International Journal of Digital Content Technology and its Applications
Volume 3, Number 3, September 2009

Improving Software Development Using Scrum Model by Analyzing Up and Down Movements on The Sprint Burn Down Chart: Proposition for Better Alternatives

Md. Junaid Arafeen[1], Saugata Bose*
[1] *Software Engineer, Vonair Inc, Dhaka, Bangladesh, email: arafeenbd@yahoo.com*
* *Lecturer, Daffodil University, Dhaka, Bangladesh, e-mail: saugata28@yahoo.com*
doi: 10.4156/jdcta.vol3.issue3.14

Abstract

Among various models, now a day, to address changing customer's needs, commercial software developers intend to use iterative processes. Agile software development processes are built on the foundation of iterative development. One of the agile development processes is Scrum, which is an iterative incremental process of software development. Besides meeting varying needs of customer, timely delivery of product is important too. To predict when all of the work will be completed the developers use Burn Down Chart which represents work left to do vs. time. Theoretically, progression of time should leave less amount of task, the chart will look like straight line and the sprint will remain constant. However, in software industry, the irregularities on burn down chart are common which are the ups and downs occurring due to introducing new story points, new technology, wrong estimation of resources, cost and schedule. These reasons in turn affect the cost of the project, which in turn pull down the quality of the product because of failed to attain the defined sprint goal. To avoid such irregularities in development the better solution might be use of sprint burn up chart or modified Phil Goodwin and Russ Rufer's burn down chart.

Keywords

Scrum, Product backlog, Sprint, Burn Down Chart, Person-Month, Burn Up Chart

1. Introduction

Computer software is the product that software professionals build and then support over the long term. It encompasses programs that execute within a computer of any size and architecture. Because software is embodied knowledge and because that knowledge is initially dispersed, tacit, latent and incomplete, software development is a learning process. The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process provides interaction between users and designers, between users and evolving tools and between designers and evolving tools. The evolving tool itself serves as the medium for communication, with each new round of the dialogue eliciting more useful knowledge from the people involved in an iterative process. This process is a framework for the tasks that are required to build high quality software. Agile Software Development is a methodology for software development process that promotes development iterations, open collaboration and adaptability throughout the life cycle of the project. The agile philosophy stresses four key issues: the importance of self-organizing teams that have control over the work they perform, communication and collaboration between team members and between practitioner and their customers. Agile process models have been designed to address each of these issues. Scrum is one of the models that are widely used in software development firms. In Scrum, work is delivered in monthly "sprints". Each sprint delivers a single, usable piece of work called the "product increment". This product increment is immediately available for evaluation and use by the customer at the end of a sprint. To monitor sprint progress, release progress, product progress, each day a spreadsheet is updated with the current work remaining. Scrum uses "burn-down" charts to monitor progress during a sprint. In a burn-down chart, remaining work is plotted on the Y-axis, and time proceeds along the X-axis. As tasks are completed, the line slopes down. Burn down charts provides an intuitive feel for the progress of a sprint. The most common burn down chart "signature" is a steady decline. But this "signature" is mostly uncommon in commercial level.

In this paper, we intend to point out the controversy happened whenever burn down chart is implemented commercially and try to find out the reasons behind ups and down movements on burn down chart and try to figure out the consequences faced by the industry by this irregularity. Then we propose two alternative solutions for prospective software developers. Initially the report contains the summary of Scrum model. Then characteristics of Burn Down Chart are defined. At the second phase, we study two Burn Down Charts which were derived from our 2 case studies and analyze the reasons of ups and down movements occur on those charts, then at the last phase we propose two alternative solutions for irregular movements on burn down chart.

2. Scrum methodology

Scrum is an enhancement of the iterative and incremental approach to delivering object-oriented software initially documented by Pittman and later expanded upon by Booch. Scrum is a management, enhancement and maintenance methodology for an existing system or production prototype. It assumes existing design and code which is virtually always the case in object-oriented development due to the presence of class libraries. Scrum will address totally new or re-engineered legacy systems development efforts at a later date.

Software product releases are planned based on the following variables:

•Customer requirements - how the current system needs enhancing.

•Time pressure - what time frame is required to gain a competitive advantage.

•Competition - what is the competition up to, and what is required to best them.

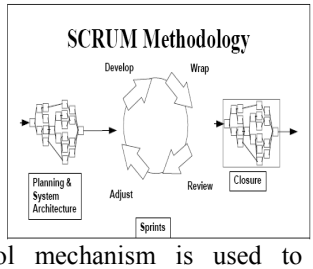
•Quality - What is the required quality, given the above variables?

•Vision - what changes are required at this stage to fulfill the system vision?

•Resource - what staff and funding are available.

These variables form the initial plan for a software enhancement project. However, these variables also change during the project. A successful development methodology must take these variables and their evolutionary nature into account. The system development process is complicated and complex.

Therefore, maximum flexibility and appropriate control is required. Evolution favors those that operate with maximum exposure to environmental change and have optimized for flexible adaptation to change. The SCRUM approach assumes that the analysis, design and development processes in the Sprint phase are unpredictable.



A control mechanism is used to manage the unpredictability and control the risk. Flexibility, responsiveness and reliability are the results.

3. Overview of scrum

Scrum has three roles Product owner, Scrum master and Project Team.

1. **Product owner** defines the features of the product, prioritize product features based on market value and prioritize them after every 30 Days, accepts and rejects work results. The release date of the product is defined by product owner. Product owner can be customer.

2. **Scrum master** is the interface point between customer, scrum team and management. Scrum master ensures that the scrum practices are followed and monitors overall project progress.

3. **Scrum project** team is a cross-functional team with 5-7 members. It includes systems analysts, programmers and quality assurance professionals and therefore it is called cross functional team.

Scrum uses three types of backlogs. These are product backlog, Burn Down chart and sprint backlog.

1. Product backlog is a prioritized queue containing all the project requirements. The product owner does the priority of backlog items but any stakeholder can add requirements to product backlog. The project progress is determined by the number of items in product backlog. Initially the size of product backlog grows as stakeholders add requirements but then it gradually decreases by the start of sprint.

2. Burn Down chart shows the work remaining in sprint. As the sprint precedes the requirements in sprint backlog decreases or burns down. A sprint is successful if sprint backlog is zero. So Burn Down chart is a useful tool for scrum teams for timely completion of sprint.

3. Sprint backlog is the output of sprint planning meeting. The requirements are broken into manageable tasks that require two days or sixteen developer hours.

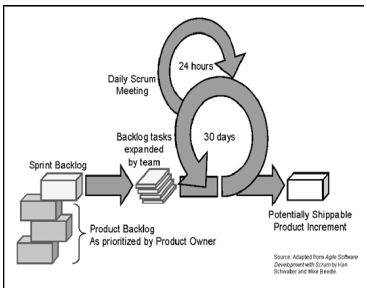
Sprint: The scrum team works for a fixed period of time. This is known as sprint. The sprint period is 30 days. During sprint, the team self-organizes and self directs. Here there can be two possible ways for scrum team.

• If the sprint goals are not achieved the scrum, team can change the functionality to be delivered by the sprint.

• Scrum team aborts the sprint if based on some new information; they feel that the sprint goal cannot be achieved.

Sprint review: Its duration is usually four hours .In the first half of sprint review demonstration of sprint progress is given to stake holders. This includes customers, product owner, scrum master, scrum team and management of software house. It provides agenda for next sprint planning meeting. The second half of sprint review meeting is with the scrum master and scrum team. The scrum team analyzes their performance in sprint and identifies positive ways of working together. They also identify weak areas and develop strategies for improvements.

Sprint Planning Meeting: All stakeholders participate in the sprint-planning meeting. They agree on the sprint goals and sprint backlogs. During sprint planning meeting, scrum team divides individual tasks to be completed during sprint.



Daily Scrum Meetings: Daily scrum meeting is a short

15 minutes meeting that takes place. every day. It is an event where scrum team members gather to exchange and share

information regarding

various issues in sprints. Customers can also participate in Scrum meeting but they are not allowed to speak. This is done to keep the meeting short.

During daily scrum meeting, each scrum member answers three questions.

1. What have you done since the last Scrum?
2. What will you do between now and next Scrum?
3. What got in your way of doing work?

In daily scrum meeting scrum Master leads the scrum team.

Characteristics of Scrum: Scrum has following characteristics.

•Small teams: Scrum team is a cross functional team comprising 5-7 members. The main motive behind scrum is that each team member should work independently but towards the same goal like sports team.

•Frequent Reviews: The project progress is monitored frequently in daily scrum meetings. Here all team members discuss various issues related to their work.

•All requirements are available in prioritized queue known as product backlog.

•The role of Scrum Master is very relationship oriented with scrum team. Scrum Master acts as a bridge between customer, software house management and scrum team. Scrum master ensures that scrum practices are followed during sprint and eliminates obstacles in the project.

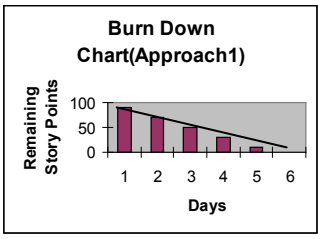
•In Scrum requirements prioritization, acceptance and rejection of sprint results and product release date is specified by product owner.

•All the stakeholders including customer are continuously updated regarding the project progress by the help of sprint review.

•Scrum focuses on continuous improvements by focusing on weak area during the sprint. This is usually done in the second half of sprint review meeting where participating audience are Scrum master and scrum team.

4. Theoretical approach of sprint burn down chart

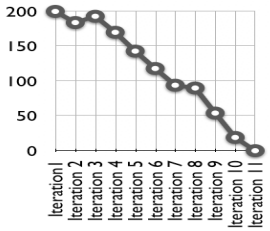
On a Scrum project, the team tracks its progress against a release plan by updating a release burn down chart at the end of each sprint. The horizontal axis of the release burn down chart shows the sprints; the vertical axis shows the amount of work remaining at the start of each sprint. Work remaining can be shown in whatever unit the team prefers--story points, ideal days, team days, and so on. There are two approaches remaining for sprint burn down chart among theorists.



Approach1: The remaining story points should reduce at every day of a sprint. Therefore, the ultimate shape of the sprint burn down chart would look

like downward straight line.

Approach2: The remaining story points would reduce at every day of a sprint. Whenever any new task is introduced, it would insert at the current sprint. Insertion of new task in turn makes upward movements of remaining work tasks on the burn down chart. But the duration of sprint should remain constant.



Burn Down Chart(Approach2)

On the burndown chart(right side), the team started a project that was planned to be eleven two-week sprints. They began with 200 story points of work. The first sprint went well and from the chart we can infer thatthey had around 180 story points of work remaining

after the first sprint. During the second sprint, however, the estimated work remaining actually burned up. This could have been because work was added to the project or because the team changed some estimates of the remaining work. From there the project continued well. Progress slowed during sprint 7 but then quickly resumed.

5. Implementation of sprint burn down chart (where the problem lies)

Case Study1

Project: Intrusion Detection System

The system will be setup in a network as a node and analyze the network packet i.e. TCP/UDP, ICMP, ARP. It will make a profile for each node and evaluate the vulnerability of the whole network which indicates the susceptibility to any intrusion. It will further detect an intrusion or any irregularity happened based on the profile.

Project length: 6 months.

Size: 5 Person-month

Technology Used: Java 6.0, Eclipse RCP (Rich Client Platform), OSGi (Open Services Gateway Initiative), JOGL (Java OpenGL Library), JPCap, NMAP, Web Services.

Team's Knowledge Base: Team members are 1 year experienced

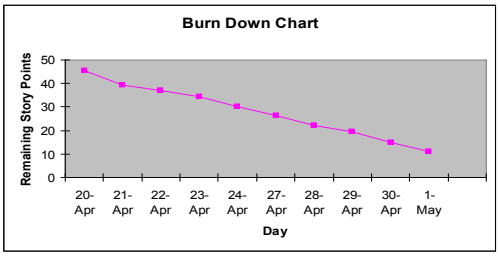
Expertise: J2EE technology like Struts, Hibernate, Servlet, JSP, SOAP, Web Services etc.

New Technology: Eclipse RCP, OSGi, JOGL, JPCap, JFreeChart

Overview: Agile-scrum was implanted from the scratch of the project. To accomplish this project we had decided to take Sprint of 2 weeks length (10 working days). To accomplish this project we have assigned 5 persons. Since we have 10 days then per sprint we had maximum 50 points to cover. Here 1 point = 1 person-day work. Sprint goal was to

1. Develop the database schema and implementation
2. Draw the UI interface.
3. Draw the 3 dimensional graph using JOGL

The following burn down chart actually represents the project's progress.



Case Study2

Project: Client Management System

The system will provide a service delivery platform to manage various CPE (Client Premise Equipment) systems. It is a plug-in based system where various plug-in will incorporate knowledgebase to the system to support/provide services to the management.

Project-length: Two years.

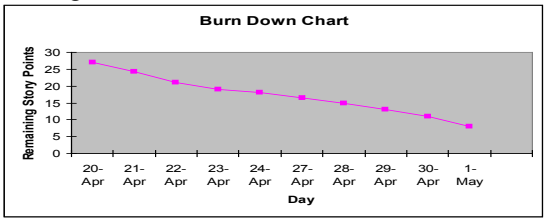
Size: 3 Person-month.

Technology Used: Java 5.0, Web Services, Axis 1.0, XML, JSP, Servlet, Java Script, EJB, JBoss Server.

Team's Knowledge Base: Team members are 1 year experienced.

Expertise: Java 5.0, Web Services, Axis 1.0, XML, JSP, Servlet, EJB, JBoss Server.

Overview: The Sprint goal was to develop the web pages for the system. The user requirement was changed at the middle of the sprint. The user wanted some widget based UI that means they wanted Web 2.0 based Web UI. Before that the developer was developing Web 1.0 based web pages. That was a significant change for the developer. The burn down chart for that particular sprint is given below



6. Analysis of the sprint burn down chart

To estimate a job correctly one has to know the answer of these three questions **what to do**, **how to do** and **what is required to do**.

"What To Do" refers the developers have to understand the customer requirements very well. The answer of the following question depends on this one. If they fail to understand "What To Do", then the answer of the subsequent two questions will be changed accordingly.

"How To Do" indicates the developers need to chop down the task into smaller one.

"What Is Required To Do" refers which language or tools or expertise or level of experience they need to accomplish the job. So Burn Down Chart can vary for any misinterpretation for the above three answers.

Considering the above case studies, we identify that there are some striking reasons that compels the burn down chart acting differently from the theoretical concept.

A. Customer Requirements Are Changing: In view of the above case studies, we find that in every

sprint customer is changing his requirements. So the number of unsolved tasks is increasing, which makes the ups and downs in the burn down chart.

B. Introducing New Technology: "What Is Required" includes technology language, tool etc. if it is new then developers can not correctly anticipate the time required to know the required knowledge to do a task. This in turn, deviates burn down chart from the ideal curve.

C. Wrong Estimation: "What To Do" and **"How To Do"** is also another point to consider. Lack of experience will hamper the correct time anticipation. A new task spawns many and only experience developer can foresee it. And thus helps to maintain the burn down line. Human factor is another point to consider. An experienced and productive developer needs less time than a fresher one. But the time estimation will be such that it will minimize the difference. The time margin should be justified irrespective of highly experienced and the regular developer.

7. Consequences of the sprint burn down chart

•The firm needs to add extra person-month for that sprint to reach the sprint goal. But it is rather impossible for a firm to employ extra persons.

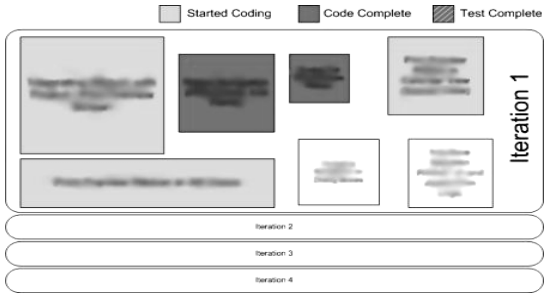
•Transfer the extra requirements to the next sprint as per discussion with the customer, which will increase the effort per person-month

•If customer forces that requirement should be fulfilled in, the current sprint then it will replace another lower priority task of a story or a full story, which will again increase the effort estimated earlier for person-month. In this way, firms fail to reach the goal promised to serve to customer.

8. Alternative proposed solution for sprint burn down chart

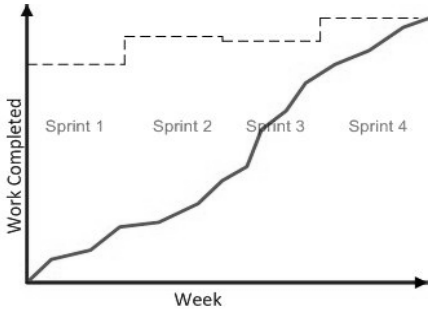
Solution1: Burn Up Chart

One of the benefits of Agile is that it focuses on small increments of functionality so that forward progress is always being made. In this paper, we like having a big printout (hand-drawn works just as well) of the following artifact in a software development firm:



Progress Snapshot (with tasks blurred). Where are we, and what do we have left?

This artifact can be easily understood at a glance, and gives a quick snapshot as to how the team is doing – right now. What it doesn't do is show how the team has been progressing over time. To show progress son a temporal basis, we like using a "Burn-up" chart. It's a modification f the Burn-down chart prescribed by many of the Agile methodologies, and simply plots the amount of work completed each week.



The Burn-up Chart. Dotted is target level.

The 3 reasons we use Burn-up charts:

1. **It passes the Walk-By test.** When it comes to charts and graphs, we are strong advocate of the KISS (Keep It Simple, Stupid) principle. Any developer walking by the office should be able to understand the chart without having to stop walking. The burn-up chart is a simple snapshot of position, velocity and acceleration. How much work has been done, how much remains to be done, how fast is work being done, and is the team gaining momentum or losing steam? The burn-up chart answers these questions at a glance.

2. **Better Estimates**. Part of being agile is revisiting estimates and refining them during the development cycle. Actions speak louder than words, and past performance speaks louder than verbal estimates. The burn-up chart is a great artifact for projecting task completion date. Draw a line through your current position, and note where this line intersects the target line – this is your projected finish date. We can change the slope of the line to match our velocity for the past week or the past 4 weeks and we can see how that changes the projected finish date if we were to maintain the velocity we've been working at recently. The key is

that that this projected finish date is based on actual team performance. Estimates at the planning stages often involve “Engineering-hours”. Estimates with the burn-up chart are not based on the performance of “2 engineers”, they are based on the performance of 2 developers, the actual people working on this project.

3. **Changes to Requirements**. The burn-up chart is based on the burn-down chart prescribed by several Agile methodologies. In a burn-down chart, the target line is at the base of the X-axis and work is shown sloping downward. One shortcoming of the burn-down chart is that that the target line is fixed. In any project, requirements change, estimates are (hopefully) refined, and unfortunately, there is always the chance of feature creep . Burn-up charts can react by moving the target line and showing at which iteration these changes occurred. It is also easy to analyze how changing the target line will affect the projected finish date. The burn-up chart embraces changes in requirements, and encourages that estimates are revisited and refined at the end of every iteration.

Strategy to Encompass Solution1

The strategy we will follow:

• We will make a list of all the items to be delivered. We might work from the Project Map, Blitz Planning or XP's planning game. Then we associate with each delivery item a relative cost. Then we try to associate with each item also a relative business benefit, so that we can discuss with the Executive Sponsor the value being delivered to the business over time. The reason for using relative estimates instead of absolute ones is that if we are, for example, 10% over on your first set of items, all the remaining items will scale accordingly. The burn charts will show this automatically.

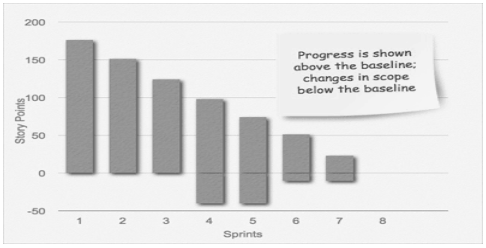
• We will sort and sequence the work items by development dependency, cost and value, and cluster them into a sequence.

• We will estimate how much can be accomplished in each iteration or delivery period. Then we will draw a line under the last item that fits in each and will umber the releases.

• At this point we will make a Burn-up chart. Then we mark either relative work units or % complete on the vertical axis, and calendar time on the horizontal. We will draw a line from the origin to the completion point, either by estimating the rate at which work can be accomplished, or as often happens, the "drop dead" delivery deadline. That shows the "planned" progress. After each iteration, we will mark how much relative work got completed.

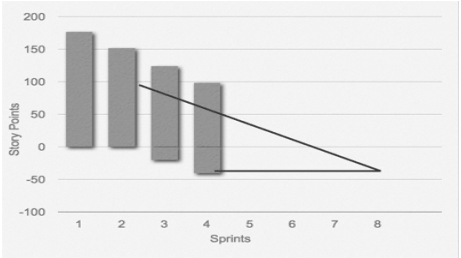
Solution2: Phil Goodwin and Russ Rufer’s Modified Burn Down Chart

The typical Scrum burn down chart shows a single value--the net change in the amount of work remaining. However, it can also mask what may be going on in a project. For example, suppose a team had expected to make progress of 40 (hours, points, whatever) last sprint but the burndown chart only shows net progress of 10. Was the team slower than expected or was more work added to the release? It's important to know the answer to this question because we cannot really predict when the release will be done without it. With this in mind, we propose the following type of burndown chart:



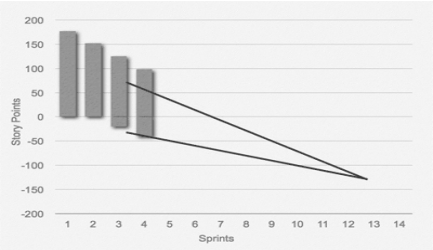
Burn Down Chart1(Proposed)

On this burn down chart, the height of each bar represents the amount of work remaining in the release. This figure shows a release with 175 story points planned in it as of sprint 1. The team finished 25 points in sprint 1, leaving 150 to go as of the start of sprint 2. There were 120 as of the start of sprint 3. So, the top of the bar is reduced by the amount of work the team finishes in a given sprint. Before the start of sprint 4, the product owner added work to the project. This additional work is shown at the bottom of the bar for the fourth sprint. We can see that the vertical height of sprint 4 goes from about -40 to about 95, or 135 points of work remaining. Forty of those 135 points are from new work. One way to predict how many sprints a project will take is to draw a trend line through the bars and extend the baseline. For example we will consider the figure below of Burn Down Chart2(Proposed) :



Burn Down Chart2(Proposed)

We can anticipate the number of sprints needed by also drawing a trend line through the changes occurring at the bottom of the bars as shown below of of Burn Down Chart3(Proposed):



Burn Down Chart3(Proposed)

9. Conclusion

Scrum is a very useful technique that can be also used for software project management. Scrum

supports both large scale projects and small scale projects. Burn down chart is an inevitable part of the SCRUM model. It helps perceive the progress about the project in which it is implemented as well as to foresee the project future. The deviation from the real curve which is an ideal straight line reflects the lack of domain knowledge, the experience and the capability of the software engineer. However, it is entirely possible that the practical burn down chart is alike the ideal curve. Therefore, if we follow either burn up chart or Phil Goodwin and Russ Rufer's modified burn down chart we can track our sprint goals without affecting our estimated

10. References

[1] Agile Software Development with Scrum, By Ken Schwaber and Mike Beedle Published by Prentice Hall, 2001
[2] Agile Software Development By Alan S.Koch
[3] SCRUM - Process Model Report By Uzair Akbar Raja, Farrakh Saeed, Mohammad Zeeshan ul Haq, Sheraz Ahmad Students-MS Software Engineering, Blekinge Institute of Technology
[4]http://www.scrumalliance.org/index.php/scrum_alliance/for_everyone/what_is_scrum/ scrum_roles
[5] Integrating Agile Development in the Real World", By Schuh, Published by Charles River Media, 2004.
[6] Agile Software Development with Scrum, By Schwaber Beedle, Published by Prentice-Hall, 2001.
[7] SCRUM Development Process By Ken Schwaber Advanced Development Methods
[8] The Challenge of "Good Enough" Software", By Bach, James, Published by American Programmer, October 1995.
[9] A Spiral Model of Software Development and Enhancement By Boehm from Proceedings of an International Workshop on Software Process and Software Environments, Coto de Caza, Trabuco Canyon, California, March 27-29, 1985.
[10] Earned-value and burn charts By Alistair Cockburn from Chapter 3 of "Crystal Clear," Published by Addison-Wesley, 2004
[11]http://www.solutionsiq.com/PDF/Sulaiman-AgileEVM.pdf
[12] http://www.mountaingoatsoftware.com
[13] http://weblogs.asp.net/jcogley
[14] Agile Project Management with Scrum By Dafydd Rees
[15] http://www.controlchaos.com
[16] http://en.wikipedia.org/wiki