Goal-Driven Backlog Development – User Story Mapping

The following document is created from a number of on-line documents all of which are fully referenced at the end of this one. At times the words are a direct copy of blog posts and so far very personal, at times they are more formal. Sometimes the paragraphs from one document will have paragraphs from another document inserted in the middle.

When a product manager is planning what the product releases will include, the goal is to deliver value for the customers and users. For both agile and waterfall processes, one of the bigger risks is that a product release will include a collection of shiny objects or features. Every release of a product should make it better than the previous release. User story mapping is a technique for assuring that each release or iteration makes the product tangibly better.

An outside-in approach to developing great products starts with market-sensing – understanding the problems that are important to your market. You don’t solve problems with features. In fact, your product doesn’t solve problems. Your users solve problems—and your goal is to create a product that helps them solve or avoid problems.

Big-bang software and IT-system releases are often specified with massive requirements documents that include long lists of stuff to do. A couple of decades ago, people realized that long lists – with often hundreds of items – are particularly difficult to consume and manage in documents, so they started moving them into spreadsheets. These long lists became easy to manage as long as you were willing to forget about delivering value and focus instead on delivering all of the items on the list.

This is one of the manifestations of bad-software-development practices that helped spur the move to agile development processes and incremental delivery – breaking that massive list into a collection of smaller lists. Working in sprints improves the execution aspects of software delivery, but if that’s all you do, you’re still just executing against a list of stuff to do, albeit potentially somewhat more or less efficiently.

However, most agile approaches simply create what can be described as a flat backlog – a list of stories that fail to provide any connection to the value that is to be delivered to users who are trying to solve problems. With user story mapping, you organize the user stories or tasks that users perform into groups that identify which stories are collectively required to deliver value.

User story mapping – a technique for assuring that you are delivering value, not just a list of features in your product – is a powerful technique for assessing the completeness of your requirements. It also gives you a tool for assuring that each release or iteration adds value by enabling a user to solve, or more effectively solve, a problem – avoiding a delivery that only solves half of a problem.

The user story map gives you a tool for organizing what is included in each release – solutions to problems. This also drives you to think about incremental releases becoming incrementally more useful at solving problems, instead of becoming incrementally less incomplete at solving problems.
Why flat backlogs don’t work
Arranging user stories in the order you’ll build them doesn’t help explain to others what the system does. Try handing the user story backlog located at Appendix A at the end of this document to a group of stakeholders or users when they ask you the question "what does the system you’re building do?" Do you think your or they will have any real idea?

Trying to understand the system - the whole system - is the difficult part of software development. One of the most common complaints from agile teams is that they lose the big picture - that is if they ever had it in the first place. You need context in order to really tell a story about the system.

For the typical project, the number of stories in a backlog is dozens at a minimum - often over a hundred. It is hard work even at the start to keep the stories at a high level. Even then it's common to end up with a 120 or so stories in a first cut of a backlog. Stepping through each one of these and making an in-out decision is tedious. Some of the most miserable hours can be spent sitting in meetings with groups prioritizing their backlog.

Building or arranging user stories into a helpful shape is more useful – a User Story Map.

Why you should build a story map
- It allows you to see the big picture in your backlog.
- It gives you a better tool for making decisions about grooming and prioritizing your backlog.
- It promotes silent brainstorming and a collaborative approach to generating your user stories.
- It encourages an iterative development approach where your early deliveries validate your architecture and solution.
- It is a great visual alternative to traditional project plans.
- It is a useful model for discussing and managing scope.
- Allows you to visualize dimensional planning and real options for your project/product.

User Story Mapping (http://www.agileproductdesign.com/blog/the_new_backlog.html)
A prioritized user story backlog helps to understand what to do next, but is a difficult tool for understanding what your whole system is intended to do. A user story map arranges user stories into a useful model to help understand the functionality of the system, identify holes and omissions in your backlog, and effectively plan holistic releases that delivery value to users and business with each release and provides a useful tool for the entire team to understand the big picture – to see the entire breadth of the system and its diverse set of users and uses.

A small story map might look something like this:
At the top of the map are "big stories." I call them user activities (borrowing the term from UX people like Larry Constantine and Don Norman). An activity is sort of a big thing that people do – something that has lots of steps, and doesn’t always have a precise workflow. If I was building an email system (which I’d never be foolish enough to do) I might have an activity called: "managing email", and "configuring email servers", and "setting up out of office responses."

A story for an "activity" might read: As a consultant I want to manage my email so I can keep up with clients, colleagues, and friends.

But that’s way too big of a story to put into an iteration or sprint.

That story breaks down into other stories like "send message," "read message," "delete message," "mark message as spam" - stuff like that. I call these user tasks. (Again a word used by UX people.) For lots of Agile people "tasks" refer to the things that developers do to finish user stories. Really a task is something that someone does to reach a goal. A user task is what users do to reach their goals, developer tasks are what developers do to create stories, Ant tasks are what ant does to... well... do whatever you're doing with Ant.

I simply arrange the small things under the big things in a bit of a grid form.

I’m always imagining time moving left to right (because it does in my western world. My counterparts from top down or right to left worlds can translate as needed). For instance when arranging stories in the map, if a person using the system typically does one thing after another, then I'll put the early thing on the left, and the later thing on the right. I do this with the big things and the little things - the activities and the tasks.

When teaching this, people often tell me "the users can perform these in any order. What order should I put them in?" I'll ask them to "explain to me what the system does at a high level - just tell me the activities." They then recite them to me. "That’s the order" I say. In fact, the order you’d explain the behaviour of the system in is the correct order. We're building a map that lets us tell a really big story about the system. Build the map in a way that helps you tell the story.

The really big stories can be considered "epics" as Mike Cohn describes them. They’re stories - just really big ones - too big to estimate and build. When an epic gets in your backlog, and it's time to discuss it in more detail, I often see people remove the epic from the backlog, decompose it, and replace the pieces they've identified. This is where I cringe. The big story was context. It was my
simple way of thinking about the whole activity that people were doing. It’s my quick way of explaining to others what the system is about.

**Test your map – have you got the big picture?**

A fully assembled map is something that can be hung on a wall or laid out on a table top and actually used as an integral part of a discussion. The map can be walked from beginning to end with a user, stakeholder, or developer and tell a story about the users of the system and what they’re doing. It is possible to skim along the top of the map, and just touch on the high points or you can dig down into the map to discuss the details.

Talking through the map with users and others helps find things that have been missed. Frequently you will hear "you've missed a couple steps here" from users when doing this. The map can be annotated with pain points or opportunities. Talking through the map with a user may elicit comments such as, "this here is really a problem with the system today."

It can be a very valuable tool for working with users. For example, when working with users they often correct me. Yesterday I was working with a potential user of a new system - she was a compliance officer for a major company. I write down a process step as she said it and placed it on the table. I'd been using green cards for big things, yellow for smaller things. She quickly corrected me - "no, that should be a yellow card." In less than an hour’s discussion she had caught onto the modelling approach and could, in very few words, supply me with a lot of information about the thing she was doing with the system. She knew yellow was smaller than green, and could tell me that the thing she was doing in the system wasn't as big as I thought it was.

Building and walking story maps leaves me more comfortable than ever before that I "get it" - that I haven’t missed something big.

**The software has a backbone and a skeleton – your map highlights that**

The big things on the top of the story map look a little like vertebrae. And the cards hanging down look a little like ribs. Those big things on the top are often the essential capabilities the system needs to have. These are the "backbone" of the software.
When it comes time to prioritize stories don't prioritize the backbone. It just "is". Prioritize the ribs - the stories hanging down from the backbone. Place them high to indicate they're absolutely necessary, lower to indicate they're less necessary. When you do this, you'll find that all the stories placed high on the story map describe the smallest possible system you could build that would give you end to end functionality. This is what Alistair Cockburn refers to as the "walking skeleton. Always try to build this first.

Plan using your backbone
When it's time to plan releases, it’s usually not important to prioritize backbone items against each other. For instance a high level backlog for a car it might look something like this:

- engine
- transmission
- brakes
- suspension
- ...

It would be stupid to ask stakeholders to prioritize that: "what's more important, the engine or the transmission?" – or "we don’t have enough time in this release, could we release without brakes and add them later?" These items are essential - and all of them are needed to deliver a minimum viable product (MVP).

Where the prioritization comes in is below this level: 4-cylinder engine or 6-cylinder engine? brakes with anti-locking or brakes without? sport suspension or not? It's how we build up those backbone items - prioritize their characteristics that matters.

When you prioritize a story map, you'll move cards or stickies up and down to indicate high or low. A long strip of masking tape can be used to creating horizontal swim lanes for each release. Stories can then be moved up and down into each lane, and even vary their height in the lane.

Keep your map displayed to communicate the big picture.
Building a story map helps you initially understand the functionality. Ask yourself when in your project do you not need to understand your functionality any longer? I know it's not a fair question, but I do find some folks that spend a great deal of time building this sort of thing to understand the problem, then tossing it all out in favour of putting stories into a flat-backlog. Cutting down the tree and loading the leaves into a mulch bag.

A story map hung as an information radiator becomes a constant point of discussion about the product that’s being built. When the project is running, it becomes the sprint or iteration planning board. Identify or mark off stories to build in the next iteration directly on the map. During the iteration place just the stories that are being worked on into a task wall to managing their development - but the story map lives on the planning wall to remind what the big picture is, and progress made.
When building software incrementally, story by story, choose them from the story map left to right, and top to bottom. Slowly move across the backbone, and down through the priorities of each rib. Slowly the system is building up, not a feature at a time, but rather by building up all major features a little at a time. That way a car is never released without brakes.

**A different backbone may be in order for adding features to an existing product**
When adding features to an existing product, it already has a backbone - and sufficient functionality to have released. I find it useful to still identify the backbone to help me get context - to help me see where new features are being placed.

On some projects adding just a few features to a large existing product, it’s been difficult to talk people into building up a story map for just a few features. In these cases, I'll simply prioritize the new features, then for each feature build a little story map to prioritize the user stories that make up that features. Each little story map may have 10 or so cards - but they're still arranged left to right, and top to bottom so we can focus on building a little-tiny-walking-skeleton of the features as early as possible.
How to create a User Story Map (http://winnipegagilist.blogspot.ca/2012/03/how-to-create-user-story-map.html)

1. Form a group of 3-5 people who understand the purpose of the product. 3-5 seems to be the magic number. Any less and you might miss some ideas. If you add more people, it slows the process down and there are diminishing returns on the quality of ideas generated.

2. Start by gathering the major user tasks of the project/application in silence - the “things people do”. Each person takes the same coloured post-it and silently writes down one user task per post-it. Once everyone has finished writing their post-its, have each person read their post-its aloud and place them on the table in front of the group. If anyone has duplicates they should be removed at this point.
   - Depending on the size of your application it can take 3-10 minutes to get all the major tasks, but you can watch the body language to see when they are done. You will see that the group goes from huddled in at the beginning to standing/leaning back at the end.
   - Likely each post-it starts with a verb. (e.g. Compose E-mail, Create Contact, Add User, etc)
   - These are the high level user stories called “User Tasks” which forms the “walking skeleton” of the map.
   - When they are done, point out to your team that in a few minutes they gathered most of the high level scope and that if they missed a user task or two, someone else probably didn’t. This might be their first ‘aha’ moment for silent brainstorming.

3. Next ask the team to group the post-its in silence. Simply ask them to move things that are similar to each other closer to each other and things that are dissimilar to each other should be moved farther apart.
   - Use silent grouping simply because it is faster than grouping them out loud.
   - If duplicates are found, remove them.
   - Groups will form naturally and fairly easily.
   - Once again, body language will help you see when they are done - usually 2-5 minutes.

4. Using another colour of post-it, name each group and put the post-it on top of the group. This step can be done out loud.

5. Arrange the groups left to right in the order a user would typically complete the tasks.
   - If the group can’t decide on an order between two or more tasks, it probably doesn’t matter.
   - The groups are called “User Activities” which form the backbone of the map.
   - You should now have the first 2 rows of the user story map - something similar to this:
     
     A1  A2  A3  
     T1 T2 T3  T4 T5  T6 T7 T8 T9

     (user activities = backbone)
     (user tasks = skeleton & timeline)

6. Now walk the skeleton to make sure you haven’t missed any major user tasks or activities. You can walk through user scenarios, or even bring in users and ask them to walk through their job functions to make sure everything is accounted for.
7. With a finished framework for the map, you can add more detailed user stories below each user task before breaking your map into releases. I like to use the same silent brainstorming technique to generate the initial set of stories for each user task and augment it with other techniques such as user scenarios and personas. Once you have a good set of stories, then you can put your release lines in the map and ask your users to prioritize top to bottom.

- I like to put the first release line on the map pretty high on the map so that it only allows for 2-3 user stories per user task in the first release. This has been an effective way to encourage prioritization and scoping.
- Because all the stories end up on small post-its, you can forgo the traditional 'as a' format and just put the story title on the post-its. This will allow you to read the map a little easier.

8. Finally, I like to take all the user stories in the first release and do some serious user story slicing to make sure we have sliced the first release as thin as possible. As a guideline, you should strive to be able to create the whole app (at least one small user story slice for each user task) in the first iteration or two.

"An example would be handy right about now" (Brian Marick)

Here is an example user story map for creating the next email client competitor:

- The second row contains the "things people do" in an email client such as Compose Email, Send Email, and Create Appointment.
- The first row contains the groupings for those things people do.
The first yellow row is the smallest user story slice for each user task. For Compose Email, that means a basic form with To, Subject, and Body text fields plus Send and Cancel buttons. Functionality for RTF support, HTML support, adding attachments, getting email addresses from your contact list etc., are not included in this user story but are included as user stories lower in the map.

The small blue and orange post-its on the larger yellow post-its indicate the status of that user story. The blue ones say “done” and the orange ones say “wip” so that you can see your project status flow down the map.

If we now build the first row left to right before working on any stories in the rest of the map we can deliver all of the basic functionality quickly. This enables us to validate our messaging architecture (i.e. send an email and then make sure you can Read it). It also allows us to test all the basic functionality in the application end to end so that we can make sure it all works together while getting feedback on whether or not it will solve the business problem. If we were building the very first e-mail client, we could release to the public at this time. Notice also that we did not have a user story for "Delete Contact" in the first release - we don’t need to deliver functionality for each "User Task" in each release.

Finally, some definitions:

- The post-its you create in Step 2 are the User Tasks (blue post-its in the diagram).
- The groups and group names in steps 3 and 4 are the User Activities (orange post-its). Jeff calls these top two rows the backbone and walking skeleton of your application.
- The user stories (yellow post-its) are organized under each User Task in order of highest to lowest priority for that User Task.
- The chronological order of how users will typically use the application goes left to right (Time).
Source Documents


The new user story backlog is a map - http://www.agileproductdesign.com/blog/the_new_backlog.html - last accessed 4 April 2013

How to create a user story map - http://winnipegagilist.blogspot.ca/2012/03/how-to-create-user-story-map.html - last accessed 4 April 2013
A Product Backlog Example

The following product backlog example sample is the product backlog written for the Scrum Alliance website in 2006. This product backlog lists everything that the product owner and Scrum team feels should be included in the software they are developing.

Profiles

- As a site member, I want to describe myself on my own page in a semi-structured way. That is, I can fill in predefined fields, but also have room for a free-text field or two. (It would be nice to let this free text be HTML or similar.)
- As a site member, I can fill out an application to become a Practitioner.
- As a Practitioner, I want my profile page to include additional details about me (i.e., some of the answers to my Practitioner application).
- As a site member, I can fill out an application to become a Trainer.
- As a Trainer, I want my profile page to include additional details about me (i.e., some of the answers to my Trainer application).
- As a Practitioner or Trainer, when I provide content to the site I want a small graphic associated with the content indicating I’m a Practitioner or Trainer. (For example, Amazon’s “Top 500 Reviewers” approach.)
- As a trainer, I want my profile to list my upcoming classes and include a link to a detailed page about each.
- As a site member, I can view the profiles of other members.
- As a site member, I can search for profiles based on a few fields (class attended, location, name).
- As a site member, I can mark my profile as private in which case only my name will appear.
- As a site member, I can mark my email address as private even if the rest of my profile is not.
- As a site member, I can send an email to any member via a form.
- As a site administrator, I can read practicing and training applications and approve or reject them.
- As a site administrator, I can edit any site member profile.

News

- As a site visitor, I can read current news on the home page.
- As a site visitor, I can access old news that is no longer on the home page.
- As a site visitor, I can email news items to the editor. (Note: this could just be an email link to the editor.)
- As a site administrator, I can set the following dates on a news item: Start Publishing Date, Old News Date, Stop Publishing Date. These dates refer to the date an item becomes visible on the site (perhaps next Monday), the date it stops appearing on the home page, and the date it is removed from the site (which may be never).
- As a site member, I can subscribe to an RSS feed of news (and events? Or are they separate?).
• As a site editor, I can assign priority numbers to news items. Items are displayed on the front page based on priority.

Courses and Events
• As a site visitor, I can see a list of all upcoming “Certification Courses.” I can page through them if there are a lot.
• As a site visitor, I can see a list of all upcoming “Other Courses” (non-certification courses). I can page through them if necessary.
• As a site visitor, I can see a list of all upcoming “Events.” (Events are things such as the Scrum Gathering, conferences, free seminars, etc.)
• As a trainer, I can create a new course or event. This includes the following information: name, description (HTML), trainer names (multiple selection from a list), start date, end date, venue name (HTML) and address, contact name, contact phone, contact email, a link for more information, and a link to register. For a certification course the name of the class is a dropdown list; for others, it is free text.
• As a trainer, when I create an Other Course or Event, I am charged a listing fee for that activity. (Note: We’ll need this to tie into credit card processing.)
• As a site administrator, I can create an Other Course (?) or Event that is not charged a listing fee. This is so that the Scrum Alliance doesn’t charge itself for Scrum Gatherings that it puts on.
• As a site administrator, I can set the listing fee per Other Course or Event.
• As a trainer, I can update one of my existing courses or events.
• As a trainer, I can delete one of my courses or events.
• As a trainer, I can copy one of my courses or events so that I can create a new one. When copying it I am asked for the date(s) of the new course or event.
• As a site admin, I can delete any course or event.
• As a site editor, I can update any course or event.
• As a trainer, admin, or editor, I can turn a course into an event or an event into a course (in case it was entered in the wrong category). (Note: making something a Certification Course will probably require selecting the name of the course from the pre-approved list.)
• As a site visitor, I have an advanced search option that lets me fill in a form of search criteria (country, state, trainer name, date range, word in description, etc.).
• As a site visitor, when I’m viewing a course I can click on the trainer’s name and be taken to the trainer’s profile.
• As a site visitor, I can subscribe to an RSS feed of upcoming courses and events.

FAQs
• As a site visitor, I can read FAQs.
• As a site editor, I can maintain an FAQ section.
• As a site member, I can do a full-text search of the FAQs. (Maybe we want this for any site visitor?)

Resource
• As a site member, I can download the latest training material and methodology PDFs.
• As a visitor, I can download presentations, PDFs, etc. on Scrum that I can use.
• ....more needed here................
Goal-Driven Backlog Development

Jobs

- As a site member (?), I can scroll through a listing of jobs. (There won’t be enough at first to justify search fields.)
- As someone who wants to hire, I can post a “help wanted ad”.
- As a site admin, I need to approve each help wanted ad before it gets to the site.
- As a site admin, I am emailed whenever a job is submitted (so that I am aware of it and can decide if I want to post it).  
- As a site member, I can subscribe to an RSS feed of jobs available.
- As a site admin, I can edit and delete help wanted ads.
- As a site admin, I want jobs to stop publishing on the site 30 days after being posted. (Note: 30 days doesn’t need to be configurable at this point. Hardcoding is fine for now.)
- As someone who wants to hire, I want to be able to extend an ad for another 30 days (repeatedly) by visiting the site and updating the posting. (Note, I can’t update it 10 times today and extend the posting 300 days today.)
- As someone who has posted an ad that is about to expire, seven days before it expires I want to be emailed a reminder so that I can go extend the ad. (Note: This means the ad could have an expiration date 37 days into the future, which is fine.)

Articles

- As a site visitor, I want to read a new article on the front page about once a week.
- As the site editor, I can include a teaser with each article. The teaser shows up on the front page for all to read.
- As a site member who has read a teaser on the front page, I want to read the entire article. (Note: We want any site visitor to read articles for now.)
- As the site editor, I can add an article to the site.
- As a site editor, I can set start publishing dates (when teaser appears on front page), old article date (when it disappears from home page), and stop publishing dates (when it’s removed from site, if ever) for articles.
- As a site editor, I want to be able to designate whether or not an article (or for that matter any piece of info) ever makes the home page… some things will not.
- As the site editor, I have pretty good control over how the article looks (include images and captions, for example).
- As a site visitor, I want the link from the article teaser to take me directly to the body of the article… not to another teaser setup.
- As a site editor, I want to be able to indicate whether an article is publicly available or for members only.
- As a site visitor, I want to be able to read some of your articles.
- As a site member, I want to have full access to all articles.
- As a site visitor, I can do a full-text search of article body, title, and author name.
- As a site visitor, I can subscribe to an RSS feed of articles. (Teasers only?)
- As a site visitor, I can post comments about articles so that others can read them.
Home Page
- As a site editor, I want to have a prominent area on the home page where I can put special announcements, not necessarily news or articles.
- As a site editor, I’d like to have some flexibility as to where things appear to accommodate different types of content.
- As a site member, the upcoming courses are what I want visitors to notice.
- As a site visitor, I want to see new content when I come to the site.
- As a site visitor, I want to have articles that interest me and are easy to get to.
- As a site editor, I have ideas on how I want the home page to look and feel.
- As a site visitor, I need to know as soon as I visit what on earth Scrum is, and why it needs an alliance.
- As a site visitor, I want to know as I glance around the home page what on earth a CSM is and why I’d want to be one.
- As a site visitor, I want to be able to get back to the home page quickly and easily.
- As a site visitor, I want to see a list of the most popular items on the site. (Note: Not everything has to be considered. For example, we don’t need to know the most popular profile but it would be useful to have a “most popular” box that listed the most popular articles, news items, or etc.)

Ratings
- As someone who successfully completed a Certification Course (becoming a Certified ScrumMaster or Certified Product Owner), I am emailed a link to a survey about the course and instructor.
- As a trainer, I want to be assured that no one can submit the same answers multiple times and skew my results.
- As a trainer, I am notified about the results of surveys about my classes. (Questions: After each survey? After a set period of time? Does the trainer get an email or just know to go to the site?)
- As a site admin, I can see the results for each trainer and averages for the class (for all trainers).
- As a site visitor who is considering attending a certification course, I want to see a trainer’s rating (either for that course or for all of his or her certification courses combined).
- As a trainer, I want my rating to show up on my profile page.

What Is Scrum?
- As a site visitor, I want there to be a section of the website that teaches me the basics of what Scrum is.
- As a site editor, I can create the content of the What Is Scrum section.

Registry
- As a site visitor, I can view lists on the site of all Certified ScrumMasters, Practitioners, Trainers, and Certified Product Owners. (The CSM list has over 5,000 names so a letter-based pagination approach is needed.)
- As a CSM, Practitioner, or Certified Product Owner, I can have my name listed in the registry without becoming a member of the site. (For example, I take a certification class but never register or let my membership lapse.)
• As a trainer who has finished teaching a Certification class, I can load an Excel file (first name, last name, email) into the site. I am prompted for the trainer names (I may not have trained alone), certification date, and type of certification (i.e., CSM or CPO). The names are loaded into a pending state and not yet added to the registry. (Note: We could have this charge $50 per person right then.)
• As a site admin, I can view all classes in a pending state.
• As a site admin who has received proof of payment from a trainer, I can move people in his or her class from a pending state to the registry.
• As a new Certified ScrumMaster or Certified Product Owner, once my name has been loaded to the registry I am sent an email welcoming me to the Scrum Alliance and containing instructions on how to register / activate my membership.
• As a site editor, I can edit the content of the email automatically sent to new Certified ScrumMasters and Product Owners.

Membership
• As a company, I can join the Scrum Alliance by paying a corporate membership fee. This will include uploading items related to corporate membership (e.g., company description, a logo of size x by y).
• As a corporate sponsor, my logo is displayed on a “corporate sponsors” page just like at http://www.agilealliance.org/portal_url/corporatemembers. (Note that the display order on that page is random.)
• As a corporate sponsor I want my logo to randomly appear on the home page. (That is, it rotates among other corporate sponsors.)
• As a CSM or CPO who has been approved for Practitioner status (by a site admin reading my submission), I am charged a fee.
• As someone about to become a trainer, I can pay an annual fee.
• As a site administrator, I can set the annual fees for members, Practitioners and Trainers.
• As someone whose membership (of any type) is about to expire, I am sent a reminder and a link through which I can renew. (Note: Think about overlapping memberships and prorating.)
• As a member with short-term memory problems, I can have the system email me a new password or a password reminder, possibly my username (unless we use email for that), and so on.

For Trainers Only
• As a trainer, I can read information of relevance only to trainers.
• As a site editor, I can post information in a trainers-only section.