

Fredrik Jons
Linköping University
Linköping, Sweden
frejo105@student.liu.se

Alexander Stolpe
Linköping University
Norrköping, Sweden
alest170@student.liu.se

ABSTRACT

Coming later

INTRODUCTION

Today a lot of people are using social media in one way or the other and it is estimated that there will be around 2.67 billion social media users around the globe by 2018[**STATISTA_SN_WORLD_USERS**]. Most of these social networks have released Application Programming Interfaces (APIs) which developers wish to integrate these networks into their software.

Motivation

Social network usage is growing and has gone from 0.97 billion users in 2010 to 2.14 billion in 2015.[**STATISTA_SN_WORLD_USERS**]. This would account for approximately 29% of the earth's population in 2015, which was 7.347 billion 2015[**WORLD_BANK_POPULATION**]. It is worth noting that this counts created user accounts and not unique users, one person can have multiple accounts over multiple networks, and accounts may not belong to an actual person, but rather companies or bots.

Because of the currently high, and still growing, number of social media users we find it highly likely that we will see an increasing number of applications that involve social media in their software in one way or the other. Out of all of the social networks existing today there are twenty that have more than 100 million active accounts[**STATISTA_LEADING_SOCIAL_NETWORKS**]. This means that if one would want to create an application that involves a lot of social networks we will have to do a lot of work just to implement all of these into our system.

Purpose

Because of this we see a need for a way to combine these social network APIs in some way to save development time and reduce the amount of duplicate code written in software. The purpose for this project is to create a library that combines the APIs and make it easier to involve social media in software.

A software library by definition is a set of pre-written code that a developer might add to a project to add more functionality or to ease the development process[**TLDP_LIBRARY_DEFINITION**].

RESEARCH QUESTION

Is it possible to create a modular library to maintain social network APIs?

LIMITATIONS

Because there are so many social networks existing today and we have a finite amount of time to complete this project we will focus on a smaller set of APIs to implement into our library. We have set up a few criteria for the APIs so we can find suitable candidates for our library, which are:

- The social network must be one of the 22 most popular[**STATISTA_LEADING_SOCIAL_NETWORKS**].
- It must be relevant for our geographical location, in this case Europe.

Other than this we will judge the API itself by how good its documentation is, its functionality and its ease-of-use.

THEORY

A lot of time and money can be saved by reusing code, of which libraries are one form. Although there are some issues with using libraries, the gains from avoiding reinventing the wheel makes writing and using libraries a common practice, in particular in open source software[**2998479020080101**].

Writing code which can be easily reused requires a deeper analysis of the problem domain, which may increase the cost and time required compared to developing the same code without reuse in mind, but can drastically decrease the cost of developing systems in the future where the code can be reused [**lim1994effects**]. This cost reduction is apparent foremost in terms of direct cost of development, but also in time-to-market, which can be argued to be even more important in the long term[**griss1993software**].

With code reuse there are also several potential issues which have to be kept in mind, both when it comes to the implementation itself but also when it comes to using the implementation. Backwards compatibility between versions is a major topic in itself[**raemaekers2012measuring**]. There is also the risk of a library being abandoned by its maintainer. This is especially true for proprietary libraries, where the source code may not be available. In this case, the library might have to be replaced, making all the effort to use the library wasted.

When designing our library we will want keep several things in mind. We want to design the library so it's easy and straightforward to use for a developer. As Henning points out in his article[**Henning:2007:ADM:1255421.1255422**], about design of APIs, it's very easy to create a bad one, but very hard to create an API that feels natural and easy to work with. APIs as we know are a kind of interface for a program to gain access to another program without direct access, and can be compared to the interface of our library.

Henning continues in his article to discuss guidelines for how an API should generally be designed. What feels most relevant to our work is how he describes how the APIs should be designed from the perspective of the user, because when it's done from the implementer's point of view the needs of the user are often forgotten. It's usually best to document first, because when it's done after the implementation the programmer, who wrote the functionality, will usually just dictate what he did, rather than make it obvious enough for others who are not as familiar with the code.

Similar work

Several similar works serve the same purpose^{1,2,3}, but none are still maintained. There are also commercial services⁴ which provide this functionality for some popular APIs, but charge money to sign up and use. As it is proprietary its inner workings are completely opaque, and thus will not be examined by this study.

¹ <http://www.agorava.org/>

² <https://github.com/gorbin/ASNE>

³ <https://github.com/socialsensor/socialmedia-abstractions>

⁴ `\url {https://cloudrail.com/}`.