7-1-2013

An IT Architecture Enabling Flexible Adjustment Of Exploration/Exploitation Trade-Off

Jörg Gottschlich
Technische Universität Darmstadt, Darmstadt, Germany, gottschlich@emarkets.tu-darmstadt.de

AN IT ARCHITECTURE ENABLING FLEXIBLE ADJUSTMENT OF EXPLORATION/EXPLOITATION TRADE-OFF

Gottschlich, Jörg, Technische Universität Darmstadt, Hochschulstr. 1, 64289 Darmstadt, Germany, gottschlich@emarkets.tu-darmstadt.de

Abstract

The trade-off between exploration of new ideas and exploitation of certainties create a need for managing a balance between those two concepts within organizations. To align with an associated strategy, we suggest an IT architecture with an embedded mechanism to manage this balance when trying new approaches. A prototype and evaluation with encouraging results show the viability of the proposed architecture design for a product recommender scenario.

Keywords: *IT infrastructure, exploration, exploitation, trade-off, recommendation agent*

1 Introduction

A branch of a large international online retailer (several billion EUR annual turnover; 30,000 employees; anonymous for confidentiality reasons) operates an online shop users can subscribe to with their Facebook profile to get product recommendations matching their interests. Since all products are offered and sold by partner stores, the operator has no user purchase history available and it becomes crucial to make proper use of the user's profile data to identify suitable products for him or her. This is a challenging task and literature is scarce on the topic of which Facebook profile data is valuable for product recommendations. Hence, there is a need for experimentation to identify successful approaches in using profile data for product recommendation, as effective product recommenders lower search cost for users and enable shop owners to better satisfy customer preferences (Hinz and Eckert, 2010). It is therefore important to display relevant results to users as early as possible in order to not turn them off from using the shopping site due to disappointing product suggestions.

In this setting, the company faces an instance of the classical exploration/exploitation dilemma described by March (1991). While exploration is important for being innovative to create new opportunities, exploitation plays an important role to benefit from already available knowledge. When competitiveness depends more and more on superior analytical capabilities over the competition (Davenport, 2006), a high flexibility to switch back and forth between experimentation and the quick usage of identified successful approaches becomes a strategic objective (Hitt *et al.*, 1998). Specifically, but not limited to, in an E-Business environment as described, IT support has influence on a company's success (Chan and Reich, 2007; Melville *et al.*, 2004) and need to be able to align with the strategic objectives of the business (Henderson and Venkatraman, 1993).

Hence, to foster the flexibility of an IT infrastructure that aligns with a strategic objective of intense exploration/exploitation cycles, we suggest an IT architecture enabling a quick switch between exploration necessities and exploitation opportunities. In order to do so, the architecture implements a Meta-System which is connected to several subsystems ("candidate systems") – one for each approach that is to be tested ("exploration"). When in operation, the Meta-System receives a user request, selects one of the candidate systems to process the request and presents the output (like the product recommendations in the example above) to the user. Which candidate system is used to produce the output cannot be determined by the user. Feedback provided by the user (e.g. explicit feedback, click-through-rate, purchase) is used to track the performance of each candidate system. Over time, as more and more user requests are processed, the Meta-System is able to identify better performing

approaches and overweigh them in the selection process to make use of their superior performance ("exploitation"). Thus, the Meta-System provides a controlling instance to continuously balance between exploration and exploitation without interrupting operations.

In a case study, we examine the viability of the proposed approach. A prototype implementation for the introductory product recommendation example has been implemented and user feedback data has been collected. We use the data to verify the functionality of the presented architecture. In a performance comparison, we show how such an architecture is able to create a potential benefit surplus.

The rest of the paper is organized as follows: In section 2, we provide the theoretical foundation for the development and understanding of our approach, followed by a methodological introduction in section 3. Section 4 introduces the architecture developed with a specific focus on the exploration/exploitation-balancing component. We show an example run in section 5. The paper concludes with a summary of the results and an outlook on future improvements of this approach.

2 Theoretical Foundation

The relationship between exploration and exploitation plays an important role in organizational development. Following March, exploration of new possibilities includes activities such as "search, variation, risk taking, experimentation, play, flexibility, discovery, innovation", while the exploitation of old certainties is characterized by terms like "refinement, choice, production, efficiency, selection, implementation, execution" (March 1991).

Striking the right balance between those two concepts is a crucial task: Focusing too much on exploitation can lead organizations to be stuck in suboptimal equilibria while engaging too much in exploration without paying attention to exploitation bears the danger of having the cost of experimentation without being able to reap the benefits (March 1991).

One aspect of the relationship between exploration and exploitation is the question of whether the two are orthogonal or continuous concepts, i.e. do they form a zero-sum game such that one can only be increased on cost of the other or can they be carried out rather independently without the need to make a trade-off? (Gupta *et al.*, 2006) The answer to this question depends, among others, on the scarcity of resources (do they compete for resources?) and the level of analysis (individual vs. complex organizations, i.e. can the tasks be spread to be carried out independently?).

In this paper, we take the perspective of exploration and exploitation being continuous concepts as we operate on a scare resource: user requests. Given the stream of incoming user requests, we need to decide if it rather serves explorative or exploiting purposes and every user request can only serve one purpose.

Another question on the interplay of exploration and exploitation is which mechanisms can be used to achieve a balance of those two concepts (Gupta *et al.*, 2006). A "synchronous pursuit of both exploration and exploitation via loosely coupled and differentiated subunits or individuals, each of which specializes in either exploration or exploitation" (Gupta *et al.*, 2006) is called "ambidexterity" (Benner and Tushman, 2003). This type of pursuit can be seen as following a parallel approach as opposed to a serial pattern which is called "punctuated equilibrium" – meaning that periods of exploration are followed by periods of exploitation to form a balanced equilibrium in time (Burgelman, 2002; Gupta *et al.*, 2006).

Why is the balance of exploration and exploitation important to organizations such as companies? Levinthal and March (1993) argue that "[t]he basic problem confronting an organization is to engage in sufficient exploitation to ensure its current viability and, at the same time, to devote enough energy to exploration to ensure its future viability." He and Wong (He and Wong, 2004) found evidence that the interaction between explorative and exploitative innovation strategies has a positive effect on sales

growth rate and conversely, an imbalance between those is negatively related to sales growth rate. Also Kim et al. (2012) conclude that firms may emphasize one of the two concepts at any time, but that over time a balance should be maintained.

Now, seeing the right balance of both exploration and exploitation as a strategic objective, the question arises how IT systems can be aligned in support of this objective. This might especially be important for E-Businesses as their organizational structure constitutes largely of IT systems by definition, yet traditional businesses also benefit from an alignment of IT and business strategy (Chan and Reich, 2007). Additionally, Ten Hagen et al. (2003) stress the importance of exploration inside recommender systems to avoid being stuck in local optima and hence they use an explorative approach to adapt to users when recommending products.

In summation, there is evidence that an architecture enabling companies to dynamically balance exploration and exploitation in IT systems is a relevant task and thus the goal of this paper is to contribute a suitable architectural design.

3 Methodology

Our goal is to create a system that helps to overcome the exploration/exploitation dilemma. As we want to create an artifact, we follow the Design Science paradigm which describes an approach rooted in the engineering sciences. A common methodology in this area is suggested by Hevner et al. (2004). They provide several guidelines which we follow in the construction of the proposed architecture (Hevner *et al.*, 2004):

- **Design as an Artifact:** Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
- **Problem Relevance:** The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
- **Design Evaluation**: The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
- **Research Contributions:** Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
- **Research Rigor:** Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
- **Design as a Search Process:** The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
- **Communication of Research:** Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

The introduction and the theoretical foundation in section 2 show the relevance of the problem of exploration/exploitation balancing. Our proposed architecture, the design artifact, addresses this problem and provides a solution in section 4 which is the result of the rigorous search for a solution to the identified problem of aligning IT to support a strategy of exploration and exploitation balance. Our design is evaluated in section 5. In our concluding remark (section 6) we summarize our research contribution. To communicate our research, we present the results in this paper.

4 Architecture

After giving a short introduction of the implementation context, we introduce our architecture design (see Figure 1).

4.1 Context & Requirements

To get a good understanding of the operating environment of our architecture recall the introductory example at the beginning. A stream of user requests arrives at the Meta-System which has a number of candidate systems attached. These candidate systems provide different implementations for a common task. The goal of the Meta-System is to track performance of the candidate systems and strive for a desired balance of exploration and exploitation.

Considering today's common technical server setups in regard to load balancing or reverse proxy systems, these might offer a convenient implementation context for the suggested architecture – or at least provide a solid technological foundation for productive implementations in a real-world scenario.

The following section introduces each part of the architecture in detail.

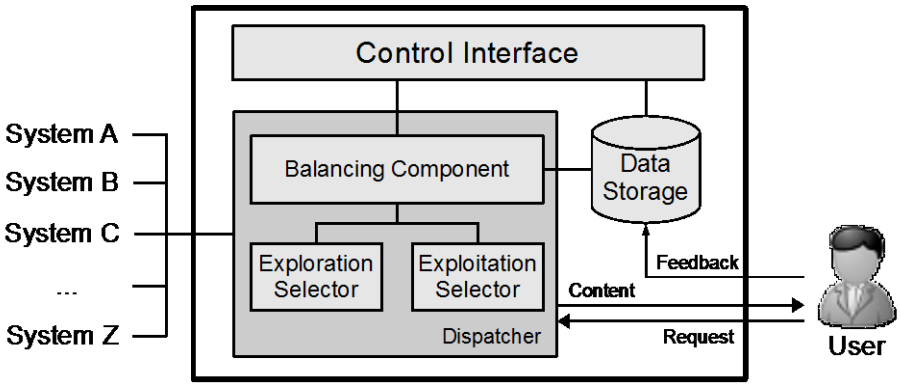

Figure 1. Architecture overview

4.2 Candidate Systems

On the input side of the system there are several candidate systems whose performance should be tested against each other. They either need to provide a uniform interface to the Meta-System or an adapter interface has to be implemented on the Meta-System side which converts the different output formats into a common structure. Please note that candidate systems may use additional data storages or remote systems but as they are perceived as closed systems by the Meta-System architecture those are not shown here.

Candidate systems do not necessarily refer to a system in the narrow sense of a dedicated server system, but can be any kind of comparable logic in a wider sense (e.g. different statistical models which are run in just one software environment or even different instances of the same system but running with different settings of performance affecting parameters).

In the example case we introduced at the beginning of this paper, the candidate systems are the different implementations for product recommenders using information from Facebook profiles (e.g. Age, Gender, Likes, Groups, Friends' Likes) to find matching products.

4.3 User Interface

On the output side there is a common interface to the user, usually (but not necessarily) sending his request over the internet. The Meta-System architecture provides a common interface for all candidate systems. This is important to not create artificial influences by the user interface design which could bias the user feedback from the between-subject experimental design. Additionally, the Meta-System is responsible for registering and storing user feedback.

For an example of the user output see Figure 2. It shows the product list that was generated based on the user's Facebook profile and provides a survey form to collect the user's feedback (in a commercial scenario one would use common performance metrics such as click-through rates, visibility time or sales). This page looks identical for every candidate system, only the selection of products changes.

4.4 Data Storage

A Data Storage connected to the Meta-System stores the assignment history of users and candidate system and collects user feedback for measuring the performance of each system. This data is needed in the Balancing Component to establish a favored trade-off between exploration and exploitation as shown below. In addition, the collected data can be viewed and evaluated manually via the Control Interface, not only to enable monitoring of the learning/adjustment process but also to enable control of the Balancing Component's parameterization.

The data collected in the Data Storage is also available later for additional analyses and can be seen as an asset to potentially provide further analytical insight.

For the example case, the Data Storage keeps the assignment between product recommender and user, the products that are shown to each user and the survey data sent back by the user to validate the recommenders' performance.

4.5 Dispatcher with Balancer Component

The Dispatcher is responsible to send an incoming user request to one of the candidate systems. Its core is the balancer component, a weighted random selection process, which determines how many user requests are forwarded to a candidate system. The balancer component hence controls the frequency a candidate system is used to serve user requests. A candidate system receives user requests to explore its performance; when the data is sufficient to make a decision, the system with the lowest performance is removed and receives no user requests anymore.

The exploration weight $w_{explore}$ is specified by the **Exploration Selector**. In principle, a simple round robin strategy would suffice to give each system an equal amount of user requests. But in a real world scenario, new systems are added at a later time, systems might have downtimes or requests are possibly processed in parallel. Therefore, a data driven approach is more reliable. Our approach computes the selection weight for a candidate system based on the difference Δ_i of test cases system i is missing compared to the system with the highest amount of test cases (Equation 2). If system A has 40 test cases, B has 30 and C 20, the selection weights for the next requests are 1 for A, 11 for B and 21 for C. The addition of 1 is needed as an initial weight and to break ties. z is the number of candidate systems available, n_x denotes the number of test cases already stored for system x.

$$\Delta_i = [\max_{j\in[1;z]} n_j] - n_i + 1 \quad (2)$$

We normalize all the candidate system's deltas, to reach at the explore weight $w_{explore,i}$ for each candidate system i:

$$w_{explore,i} = \frac{\Delta_i}{\sum_{j\in[1;z]} \Delta j} \quad (3)$$

Doing so, the highest weight is put on the system farthest behind in the number of test cases. When a user request comes in, a system is chosen by feeding those weights to the random selection process. The reason we still use a random selection instead of just choosing the system with the highest weight is to be less vulnerable to potential systematic biases in the experimental design.

Depending on the application scenario, other weighting mechanisms can be established. This might especially be appropriate if a new candidate system is attached next to some long running systems which have a long history of test cases. Given the max-distance approach just shown, this system would be likely to take over all user requests from the other systems. This can be intended, but

attention should be paid on the desired approach to exploration weighting when new candidate systems are added.

The **Exploitation Selector** tries to identify and overweigh the most performing system(s) to quickly reap the benefits associated with the use of a high-performing system.

The basis of the performance comparison is an adequate performance metric S – e.g. click rates, Facebook "Likes" or sales (see (Page, 2008) for more examples) – which will be chosen according to the individual application domain.

This metric is checked frequently to evaluate performance development. In our design, the performance is re-evaluated after each "round", i.e. as soon as all systems reach a new common number of test cases (e.g. suppose every system has 4 users, when each of them reaches the 5^{th} user, the performance of every system for users 1 to 5 is evaluated – compare x-axis in Figure 3).

The exact method of selecting and switching recommenders depends on the individual goal of the experiment. This paper focuses on identifying and quickly exploiting one approach out of several possible solutions and hence the overall goal is to quickly increase performance. In different settings, one might rather be interested in the actual differences of systems than a winner or loser decision. Instead of making the claim to provide a one-fits-all approach, we rather suggest that decision process provided here should be adapted to individual needs according to the specific application scenario.

In order to detect the best (or worst) performing system, we test the system with the highest (lowest) performance mean against the performance mean of all other systems within each round. If the difference is significant on a predefined level, it is possible to make the decision and either start using the top performing system exclusively or removing the worst performing system from the selection set (i.e. distribute users only to the remaining systems and continue the selection process). Executing the latter approach repeatedly, also leads to the identification of the best performing system(s) eventually – so be it with a more precautious approach.

Comparing only the lowest and highest mean to the mean of the remaining systems has some advantages:

- Complexity reduction: Instead of $c \cdot (c-1)/2$ tests when comparing each candidate system with each other, we only need to perform two statistical tests: one to test for a potential winner and one to test for a potential loser. Additionally, we avoid the alpha error inflation (and the associated corrections) resulting from the application of multiple tests for pairwise comparisons.
- Quick identification of distinct winners/losers: If the candidate systems' performance spread early into different levels, we can make an early decision while one-by-one comparison might lead to more indifferent findings requiring more test cases to increase test power (Bortz, 2005).

To test the difference of the mean for significance, we use the Wilcoxon-Mann-Whitney test (WMW test) (Mann and Whitney, 1947; Wilcoxon, 1945) which is a non-parametric test and therefore doesn't put restrictions on the distribution of the user feedback. This comes to the price of slight loss in statistical power compared to a common t-test, but increases flexibility and prevents the need and danger of making prior distribution assumptions (we rather collect a few more test cases than making an unjustified decision). However, if reasonable judgment about the distribution of the user feedback is possible, the test used can be adapted accordingly.

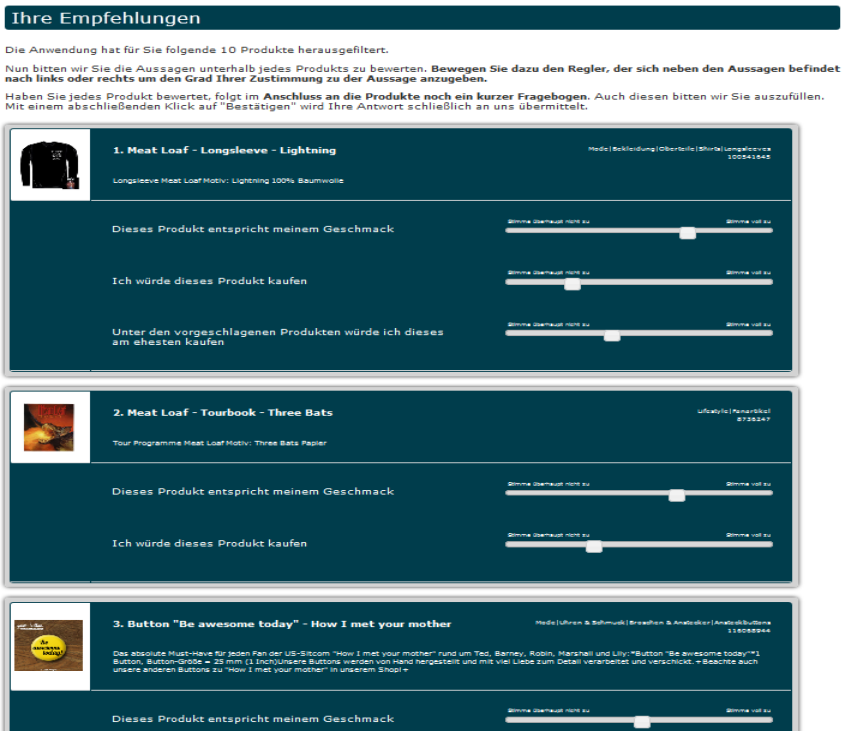

Figure 2. *The User Interface of the Prototype presenting the result of a candidate system to the user*

The null hypothesis of the test is that the two samples are from the same population of performance values. A significant test result indicates that the mean performance of the system in focus differs meaningfully from the mean performance of the alternative systems, thus the judgment over the high or low performance of the candidate system is considered as justified.

If several systems cluster at the bottom or at the top of the distribution (i.e. if their mean performance is similar) the test will not lead to a decision in the current round as, by definition, the systems' performances are considered equal. With increasing number of test cases, the performance differences can still rise to significant values in later rounds as sample size and test power increases (Bortz, 2005).

In section 5 we present an example run which shows how the exploration decision process works over time.

4.6 Control Interface

In order to set the parameters of the iterative selection process (such as confidence requirements), add new candidate systems or monitor the adaption behavior of the Meta-System, a Control Interface is provided that serves administrative purposes.

5 Case Study

To demonstrate the functionality of the architecture presented, we are going to show a sample case in this section. It is based on the introductory example of product recommenders based on Facebook profiles. Of course, it is possible to use other social media user data to create product recommendations (and e.g. let those candidate systems compete against the Facebook-based ones), but for this prototype, we focused on using Facebook profiles only.

5.1 Experimental setup

We built a system that allowed users to log in with their Facebook profile and receive 10 product recommendations based on their profile from a database of roughly 2 million products. The product recommendations are presented to the user as shown in Figure 2. For this experiment, users were explicitly asked via questionnaire to state their satisfaction with the recommended products on 100-points-Likert scales. In a real-world scenario, the collection of user feedback would be done rather implicitly, e.g. via click-through rates, sales figures or similar measures.

The experimental run of the prototype was conducted during July/August 2012 and yielded 162 responses available for analyses. Those 162 respondents were spread among 6 candidate systems (labeled A-E and Z), yielding 27 respondents per recommender.

5.2 Results

Figure 3 shows the average user rating per candidate system. We sum up all of the single ratings of each user (maximum is 3100 points per user) and take the average rating as our performance measure for the candidate systems. It is iteratively recalculated in every round with a growing number of observations. For example, in round 10, the average is computed from 10 observations per recommender while in round 20 it is based on 20 observations, meaning the measure gets more stable over time (compare Figure 3).

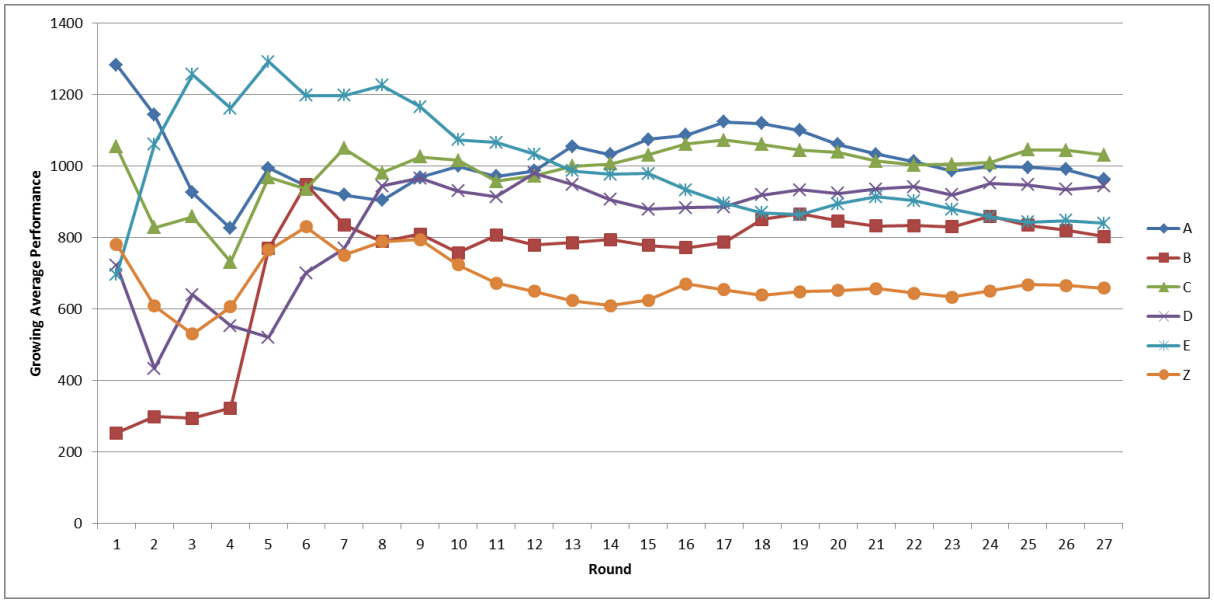

Figure 3. Average Performance of all candidate systems over 27 rounds

Table 1 shows the results of the exploitation checks. Usually the testing for exploitation possibilities starts when a predefined level of minimum observations per candidate system (depending on the specific application domain) is available. For this case, we already start right from the beginning for illustration purposes, to show how significance evolves.

For small N (i.e. in early rounds), the test results are of course unreliable. However, we see how in later rounds the test stabilizes on Z as the lowest performing system. A minimum sample size should be defined before using the test results for a selection decision. In this case Z is the most likely candidate to be removed from the set, as it stabilizes at a low performance level. If Z no longer belongs to the set of candidate systems, user requests are only distributed to the remaining five systems. By doing so, the remaining systems receive a higher amount of user requests, which

accelerates the collection of further data to make a robust decision about the performance of the remaining systems.

Round	Lowest Performer		Common Mean of others	Wilcoxon-Mann-Whitney test			
	System	Mean		p value	conf. int. low	conf. int. high	conf. int. size
1	--	--	--	--	--	--	--
2	B	299	814	0.1212	-200	1174	1374
3	B	294	841	0.0172	91	1083	992
4	B	323	775	0.0100	84	768	684
5	D	521	957	0.1217	-50	948	998
6	D	700	971	0.2502	-271	761	1032
7	Z	750	954	0.5516	-276	628	904
8	B	788	968	0.0891	-68	767	835
9	Z	794	986	0.5221	-233	552	785
10	Z	723	955	0.3360	-151	587	738
11	Z	672	942	0.1939	-97	588	685
12	Z	649	950	0.1092	-55	602	657
13	Z	624	954	0.0527	-4	603	607
14	Z	609	942	0.0357	26	587	561
15	Z	624	948	0.0343	27	566	539
16	Z	670	947	0.0660	-20	519	539
17	Z	653	952	0.0356	24	535	511
18	Z	638	963	0.0166	66	555	489
19	Z	648	961	0.0180	58	529	471
20	Z	651	952	0.0181	46	502	456
21	Z	657	945	0.0194	43	482	439
22	Z	644	938	0.0120	73	477	404
23	Z	633	924	0.0100	70	461	391
24	Z	650	935	0.0111	62	454	392
25	Z	668	933	0.0190	39	431	392
26	Z	665	927	0.0152	43	421	378
27	Z	658	915	0.0145	43	408	365

Table 1. Iterative application results of the WMW test (α=5%)

5.3 Estimate of potential benefits

The prototypical character and the limited number of observations available so far are hardly a base for providing solid measures of benefit. Nevertheless, we like to provide an estimate of potential benefits which we conduct as follows:

- We assume, it is decided in round 27 to remove Z from the selection set
- We use the existing observations to simulate a second pass, but this time assigning Z the average raw performance of all other systems except Z in each round. The logic behind this assumption is that after the removal of Z the users previously sent to Z would rate the other systems similar to the users originally sent to A-E on average. So the second pass reuses the original data set, but for each round the values of Z are replaced by the average rating of its peers in that round.
- We compare the total sum of user ratings of the first pass with the second pass to derive a potential benefit from the decision to remove Z in round 27. Please note that we are using absolute user rating values here (as to measure the total gain from operating the system), while we use average rating to control the candidate system's performance in the iterative exploitation process.

The results can be found in Table 2.

	First pass (Round 1-27 as shown)	Second pass (Simulated round 28-54)	Sum of User Ratings	
			absolute	in %
Base case (Z is not removed)	141,333	141,333	282,666	100.0
Simulation (Z removed after first pass)	141,333	148,296	289,629	102.5

Table 2. Potential benefit simulation results

So under the given assumptions, we see a total surplus of 2.5% (over both passes) by the decision to switch off Z at the end of round 27. It is created by sending users not to Z anymore but rather to one of the other, better systems.

6 Conclusion and Outlook

In this paper, we proposed an architecture providing the capability to dynamically balance between exploring new approaches and exploiting the opportunities discovered by this exploration. We introduced und explained the architecture design and showed an example run with first available data and encouraging results. By an automated in-process control of user request distribution, the user satisfaction can potentially be increased compared to a static setup.

While the task of exploration and exploitation could theoretically be done manually, the suggested architecture enables extensive "online" experimentation without interruption of service – as the user requests are dispatched by the Meta-System, candidate systems can easily be added or removed. Additionally, concise predefined success criteria provide for a proper test setup meeting statistical requirements for a sound methodology. As a positive side effect, an operational data repository grows along with the running system which can serve additional analytical purposes (e.g. traffic or response time analyses, see also (Palmer, 2002)). Thus, the application of this architecture contributes in multiple ways to support a strategy of balancing exploration and exploitation.

The product recommender case shown in this paper and implemented as a prototype is only one possible application from many. For any scenario, where a number of alternative methods exist and user feedback indicates perceived quality by the user, the architecture offers a solution to the exploration/exploitation dilemma. For example when looking at online advertising, there are different approaches of choosing the right banners for a web page visitor – e.g. based on known user characteristics or navigation behavior. Using the proposed architecture, a website operator could implement different models and let them compete against each other, using the click-through rate as a success measure.

The most important next step would be to verify the functionality of this architecture in a larger environment, e.g. within a large company, to create a bigger set of test cases, hence more stable test results and a better estimate of potential (or realized) benefits.

Even though introduced in a setting of generating product recommendations to display on a website, the approach can also be modified to suit other implementation contexts. Whenever there is a need for experimentation on a stream of user requests, an implementation should be considered. The architecture is not limited to interface with the user directly, but can also feed into another intermediate system.

References

Benner, M. J. and Tushman, M. L. (2003). Exploitation, Exploration, and Process Management: The Productivity Dilemma Revisited, *The Academy of Management Review* 28(2): 238.

Bortz, J. (2005). *Statistik für Human- und Sozialwissenschaftler (in German)*, 6th ed., Heidelberg: Springer Berlin Heidelberg, p. 882.

Burgelman, R. A. (2002). Strategy as Vector and the Inertia of Coevolutionary Lock-in, *Administrative Science Quarterly* 47(2): 325.

Chan, Y. E. and Reich, B. H. (2007). IT alignment: what have we learned?, *Journal of Information Technology* 22(4): 297–315.

Davenport, T. H. (2006). Competing On Analytics, *Harvard Business Review* 84(1): 98–107.

Gupta, A. K., Smith, K. G. and Shalley, C. E. (2006). The Interplay Between Exploration and Exploitation., *Academy of Management Journal* 49(4): 693–706.

He, Z.-L. and Wong, P.-K. (2004). Exploration vs. Exploitation: An Empirical Test of the Ambidexterity Hypothesis, *Organization Science* 15(4): 481–494.

Henderson, J. and Venkatraman, N. (1993). Strategic alignment: Leveraging information technology for transforming organizations, *IBM systems journal* 32(1): 472–484.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design science in information systems research, *Mis Quarterly* 28(1): 75–105.

Hinz, O. and Eckert, J. (2010). The Impact of Search and Recommendation Systems on Sales in Electronic Commerce, *Business & Information Systems Engineering* 2(2): 67–77.

Hitt, M. A., Keats, B. W. and DeMarie, S. M. (1998). Navigating in the new competitive landscape: Building strategic flexibility and competitive advantage in the 21st century., *Academy of Management Perspectives* 12(4): 22–42.

Kim, C., Song, J. and Nerkar, A. (2012). Learning and innovation: Exploitation and exploration trade-offs, *Journal of Business Research* 65(8): 1189–1194.

Levinthal, D. A. and March, J. G. (1993). The myopia of learning, *Strategic Management Journal* 14(S2): 95–112.

Mann, H. B. and Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other, *The Annals of Mathematical Statistics* 18(1): 50–60.

March, J. G. (1991). Exploration and Exploitation in Organizational Learning, (W. H. Starbuck & P. S. Whalen, Eds.)*Organization Science* 2(1): 71–87.

Melville, N., Kraemer, K. and Gurbaxani, V. (2004). Review: information technology and organizational performance: an integrative model of IT business value, *MISQ* 28(2): 283–322.

Page, R. (2008). Web Metrics 101 – What DO all these Terms Mean?, *Blog*. Retrieved August 12, 2012, from http://www.makeuseof.com/tag/web-metrics-101-what-do-all-these-terms-mean/

Palmer, J. W. (2002). Web Site Usability, Design, and Performance Metrics, *Information Systems Research* 13(2): 151–167.

Ten Hagen, S., Van Someren, M. and Hollink, V. (2003). Exploration/exploitation in adaptive recommender systems, *Proceedings of Eunite 2003* (634): 10–12.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods, *Biometrics Bulletin* 1(6): 80.