The Viking

A Direct Marketing System

Mark L. Fussell
ChiMu Corporation
mark.fussell@chimu.com

Introduction

The Viking is a system that maintains information about customers and can send them letters tailored with important, relevant, and time-critical information. The Viking is the core of a direct marketing system: it will directly interact with users to configure the campaigns and will work with other system components to accomplish the mailing itself.

The functionality needed from The Viking is described through requirement stories (informal use cases). Each story describes an interaction that the system must be able to support, and the collection of stories provides a prioritization of system capabilities. The more high-priority stories your design or implementation of The Viking supports, the bigger and better a Viking you will have created and the happier your company will be with your performance.

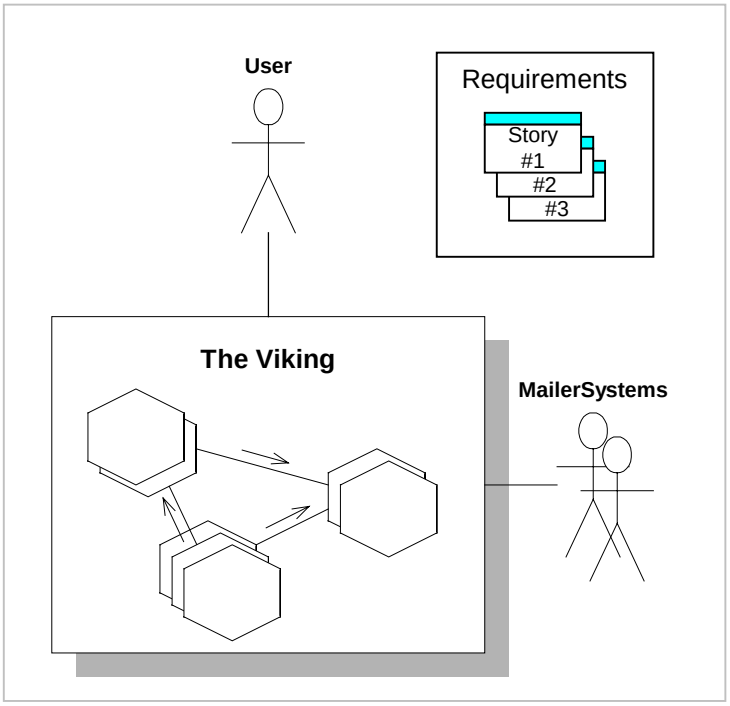
The Viking's requirements/capabilities are described in the attached stories, but the following provides a short summary of these details. Ultimately, **The Viking** must be able to:

- Maintain any amount of information related to a **Customer** (either directly or indirectly) and make that information accessible to the **Letter** generation process.
- Select a group of customers that the **User** desires to send letters to.
- Generate letters from **Templates** that describe both the common, static information for the letter and the pluggable, dynamic information that must be retrieved from customer information or other information (like today's date). A single template can be used to generate appropriate letters for thousands of customers.
- Deliver these letters through external **MailerSystems** that send electronic email or to print the letters and envelopes for automated delivery. Supporting the electronic delivery is more important.
- Maintain the status of letters: whether they were sent to customers, whether delivery was successful or rejected, and whether the user responded to the letter.

Further details of the capabilities will come out in the stories below. Together these capabilities enable The Viking to support the direct marketing functionality your company requires.

System Context

The Viking is a system that needs to interact with two kinds of external actors: human Users and MailerSystems. Users will be interacting with The Viking to examine customer information, generate letters, send them to the MailerSystems, and monitor the status of the sent letters. The Viking will interact with the MailerSystems to send letters and the MailerSystems will interact with The Viking to give status about sent letters or to receive replies. This paragraph is just an overview; see the Requirement Stories for more detailed requirements.



Requirement Stories

The following pages contain The Viking's requirements expressed as stories (informal use cases). Each story has a header area containing an identifier, a name, and a date: these are just for reference purposes. More importantly, each story is given a priority. The Viking **must** support higher priority stories (lower numbers) before it can be considered to support lower priority stories: that is, if The Viking supports stories with priority (#1,#2,#5,#6) it effectively only supports two stories (stories with priority #5 and #6 are ignored). The rest of the header fields (or any additional fields) can be used as a team desires.

The next section of a Story is a description of the functionality The Viking needs to provide to support that Story. Following this is a section for requirement notes that augment the description. Finally, there is a section for any other notes the team wishes to associate with the story.

Example Templates

Some example templates are mentioned within the Requirement Stories. Note that the format of these templates is meant to represent functionality and ease of use. It is not required that The Viking use the same special delimiters or other characteristics, but it must be easy for a user to edit and preview the templates both with and without sample data.

Final Notes

Bonus credit will be awarded to teams who can correctly identify the reason The Viking is so named.

Story ID	**1**	**Date**	**2000-04-01**	**Priority**	**1**
Name	**Letter Generation**				
		Development Risk		**Estimate**	

Description:

A user selects a set of customers to whom they wish to send letters and a template that defines the letter format. The Viking then generates a letter per customer that is based on filling in the "pluggable" information for the template with customer-related information. The user then previews the result of expanding the template for each of the customers.

Requirement Notes:

For this story, a template like:

> Dear «Proper Salutation» «Customer's Name»,
>
> Thank you very much for ordering from us on «Date of Most Recent Order». We recently received several thousand cans of the very special ingredient, which has returned from a luxurious trip to Hawaii, and we can make a one-time offer of just US$99.99 per case. If you are interested, just call 1-800-555-1234 or click to:
> http://www.TheHappyViking.com/SpecialOrder/«Special Order Number»/order.html
> If you contact us by phone, mention special order #«Special Order Number» to get this extra special discount.
>
> Thanks again from all of us at The Happy Viking. We hope you have many days and nights enjoying this wonderful delicacy.

should be considered representative. As a first stage you could even consider just the pluggable information on the first line.

Notes:

Story ID	2	Date	2000-04-01	Priority	3
Name	Adding Customer Information				
		Development Risk		Estimate	

Description:

A user knows of a new customer that she wants to add to The Viking. The user can create a new customer entry and record relevant information (Name, Salutation, Address, Recent Purchase, and anything else needed by other parts of The Viking) for that customer.

Requirement Notes:

Ultimately, it will be necessary to support arbitrary information associated with a customer, but for this story the information necessary can be "hard-coded" to be just the information mentioned in the description and augmented as necessary to satisfy other stories.

Notes:

Story ID	3	Date	2000-04-01	Priority	2
Name	Sending Letters				
		Development Risk		Estimate	

Description:

A user chooses from among the generated letters and decides which ones to send out and by which MailingSystem to send them. The Viking should already have information associated with each customer so it can properly distribute the letter by a particular MailingSystem: the user should not need to enter this information as part of the sending process.

Requirement Notes:

To support this story alone, it is only necessary to send by SMTP or similar for email, and to print out the letter (with an address header) for postal mail. Future stories may increase the number of MailingSystems.

Notes:

Story ID	4	Date	2000-04-01	Priority	6
Name	Customer Selection by Criteria				
		Development Risk		Estimate	

Description:

Upgrade the template from Story#1 to support selecting the set of customers by various search criteria. For example, select all customers who spent more than US$100 on their most recent order, or all customers who have ever bought a particular product.

Requirement Notes:

Notes:

Story ID	5	Date	2000-04-01	Priority	8
Name	Add New Property Types				
		Development Risk		Estimate	

Description:

Allow a user to augment The Viking's information model with additional property types for customers. A user should be able to enter property values for each of these new property types for any given customer, and to access the property types within the templates. No information currently within The Viking can be lost when recording additional property types. You can assume new properties are always simple strings.

Requirement Notes:

An example property type might be "Pet's Name" and "Pet's Pronoun" (He, She, or It). A corresponding template might be:

Dear «Proper Salutation» «Customer's Name»,

Thank you very much for ordering from us on «Date of Most Recent Order». We are sure «Customer's "Pet's Name"» will love the delicacy. Please give «Customer's "Pet's Pronoun"» our best wishes and if «Customer's "Pet's Pronoun"» would like to order some more, call us at: 1-800-555-1234 or connect to:
http://www.TheHappyViking.com/

Thanks again from all of us at The Happy Viking. We hope «Customer's "Pet's Name"» has many days and nights enjoying this wonderful delicacy.

Notes:

Story ID	6	Date	2000-04-01	Priority	9
Name	Template Grouping				
		Development Risk		Estimate	

Description:

Instead of only having a single template selected by a user, enable the user to create/store/recall a named group of templates that will determine a single template to use based on information from a customer. So the user can pick a template group and a thousand customers and the appropriate template (based on the users criteria captured within the group) will be used for each customer. Only a single template will ever be selected (so a single letter generated) per customer, so there must be a deterministic set of (user controllable) rules to make the selection within the group.

Requirement Notes:

For example, the group of templates might be in French, English, Chinese, etc. And each customer has a "Preferred Language" property that will be used to choose amongst the templates. After that, the template evaluation can be the same as always.

Notes:

Story ID	7	Date	2000-04-01	Priority	7
Name	List Information Support				
		Development Risk		Estimate	

Description:

Upgrade the template from Story#1 to support multiple (list) information associated with a customer.

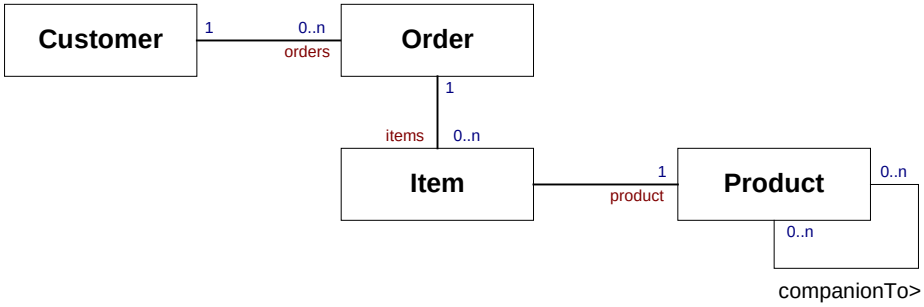
Requirement Notes:

For this story, a template like:

> Dear «Proper Salutation» «Customer's Name»,
>
> Thank you very much for ordering from us on «Date of Most Recent Order». Given your order, can we recommend the following companion products:
>
> «List of companion products and their prices based on the items within the recent order»
>
> All of these items are being offered at special discount prices, just for you. If you are interested, just call 1-800-555-1234 or connect to:
>
> http://www.TheHappyViking.com/SpecialOrder/«Special Order Number»/order.html
>
> If you contact us by the phone, mention special order #«Special Order Number» to get this extra special discount.
>
> Thanks again from all of us at The Happy Viking. We hope you have many days and nights enjoying this wonderful delicacy.

should be considered representative. The format of the list can initially (for this story alone) be automatic.

To support this story, The Viking will have to associate customers with the products that they purchased and then products with companion products. An example information relationship might be:



Notes:

Story ID	8	Date	2000-04-01	Priority	5
Name	Delivery Monitoring				
		Development Risk		Estimate	

Description:

A user reviews the letters to see which have been sent and whether any of them have "failed delivery". If so, the user can choose to resend them either by the same MailingSystem or by a different MailingSystem. Also, to support this story, the MailingSystem must be able to tell The Viking which sent letters "failed delivery". It is up to The Viking to encode enough information in a sent letter to be able to correctly identify (or support a human to identify) which sent letter has "failed delivery".

Requirement Notes:

For the case of postal mail, a person will represent the MailingSystem and will need to be given a UI to find a sent letter and say that it "failed delivery". For the case of electronic mail, the MailingSystem should be able to talk directly with The Viking (i.e. through a non-UI API).

Notes:

Story ID	9	Date	2000-04-01	Priority	4
Name	Template Creation				
		Development Risk		Estimate	

Description:

A user creates a new template either from scratch or by copying an existing template. A template needs to support both constant and “pluggable” information, and a user should be able to create a template and preview its appearance. The editing of a template does not need to be WYSIWYG, but it could be.

Requirement Notes:

It would be nice if the previewing capability included being able to see the template with and without example data, but this is not required.

Notes:

Story ID	10	Date	2000-04-01	Priority	10
Name	Multiple Final Formats				
		Development Risk		Estimate	

Description:

The Viking needs to have the ability to generate multiple final formats (e.g. PDF, ASCII, HTML, etc.) from a single "content template". The "content template" should describe the information, wording, and structure of the resulting letter. A set of "formatters" should take this resulting generic letter and be able to convert it into PDF, ASCII, HTML with various styles, and so on. Basically this is just a content vs. format separation so someone focused on presentation issues can deal with the difficulties of HTML styling and the other format issues. The "content template" writer can focus on what information needs to be communicated to the customer independently of the styling.

Requirement Notes:

Notes:

CodeFest Description

The problem description of The Viking for CodeFest is the same as for DesignFest. Depending on your CodeFest team, you may also have DesignFest deliverables that can help you build The Viking.

The Viking can be built incrementally: it is acceptable to support just the first few highest-priority stories. But to successfully complete functionality for The Viking you must have Functional Tests that show the system can support the particular stories. Some of the Functional Tests could include a human script for GUI interaction, but most of the tests should be automated. Although you may not have a Subject Matter Expert (SME) available within the CodeFest team, you should have someone in the team represent the SME and make sure the functionality of the stories is within the system.

The Viking can be built using any third party applications and libraries that together total less than US$800 per seat. For demonstration purposes, you can use evaluation copies of products. A third party product should not interfere with enhancing The Viking's core functionality, so ideally the product would be open source or have a very well defined responsibility that is unlikely to grow beyond the product's abilities.

DesignFest Submission Notes

I think The Viking is an intermediate or introductory design problem. It is a little "larger" than The Vending Machine problem, but it is definitely doable by a mostly beginner or intermediate team. I recently used this problem as part of an OO training session with a team of programmers new to OO and they could complete a CRC, Java Design, and a Java domain implementation (no GUI but verified by Functional/Scenario tests) in a single day. The Viking definitely does not have any particularly difficult or obscure domain issues that would make it hard.

The biggest drawback to this design problem is that The Viking is a dangerous tool and I do not want tools like this falling into the wrong hands.

Difficulty Adjustments

The difficulty of The Viking as a design problem is partially dependent on how much "Analysis" is provided. In this first submission I left any real analysis deliverables out (for example, there is no abstract system model or a conceptual model). We could put that in as an optional section if people desired, but DesignFest problems have tended to be pre-Analysis in the past and teams may like that. We might make it optionally available to the team. Note that in my submission I intentionally lowercased words like 'customer', 'user', and 'letter'. Just uppercasing these words might make the design easier for beginners.

Obviously a major difficulty adjustment is the number of Stories a team's Viking can support. The priority of the stories is meant to provide this path of increased difficulty. We might also want to group stories into sets that are given out at different times to make them less intimidating (or to test system requirement changes on a team). This may have been tried on previous DesignFests, but I do not remember the results.

CodeFest

I think this could also be a good CodeFest problem: it is very incrementally completable and over a couple days of work the completed functionality could vary between:

- A fully functional system with a reasonable GUI to manage Customer information, the ability to generate interesting Letter formats (HTML, RTF), and even to send them by SMTP.
- A system pre-configured with Customer data, which can be executed with Functional tests, and which sends some ASCII letters into a black-hole SMTP façade or out to a transcript/log.

In any case within these extremes, a system could be demonstrated.

Because of the potential for very different capability levels without excessive time commitment, this might be a reasonable problem to try to get experienced coders (normal conference attendees) to take a shot at. The major drawback is that the result is not at all useful: it might be more interesting to do a WikiWiki update or something. The trade is that the problem is simpler to describe to a wide audience without it being something everyone has already done.

Similarly, I think this problem lends itself to both an XP approach and a pre-Designed approach over a similar two-day timeframe. It might be a way to get a large (informal) comparison of the XP approach to the pre-Designed approach. Probably the biggest problem with this is that skill and experience levels will vary too much to be a useful comparison.

Finally, the language and tools used for implementation would make a dramatic difference in completed functionality. Combining tools like an enhanced XSLT processor (Cocoon), XMetal, a language with Serialization capability or good DB access, a GUI builder, and a built-in SMTP library would do more than 50% of the system. Not really a problem, but just a note.

A Happy Viking has many stories to tell, many songs to sing, but only one food to eat.

Submitted by

Mark L. Fussell ChiMu Corporation
mark.fussell@chimu.com www.chimu.com