技術情報源の効果的共有手段の考察

電気通信大学 X680x0 同好会 tenpre(11 年度入学)

概要

技術書とは、持っているだけでプログラマのステータスとして認識される魔術書である [**要出典**]。しかし昨今ではインターネット上のサイトでもかなり多くの情報を得られることもあり、紙媒体に頼るということも少なくなってきたように思える。その一方で技術書に頼らざるを得ない状況も依然として多く、結果として複数の情報媒体を駆使しないといけない状態になっている。本駄文では技術情報を探す上で必要となる事柄について考察し、複数の媒体の情報を統一して管理できる情報共有サービスを提案する。

1　はじめに

みなさんはプログラミングの勉強の際に技術書を買うことはあるだろうか。インターネットが日常において当たり前のものとなった現代において、何かを勉強するために書籍を買うというのは、紙媒体か電子書籍かどうかに関わらず機会が減っているのではないだろうか。というのも情報技術は勿論のこと、現在のインターネットには様々な分野についての情報源が多数存在するからである。少し何かを勉強したければ検索をすれば大抵のことは調べられてしまうのだ。

では技術書なんてものは必要なく、インターネット上の情報だけで全て事足りるのだろうか。わざわざお金を払って技術書を買う理由とは一体何だろうか。ここでは技術書は必要なのかどうか、どのようなときに便利なのか、そしてどのようにしたら技術書とインターネットの情報源を上手く使いこなせるのか、最近私が思うことを織り交ぜながら述べていきたいと思う。

2　技術書が必要になるとき

まず、技術情報源が必要になるのはどのようなときだろうか。技術書というのは基本的にプログラミング言語についてや実装等について述べられたものであり、それらを学ぶときになって初めて必要となる。では技術を学ぶときに情報源に求められるものとは何だろうか。個人的な結論から述べれば、それはひとえに**「方針や主張に一貫性があり、体系だった理論について最低限必要な事項を不足無く網羅できるもの」**だと考える。

例えば C 言語の入門的知識の学習で考えてみるとすると、まずあなたはどのような情報源を探すだろうか。文法について述べているものだろうか。しかし文法だけ分かっても基本的にソースコードは書けない。それは各プログラミング言語やそれが属するパラダイムには、その流儀に沿った作法、仕様があるからだ。C 言語について言えば、

1. 処理は main 関数内に書く
2. 多くの関数はヘッダーを include しないと使えない
3. 変数は使用する前に宣言しないといけない
4. malloc 等で動的に確保したメモリ領域は free で開放しなければいけない

といった仕様が存在する。勿論これは他の言語についても言えることであり、例えば Haskell では参照透過性を確保するために、関数内で外部の環境に対して直接破壊的な処理は出来ない。つまり文法を学ぶだけではそういった作法や流儀を理解することは出来ず、場合によってはその仕様によって良からぬ事態を引き起こすかもしれない（C 言語におけるメモリリークが良い例だろう）。逆にそういった作法の知識だけでは実際のソースコードの書き方が分からず、既存のソースコードを読むにあたっても細かな文法を理解することは不可能である。

つまり技術情報源に必要な事を具体的に表すと、一連のパラダイムを説明する上で具体的な実装をしっかりと述べられているものが望ましい[*1]。書籍であろうとインターネット上のサイトであっても、これについては変わらないと思われる。

では、得るためには金銭を支払う必要がある技術書が、手軽に調べられるインターネット上の情報源に勝るものとはなんだろうか。それは「複数の有識者によって、その内容が編纂され、ある程度の量が纏められている」という保証ではないだろうか。

インターネット上の情報というものは、分散されている傾向があり、実装の一部について詳しく述べられていても、全体を網羅できるものはなかなか見つからない事が多い。勿論、しっかりと全体が網羅されており、理論と実装が学べるものであれば問題はないが、全ての情報に対して十分な情報源が存在するとは限らないだろう。特にこれは技術のレベルが高いほど言えることで、初学者向けのものはインターネット上にも有用な情報源が多く存在するものの、少し特殊な実装、理論に関しては一部分のトピックしか述べられないものも多い。学ぶ側としては同じレベルの情報が纏まっている書籍に頼った方が学びやすいといえる。

一方で、インターネット上の情報源には技術書には無い、「伝達の早さ」という利点がある。まだどの書籍メーカーも纏められていないような情報に関しては、インターネット上のフォーラムに頼った方が早いだろう。

以上から技術書は、「ある程度成熟した環境、理論を学ぶとき」に最も有効であると考える。そして私が思うのは「余程インターネット上に優秀な情報源がある訳でなければ、潔く技術書を買うべきである」ということである。余程内容が古くない限り、インターネットで情報を探して時間を費やすよりは、金銭を支払って技術書を購入したほうが損をしない、と私は考える。

3　情報源の探し方とその難しさ

技術書の優位性を述べてきたが、インターネット上の情報でも良いものは多く存在する。例えば C++ においてならば、有名なサイトとしては「ロベールの C＋＋教室」[1] がある。このサイトの情報は既に書籍化されており、このような後々書籍になったサイトというのは書籍の評価を参考に出来るので評価がし易いという面もある。サイトで十分ということもあるので、インターネットの情報を軽視することは出来ない。

しかし情報源を探して Google 検索で探してみたが時間の無駄だった、というケースも少なくはないだろう。これは書籍でも起きる事だ

[*1] 例外もあるが大抵は実際にコードを少し書いたことがある事が前提となるようなものであり、事前知識で不足しているところを補わなければならないものが殆どである。

が、数がとても多いインターネットのサイトではより顕著に起きる。特に特定の概念やフレームワークを勉強する時は、しっかりとその土台から勉強する必要があり、先走って付け焼き刃的な知識でプログラムを組みだすと後々後悔するようなことも多い。書籍クラスのしっかりとした情報源があるに越したことはないのである。

ネット上の情報となると、Q&A サイトの様なフォーラムが最近では最も活発なのではないだろうか。フォーラム系のサイトでは Qiita[2] や Stack Overflow[3] 等が有名である。このようなサイトは各々が情報を更新出来るため更新速度が早く、インターネットの利点を最大限に有効活用できる。一方でインターネット特有である情報の分散が起きやすいため、分野の初学者が参考にするには適さないという面もある。

ここで私は以上のような複数の情報源を整理する手段として、「複数の情報源を複合的に纏められる情報源整理共有サービス」を提案したい。上で述べたとおり、技術書かインターネットのサイトか、どちらか一方が優れているという事はないのだから、その両方を扱えるサービスが望ましいのである。ひと目でどの情報源を当たればいいのかが分かれば、人は情報を探す時間をもっと有意義なことに費やすことが出来るだろう。

また１つの事を勉強する上で、見るべき情報源が１つだけというのはあまり無いことだろう。情報量が少ない場合に複数あたるのは勿論、複数の情報源をあたることでより深い理解が得られる、ということも十分考えられる。そのためカテゴライズのような、複数の内容を纏めることが出来る必要が出てくるのである。

しかし情報源を漁る上ではもうひとつ重要なことがある。それは「その技術を理解するのに十分な基礎知識を身につけているか」ということである。当たり前だが幾ら技術に関する説明やソースコードがあっても、それを使う方法が分からなければ何の意味も無い。技術情報源を見るにあたって**「先に読んでおくべき情報源」**が明示されていることも望ましい。

加えて情報の編纂が自由に出来ると良いだろう。このようなサービスを作成するならインターネットの速度を利用しない理由はない。

以上のことを纏めると、理想的なサービスは次のようになる。

1. 書籍とサイトを統一して扱えること
2. 教えている内容に則したカテゴライズができること
3. 他の情報源との関連性が順序付けができること
4. 誰でも編集可能であること

これらの機能を満たしたサービスとして次のようなものを提案したい。

4 提案手法

4.1 情報の管理

まず一つの情報源については以下の様な記述が出来るようにする。

情報源の種類 情報源には書籍媒体、Web サイトのどちらも記述できるようにする

評価とコメント 情報源にはコメント付きの 10 段階評価をつけられる

大筋は Amanon[4] の様なショップサイトにあるコメントを思い浮かべて貰えれば十分である。評価をつけるだけなら基本的にはそのような方法で十分だろう。

更に上のような情報源をカテゴライズ出来るようにする。カテゴリは基本的に「学びたい内容」や「対象となる読者層」について特徴付けるものにし、それを学びたい人たちが一目でどれを読めばいいのか分かるようにする。またカテゴリは一つに情報源に対して複数設定できるようにし、単一の情報源が一つのカテゴ

リの中で埋もれてしまわないようにする。

4.2 **情報管理** SNS

以上のような記述を定義したとき、これらをどのような風に管理するという問題がある。ここで順序付けとして**順序付きグラフ**のようなものを利用したい。

まず以上のようなカテゴリ郡をノードとしたグラフを作成する。エッジはあるカテゴリから別のカテゴリへの関連性を示すもので、順序付けがされているので**「どの情報源を学んだ上で、どういった情報がほしい時にはどれを読めばいいのか」**というのがグラフィカルに表現することが可能になる。

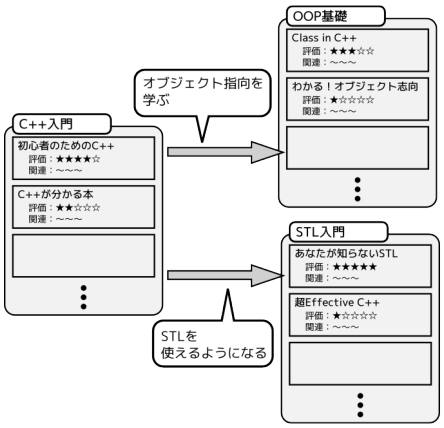

図 1　カテゴリ分けされたグラフのイメージ

しかしカテゴリ間ではなく、情報源の間で関連性を持たせたい時もあるだろう。それに関してはカテゴリだけではなく、情報源の間でも関連付けが出来るようにし、逆にカテゴリの関連性にはそれだけでなく、内部の情報源の他の情報との関連性も加味したエッジを張れるようにする。こうすることで特定の情報源の間での関連性を確保したまま、カテゴリの関連性の正確さを強化することが可能となるのである。

このようなサービスが SNS のような形態を取っているサービスで展開されれば良いだろう。書籍等の評価というのはシビアなもので、一人がネガティブなものを発すれば全体の評価はそれに大きく引っ張られることが多い（逆はあまり無いように思える）。従って評価付けをした人間を特定できるということは重要である。

5　今後の方針

ここで何らかのモックアプリが提示出来れば良かったが、残念ながら時間的余裕が無いため、今回は断念した。仮に時間があったとしても、筆者個人にサーバや Web アプリに関する知識が無いため、しっかりとしたサービスとして作ることも難しいだろう。筆者の無能さが悔やまれる。

個人的には上にも述べた通り、最近は SNS やフォーラムサイトも活発であるので、こういった「一度生み出された情報源を整理するサービス」は、そのようなサイトのサービスとして提供されていても良いと思う。そのサービス自体や Wiki でも出来なくはないが、グラフィカルなものが出来れば個人的には見やすくて嬉しい。もしそういうサービスが既にあるなら筆者にこっそりと教えていただけるとこの上ない。

6　参考文献

参考文献

[1] ロベール 「ロベールのＣ＋＋教室」 <http://www7b.biglobe.ne.jp/~robe/cpphtml/>

[2] Qiita - プログラマの技術情報共有サービス <https://qiita.com/>

[3] StackOverflow <http://stackoverflow.com>

[4] Amazon.co.jp <http://http://www.amazon.co.jp>