stopped in his character of 'the Captain', gallantly shot him through the head and rode away; the mail was waylaid by seven robbers, and the guard shot three dead, and then got shot dead himself by the other four, 'in consequence of the failure of his ammunition': after which the mail was robbed in peace; that magnificent potentate, the Lord Mayor of London, was made to stand and deliver on Turnham Green, by one highwayman, who despoiled the illustrious creature in sight of all his retinue; prisoners in London gaols fought battles with their turnkeys, and the majesty of the law fired blunderbusses in among them, loaded with rounds of shot and ball; thieves snipped off diamond crosses from the necks of noble lords at Court drawing-rooms; musketeers went into St Giles's, to search for contraband goods, and the mob fired on the musketeers, and the musketeers fired on the mob; and nobody thought any of these occurrences much out of the common way.
5. In the midst of them, the hangman, ever busy and ever worse than useless, was in constant requisition; now, stringing up long rows of miscellaneous criminals; now, hanging a housebreaker on Saturday who had been taken on Tuesday; now, burning people in the hand at Newgate by the dozen, and now burning pamphlets at the door of Westminster Hall; to-day, taking the life of an atrocious murderer, and to-morrow of a wretched pilferer who had robbed a farmer's boy of sixpence. All these things, and a thousand like them, came to pass in and close upon the dear old year one thousand seven hundred and seventy-five. Environed by them, while the Woodman and the Farmer worked unheeded, those two of the large jaws,

and those other two of the plain and the fair faces, trod with stir enough, and carried their divine rights with a high hand. Thus did the year one thousand seven hundred and seventy-five conduct their Greatnesses, and myriads of small creatures – the creatures of this chronicle among the rest – along the roads that lay before them.

Translations

1. Era el mejor de tiempos, era el peor de tiempos, era la edad de sabiduría, era la edad de tontería, era la época de creencia, era la época de incredulidad, era la temporada de Luz, era la temporada de Oscuridad, era la primavera de esperanza, era el invierno de desesperación, no teníamos todo antes de nosotros, no teníamos nada antes de nosotros, íbamos todos directo al Cielo, íbamos todos directos el otro camino – en resumen el período hasta ahora pareció al período presente, que algunas de sus autoridades más ruidosas insistieron en el que recibe, para siempre o para el mal, en el nivel superlativo de la comparación sólo. Había un rey con una mandíbula grande y una reina con una cara clara, en el trono de Inglaterra; había un rey con una mandíbula grande y una reina con una cara justa, en el trono de Francia. En ambos países estaba más claro que el cristal para los señores de los vedados(conservas) estatales de los panes y peces, que las cosas en general se colocaron para siempre. Era el año de Nuestro Señor mil setecientos setenta y cinco.
2. Revelaciones espirituales se concedieron a Inglaterra en ese período favorecido, como en este. La señora Southcott había logrado recientemente su cumpleaños bendito cinco-y-vigésimo, de quien un privado profético en la guardia de la vida había anunciado el aspecto sublime anunciando que se hicieron arreglos para la deglución de Londres y Westminster. Incluso el fantasma de Cock-lane había sido colocado solamente una docena redonda de años, después de criticar mordazmente sus mensajes, como los espíritus de este mismo año pasado pasado (sobrenaturalmente deficientes en originalidad) golpeó a ellos. Meros mensajes en el orden terrenal de los hechos habían llegado últimamente a la corona inglesa y la gente, de un Congreso de súbditos británicos en América: que extraño relacionar, resultó más importante a la raza humana que cualquier comunicación todavía recibió a través de cualquiera de los pollos de la cría del gallo-lane. Francia, menos favorecida en conjunto en cuanto a los asuntos espirituales que su hermana del escudo y Tridente, rodado con superior suavidad cuesta abajo, hacer dinero y gastarlo. Bajo la dirección de sus pastores cristianos, entretuvo, además, con tales logros humanitarios como condenar a un joven para tener sus manos cortadas, su lengua arrancada con pinzas y su cuerpo quemado vivo, porque él no había se arrodilló en la lluvia para hacer honor a una procesión sucia de los monjes que han aprobado dentro de su punto de vista, a una distancia de unos cincuenta o sesenta metros.
3. Es bastante probable que, enraizada en los bosques de Francia y Noruega, no crecían árboles, cuando esa víctima fue condenado a muerte, ya marcado por la

Woodman, Fate, a bajar y ser aserrada en tablones, para hacer un determinado móvil marco con un saco y un cuchillo en él, terrible en la historia. Es bastante probable que en las dependencias aproximadas de algunos labradores de las tierras pesadas adyacentes a París, allí estaban protegidas del tiempo, ese mismo día, carros groseras, salpicado de fango rústico, apagaron sobre los cerdos y las aves de corral en roosted, que el granjero, Muerte, ya se había apartado para ser sus carretas de la Revolución. Pero, eso Woodman y que Farmer, aunque trabajan sin cesar, trabajan en silencio, y nadie los escuchó, ya que se ocuparon de la banda de rodamiento amortiguado: el lugar, por cuanto para entretener a cualquier sospecha de que estaban despiertos, era ser ateo y traidor. En Inglaterra, apenas había una cantidad de orden y protección para justificar tanto la jactancia nacional.

4. Robos audaces por parte de hombres armados, y robos de la carretera, se llevaron a cabo en la capital misma todas las noches, las familias se advierte públicamente a no salir de la ciudad sin necesidad de retirar sus muebles a los almacenes tapiceros para la seguridad, y el salteador de caminos en la oscuridad era un comerciante de la ciudad en la luz, y, de ser reconocido y cuestionado por su compañero de comerciante a quien se detuvo en su carácter de ' capitán ', galantemente le dispararon en la cabeza y se alejó, el correo fue atacado por siete ladrones, y el guardia disparó tres muertos, y luego le dispararon muerto a sí mismo por los otros cuatro, ' como consecuencia del fracaso de su munición Â » : después de que el correo fue robado en paz, esa magnífica potentado, el alcalde de Londres, fue hecho para soportar y cumplir Turnham ladrones cortadas con tijeras; verdes, por un salteador de caminos, que despojó al ilustre criatura en presencia de todo su séquito, prisioneros en cárceles de Londres luchó batallas con sus carceleros, y la majestad de la ley disparó trabucos en lugar de ellas, cargada con cartuchos de tiro y la pelota off cruces de diamantes de los cuellos de los nobles señores en los salones de la Corte; mosqueteros entraban en St Giles, para buscar artículos de contrabando, y la multitud dispararon contra los mosqueteros, y los mosqueteros dispararon contra la multitud, y nadie pensó alguna de estas repeticiones mucho fuera del camino común.
5. En medio de ellos, el verdugo, siempre ocupado y cada vez es peor que inútil, se mantenía en constante solicitud; ahora, el tendido de largas filas de varios criminales; ahora, colgando un housebreaker el sábado que habían sido tomadas el martes; ahora, quema de personas en la mano en Newgate por decenas, y quema actualmente folletos a la puerta de Westminster Hall; a día, tomando la vida de una atroz asesino, y después de un desafortunado golpe que le había robado de un agricultor joven de apenas unos centímetros. Todas estas cosas, y mil al igual que ellos, vino a pasar en y cerca de la vieja y querida año de mil setecientos setenta y cinco. Colamente por ellos, mientras que el Woodman y el agricultor ha trabajado sin tregua los dos de los grandes mandíbulas, y los otros dos de la llanura, y enfrenta la feria, pisaron con agitar lo suficiente, y llevaron sus derechos divinos con una mano alta. Por lo que el año de mil setecientos setenta y cinco realizar sus

Step	Time	Total time
Define the macro-task and design the flow composition	06:14	06:14
Publish split task	00:15	06:29
Approve split task and published verification task	00:13	06:42
Approve split validation task, approve split task text	00:03	06:45
Create and publish translation tasks based on split	00:30	07:15
Approve translation tasks AND publish translation verification tasks	00:30	07:45
Approve/ reject translation verification tasks, re-publish rejected tasks	00:14	07:59
Approve translation tasks AND publish translation verification tasks	00:08	08:07
Approve translation validation task, approve split task text	00:09	08:16

Table C.3.: Exp. 1 A II: execution time - WebApplication Approach

Greatnesses, y multitud de pequeñas criaturas, las criaturas de esta crónica entre el resto, a lo largo de los caminos que les.

Run 2

Split the original text into eight parts. Including translation validation.

WebApplication Approach Split text

1. It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way – in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
2. There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever. It was the year of Our Lord one thousand seven hundred and seventy-five. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster.
3. Even the Cock-lane ghost had been laid only a round dozen of years, after rapping out its messages, as the spirits of this very year last past (supernaturally deficient

in originality) rapped out theirs. Mere messages in the earthly order of events had lately come to the English Crown and People, from a congress of British subjects in America: which, strange to relate, have proved more important to the human race than any communications yet received through any of the chickens of the Cock-lane brood. France, less favoured on the whole as to matters spiritual than her sister of the shield and trident, rolled with exceeding smoothness down hill, making paper money and spending it. Under the guidance of her Christian pastors,

4. she entertained herself, besides, with such humane achievements as sentencing a youth to have his hands cut off, his tongue torn out with pincers, and his body burned alive, because he had not kneeled down in the rain to do honour to a dirty procession of monks which passed within his view, at a distance of some fifty or sixty yards. It is likely enough that, rooted in the woods of France and Norway, there were growing trees, when that sufferer was put to death, already marked by the Woodman, Fate, to come down and be sawn into boards, to make a certain movable framework with a sack and a knife in it, terrible in history. It is likely enough that in the rough outhouses of some tillers of the heavy lands adjacent to Paris, there were sheltered from the weather that very day, rude carts, bespattered with rustic mire, snuffed about by pigs, and roosted in by poultry, which the Farmer, Death, had already set apart to be his tumbrils of the Revolution.
5. But, that Woodman and that Farmer, though they work unceasingly, work silently, and no one heard them as they went about with muffled tread: the rather, forasmuch as to entertain any suspicion that they were awake, was to be atheistical and traitorous. In England, there was scarcely an amount of order and protection to justify much national boasting. Daring burglaries by armed men, and highway robberies, took place in the capital itself every night; families were publicly cautioned not to go out of town without removing their furniture to upholsterers' warehouses for security;
6. the highwayman in the dark was a City tradesman in the light, and, being recognised and challenged by his fellow-tradesman whom he stopped in his character of 'the Captain', gallantly shot him through the head and rode away; the mail was waylaid by seven robbers, and the guard shot three dead, and then got shot dead himself by the other four, 'in consequence of the failure of his ammunition': after which the mail was robbed in peace; that magnificent potentate, the Lord Mayor of London, was made to stand and deliver on Turnham Green, by one highwayman, who despoiled the illustrious creature in sight of all his retinue; prisoners in London gaols fought battles with their turnkeys, and the majesty of the law fired blunderbusses in among them, loaded with rounds of shot and ball; thieves snipped off diamond crosses from the necks of noble lords at Court drawing-rooms;
7. musketeers went into St Giles's, to search for contraband goods, and the mob fired on the musketeers, and the musketeers fired on the mob; and nobody thought any of these occurrences much out of the common way. In the midst of them, the

hangman, ever busy and ever worse than useless, was in constant requisition; now, stringing up long rows of miscellaneous criminals; now, hanging a housebreaker on Saturday who had been taken on Tuesday; now, burning people in the hand at Newgate by the dozen, and now burning pamphlets at the door of Westminster Hall; to-day, taking the life of an atrocious murderer, and to-morrow of a wretched pilferer who had robbed a farmer's boy of sixpence.

8. All these things, and a thousand like them, came to pass in and close upon the dear old year one thousand seven hundred and seventy-five. Environed by them, while the Woodman and the Farmer worked unheeded, those two of the large jaws, and those other two of the plain and the fair faces, trod with stir enough, and carried their divine rights with a high hand. Thus did the year one thousand seven hundred and seventy-five conduct their Greatnesses, and myriads of small creatures – the creatures of this chronicle among the rest – along the roads that lay before them.

Translations

1. *Accepted* Fue el mejor de los tiempos, fue el peor de los tiempos, fue la edad de la sabiduría, fue la edad de la tontería, fue la época de la creencia, fue la época de la incredulidad, fue la temporada de Luz, fue la temporada de Oscuridad, fue la primavera de esperanza, fue el invierno de desesperación, teníamos todo delante de nosotros, todos íbamos directo al cielo, todos íbamos directo al otro lado -en pocas palabras, el período fue hasta el momento como el período actual, que algunos de sus más escandalosas autoridades insistieron en que fuera recibido, por bueno or por malo, en el grado superlativo de comparación solamente.
2. *Rejected* There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever. It was the year of Our Lord one thousand seven hundred and seventy-five. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster.

Accepted Había un rey con una gran quijada y una reina con una cara plana, en el trono de Inglaterra, había un rey con una gran quijada y una reina con un bello rostro, en el trono de Francia. En ambos países era más claro que el cristal para los señores de los cotos estatales de los panes y los peces, que las cosas, en general, se establecieron para siempre. Era el año de Nuestro Señor un mil setecientos setenta y cinco. Revelaciones espirituales fueron concedidas a Inglaterra en ese período favorecida, como en este. Sra. Southcott

había alcanzado recientemente su cumpleaños bendito cinco y veinte, de los cuales un profética privado en la Guardia Real se había anunciado la aparición sublime con el anuncio de que se hicieron los arreglos para la deglución de Londres y Westminster.

3. *Accepted* Incluso el fantasma Cock-lane se había establecido sólo una docena ronda del año, después de rapear sus mensajes, como los espíritus de este mismo año pasado pasado (sobrenaturalmente deficiente en la originalidad) rapped fuera suyo. Mensajes Mere en el orden terrenal de los acontecimientos habían llegado últimamente a la Corona y el pueblo Inglés, a partir de un congreso de súbditos británicos en América: que, por extraño que parezca, han demostrado ser más importante para la raza humana que cualquier comunicación ha recibido a través de cualquiera de los pollos de la nidada Cock-lane. Francia, menos favorecida en general como a los asuntos espirituales que su hermana del escudo y tridente, rodó con gran suavidad cuesta abajo, por lo que el papel moneda y gastarlo. Bajo la guía de sus pastores cristianos
4. *Accepted* ella se entretenía, además, con este tipo de logros humanos como sentenciar a un joven para tener las manos cortadas, con la lengua arrancada con tenazas, y su cuerpo fue quemado vivo, porque no se había puesto de rodillas bajo la lluvia para honrar a una procesión sucia de monjes que pasaban dentro de su punto de vista, a una distancia de unos cincuenta o sesenta metros. Es bastante probable que, enraizada en los bosques de Francia y Noruega, no crecían árboles, cuando esa víctima fue condenado a muerte, ya marcado por la Woodman, Fate, a bajar y ser aserrada en tablones, para hacer un determinado móvil marco con un saco y un cuchillo en él, terrible en la historia. Es bastante probable que en las dependencias aproximadas de algunos labradores de las tierras pesadas adyacentes a París, allí estaban protegidas del tiempo, ese mismo día, carros groseras, salpicado de fango rústico, apagaron sobre los cerdos y las aves de corral en roosted, que el granjero, Muerte, ya se había apartado para ser sus carretas de la Revolución.
5. *Accepted* Pero, eso Woodman y que Farmer, aunque trabajan sin cesar, trabajan en silencio, y nadie los escuchó, ya que se ocuparon de la banda de rodamiento amortiguado: el lugar, por cuanto para entretener a cualquier sospecha de que estaban despiertos, era ser ateo y traidor. En Inglaterra, apenas había una cantidad de orden y protección para justificar tanto la jactancia nacional. Robos audaces por parte de hombres armados, y robos de la carretera, se llevaron a cabo en la capital misma todas las noches, las familias se advierte públicamente a no salir de la ciudad sin necesidad de retirar sus muebles a los almacenes tapiceros 'para la seguridad;
6. *Accepted* the highwayman in the dark was a City tradesman in the light, and, being recognised and challenged by his fellow-tradesman whom he stopped in his character of 'the Captain', gallantly shot him through the head and

rode away; the mail was waylaid by seven robbers, and the guard shot three dead, and then got shot dead himself by the other four, 'in consequence of the failure of his ammunition': after which the mail was robbed in peace; that magnificent potentate, the Lord Mayor of London, was made to stand and deliver on Turnham Green, by one highwayman, who despoiled the illustrious creature in sight of all his retinue; prisoners in London gaols fought battles with their turnkeys, and the majesty of the law fired blunderbusses in among them, loaded with rounds of shot and ball; thieves snipped off diamond crosses from the necks of noble lords at Court drawing-rooms;

7. *Accepted* mosqueteros entraban en St Giles, para buscar artículos de contrabando, y la multitud dispararon contra los mosqueteros, y los mosqueteros dispararon contra la multitud, y nadie pensó alguna de estas repeticiones mucho fuera del camino común. En medio de ellos, el verdugo, siempre ocupado y siempre peor que inútil, estaba en constante requisición; ahora, encadenar largas filas de delincuentes diversos; ahora, colgando de un ladrón de casas el sábado que había sido llevado el martes, ahora, la quema de las personas en la mano a Newgate por docenas, y ahora quema de panfletos en la puerta de Westminster Hall; a día, tomando la vida de un asesino atroz, y mañana de un ladronzuelo miserable que había robado el hijo de un granjero de seis peniques.
8. *Rejected* Todas estas cosas, y mil como ellas, vinieron a pasar y atravesar el año mil setecientos setenta y cinco. Rodeados por ellas, el leñador y el granjero trabajaron sin parar, los dos de mandíbulas grandes, y los otros dos con sus rostros planos y justos, caminando con firmeza, cargando sus derechos divinos con la mano derecha. Así que el año mil setecientos setenta y cinco fue de grandeza y milagros y criaturas pequeñas - las criaturas de las crónicas - a lo largo de los caminos que los esperan.

Accepted Todas estas cosas, y mil como ellos, aconteció en y cerca de la querida año un mil setecientos setenta y cinco. Environed por ellos, mientras que el Leñador y el Farmer trabajaron en letra muerta, los dos de los grandes mandíbulas, y los otros dos de la llanura y las caras justas, pisó con agitación suficiente, y lleva a sus derechos divinos con mano poderosa. Así fue como el año un mil setecientos setenta y cinco conducen sus Grandezas, y miríadas de criaturas pequeñas - las criaturas de esta crónica entre el resto - a lo largo de los caminos que se extendía ante ellos.

Manual Approach Split text

1. It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way –

Step	Time	Total time
Define and publish the split task	01:23	01:23
Approve split task	00:10	01:33
Evaluate split task result	04:02	05:35
Re-publish assignment	00:12	05:47
Approve split task	00:13	06:00
Download and store split text	01:29	07:29
Prepare and publish the text splitting validation task	03:12	10:41
Approve text splitting validation task	00:11	10:52
Publish the translation tasks	04:48	15:40
Approve the translation tasks	01:50	17:30
Fetch the translated texts and “cache” them	03:55	21:25
Publish the translation validation tasks	06:45	28:10
Approve translation validation tasks	01:32	29:42
Analyse validation results	03:26	33:08
Re-publish rejected assignments	01:50	34:58
Approve the translation tasks	00:33	35:31
Fetch the translated texts and “cache” them	02:43	38:14
Publish the translation validation tasks	04:07	42:21
Approve translation validation tasks	00:35	42:56
Analyse validation results	01:58	44:54
Re-publish rejected assignments	00:58	45:52
Approve the translation tasks	00:11	46:03
Fetch the translated texts and “cache” them	01:01	47:04
Publish the translation validation tasks	01:57	49:01
Approve translation validation tasks	00:09	49:10
Analyse validation results	00:26	49:36
Merge the translated texts	01:31	51:07

Table C.4.: Exp 1 A II: execution time - Manual Approach

in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

2. There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever.
3. It was the year of Our Lord one thousand seven hundred and seventy-five. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster.
4. Even the Cock-lane ghost had been laid only a round dozen of years, after rapping out its messages, as the spirits of this very year last past (supernaturally deficient in originality) rapped out theirs. Mere messages in the earthly order of events had lately come to the English Crown and People, from a congress of British subjects in America: which, strange to relate, have proved more important to the human race than any communications yet received through any of the chickens of the Cock-lane brood.
5. France, less favoured on the whole as to matters spiritual than her sister of the shield and trident, rolled with exceeding smoothness down hill, making paper money and spending it. Under the guidance of her Christian pastors, she entertained herself, besides, with such humane achievements as sentencing a youth to have his hands cut off, his tongue torn out with pincers, and his body burned alive, because he had not kneeled down in the rain to do honour to a dirty procession of monks which passed within his view, at a distance of some fifty or sixty yards. It is likely enough that, rooted in the woods of France and Norway, there were growing trees, when that sufferer was put to death, already marked by the Woodman, Fate, to come down and be sawn into boards, to make a certain movable framework with a sack and a knife in it, terrible in history.
6. It is likely enough that in the rough outhouses of some tillers of the heavy lands adjacent to Paris, there were sheltered from the weather that very day, rude carts, bespattered with rustic mire, snuffed about by pigs, and roosted in by poultry, which the Farmer, Death, had already set apart to be his tumbrils of the Revolution. But, that Woodman and that Farmer, though they work unceasingly, work silently, and no one heard them as they went about with muffled tread: the rather, forasmuch as to entertain any suspicion that they were awake, was to be atheistical and traitorous.

7. In England, there was scarcely an amount of order and protection to justify much national boasting. Daring burglaries by armed men, and highway robberies, took place in the capital itself every night; families were publicly cautioned not to go out of town without removing their furniture to upholsterers' warehouses for security; the highwayman in the dark was a City tradesman in the light, and, being recognised and challenged by his fellow-tradesman whom he stopped in his character of 'the Captain', gallantly shot him through the head and rode away; the mail was waylaid by seven robbers, and the guard shot three dead, and then got shot dead himself by the other four, 'in consequence of the failure of his ammunition': after which the mail was robbed in peace; that magnificent potentate, the Lord Mayor of London, was made to stand and deliver on Turnham Green, by one highwayman, who despoiled the illustrious creature in sight of all his retinue; prisoners in London gaols fought battles with their turnkeys, and the majesty of the law fired blunderbusses in among them, loaded with rounds of shot and ball; thieves snipped off diamond crosses from the necks of noble lords at Court drawing-rooms; musketeers went into St Giles's, to search for contraband goods, and the mob fired on the musketeers, and the musketeers fired on the mob; and nobody thought any of these occurrences much out of the common way. In the midst of them, the hangman, ever busy and ever worse than useless, was in constant requisition; now, stringing up long rows of miscellaneous criminals; now, hanging a housebreaker on Saturday who had been taken on Tuesday; now, burning people in the hand at Newgate by the dozen, and now burning pamphlets at the door of Westminster Hall; to-day, taking the life of an atrocious murderer, and to-morrow of a wretched pilferer who had robbed a farmer's boy of sixpence.
8. All these things, and a thousand like them, came to pass in and close upon the dear old year one thousand seven hundred and seventy-five. Environed by them, while the Woodman and the Farmer worked unheeded, those two of the large jaws, and those other two of the plain and the fair faces, trod with stir enough, and carried their divine rights with a high hand. Thus did the year one thousand seven hundred and seventy-five conduct their Greatnesses, and myriads of small creatures – the creatures of this chronicle among the rest – along the roads that lay before them.

Translations

1. *Rejected* Fue el mejor de los tiempos, era el peor de los tiempos, era la edad de la sabiduría, era la edad de la estupidez, era la época de la fe, era la época de la incredulidad, era la estación de la Luz, que fue la época de la oscuridad, era la primavera de la esperanza, era el invierno de la desesperación, teníamos todo ante nosotros, no teníamos nada ante nosotros, todos íbamos directo al cielo, todos íbamos directo a la inversa - en resumen, el período fue tan lejos como el actual período, que algunos de sus autoridades más ruidosos insistieron en que se recibieron, para bien o para mal, en el grado superlativo de la única comparación.

Accepted Fue el mejor de los tiempos, era el peor de los tiempos, era la edad de la sabiduría, era la edad de la estupidez, era la época de la fe, era la época de la incredulidad, era la estación de la Luz, que fue la época de la oscuridad, era la primavera de la esperanza, era el invierno de la desesperación, teníamos todo ante nosotros, no teníamos nada ante nosotros, todos íbamos directo al cielo, todos íbamos directo a la inversa - en resumen, el período fue tan lejos como el actual período, que algunos de sus autoridades más ruidosos insistieron en que se recibieron, para bien o para mal, en el grado superlativo de la única comparación

2. *Accepted* Había un rey con una gran quijada y una reina con una cara plana, en el trono de Inglaterra, había un rey con una gran quijada y una reina con un bello rostro, en el trono de Francia. En ambos países era más claro que el cristal para los señores de los cotos estatales de los panes y los peces, que las cosas, en general, se establecieron para siempre.
3. *Rejected* Era el año de Nuestro Señor un mil setecientos setenta y cinco. Revelaciones espirituales fueron concedidas a Inglaterra en ese período favorecida, como en este. Sra. Southcott había alcanzado recientemente su cumpleaños bendito cinco y veinte, de los cuales un profética privado en la Guardia Real se había anunciado la aparición sublime con el anuncio de que se hicieron los arreglos para la deglución de Londres y Westminster.

Rejected Fue el año de Nuestro Señor de mil setecientos setenta y cinco. Las revelaciones espirituales son concedidas a Inglaterra en que favoreció al este. La Sra. Southcott habían logrado recientemente sus cinco-y-vigésimo cumpleaños feliz, de los cuales una profética en la vida privada guardias habían anunciado la sublime aparición mediante el anuncio de que se tomaron medidas para la deglución de Londres y Westminster.

Accepted Era el año de Nuestro Señor un mil setecientos setenta y cinco. Revelaciones espirituales fueron concedidas a Inglaterra en ese período favorecida, como en este. Sra. Southcott había alcanzado recientemente su cumpleaños bendito cinco y veinte, de los cuales un profética privado en la Guardia Real se había anunciado la aparición sublime con el anuncio de que se hicieron los arreglos para la deglución de Londres y Westminster.

4. *Rejected* Incluso el fantasma Cock-lane se había establecido sólo una docena ronda del año, después de rapear sus mensajes, como los espíritus de este mismo año pasado pasado (sobrenaturalmente deficiente en la originalidad) rapped fuera suyo. Mensajes Mere en el orden terrenal de los acontecimientos habían llegado últimamente a la Corona y el pueblo Inglés, a partir de un congreso de súbditos británicos en América: que, por extraño que parezca, han demostrado ser más importante para la raza humana que cualquier comunicación ha recibido a través de cualquiera de los pollos de la nidada Cock-lane.

Accepted Incluso el gallo-lane fantasma sólo se había establecido una ronda decena de años rapeando, después de sus mensajes, ya que los espíritus de este mismo año últimos (sobrenaturalmente deficiente en originalidad) diatriba de la suya. Sólo los mensajes de la orden de los eventos terrenales últimamente han llegado a la Corona Inglesa y de los Pueblos, de un congreso de súbditos británicos en América: que extraño se refieren, han demostrado ser más importantes para la raza humana de las comunicaciones ha recibido a través de cualquiera de los pollos del gallo-lane camada.

5. *Accepted* Francia, menos favorecida en general como a los asuntos espirituales que su hermana del escudo y tridente, rodó con gran suavidad cuesta abajo, por lo que el papel moneda y gastarlo. Bajo la guía de sus pastores cristianos, se entretenía, además, con este tipo de logros humanos como sentenciar a un joven para tener las manos cortadas, con la lengua arrancada con tenazas, y su cuerpo fue quemado vivo, porque no se había puesto de rodillas en el lluvia para honrar a una procesión de monjes sucia que pasó a su juicio, a una distancia de unos cincuenta o sesenta metros. Es bastante probable que, enraizada en los bosques de Francia y Noruega, no crecían árboles, cuando esa víctima fue condenado a muerte, ya marcado por la Woodman, Fate, a bajar y ser aserrada en tablones, para hacer un determinado móvil marco con un saco y un cuchillo en él, terrible en la historia.
6. *Accepted* Es bastante probable que en las dependencias aproximadas de algunos labradores de las tierras pesadas adyacentes a París, allí estaban protegidas del tiempo, ese mismo día, carros groseras, salpicado de fango rústico, apagaron sobre los cerdos y las aves de corral en roasted, que el granjero, Muerte, ya se había apartado para ser sus carretas de la Revolución. Pero, eso Woodman y que Farmer, aunque trabajan sin cesar, trabajan en silencio, y nadie los escuchó, ya que se ocuparon de la banda de rodamiento amortiguado: el lugar, por cuanto para entretener a cualquier sospecha de que estaban despiertos, era ser ateo y traidor.
7. *Accepted* En Inglaterra, apenas había una cantidad de orden y protección para justificar tanto la jactancia nacional. Robos audaces por parte de hombres armados, y robos de la carretera, se llevaron a cabo en la capital misma todas las noches, las familias se advierte públicamente a no salir de la ciudad sin necesidad de retirar sus muebles a los almacenes tapiceros para la seguridad, y el salteador de caminos en la oscuridad era un comerciante de la ciudad en la luz, y, de ser reconocido y cuestionado por su compañero de comerciante a quien se detuvo en su carácter de 'capitán', galantemente le dispararon en la cabeza y se alejó, el correo fue atacado por siete ladrones, y el guardia disparó tres muertos, y luego le dispararon muerto a sí mismo por los otros cuatro, 'como consecuencia del fracaso de su munición Â»: después de que el correo fue robado en paz, esa magnífica potentado, el alcalde de Londres, fue hecho para soportar y cumplir Turnham ladrones cortadas con tijeras; verdes

, por un salteador de caminos , que despojó al ilustre criatura en presencia de todo su séquito , prisioneros en cárceles de Londres luchó batallas con sus carceleros , y la majestad de la ley disparó trabucos en lugar de ellas , cargada con cartuchos de tiro y la pelota off cruces de diamantes de los cuellos de los nobles señores en los salones de la Corte ; mosqueteros entraban en St Giles , para buscar artículos de contrabando , y la multitud dispararon contra los mosqueteros , y los mosqueteros dispararon contra la multitud , y nadie pensó alguna de estas repeticiones mucho fuera del camino común . En medio de ellos, el verdugo , siempre ocupado y siempre peor que inútil , estaba en constante requisición ; ahora , encadenar largas filas de delincuentes diversos ; ahora , colgando de un ladrón de casas el sábado que había sido llevado el martes , ahora , la quema de las personas en la mano a Newgate por docenas, y ahora quema de panfletos en la puerta de Westminster Hall; a día , tomando la vida de un asesino atroz , y mañana de un ladronzuelo miserable que había robado el hijo de un granjero de seis peniques .

8. *Accepted* Todas estas cosas, y mil como ellos, aconteció en y cerca de la querida año un mil setecientos setenta y cinco. Environed por ellos, mientras que el Leñador y el Farmer trabajaron en letra muerta, los dos de los grandes mandíbulas, y los otros dos de la llanura y las caras justas, pisó con agitación suficiente, y lleva a sus derechos divinos con mano poderosa. Así fue como el año un mil setecientos setenta y cinco conducen sus Grandezas, y miríadas de criaturas pequeñas - las criaturas de esta crónica entre el resto - a lo largo de los caminos que se extendía ante ellos.

C.1.2. Macro-Task B: Artwork Assignment

Run 1

Find the artist of five paintings, execute majority voting after three assignments

Step	Time	Total time
Define the macro-task and design the flow composition	03:58	03:58
Publish the image tagging tasks	00:54	04:52
Approve the tagging results, execute majority voting	01:04	05:56

Table C.5.: Exp. 1 B I: Execution time - WebApplication Approach






Image	Painter	Topmost answer	Topmost answer share	Topmost answer evaluation
	Michelangelo	Michelangelo	66%	Correct
	unknown	Rembrandt	33%	Incorrect
	unknown	Michelangelo	33%	Incorrect
	Michelangelo	Michelangelo	100%	Correct
	Pablo Picasso	Pablo Picasso	66%	Correct

Table C.6.: Exp. 1 B I: Answers - WebApplication Approach

Step	Time	Total time
Publish the image tagging tasks	04:43	04:43
Approve the image tagging tasks	01:23	06:06
Analyse the tagging assignments and execute majority voting	03:52	09:58

Table C.7.: Exp. 1 B I: Execution time - Manual Approach






Image	Painter	Topmost answer	Topmost answer share	Topmost answer evaluation
	Michelangelo	Michelangelo	100%	Correct
	Rembrandt	Rembrandt	66%	Correct
	Michelangelo	Michelangelo	100%	Correct
	Michelangelo	Michelangelo	66%	Correct
	Pablo Picasso	Pablo Picasso	66%	Correct

Table C.8.: Exp. 1 B I: Answers - Manual Approach

Run 2

Find the artist of ten paintings, execute majority voting after five assignments

Step	Time	Total time
Define the macro-task and design the flow composition	02:28	02:28
Publish the image tagging tasks	00:14	02:42
Approve the tagging results, execute majority voting	00:30	03:12

Table C.9.: Exp. 1 B II: Execution Time - WebApplication Approach











Image	Painter	Topmost answer	Topmost answer share	Topmost answer evaluation
	Michelangelo	Michelangelo	70%	Correct
	Claude Monet	Claude Monet	60%	Correct
	Rembrandt	Rembrandt	100%	Correct
	Pablo Picasso	Pablo Picasso	60%	Correct
	Claude Monet	Claude Monet	80%	Correct
	Rembrandt	Rembrandt	80%	Correct
	Michelangelo	Michelangelo	n\ a	Correct
	Claude Monet	Claude Monet	100%	Correct
	Claude Monet	Claude Monet	80%	Correct
	Leonardo Da Vinci	Leonardo Da Vinci	80%	Correct

Table C.10.: Exp. 1 B II: Answers - WebApplication Approach

Step	Time	Total time
Publish the image tagging tasks	05:40	05:40
Approve the image tagging tasks	02:36	08:16
Analyse the tagging assignments and execute majority voting	09:05	17:21

Table C.11.: Exp. 1 B II: Execution Time - Manual Approach










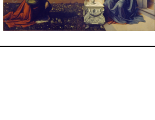
Image	Painter	Topmost answer	Topmost answer share	Topmost answer evaluation
	Claude Monet	Claude Monet	90%	Correct
	Claude Monet	Claude Monet	70%	Correct
	Rembrandt	Rembrandt	90%	Correct
	Pablo Picasso	Pablo Picasso	80%	Correct
	Claude Monet	Claude Monet	70%	Correct
	Rembrandt	Rembrandt	90%	Correct
	Michelangelo	Michelangelo	90%	Correct
	Claude Monet	Claude Monet	80%	Correct
	Claude Monet	Claude Monet	80%	Correct
	Leonardo Da Vinci	Leonardo Da Vinci	100%	Correct

Table C.12.: Exp. 1 B II: Answers - Manual Approach

C.2. Experiment Two: Crowd-Sourced MacroTask Composition

C.2.1. Experiment Two A: Multiple Workers

Step	Time	Total time	Comments
Create a new workspace	06:19	06:19	nice design and everything is understandable,easy to use.create task very simple here.
Create a data-source with a web service supplying images of artwork	01:20	07:39	Created datasource titled “Paintings” of WEBSERVICE type.
Add a categorization task	54:48	1:02:27	I really like the user interface. It was a little unclear at first as I was learning the system, but I liked it once I got used to it. It could use more options for setting the number of choices the worker has or other customizations. Descriptions of some features might be helpful. I personally would prefer the metadata to be on the side with the message log so you can view the workspace and diagrams in the center more clearly. It also is not entirely clear from the project description what needs to be done. When I got into the system there had already been two tasks created by someone else that were named something similar. I left them there and created my own at .02 USD. I connected this to the data source and the results, but it was unclear if this is was what was desired. The video mentions a validation, but the project description says only to create the categorization step, so that is what I did. Overall it is a very useful system. I look forward to seeing where it goes.
Add a merger	12:04	1:14:31	I did my best ,

Table C.13.: Exp. 2 A: Execution Time

C.2.2. Experiment Two B: Single Worker

Step	Time	Total time	Comments
Define the entire workflow	09:17	09:17	

Table C.14.: Exp. 2 B: Execution Time