**RC25244 (WAT1204-079) April 25, 2012**
**Computer Science**

# IBM Research Report

## AdaptWatson: A Methodology for Developing and Adapting Watson Technology

**David Ferrucci, Eric Brown**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

**Research Division**
**Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich**

# AdaptWatson: A methodology for developing and adapting Watson technology

IBM Research Report, RC25244
December, 2011

David Ferrucci, Eric Brown
IBM T. J. Watson Research Center
19 Skyline Dr.
Hawthorne, NY 10532

## Abstract

The DeepQA team built Watson in under four years by adopting a metrics-driven research and development methodology, which we call *AdaptWatson*. This methodology relies heavily on disciplined integration of new and improved components, extensive experimentation at the component and end-to-end system level, informative and detailed accuracy analysis, and the efficacious application of highly skilled researchers and engineers to develop, innovate, and invent technologies that together deliver the level of performance required to solve the problem at hand. We present this methodology, highlight the key elements, and briefly compare it to the Agile software development methodology.

## Introduction

The DeepQA team at IBM Research began the development of Watson in early 2007. By the end of 2007 we had an initial implementation of the DeepQA framework. It admitted plug-ins from the team of researchers and performed the full end-to-end question-answering task on Jeopardy! clues. While its performance was poor, the team was in a position to incrementally advance core algorithms, measure results and, based on those results, come up with new ideas and iterate.

As a question answering and decision support system, Watson relies on a combination of sophisticated analytics and relevant content (e.g., books, documents, articles, reports, and other unstructured, semi-structured and structured resources) to analyze the input problem, generate solution hypotheses, and analyze supporting evidence for those hypotheses to compute scores and confidences. With over two dozen researchers and engineers working together to develop analytics and integrate content, it was clear that an ad-hoc approach to developing the system would not succeed. As the project continued into 2008, a rigorous methodology emerged for

rapidly advancing the research, development, integration, and evaluation of over 100 core algorithmic components. We named this methodology *AdaptWatson*.

At a high level, AdaptWatson starts with a traditional software development methodology based on iterative platform development and optimization, and extends it with an iterative training and domain adaptation process. This high level process is shown in Figure 1. The training and domain adaptation process produces new algorithms and content requirements, which drive both the traditional core platform development cycle and the content acquisition and integration process. New algorithms, growing content, and core platform development can generate requirements for system optimization, while system optimization can enable larger data volumes and more computationally demanding techniques. Similarly, core platform development may extend the underlying architecture and framework such that new algorithms and approaches become feasible. New algorithms developed during training and domain adaptation may generate new content requirements, and analysis of new content can produce new knowledge and inspire new algorithms.

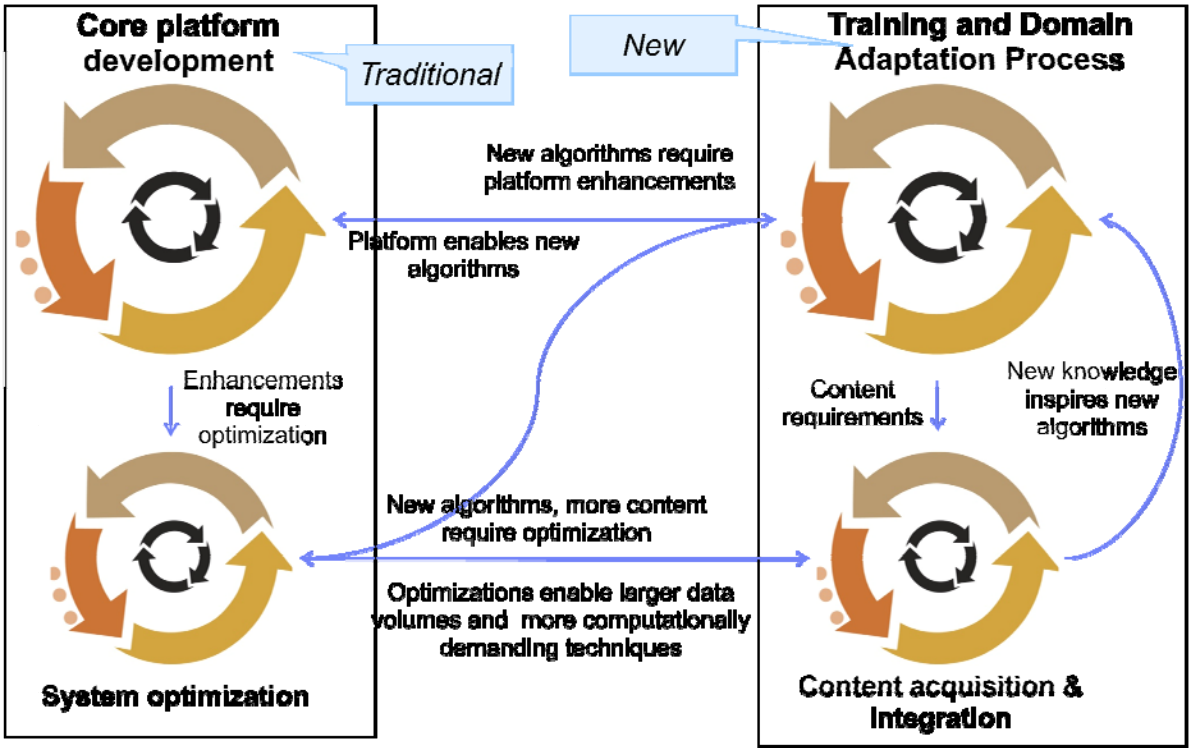


**Figure 1 High level Watson development and adaptation process**

## AdaptWatson

The core of the AdaptWatson methodology is the training and domain adaptation process. The high-level view of training and domain adaptation in Figure 1 expands out to a series of iterative steps, detailed in Figure 2. This figure shows the main steps of the methodology, the inputs and outputs for each step, and the roles or skills of the people required to perform the step. The figure also highlights three key points in the process where various kinds of learning take place, and it notes the steps that make up the core experimental research process.

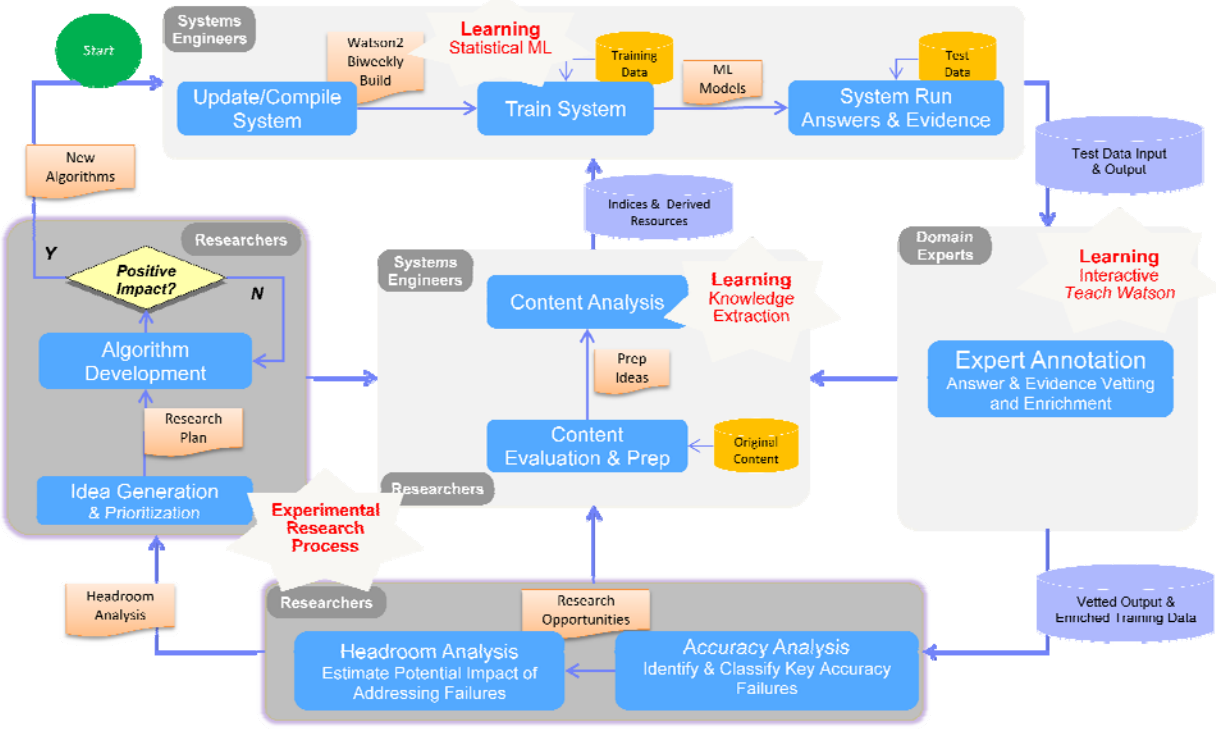


**Figure 2 AdaptWatson training and domain adaptation process**

The development process starts in the upper left corner with a sequence of steps performed by Systems Engineers, who are responsible for configuring, deploying, and running the software. Before the overall methodology can even be invoked, there must be a baseline version of the system to run. Once this bootstrapping process of creating the initial baseline version is complete, the first step in the adaptation process is to update and compile the system, which involves integrating the current release versions of all of the system components and building the system.

After the Systems Engineers have successfully compiled and configured the system, they train the models used by the statistical components in the system, which includes any individual analytics implemented with machine learning algorithms and the final merger component that weighs and combines the results of all of the analytics in the system [Gondek et al., 2012].

Next, the Systems Engineers launch the system run on test data. Test data typically consists of development data that is currently being analyzed by the team to improve system accuracy, and blind data that has not been seen by the team and is used to generate statistically significant performance results. Note that test data is never used to train the system – training and test data must always be kept separate. The output of the system run step is a large volume of detailed results and logging information produced by the system, including, for each question, a ranked list of responses with confidence scores, evidence profiles, scores produced by individual analytics, and supporting evidence.

To accurately evaluate the system run output, the results must be vetted by domain experts. In the expert annotation step, domain experts review the system output and ensure that the answer key used to judge the results for each question is accurate and complete. This step is required because Watson may find alternative answers or answer variants that are correct responses to the input problem, but which were not originally contemplated by the answer key developer. We can also apply this vetting process to the evidence to ensure that the evidence used by Watson to justify the solution is in fact relevant and appropriate.

Once the answers and evidence have been "vetted" by the domain expert, we can accurately measure the performance of the system and analyze issues in the accuracy analysis step. Researchers perform this task using accuracy analysis tools that score the overall test run using a variety of metrics and provide mechanisms for drilling down into the errors. A key element of this process is identifying different error classes and grouping Watson's incorrect responses into these classes. The error classes typically represent a key metric (e.g., binary recall, which suggests a problem with primary search or candidate generation), a processing stage in the pipeline (e.g., candidate generation or lexical answer type detection), a core approach or algorithm (e.g., parsing or term matching), a particular answer scorer (e.g., type coercion), or a class of answer scorers (e.g., passage scoring).

Once a class of problem has been identified, the researchers perform an analysis of the impact of addressing that problem in the headroom analysis step. This analysis produces a "headroom" estimate, which shows the potential impact on overall system performance if the current problem class is solved. This is typically an ideal estimate, since creating a solution that solves the entire class is rare. Nevertheless, these estimates are useful for comparing the potential impact of different solutions competing for resources (mainly researcher time) and deciding which approaches are worthy of investment. This step also helps avoid "mole whacking," where a solution is built that addresses a very narrow problem, often in a very domain specific way. We always prefer solutions that scale to as large a class of problem as possible and try to avoid mole whacking. If the headroom analysis shows insufficient potential impact, development does not continue to the next step in the lifecycle.

Accuracy and headroom analysis may lead to the process in the middle of Figure 2, which addresses content. In this step, researchers explore and evaluate potential content sources to add to the system as primary search and evidence sources. This exploration includes assembling content to address recall (i.e., coverage) issues, as well as issues around providing sufficient evidence to justify a result. Researchers must decide what sort of analytics to apply to new content sources to achieve the most value from the source, and the application of analytics may involve a learning step to train the knowledge extraction components.

New content may contain more information than just textual content. For example, the content may have rich tabular information, link structure based on cross references or citations, and semantic information based on the organization of the text (e.g., separating discussion of a disease into description, diagnosis, and treatment sections). Researchers will explore and invent ways to leverage this non-textual information. Once the content preparation and analysis steps have been finalized, the systems engineers execute the actual content analysis step. This produces indices over the content, as well as any derived resources that result from deeper analysis and knowledge extraction over the text.

After accuracy and headroom analysis, the second phase of the experimental research process is idea generation and algorithm development. This task is conducted by researchers on the team and typically requires highly trained researchers with expertise in a variety of areas, including natural language processing, knowledge representation and reasoning, machine learning, and information retrieval. The idea generation step results in a research plan, and the algorithm development step is an iterative process where ideas in the research plan are implemented and evaluated. If an algorithm is measured to have positive impact, it is added to the next baseline configuration of the system. If an algorithm does not have positive impact, the algorithm development step iterates and the researchers make improvements until either the algorithm has positive impact, or the idea is shelved. After all of the new algorithms have been added to the baseline system, the overall adaptation process repeats from the beginning with the new baseline.

The overall adaptation process iterates until the performance targets of the driving application are met. Over time as the inventory of analytics and algorithms grows we expect that the experimental research process will become a smaller and smaller portion of the overall process. Since the experimental research process is the most expensive part of adaptation, reducing the effort and time spent in that phase will be the key to scaling Watson to new domains.

## Comparison to Agile

The Watson training and adaptation process has some similarities with the popular Agile development process. Table 1 gives a comparison between the Traditional (e.g., Waterfall) software development model, Agile development model [Cockburn, 2006], and the Watson training and adaptation process. Compared to Agile, the Watson adaptation process is driven by

accuracy analysis on large statistically significant blind data sets, not simply to correct defects but rather to inspire a new round of algorithmic generalizations (i.e. *Research*). Keys to successful application of the Watson adaptation process include meaningful and measurable target accuracy requirements and metrics, and access to sufficient quantities of data for training, development, and testing. Sufficient data is absolutely key to training meaningful models and generating statistically significant performance results on blind test data. Without such data, it is very hard, if not impossible, to develop general algorithms in the problem domain.

**Table 1 Watson adaptation versus Agile development**

| | Traditional | Agile | Watson |
|---|---|---|---|
| **Requirements and Design** | Negotiate with customer to create a contract, design specification, and comprehensive documentation | Iterate with customer with **working prototypes** to constantly refine requirements | Establish **target accuracy requirements and metrics** over representative data.<br>**Sufficient quantities of training data** required for learning. |
| **Implementation** | Leverage formal processes and development tools driven by a comprehensive plan | Leverage small teams, extreme programming, and **quick implementation cycle times** to respond to changing requirements | **Rapid research cycle -** Failures found in **accuracy analysis** lead to generalizations and development of **new algorithms.**<br>**Experiments** validated on large enough blind data-sets. |
| **Measurement** | Validate that software meets design specification using testing and benchmarks | Iterate with customer to **evaluate working prototype** and validate requirements are met<br>**Defect triage** used to assign **severity** and **priority** to bugs | **Accuracy metrics** dominate over traditional robustness, speed and scale.<br>**Headroom analysis** used to estimate impact of new or improved analytics and guide investment |

## Conclusions

AdaptWatson defines a methodology for adapting Watson to new domains by starting with a rapid prototyping approach similar to Agile and adding to that an experimental process based on crisp end-to-end performance metrics, accuracy analysis to classify performance issues and guide development and research investment, and explicit allowance for a research phase where new algorithms and solutions must be developed to solve new problems in the domain. Underpinning this methodology is the application of learning to leverage training examples and human expert data annotation to automatically train both analytic components and the end-to-end system.

Unlike traditional software development where requirements analysis leads to a complete specification that describes the solution, adapting Watson to a new domain requires inventing

new technologies for problems that do not have straight forward solutions. This is inherent in any computer technology that attempts to analyze, understand, and leverage human language.

Over time as more analytics are developed, the technology matures, and a foundation of Watson technologies is created in each domain, we expect that the investment required in the Experimental Research Process will taper off. This trend will be essential for the continued growth and scale of the technology.

## References

D. C. Gondek, A. Lally, A. Kalyanpur, J. W. Murdock, P. Duboue, L. Zhang, Y. Pan, Z. M. Qiu, and C. Welty, "A framework for merging and ranking of answers in DeepQA," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 14, pp. 14:1–14:12, May/Jul. 2012.

A. Cockburn, *Agile Software Development: The Cooperative Game (2nd Edition)*, Addison-Wesley Professional, 2006.