28TH ANNUAL
PACIFIC NW
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010

ACHIEVING QUALITY IN A COMPLEX ENVIRONMENT

Conference Paper Excerpt from the

CONFERENCE PROCEEDINGS

User Experience Grading via Kano Categories

Matt C. Primrose
Visual Computing Group
Intel Corporation
Portland, OR. USA
matt.c.primrose@intel.com

***Abstract*— A consistent challenge in product development is determining whether a product meets its usability requirements. In particular, accurate, meaningful usability information is hard to obtain before product release. This paper describes a method to classify use cases, features, and requirements into Kano model categories and then grade each based on how well is has been implemented from a usability perspective. Using this method during product development can provide early, regular data on the status of product usability, help to determine where resources are spent, and perform competitive analysis based on product features and usages. This method was tested on a single case study, and in particular for the requirements of the user interface for a discrete video card.**

User Experience; Kano; Usability; Requirements Verification;

I. INTRODUCTION

Usability is by nature a subjective measurement. Each person may find a product more or less usable than others based on his or her own perceptions and experience. The best that developers can hope for is to create a design that the majority of users will find easy to use while still meeting all of the functional requirements.

The method this paper describes is not a panacea for the problem of measuring usability but instead can be used by development, validation, and product marketing to help guide developers in making good choices when it comes to allocating resources, cutting or keeping features, and whether to add that "one generation ahead" feature or leave it out until the next version.

The process flow [Fig. 1] outlines how this method works. The shaded boxes in the flow are where this method adds value to the standard planning and execution flow and will be the focus of this paper. Much of this process work happens during the product planning phase. There is also a small grading effort and usability assessment required during development.

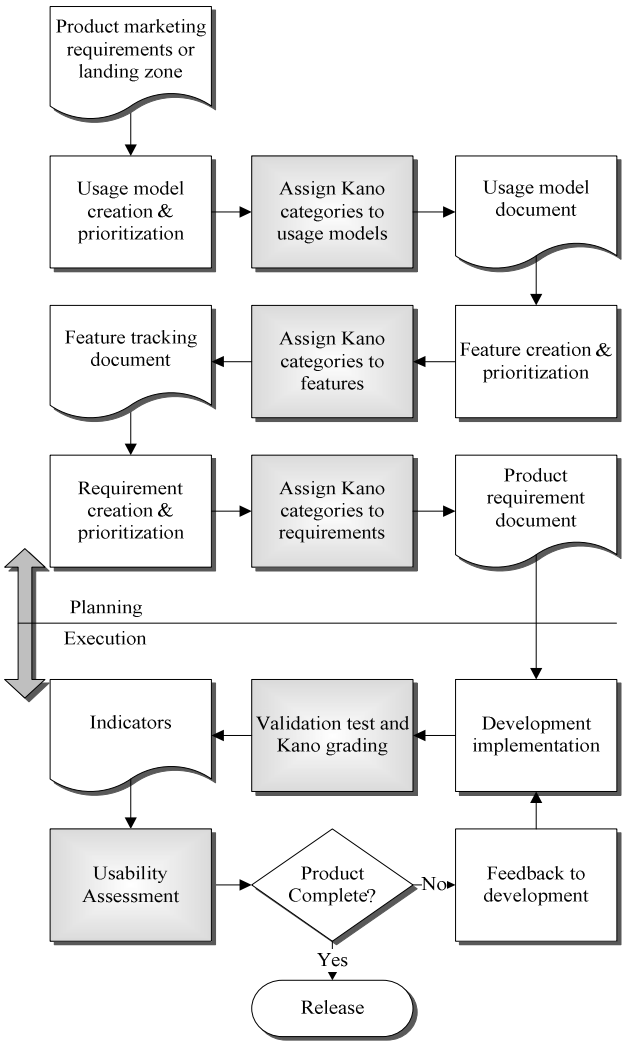

Figure 1: Process flow

II. KANO MODEL

The method of grading and prioritizing development work described here is based on the categorization of use cases, features, and requirements using the Kano model [1]. The Kano Model of Customer (Consumer) Satisfaction classifies product attributes based on how they are perceived by customers and their effect on customer satisfaction.

These classifications are useful for guiding design decisions in that they "indicate when good is good enough, and when more is better" [2]. Following the principals of a simplified version of the Kano model [Fig. 2], each usage model, feature, and requirement is categorized into one of three categories: Must Have, Desired, or Differentiator.

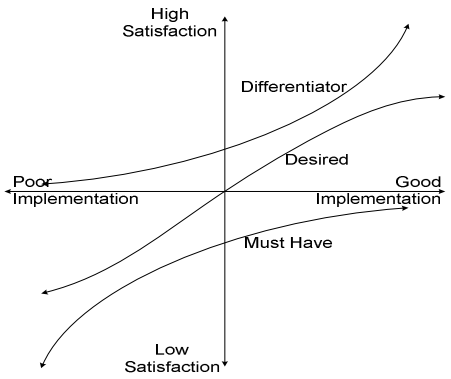

Figure 2: Kano graph

The "Must Have" category includes those components that customers expect to be available for the product being designed. A product missing one or more "Must Have" usages, features, or requirements can result in customer dissatisfaction and leave customers thinking that the developer did not understand the needs of their consumers.

"Desired" usages, features and requirements are related to performance measurements. These could include how fast a product responds to user input or how much of a usage, feature, or requirement is available to the user. The more that the user experience exceeds the minimum threshold the more customer satisfaction will be generated by the product - until a point of diminishing returns is reached. The threshold value as well as the diminishing returns value should be accounted for in the product requirements to ensure that the right amount of development effort is spent.

"Differentiator" capabilities are what uniquely differentiate one product from other similar products. "Differentiator" usages, features, or requirements tend to be transitory in nature. Once consumers have accepted these capabilities, they quickly come to desire and expect them to be available in future products. Therefore, a new usage, feature, or requirement classified in the "Differentiator" category can expect to be there for one generation of the product and then, as customers desire this capability and come to rely on it, may be re-categorized as "Desired" and then "Must Have".

III. The Planning Stage

Product development can begin with a business opportunity, compelling new technology, innovative usages, or any combination of the three. In order for a product to achieve broad success, however, it needs to balance all three of these perspectives [3]. To this end, product planning must include business analysis, new technology architecture, and a description of the intended usages for the product. These intended usages are at the core of what this usability grading method uses to provide feedback to developers.

Based on the fundamental business, technology, and usage data, the product features and requirements are written and prioritized so that developers know exactly what the final product must (and in some cases, must not) do. Once features and requirements are specified, those that affect the usage models are categorized into Kano groups in order to drive the usability assessment.

A. Applying the Process

Categorizing the usages, features, and requirements into Kano categories takes roughly the same amount of time as it takes to assign traditional priority. The advantage of using Kano categorization instead of prioritization is that simple prioritization is one dimensional whereas Kano categorization is able to provide important information in addition to priority and motivation at the usage, feature, and requirement level.

Unless a requirement, feature, or usage is removed from the product, all requirements should be implemented before a product can be considered feature complete. By using Kano categories instead of simple prioritization, developers can implement requirements based on related features and how those requirements affect the feature. This enables a logical development flow that groups requirements together based on feature, and prioritizes the requirement implementation. Requirements critical to consumer acceptance of a feature are implemented first, with enhancements to those features being done only after the minimum levels of achievement that define success have been reached.

To further explain how this development prioritization process works, consider the example of a product that adds value to three usage models. The first usage is categorized as a "Must Have", the second usage is classified as a "Desired", and the third is considered an "Differentiator". Each of these usages will have associated features that have their own Kano categorization. For the "Must Have" usage [Fig. 3], each feature could be a "Must Have", "Desired", or "Differentiator" feature depending on how that feature relates back to the usage. In addition, for each feature, the requirements supporting the feature will have their own Kano categorizations which may not necessarily match up to the feature categorization but instead may be based on how the requirement relates to the feature. The priority by which each requirement is implemented is based on the criticality of the requirement to the feature and then the feature to the usage.

Generally, the "Must Have" category has the highest priority. Requirements in this category provide the base level of functionality for the features and usages and in most cases should be developed first in order to establish the product framework. Requirements in the "Desired"

category have the next highest priority until they meet their minimum threshold. Once that threshold has been met, the priority for additional development for the "Desired" requirements lessens until the maximum threshold (point of diminishing returns) is reached, at which point the priority is reduced to nothing. "Differentiator" requirements tend to start out with a lower priority compared to "Desired" requirements, but once the "Desired" requirements have reached the minimum threshold, the priority for developing "Differentiator" requirements increases and surpasses the priority for additional development of "Desired" requirements. Time permitting, additional development of "Desired" requirements should happen until the maximum threshold is reached. The example [Fig. 3], shows a number in the requirement box which represents the order in which the requirements would be implemented. Exceptions can be made due to cost associated with implementation, technology readiness, or a staged release process, but in general this is the way the category priorities work best.

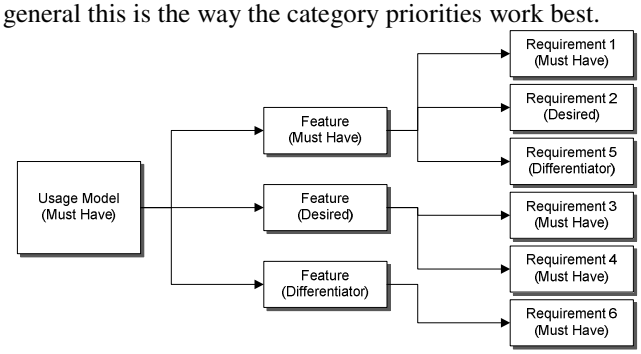

Figure 3: Categorization tiers

Once product development has moved into the development phase, developers can use the Kano-categorized usages, features, and requirements to prioritize the development and validation work. Knowing which requirements relate to which feature provides the ability to assign similar work to the same people, increasing development efficiency. Well-defined requirements aid this efficiency by describing what needs to be implemented and, in the case of "Desired" requirements, the minimum and maximum amount of implementation required.

In some cases, features may relate back to multiple usage models, or requirements may relate back to multiple features. In these cases the Kano category for how each requirement as it relates to each feature and each feature as it relates to each usage model needs to be tracked. Having this information available at the feature and requirement level allows for better decision making, resource allocation, and prioritization. If requirements are altered or removed during the planning or execution stages, a clear impact assessment can be made for each feature and usage related to the requirement being changed.

B. Planning Validation

Kano categorization during the planning stage is not just preparation work for the execution stage of development. Comparing the product's planned usages and features to competitive products can be done to determine if the current usages and features described will deliver a compelling product. By performing a feature analysis using Kano categories of a current product and comparing that to the features of the planned product, a competitive analysis between the two can be performed. Taking into consideration that competing products are likely to be enhanced during the product development period, features, requirements, and usages can be modified to ensure that the product being developed does not just meet the current competition but can compete with potential products that will be available at launch.

IV. THE DEVELOPMENT STAGE

Validating that the product being developed meets requirements is an important part of any product development. In most cases this validation work happens in conjunction with development work to minimize time spent at the end of the project fixing bugs or improving usability. Kano grading fits in well with this form of testing by forcing validation teams to look at each requirement, feature, and usage to verify both *if* it is implemented and *how well* it is implemented. As an added benefit this level of examination quickly exposes any unclear requirements and provides opportunity to clarify confusion between the development, validation, and marketing teams.

Generally, developers implement requirements for a given feature at the same time. These requirements are usually related to one another and verifying that they work together to enable the feature is easier when they are developed simultaneously. As features are enabled in a product, the validation team needs to test to verify that the feature as a whole is meeting expectations. While requirement testing is more geared for verifying that each individual requirement is implemented correctly, testing at the feature level will mostly find functional and usability issues and should be designed to test for these possible issues. Assessment at the feature level is a great opportunity to find these issues and make adjustments before the product moves too far down the path of development.

While most testing is done at the requirement and feature level, assessing overall product usability must be done at the usage level. The usages are what customers of the product most relate to and usages provide the root-level rationale for the user-based features and requirements. If the requirements for the product are written and implemented correctly and any usability issues at the feature level have been resolved, performing the usage-based usability assessment should find few new major issues: issues found at this level mostly center around interoperability between features and minor tweaks to overall usability of the product. If major usability issues are discovered at this level, then either something was likely missed in the planning phase or the market changed significantly in a way

that renders the current product implementation at least partially obsolete.

Usage grading generally lags behind feature assessment much as that feature assessment lags behind requirement testing. As features and usage models are implemented, validation is able to ramp up the testing in these areas and reduce the overall amount of requirement testing [Fig. 4]. As requirements for a given feature are enabled, feature testing can ramp up and replace many of the requirement tests that were previously done. The same transition occurs for feature tests being replaced by usage model tests. Because of these factors, as the project progresses the focus of the testing is more complex and closer to what end users will experience. Over time, an increasingly accurate assessment of usability is gained. However, a base level of requirement and feature testing still needs to be maintained to catch regressions.

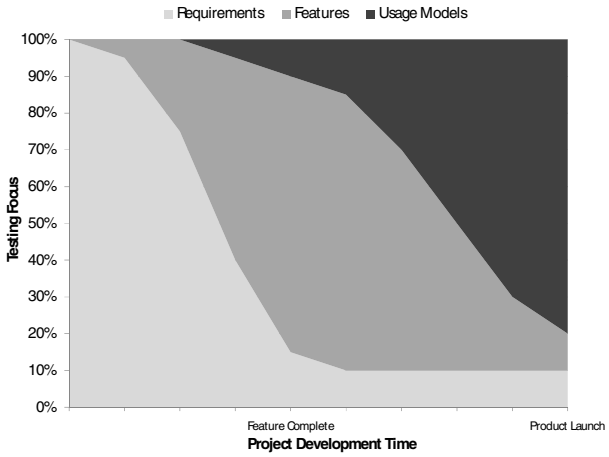

Figure 4: Validation testing mix

A. *Validation Grading*

Validating whether a usage, feature, or requirement is implemented is fairly straightforward with a standard pass/fail result; however, to measure *how well* a usage, feature, or requirement is implemented requires a more robust method. A grading scale is able to handle this in a way that allows for quantification of a subjective assessment. The grading scale this method uses categorizes the usages, features, or requirements into four categories: Not Implemented, Partially Implemented, Fully Implemented, or Implemented Beyond Minimum Requirement. Each category is assigned a numerical value [Fig. 5] which is then used to generate the reporting indicators.

Assessment Result	Point Value
Not Implemented	0.0 points
Partially Implemented	0.5 point
Fully Implemented	1.0 point
Implemented Beyond Minimum Requirement	1.25 points

Figure 5: Kano grading

Usages, features, and requirements not yet implemented score a zero. Partially implemented items score a half point. A "Partially Implemented" assessment may be due to the requirement, feature, or usage being dependent on other requirements, features, and usages. Those requirements, features, and usages that are fully met are given one full point. Items that exceed the minimum, for example a product that performs faster than the minimum customer requirement, are scored at 1.25 points.

Each usage, feature, or requirement is graded separately. Each must be fully implemented or have an approved exception in place before the project is complete. For example, four requirements that are scored at 1.25 do not make up for a single requirement scored at 0. The scores help to evaluate how the project is progressing, identify what areas of the project require more development, and aid in assessing the usability of the product in order to meet project goals.

Generating test results for Kano categorized requirements is different from a standard pass/fail test. For results that would normally be a "fail", tests must be differentiated between "Not Implemented" and "Partially Implemented" results. And "passing" results must be refined into "Fully Implemented" and "Implemented Beyond Minimum Requirement". For example, a test for assessing the requirement "The user interface must support a minimum of 7 profiles and a maximum of 35 profiles" could have one of four possible results. The "Not Implemented" result would be if the user interface did not have profile support at all. The test would be marked as "Partially Implemented" if the user interface had profile support but either the profile support was not currently working (placeholder) or it supported less than the minimum of 7 profiles. If the testing showed that the user interface supported 7 working profiles then a result of "Fully Implemented" would be recorded. And finally, if the user interface supported more than 7 working profiles, then a result of "Implemented Beyond Minimum Requirement" would be given. If more than 35 profiles were supported, the assessment would stay the same; however, customer satisfaction would not be significantly improved by going beyond 35. Each result is then converted to its numerical equivalent [Fig. 5] and can then be used to generate testing indicators.

While performing the grading it may be that "fully implemented" or "partially implemented" scores decrease from one test run to another. Usually this is a sign that a regression has occurred in the product and should be treated just as if any other test had failed. An "Implemented Beyond

Minimum Requirements" assessment result that drops is not cause for a failure report (unless it drops below "Fully Implemented"), however it may indicate that a performance issue is occurring and could warrant additional investigation. As with other forms of testing, the tests used in these assessments must be kept aligned with the current requirements, features, and usages - otherwise false results could be recorded, or defects could be overlooked.

B. Assessing Usability

Usability assessment is a mix of subjective and objective measures. Additional changes can always be made to satisfy some users, but those same changes could negatively affect other users. The usability goal for designing any product is to make the product the most usable for the target users while providing the desired feature set.

To help achieve this goal, the scores are used to assess how well the currently planned requirements, features, and usages are implemented. If the product planners did a good job of identifying the important usages and the requirements were well written, the scores should provide an accurate baseline for the product usability. Increasing scores indicate a more complete and more usable product. For the "Must Have" category, the highest possible score is 1.0 and the average score for all "Must Have" usages, features, and requirements must be at 1.0 before a product launches. Exceptions are allowable with the understanding that if there are "Must Have" requirements, features, and usages that are not scored at 1.0, then customer satisfaction will be greatly affected. For "Desired" and "Differentiator" categories, the maximum score is 1.25. In order to ensure good usability for users, the average validation score needs to be between 1.0 and 1.25 across all requirements, features, and usages. If one of these items is not scored at 1.0 or higher in time for product launch, then customer satisfaction for that feature or usage will be negatively affected and product adoption could be slowed or narrowed. Missing a grade of 1.0 for any requirement, feature, or usage should be considered a test failure and appropriate bug reports filed.

In addition to validation grading, competitive analysis can also be performed using the same Kano category grading technique. The grading of a competitor's product is easier to perform at the feature or usage level as access to their development requirements is typically unknown. By comparing the grading results of the current competition product and the current in-development product, a competitive analysis can be made. Of course, competition does not stand still waiting for other products to be developed, so it is good to perform this competitive analysis every time a competitor releases a new version of their product to keep abreast of new features and usages.

C. Feedback to Development

The results of the validation testing need to be rolled up to the development team to make them aware of any issues found in the testing. As issues are found and fixed, tests must be re-run to verify that the fixes were implemented properly and new issues are not created. Results from competitive analysis can cause shifts in the product priorities. Working together with marketing and development teams, these shifts must be managed to keep the product relevant to customers without significant impact to development schedules.

As a product goes through validation cycles opportunities can arise to root-cause usability issues that were unforeseen when the product requirements, features, and usage models were initially developed. This is where qualitative measures potentially will have to be used in order to identify if the usability issue being seen is something that should be changed or is serving the intended purpose. It can be difficult to fully comprehend how a design will look and feel without being able to interact with it. Leaving open the possibility for small design changes during implementation allows the developer, marketing, and validation people the ability to adapt to a changing market.

V. CASE STUDY

The User Experience Grading process was developed and implemented by a validation team working closely with software development and product marketing. The project was a control panel user interface for a discrete video card. The development team used the Scrum method of Agile development, which worked well with the User Experience Grading process described in this paper. The sprint cycles allowed for regular grading checks and feedback throughout the development process. Due to the timing of the project and the point in which the validation team was engaged, the User Experience Grading process was only used at the requirement level (features and usages were not graded), and only after working code was available to test. Even with these limitations to the process, the value of using the User Experience Grading process over traditional validation methods paid dividends by catching and resolving issues earlier in the development process, forced better requirements management, and reduced development team overhead.

Overall, the case study included 266 requirements graded over 11 sprints (10 months). To date, the User Experience Grading process has helped find over 400 functionality and usability issues with the control panel software. The product this process was used on was an internal-only version of the video card control panel; however, many of the usability issues found on the internal version of the software were used to improve the external version of the same software design which received positive reviews by industry press. One review stated "*[Company] has also been working to improve the user interface of the graphics control panel. What we saw looked much improved over the existing [company] control panels.…"* The ability for validation to engage earlier and provide feedback to the developers in a way that highlighted usability enabled solutions to be developed that could then be used on similar products being developed in parallel.

VI. BENEFITS

Despite applying the User Experience Grading process only to requirements, there were several observed benefits.

Early Detection: Making use of the User Experience Grading process the validation team was able to identify several issues during the pre-alpha phase of software development that could have potentially been missed until much later in the validation cycle. Using the process required the validation team to look at each requirement and grade how well the current functionality compared to the wording of the requirement at the same time subjectively assessing usability for every sprint cycle. Performing the checks as individual requirements were implemented in code by the development team (even before development had fully implemented a feature) enabled early detection and correction of potential issues.

For example, the user interface aspect of a new feature was introduced during a sprint cycle without the underlying hooks into the driver. Validation used the associated testing cycle to assess this new code against the documented requirements and for usability. Validation found logical flow issues that could have turned into significant usability issues. This early identification allowed the development team to make a relatively easy change to the code and resolve the issue before the impact of such a change could become costly.

Better Requirements Management: Another benefit found during the case study was the ability to find and adjust requirements whose wording needed to be updated due to changes in project scope. By regularly grading the code against the requirements and reviewing the grading with development, the team was able to quickly find places where an implementation decision had been made but the requirement had not been updated to match. This constant checking of the code against the requirements naturally created a forcing function to keep the requirements up to date. Of course, when a discrepancy between requirements and implementation was not due to an approved change, it resulted in a defect report rather than a requirements update.

Reduced Development Team Overhead: Before the validation team began using the User Experience Grading process, the development team had kept its own set of the tasks and features they were working on at any given time. Using this list, they determined what parts of the code were complete and what parts were not complete. This work was overhead that the development team had to track on a sprint by sprint basis and realistically should be done by the validation team. By using the User Experience Grading process, the validation team reduced the overhead for the development team while at the same time increasing the quality of the validation feedback.

VII. LESSONS LEARNED

Future projects: It is suggested to use the User Experience Grading process earlier than was possible in the case study. Making use of the Kano categories and competitive analysis in the planning phase would have helped to determine better prioritization and focus on usages and features most important to the target users of the case study product.

Using only the requirements for grading limited the scope of the validation team's feedback to the development team. Expanding the grading to include both the features and the usages would enable validation to take a more holistic approach to evaluating the product's usability. In the case study, feature and usage level usability validation had to be done using "gut instinct" and expensive semi-annual user studies. Identifying and changing subtle usability issues was difficult without having user study feedback available. Having a validation process in place like the User Experience Grading process that is equipped to evaluate the usages, features, and requirements would have reduced the need to host as many user studies and would have enabled faster turn-around time for resolving usability issues.

REFERENCES

[1] N. Kano, et. al., "Attractive and Normal Quality", *Quality*, vol. 14, no. 2, 1984

[2] http://www.ucalgary.ca/~design/engg251/First%20Year%20Files/kano.pdf

[3] E. Simmons, "The Usage Model: Describing product Usage during Design and Development", *IEEE Requirements Engineering*, 2005